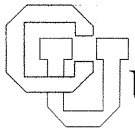
Artists – A New Population of User Interface Designers?

Bent Nielsen

CU-CS-397-88



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Artists-A New Population of User Interface Designers?

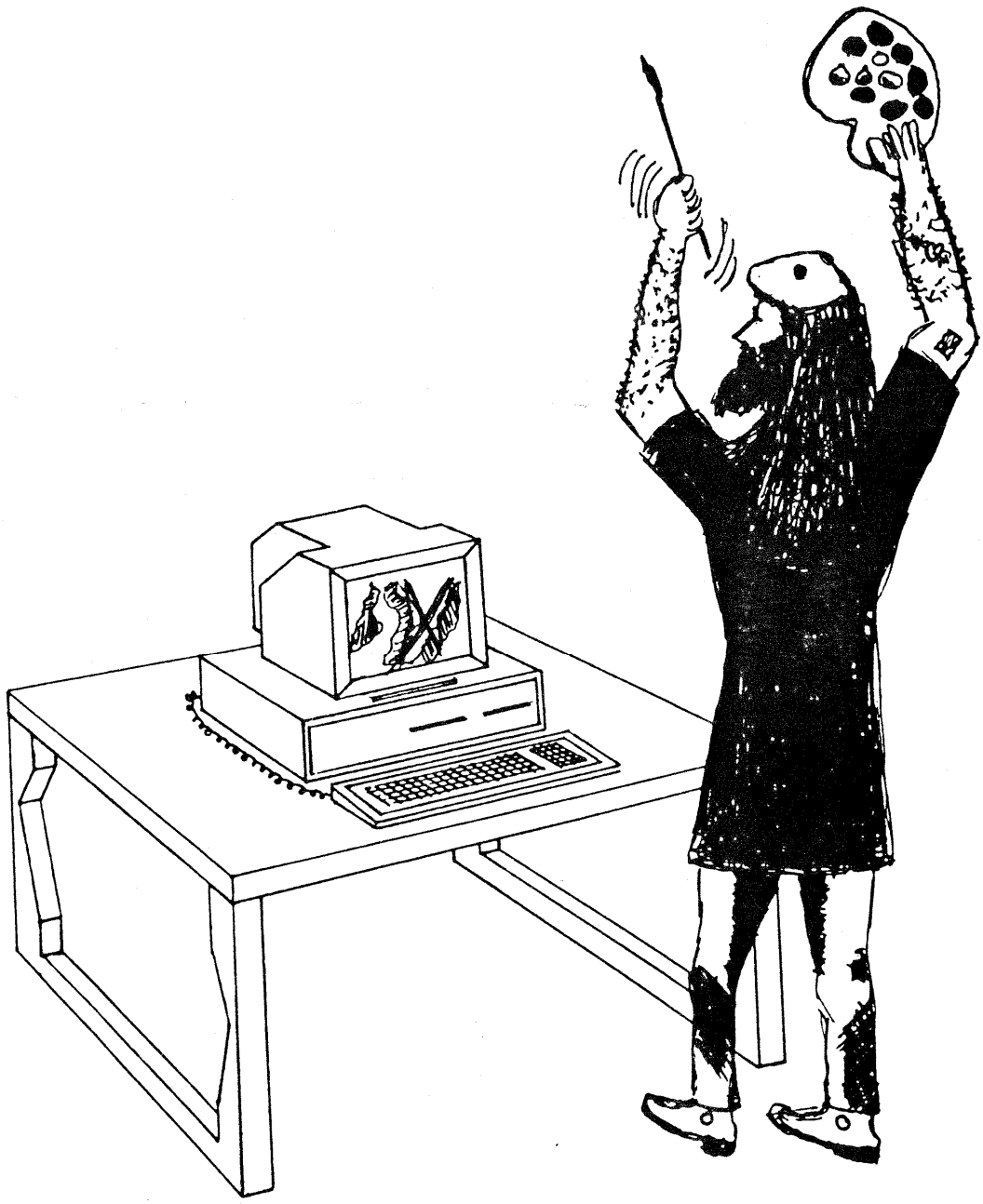
Bent Nielsen

CU-CS-397-88 May 1988

Department of Computer Science
Campus Box 430
University of Colorado,
Boulder, Colorado, 80309

ARTISTS

- A new population of user interface designers



Bent Nielsen

This is the final report to meet the requirements for

the Master of Engineering degree

at the University of Colorado, Boulder

By Bent Nielsen

Spring 1988

ABSTRACT

This report explores the possibility that artists could make a useful contribution to the design of user interfaces. Five artists were asked to use a prototype of a software tool intended to make user interface design accessible to nonprogrammers. Thinking-aloud protocols were collected. The results of this test, together with interviews of the participants, suggest that while artists are willing and able to use computerized design tools they may lack the specialized knowledge to contribute to user interface design. Further, the particular tool used in the test did not seem to provide a good match to the artists' conception of their task.

TABLE OF CONTENTS

1.0 INTRODUCTION

The prices on computers dropped dramatically in the late seventies and the development of the PC made computers affordable to a broader audience. This created a new user class, in which computers were no longer used only by specialists but also by "just folks". This has led to increased emphasis on the design of systems that are easier to learn and to use.

Where will the new design ideas needed for these systems come from? While a variety of approaches to user interface design have been developed, for example by [NORMN83], [HECKL84], [GOULD85] and [KOZAR86], the consensus remains that to a considerable extent user interface design is an art [HECKL84]. Many of the issues involved, such as selection of effective visual representations and choice of appropriate information on which to focus, seem related to problems in the visual arts[IVES82]. Further, one can speculate that aesthetic considerations will play an increasingly important role in consumer decisions in the computer marketplace, and therefore in user interface design.

Against this background the question naturally arises, can artists make a useful contribution to user interface design? If so, what kind of tools would be needed to support their work?

[LEWIS87] describes a prototype software tool, NoPumpG, which uses spreadsheet concepts to permit the creation and control of interactive graphics and animation. One goal of this system is to permit

nonprogrammers, including artists, to create user interfaces without working through intermediaries. This system provides an opportunity to explore the role artists could play in user interface design, as well as to obtain specific feedback on whether the particular approach taken in NoPumpG is viable.

The exploratory study reported here used the NoPumpG prototype as a vehicle to address the following questions, which include both quite general issues as well as some specific to NoPumpG and its design. (1) Would artists be able to contribute to user interface design, given appropriate tools? [IVES82] notes that artists may lack the necessary understanding of the context in which a user interface must function. Further, he notes that communication between system managers and artist-designers may be difficult. (2) Are the concepts in the NoPumpG design natural and easy to grasp for artists? (3) If not, what can be said about the characteristics a user interface design tool for artists should have? [IVES82] raises doubts that artists would embrace any computerized design tool.

1.1 The approach

To explore these questions I asked a group of artists to perform a simple task using the NoPumpG prototype. I collected "thinking aloud" protocols [LEWIS82], in which participants were encouraged to make verbal comments about the task, the tool, and their approach to the problem, as they worked. I also interviewed the participants informally

following the thinking-aloud test, giving them an opportunity to comment more generally on their experience in the test, and their thoughts (if any) on user interface design.

The participants were four professors in Fine Arts and one graduate student in that department. I chose to use professors for two reasons: they were easily accessible in the university setting in which I work, and I felt that their teaching experience might make it easier for them to make useful comments about the study.

None of the participants had any experience in user interface design, or any specific knowledge about it. This results in a bias in the study against finding that artists can contribute to user interface design. That is, if the artists in this group provide evidence that artists can make a contribution, one can assume that this conclusion would only be stronger if more experienced participants were studied. On the other hand, if these participants do not provide positive indications one must allow for the possibility that a specially selected group of participants would give more positive results. This negative bias was accepted because to the fact that very few artists today have experience in user interface design, and so the finding that artists without experience can contribute would have greater practical impact.

1.1.1 The graphics tool

As already mentioned, the NoPumpG system used in the study is intended to permit nonprogrammers to create interactive graphics. It has been

used to a limited extent to create graphical user interfaces, though not by nonprogrammers.

NoPumpG is based on the principle of the spreadsheet. This design was motivated by the observation that spreadsheets have proved to be natural and easy to use for a wide range of users, many of whom are not programmers. The size and position of graphic images are specified by cells in the spreadsheet. When a graphic image is moved, associated spreadsheet cells are automatically updated, permitting graphical input to the spreadsheet. Conversely, if the value of a spreadsheet cell associated with an image is changed, the image is adjusted. This permits graphical output from the spreadsheet. Animation is obtained by making a cell that controls the position of an image depend on a special cell containing a clock.

Consider a line segment; in NoPumpG it will be defined by five cells, one for each of the coordinates of the endpoints of the segment and one extra cell to control the visibility of the line segment.

NoPumpG is currently implemented on the Apple Macintosh using Turbo Pascal and on the HP 9000/300 using common LISP. The version used in these tests was implemented on the Macintosh.

1.1.2 The procedure

I used the "thinking-aloud method" for studying the participants as they used the graphics tool. When using this method the participants are asked to comment on their current activities and thoughts while they work. This method has proven to be very useful in evaluating user interfaces [LEWIS82, MACK82]. I chose the method for this study not only because of its value in providing specific information about the value of the NoPumpG tool for these users, but also because the comments participants make might shed light on the more general question of the ability of the participants to contribute to user interface design. The informal nature of the thinking aloud comments might make them more revealing than the answers to direct questions. A further consideration favoring the method is that it permits a great deal of information to be collected from each participant, in contrast with methods which emphasize measurement of performance, which gather less information from each participant and consequently rely on the availability of a large sample of participants.

The method has drawbacks which must be weighed against these advantages. Making comments while working is not natural, and the presence of an observer adds to the artificiality of the testing situation. The data resulting from the method are poorly structured and difficult to summarize. Traditional statistical methods cannot normally be used, especially when the sample of participants is small. Taking advantages and disadvantages together, the method makes it possible to gather some information about people's performance and factors influencing it, even

when only limited testing is possible, and hence seems appropriate for this study.

Because the thinking aloud procedure focuses mainly on participant's activities during the test, and not on their general attitudes and views, I conducted informal interviews following the test. The interview started with discussion of specific choices participants made during the test, and then was broadened to solicit more general comments about user interface design.

1.1.3 The task.

Participants were asked to create a picture of a boat and make it move across the screen. This exercise required the use of most of the basic features of NoPumpG. A second task, more representative of actual user interface design problems, was prepared but not used: participants were unable to complete the simpler, introductory task quickly enough to go on to the second task. As a result, the test provides information about how easily the artists could work with NoPumpG but not about how they might deal with user interface design problems specifically. Thus the interviews, rather than the test, must be relied upon for judgments on this more general issue.

2.0 METHOD

2.1 Participants

The participants were faculty and one student from the Department of Fine Arts at the University of Colorado, Boulder. Their background are described in Table 2.1. They ranged in age from about 25 to about 55. They were recruited by personal contact, following suggestions from a member of the department faculty, and volunteered to serve.

A:	Graduate student at the department of fine art. Some experience with word processors.
B:	M.F.A, associate professor. Experience with word processing and computer graphics, several years teaching experience.
C:	M.A, professor. Some experience with specifying software for computer graphics and work with computer graphics, several years of teaching experience.
D:	M.A, associate professor. Some experience with computer graphics, 7 years of teaching experience.
E:	M.F.A., associate professor. Experience with word processing and computer graphics. 5 years experience as a professional graphics designer and 7 years experience with teaching.

table 2.1 Summary of five participants' working experience.

2.2 Procedure.

2.2.1 Setting.

The participants were tested individually in an office setting. Two participants asked to be tested in their own offices, while the remaining three were tested in a Computer Science faculty office.

2.2.2 Instructions

As an introduction to the test I explained the thinking aloud method to the participants. I explained that it was important that they comment on their activities and thoughts as they worked, that I was not interested in their "secret thoughts" but only the thoughts which were related to this test, and that I would not help them if they had problems but rather observe them as they tried to solve those problems. I made it clear that this test was neither a test of them nor a test of their abilities to learn how to use a specific piece of software. I explained that it was strictly a test of the software and the more critical they could be the better. I made a complete video recording of the first test, and thereafter made handwritten notes of participants' actions and comments.Notes were taken during the test and video equipment used at one occasion.

2.2.3 Materials and training.

All participants had access to a reference manual describing the NoPumpG software, reproduced in Appendix A. Observation of the performance of the first participant suggested that further material would be needed, so a more discursive tutorial description was prepared and given to the four remaining participants. This tutorial is included as Appendix B.

Since an objective of the test was to see how easily artists would master NoPumpG, only a brief introduction to the system was provided. The general spreadsheet philosophy was explained and two short demonstrations were given. Participants who had not worked with the Macintosh were given a brief introduction, including an explanation and demonstration of the use of the mouse.

2.2.4 The task

The participants were asked to create a boat on the screen and make it move across the screen, and then to attach a line to the boat that would stretch as the boat moved.

2.2.5 Test procedure.

When the instructions had been given participants were free to work on the task in their own way. A video record was made of the first participant's work, and handwritten notes of comments and actions were made during the subsequent tests. I did not intervene with suggestions or help, unless the participant was unable to continue or was having difficulty with the mechanics of the Macintosh. Participants were asked to spend about an hour on the test, but two chose to continue to work for an additional 15 minutes.

2.2.6 Interview.

Following the test I asked the participants to comment on particular problems I had observed or on choices they had made while working. I then invited the participants to suggest improvements to the NoPumpG tool, and to comment on the broader issue of the contribution of artists to user interface design, if they wished to do so.

3.0 RESULTS

3.1 Overall performance.

All five participants were able to complete the test task, but three required significant assistance from the experimenter to do so.

3.2 Using NoPumpG.

3.2.1 Creating an image.

NoPumpG provides a simple bitmap editor, which can be used to create small images like that of the boat in the test task. Participants B and C were able to use this editor without assistance, though they did not discover some of its useful features. In particular, they worked by setting one bit at a time, though the editor supports the setting of many bits in an operation, as in drawing a line. The remaining participants needed help with the editor. All participants were ultimately able to create a reasonable image of a boat.

3.2.2 Use of coordinates.

NoPumpG controls graphical objects by specifying their coordinates on the screen. There were a number of indications that this use of coordinates was unnatural for participants.

If the coordinates of an object become larger than the size of the screen, as happened at least once for each of the participants, the object is no longer visible, but participants were confused by this.

A specific way in which coordinates came to be out of range was that the bitmap editor, used to create small images, required users to click the mouse to indicate where the image should be displayed. Because this requirement was not indicated explicitly, two participants, B and E, unknowingly clicked the mouse in a location outside the display area. The resulting image had coordinates out of range, and although the coordinates were displayed B and E did not understand why the image was invisible, or what to do to move it into view.

Two participants (B and D) experimented with changing the contents of cells containing the coordinates of graphic objects, but the values they selected indicated a lack of understanding of the role of the coordinates. For example, B entered a value of 1000, which is far out of range for the display area.

Participant D had difficulty distinguishing the x and y coordinates of objects, indicating unfamiliarity with the analytic geometry concepts on which NoPumpG relies. Participant B said explicitly, "I do not think in terms of coordinates when I'm designing."

3.2.3 Use of formulae.

While the values of cells can sometimes be specified as constants, nontrivial constructions require that the value of cell be given by a formula, which indicates how to compute the value from the values of one or more other cells. In the test task it was necessary to use a formula to relate the x coordinate of the boat to a cell containing a clock. All participants had difficulty doing this.

Some of the problems may reflect trouble with the way formulae are specified in NoPumpG, in which the operator in a formula must be selected before any operands, or with the sequence of mouse selections needed to enter a formula in a cell. Participant A had to be shown how to enter a formula, but was unable to recall how to do it. Participant C also needed help with the procedure for entering formulae, but then was able to do it. Participant D was able to enter the needed formula without assistance, but only after a number of failures. Participant E had to be shown how to enter the formula, and did not try to repeat the operation later.

Participant B exhibited a deeper problem. Rather than indicating that the x-cell should simply be equal to the clock cell, B attempted first to add the value of the clock to the current value of the x-cell. Unfortunately the attempts to do this resulted in a variety of system failures, since the desired formula refers to itself and leads to loops in the implementation. Some of the attempts involved placing a formula in the clock cell, rather than the x-cell, indicating a basic confusion

about the role of the formula. However, B eventually abandoned this approach and succeeded in discovering the correct method.

3.2.4 Specifying motion.

As just discussed, participants had trouble animating their boat by connecting it with the clock. Participant A indicated explicitly that this seemed unnatural. Sweeping out a path with the mouse, A said, "I just want to specify the motion like this."

3.2.5 Connecting objects.

Participants were asked to connect a line to their boat so that the line would stretch as the boat moved. None of the participants was able to do this in the intended manner, which is to use formulae to compute the coordinates of one end of the line from the coordinates of the boat. Two participants, B and D, were able to get the desired effect, but only be relating both the line and the boat to the clock.In the interview, participant B indicated that there should be a way to connect objects simply by indicating them, rather than having to manipulate formulae.

3.2.6 Miscellaneous problems.

The test revealed a number of specific problems with the NoPumpG prototype, besides those described above. All participants had difficulty with the requirement to indicate an initial position for an image before using the bitmap editor. All participants also had trouble with the operations required in selecting operands in formulae. After an operator is selected the cursor changes form, which is intended to indicate that the user should select an operand. But the participants were confused by this. They expected something to happen when the operator was selected, but not this. Further specific problems are described in Appendix C.

3.3 Suggestions for design tools

Participants made a number or comments during the test or in the interviews that indicate characteristics they believe would be desirable in a tool for their use. As mentioned earlier, one participant said that the use of coordinates, as required in NoPumpG, is unnatural. Another indicated that motion should be specified by sketching a path rather than by relating coordinates to the clock. Participant B had experience with "paintbrush" programs and said that that kind of system was easier to understand than NoPumpG. Participant D also indicated that NoPumpG was not easy to understand, but without suggesting an alternative. Participants B,C, and E indicated that NoPumpG required too much memorization, in that they could not remember how to enter formulae.

Participant B wanted to be able to connect objects graphically, and to manipulate overlapping objects. Participant D suggested that a tool should provide the ability to control font size and character forms.

3.4 The artist as a user interface designer

Since the test included only an elementary exercise with NoPumpG, and not a realistic user interface design task, comments bearing on the contributions artists might make to user interface design came mainly from the interviews. In presenting these comments, I will focus on issues raised by [IVES82].

3.4.1 Do artists understand the context in which user interfaces must function?

None of the participants indicated that they had given any thought to problems of user interface design. Participant E said, "I don't think I can help you with this question [can artists contribute to user interface design] before I have actually tried it in a real project." Participant C said explicitly that he could not contribute to user interface design: "No...I cannot program. I have someone else to do the programming. I specify what I need and they program it....Besides, I don't understand how the program works." But on the other hand, further discussion revealed that the specification C referred to covered a

sophisticated graphics package, for digitizing photographs, that C had designed. So C's self-assessment may not be accurate.

3.4.2 Would artists embrace any computerized design tool?

Four of the five participants in fact had experience with computer graphics, contrary to Ives' speculation. While none of them had an easy time with NoPumpG, aspects of their performance was encouraging. All participants were able to create satisfactory images. Participants B and D were quick to exploit NoPumpG's ability to place cells in any desired position on the screen.

4.0 DISCUSSION

Can artists contribute to user interface design? As noted, the limitations of the test rule out any strong conclusion on this point. The participants were not optimistic, but their appraisal may not be accurate. If they were placed in a situation in which they had an incentive to contribute they might be able to do so.

The NoPumpG tool does not seem to be a natural one for these participants. Its use of coordinates and formulae seemed to cause difficulties, though it is possible that improving the mechanics of the handling of formulae would help the situation. In general, the participants indicated a desire for a tool in which more could be done by graphical operations, where NoPumpG often requires the use of formulae.

It is of course easier to ask for a tool in which more operations are graphical than it is to create one. It is not obvious that a fully natural tool for these participants is possible. For example, indicating animation by sketching a trajectory is workable when the trajectory is known in advance, but what about cases when motion is determined by factors, such as user input, that cannot be known ahead of time? It is possible that some of the unnatural abstraction in NoPumpG is necessary to handle cases like this.

If so, two approaches are possible. First, these participants might be able to adapt to a NoPumpG tool, given more training and practice.

Participants A, B, and C suggested this. Second, one could consider trying to provide more artistic and creative skills for existing interface designers, rather than trying to get artists to adapt to an unfamiliar and unnatural technology. This might run into a problem complementary to the one encountered in the study: people who are comfortable with abstract tools may be uncomfortable dealing with artistic creativity.

4.5 Problems and Limitations of the study

I will in this section discuss problems I encountered during this project and how they limited my exploration of artists' potential contribution to user interface design and their requirements to a design tool.

4.5.1 The participants

I used five people in my tests, but one person did not contribute much interesting information. Thus most of the data are drawn from only four persons. Even within this narrow sample I found differences in experience with computers (see section 2.1) and motivation for working with computer graphics. Some artists were just curious to explore the possibilities of computer graphics whereas others had well defined goals (like the digitizing of photographs). As noted, none of the participants had strong motivation to explore user interface design.

4.5.2 Training of the participants

The results show that the participants would need more training with NoPumpG to be able to tackle problems in user interface design. As a result my findings are limited mostly to observations of basic problems with NoPumpG for this audience, and general comments gleaned for the interviews,

4.5.3 The procedure

Conducting the test on my own created problems. I found it difficult to observe the participants and take notes simultaneously.

4.6 Improving on this study

I explain here how I would attack the problems just discussed in following up this study.

I would test more participants (10 or more would be desirable). A consequence of this decision would most likely be that I would have to find volunteers outside the university community because I could not expect to find 10 artists in the department of Fine Arts who would be interested in participating. A disadvantage of this is that I no longer could take advantage of the participants' teaching experience (see section 1.1). An advantage would be that artists from outside the

university may have more practical experience with computers and thus be able to contribute to the test with more information about their potential as user interface designers.

I would offer an incentive to participants for serving in the study. Participants in this study served only out of curiosity, or to be helpful, and had no clear reason to attempt to master to tool or task presented.

I would also provide a training session with the participants to familiarize them with NoPumpG and its concept. This training session should enable the participants to build small applications with NoPumpG or at least show them how it could be done.

Following this training I would ask participants to perform more realistic tasks than was done in this study. In particular, I would ask them to create an actual user interface give simple requirements I would provide.

I would still use the thinking-aloud method when testing the participant. I would make video recordings of all sessions to avoid loss of information.

5.0 CONCLUSION

Artists may have the potential for becoming user interface designers if they get the right tools and if they can be motivated to learn about the context in which user interfaces must function. Appropriate tools would permit them to work as much as possible in the graphical domain in which they feel at home. NoPumpG is, in my view, a step in the right direction, despite the problems the artists had with it, but more training of the participants, and some specific improvements to the interface, would be needed to assess this.

Acknowledgements.

I thank Professor Clayton Lewis, Professor Kenneth Kozar and Nenny Panourgia for many helpful suggestions about the study and the report. I am also grateful to the anonymous participants whose cooperation made this study possible.

REFERENCES

[ADAMS86] Adams, J. L. Conceptual Blockbusting. 3rd edition. Addison-Wesley publishing company, inc. 1986

[GOULD85] Gould, J. D. & Lewis, C. H. Designing for Usability. IBM Thomas J. Watson Research Center. 1982.

[HECKL84] Heckel, P. The Elements of Friendly Software Design. Warner Books, 1984

[IVES82] Ives, B. Graphical User Interfaces for Business Information Systems MIS Quarterly december 1982, special issue.

[KOZAR86] Kozar, K. A. Humanized Information Systems Analysis and Design. forthcoming from: Mc Craw-Hill Book Company New York, 1989

[LEWIS87] Lewis, C. H. NoPumpG: Creating Interactive Graphics with Spreadsheet Machinery. Dept. of Computer Science University of Colorado, Boulder 1987.

[LEWIS82] Lewis, C. H. Using the "Thinking-aloud" Method in Cognitive Interface Design. IBM Thomas J. Watson Research Center. 1982.

[MACK82] Learning to Use Word processors: Problems and Prospects. IBM Thomas J. Watson Research Center 1982.

[NORMN83] Norman, D. A. Design principles for human-computer interfaces. Dept. of Psychology and Institute for Cognitive Science C-015 University of California, San Diego 1983.

Appendix A

APPENDIX: USING THE NOPUMPG PROTOTYPE

Comments, suggestions, and reports of related work are eagerly solicited. Please pass the software on to any interested people. Every conceivable disclaimer applies to the software: it is a very rough draft!

1 Getting started. Double click the NoPumpG icon. An information screen will appear. Press RETURN to begin work. Select **Merge** from the **File** menu to bring in a demo from the disk. Most of the demos do something when you start the clock, using the **Clock** menu.

2 Cells. To **create** a cell, select cell from the New menu. You will be prompted for a name. Cells have three fields: name, formula, and value, displayed as a stack of boxes. To **move** a cell drag in the name field. To edit a **formula** (or put one in where there is none) click on the formula field. This will enable the Ops menu. Select an operator (see below). If an argument is needed the cursor will change to a cross. Click on the cell you want for an argument. If more arguments are needed the cursor will stay as a cross until you click on enough other cells. Note that the prototype does not support nesting of operators. To build up formulae with more operators you will have to create additional cells. Also, constants cannot be entered as parts of formulae. You must create a cell and put the desired number in it. To edit the **value** of a cell click on the value field. You will be prompted for a new value. To **hide** a cell click on its little go-away box in the name field. To **bring a hidden cell back** select its name on the **Show** menu.

3 Operators. Most of the operators are straightforward. Here are descriptions of the odd ones. The **eq** op just indicates that this cell's value will be the same as that of the argument cell you click. The **const** operator locks in the current value of the cell. The value cannot be changed, by editing or by moving an associated line, until the const operator is removed. The **null** op puts in an empty formula. The ∫ op approximates the time integral of the argument cell; see discussion of clock below. The lc operator is a feature which permits a cell to be updated in either of two ways. The name stands for "last changed", meaning that the cell takes its value from whichever input source changes more recently. If the cell is a control cell (see below) the lc op takes one cell as an argument. The value of the lc will be either the value dictated by the position of the associated line, as for an ordinary control cell, or the value of the cell selected as the argument, whichever has been updated more recently. If lc is used in an ordinary cell it takes two cells as arguments and its value is the value of whichever of these has been updated more recently. Lc can be used to get some of the effects produced by true contraints. For example, to tie two control cells together so that they are equal but either can be updated by dragging, place an lc in each with the other as argument. This is shown in the demo "or lines", where it is used to keep the ends of two movable lines aligned. The **if** op takes three arguments. The first argument is tested. If it is positive the value is the second argument; otherwise it is the third argument. The **?=** op has value 1 if its operands are equal and zero otherwise.

4 Lines. To **create** a line select Line from the New menu. A line comes with 5 cells, called **control cells**, four of which hold the coordinates of its ends, while the fifth determines whether the line is visible. They have names indicating what coordinate they control. The ends can be **dragged**. If one of the control cells for the line has a formula in it that coordinate cannot be changed by dragging but will be determined by the formula. For example, if you use the eq op to make the second y coordinate of a line equal to the first y coordinate, the line will be constrained to be horizontal. The first end can be dragged in any direction, but the first end can only be dragged horizontally.

5 Text. To **create** a piece of text select Text from the New menu. The cursor will change to a text cursor, inviting you to click a location on the screen where you wish to type. Type in the desired text. You can backspace to correct mistakes. Typing is terminated whenever you make your next mouse selection (hitting Return or Enter has no effect.) You get three **control cells** with your text, two of which contain (and control) the coordinates of the beginning of the text string. You can use these to move the text, tie it to another object, etc. Text can be **dragged** if the mouse is placed near the beginning of the text string.

5 Sketches. A sketch is a little picture created by editting a bitmap. To **create** one select Sketch from the New menu, and click the location on the screen where you wish the sketch to appear. An **editing window** will appear on the screen, showing a grid, each cell of which represents a bit in the sketch. To permit you to make a series of related sketches the grid will be initialized to the last sketch you made (or loaded from disk.) Click the mouse to flip a bit on or off. Dragging will flip a series of bits on or off depending on what happened to the cell in which the dragging started. That is, if you wish to turn a series of bits on by dragging you must begin the drag in a cell that is off, and vice versa. A box to the right of the editting grid shows you what your picture will look like on the screen. When you are satisfied, click the go away box on the editing window. Sketches have two **control cells** which can be used to move them, tie them to other objects, and so forth, and a third which controls whether the sketch is visible. Sketches can be **dragged** if the mouse is placed near the point corresponding to the center of the editting grid.

6 Clock. There is a clock cell provided by the system. Use the Clock menu to start it and stop it. You can obtain animation by using the clock as an argument in a formula that computes a coordinate of a line or other object. The demo "**bird**" shows this.

The clock is implicitly used by the ∫ **operator**. When the clock runs, the cell with the ∫ operator is updated by adding the product of the clock time since last update and the current value of the argument cell. This implements an approximation to the time integral of the argument cell. For example, if you create a cell containing the velocity of an object (possibly controlled by a graphical controller) the ∫ op can be used to calculate the position of the object. This is shown in the demo "**igtst**".

7 Visibility control cells. Besides animating things by giving them time-dependent positions, you can devise frame-sequence animation by using the **visibility control cells** of a group a related sketches. To do this you put all the sketches at the same position, but arrange things so that only one is visible at a time in sequence. The demo "**butterfly**" shows this: there are two pictures of the butterfly shown alternately.

Lines and text as well as sketches have these visibility control cells. In each case the associated object is displayed only when the visibility control cell is positive.

8 Deleting objects. Cells can be hidden using their go-away box, and other objects can be hidden using their visibility control cells. But you may want to get rid of something entirely, especially if you intend to export your work, as discussed below. Delete on the Edit menu lets you click on an object to delete it. The object will refuse to go away if it has something else depending on it or on one of its associated control cells, or if it is the clock.

Note: The way dependencies are determined is crude; you won't be allowed to delete something if its control cells depend on each other. You can deal with this by editing the formulae in these cells to remove the dependency.

Note: Deleted objects will still show up in the **Show** menu, but will not be enabled. Try to ignore them.

9 Getting rid of everything. Selecting **New** on the **File** menu flushes all work so you can start afresh.

10 Exporting and importing work. The current contents of the system can be stored on disk by selecting **Save as** from the **File** menu, or by responding yes when you are asked whether you want to save what you have before quitting. Work can be reloaded by selecting **Open**, which will cause what you load to **replace** what you have, or **Merge**, which will **add** what you load to what you have. The dialog is rocky; someday it will be more Mac-like.

You can **Merge** in the same material as many times as you wish. Thus you can save a useful subassembly and then merge it into later work as needed.

Nothing is done about making names unique, so if you do merge in several copies be prepared to find that you have several objects with the same name.

11 Stopping work. Choose **Quit** from the **File** menu. You'll be given a chance to save what you've done on the way out.

12 Housekeeping. When new cells are created, or hidden cells revealed, they may overlay other cells, creating what looks like garbage. You can usually sort things out by moving one of the cells off of the others. To minimize this effect new cells are

created in an area at the bottom left of the screen in such a way that you can create five cells before getting overlap. It is a good idea to move cells out of this area, or hide them, as soon as convenient. A similar approach is used for lines, so it is a good idea to drag a new line away, or otherwise change its position, before creating too many other ones.

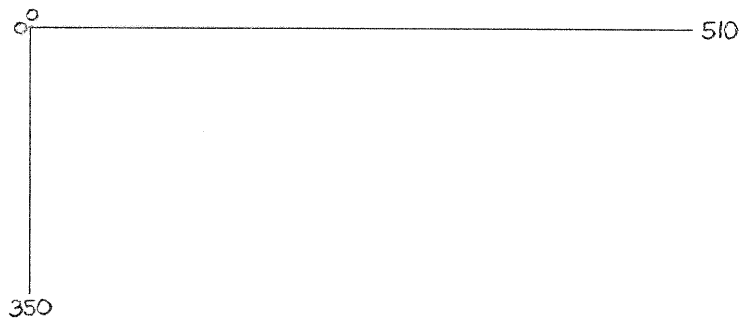
It may also happen that you find it impossible to drag an object that overlaps others. What is happening is the your mouse click is being handled by some object that itself won't move. If you are really stuck you may have to edit some control cells to break the impasse.

APPENDIX B

TUTORIAL

Here is a brief introduction to NoPumpG. This intro is not complete and does not show all the capabilities of the system but shows the basics of the system.

The screen on the Macintosh computer looks somewhat like a peace of paper but consist of dots just like a photography. It has 460 dots along the horizontal axis and 350 on the vertical axis (see fig.).



These dots can be viewed as coordinates or positions on the screen. This means that you can decide where you want to position any image you have created. For example: suppose you have changed the color of "one dot" to say black and you want to position it right in the middle of the screen. This position is half the way from the left of the screen to the right side thus the value is 460/2 = 230 "dots" and is half the way from the bottom to the top of the screen, 350/2 = 175 "dots". These "positions are also referred to as x and y coordinates. The x coordinates are the number of dots in the horizontal direction and the y coordinates in the vertical direction thus in this example x = 230 and y = 175 or (x,y) = (230,175). This is a very important and power full feature of NoPumpG and more examples of this will follow.

On the top of the screen you can see a lot of "words" or "OPTIONS" like: File, Edit, Make, Show, Ops and Clock. This "line" of options is the menu bar. The menu bar contains the commands you can choose to create images in NoPumpG. If you are not familiar with this kind of menu bar move the "arrow" or cursor on the screen to point to one the OPTIONS and click the mouse, you will then see a lot of options showing up on the screen, this is called a pulldown menu. Examples of selecting options from the pulldown menus are given below. You can also look in the appendix for a description of these options.

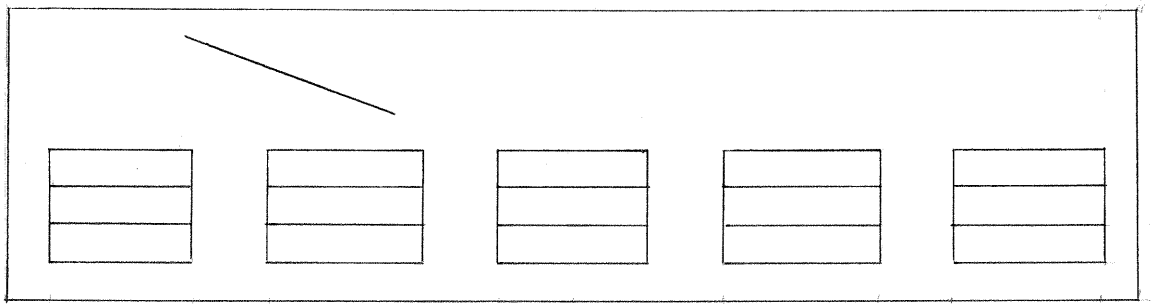
CREATING LINES AND SELECTING OPTIONS FROM THE PULLDOWN MENUS

Point to the option called "Make" and click the mouse, hold DOWN the mouse button and move the cursor down and you will see that the options in the pulldown menu are being highlighted. You select an option by releasing your finger at the highlighted option.

Creating a line:

select Make form the menubar, hold down the mouse button and move the cursor to the option called "line" (the word line is highlighted) and release your finger.

You will then see the following:



The line appears in the corner and five "cells" on the bottom of the screen.

The cells:

The cells have a Name field, a Formulae field and Value field see fig.

Name field
Formulae field
Value field

The Name field is the name of the particular cell so you can find it and reference it any time you want to, just by that name. The name may look like this L1.X1, this means line number 1 and the first coordinate in the x direction (see the description of the coordinates above for an explanation). All cells with L1 before the . (period) belongs to line 1 (the first line you have created).

The Name field is constant and cannot be changed because it refers to a particular line whereas the Formulae field and the Value field can easily be changed. In fact that is how you make an images on the screen to move.

The Formulae field can be a formulae which determines the number in the value field but can also be blank. An example: suppose we look at the

Formulae field in the cell named: L1.Y1 and we want the value to be half the value of L1.X2 we simply say: L1.X2/2. A more elaborated example is given below.

The Value field consists of a number or a position in either the y- or the x-direction (depending on the letter after the dot). An example: L1.Y1, the Y refers to the y-direction.

As you can see there is five cells which belongs to one line: L1.X1, L1.Y1, L1.X2, L1.Y2, L1.V. One line is "described" in NoPumpG by its end points. This means that L1.X1 and L1.Y1 determines the position of the "first" end of line 1 and L1.X2 and L1.Y2 to other end of the line. Try to identify which cell determines which end of the line.

L1.V determines the visibility of the cell. If the Value field contains a negative number the cell becomes invisible otherwise it is visible.

Changing the value in the Value field

This example shows you how you change the value in the Value field (from now on referred to as the value of the cell).

Point to the Value field of any of the cells which determine the position of the cell (that is, the cells with a letter X or Y after the . (period))

Press the mouse button and you will see the question: "New value" in the upper left corner. Type in a new number say 205 and press the RETURN key.

You will then notice that the line moved and that the value of the cell you pointed at (clicked on) has changed to 205.
What happen when you pressed the RETURN key was that NoPumpG realizes that a value has been changed, and that it therefore must show the new value in that particular cell. Since the value of a cell determines the position of the line on the screen (except the .V cells) NoPumpG "moves" the line to the new position. In other words: every time you change a value of a cell NoPumpG automatically updates the cell and moves the line.

Now try to change the value of the L1.V cell (.V) to -1 and see what happen, then try to change it back to a positive number.

THE CLOCK

One of the greatest advantages of NoPumpG is the ability to make the images move (create animation). The way this is done in NoPumpG is using the time to make images move on the screen. The time is controlled by the Clock. You can specify that any line on the screen should move accordingly to the clock. If you want to use the clock you can make it visible by using the Show option on the menu bar.

Try to select the Show option from the menu bar and move the cursor down to the clock option and release the mouse button. You will then see a cell appearing on the screen, this is the clock and the value of the clock is 0. In order to start the clock select the Clock option from the menu bar and chose the START option and the value of the clock cell will increments. You can stop the clock by choosing the STOP option from the clock pulldown menu. When you start the clock again it will automatically start from 0 again.
You will see how the clock can be used in some of the following examples.

ENTERING A FORMULAE IN THE FORMULAE FIELD

In this example you will see how you take advantage of the formulae field and how to use the clock.

Point to the formulae field (the middle field of the cell) of a cell say L1.X1 and click the mouse. You will notice that all the options on the menu bar except the Ops option becomes gray or almost invisible, this means that you cannot select any of these options (you can try it if you want to).

You can only select any of the options from the Ops pulldown menu, these options are the "operators" you can use in your formulae like +, - signs and so on. Try to select = from the Ops pulldown menu (hold down the mouse button and move down the cursor until you have selected the = sign and release the button). You will now notice that the cursor has changed to something which looks like a plus sign (this has nothing to do with addition. it only shows you that you are now changing a formulae in a cell and that NoPumpG wants more input from you). When you have done this move the "plus" sign (cursor) to the clock cell (the cell named clock, read the previous section to learn how to make the clock visible) and click the mouse on that cell. You will then notice that the formulae field looks like: • = clock •.
This tells NoPumpG that, whatever value the clock cell has, put that value in this cell (in this example L1.X1).

Now try to start the clock (read the CLOCK paragraph in order to see how that is done) and notice what happens.

If you let the clock run you will notice that the line is getting longer and longer until it is longer than the screen and it looks like something is wrong. Don't worry part of the line is just "out of bounce" and you can start all over again by stopping the clock and starting it again.

NOTE: if you are about to change the Formulae field and you regret it you must complete the change! And then change it back again if you are not satisfied with the result.

By now you have completed the first part of making "something to move on the screen" let us continue with this in the next example.

CREATING CELLS AND MAKING ANIMATION

Try to select the "cell" option from the pulldown menu at the Make option (point to Make on the menu bar, hold down the mouse button until the word CELL is highlighted and release the button). You will see a new question appear in the upper left corner asking you for a name of the cell (this is necessary because this cell does not belong to a line in that you created the cell and NoPumpG needs a name for it) you can give it any name not used already say MYCELL.

Then change the value of your new cell to 10.

Then try to change the Formulae field in L1.X2 by clicking on that field. Select the plus sign from the pulldown menu at the Ops option on the menu bar and click on your new cell (MYCELL) and the cell named: L1.X1 and notice the change in the Formulae field, the Value field and the position of the line. Notice that you have to click on two cell before the "plus" sign disappears the reason for this is that it takes two numbers for addition (like: 4 + 5) as opposed to one number when using the = sign (as shown before).

As you can see, every time you change the formulae the value gets changed immediately! What happen here is that NoPumpG takes the value in the cell you created (MYCELL), adds it to the value in L1.X1 and puts the result in the Value field of cell L1.X2. Suppose the value of MYCELL gets changed then NoPumpG will automatically realize the change and change the value of MYCELL and also change the value of L1.X2 (since the value of MYCELL has changed), try to change the value of MYCELL to 20.

Now try to start the clock and you should see that the line is moving across the screen until it is "out of bounce" (then just stop the clock, start it again and then stop it again immediately and the line will be back on the screen again).

This is just one example of what you can do in NoPumpG, try to think about other combinations if you want to.

MOVING THE CELLS AND MAKING THEM DISAPPEAR

You can move the cells by clicking on the name field of the cell, hold down the mouse button and move the mouse, release the button when you have moved the cell to a new position.

You can make the cells disappear by clicking in the small square in the Name field of the cell (this is convenient if you have a lot of cells on the screen). You can make them visible again by selecting the Show

option from the menu bar and selecting the cell you want to make visible again (just like the way you made the clock cell visible.

MAKING A SKETCH

Select sketch from the pulldown menu at the Make option on the menu bar and the "bitmap editor" appears. The bitmap editor consists of a lot of squares, each square correspond to a dot on the screen.

A click in a square changes the color of the square: if the color is white it becomes black and vise versa. A copy of the sketch you create is shown in the small window in the upper right corner.

When you have completed your sketch simply click the mouse in the black square in the upper left corner of the editor and it will go away leaving the sketch on the screen.

MORE

There is much more you can do with NoPumpG and these options are described in the appendix.

APPENDIX C

I list here specific comments about NoPumpG that are not central to the discussion in the body of the report, but that may be useful in improving the NoPumpG design.

Some of the participants, especially those who seemed to get the idea of using the formula field, found it hard to understand why it was necessary to create another cell if they wanted a constant value in a formula in one cell. Example: To create the formula 10 + L1.X1, 10 MUST be defined in an additional cell. The solution suggested was to type in the value directly.

When you select an operator the cursor changes shape to a "cross", or a plus sign as it was called by most of the participants. They could not understand why the cursor changed to a plus sign especially when they had selected the "=" from the Ops-menu. They seemed to expect a sign corresponding to the operator they had chosen.

One participant misinterpreted the situation when he tried to use the delete option in the edit-menu. Here the cursor also changed shape to the "plus sign" when he selected operands. He spontaneously said: "No no I don't want to set my cell equal to the clock... I want to delete the cell". He obviously thought that he was about to change the formula in the cell he had just selected, and remembered that the last time he saw the "plus sign" was about to set the cell equal to the clock. It seems

necessary to have different cursors for different modes indicating what is about to be done in the particular mode.

Other comments were about highlighting a selected cell so one always remembered which cell one modified. Also an optional ruler on the screen to remind one of the size of the coordinate system was suggested.

Recursive formulae cause system failures in NoPumpG. The user should not be allowed to create them.

It seems unnecessary to allow the user to change the formula in the clock cell.

It was not obvious to the users how many arguments were needed for a particular operator (even though it was obvious when they thought about it); some indicator of the number of arguments might be helpful.

Overlapping cells were very confusing for everybody. One participant expressed a desire for some easy way to locate particular cells.

It seems to be difficult to understand that an object can move outside the screen. Something which indicated that the object was still "alive" could help.

The system accepts almost any kind of value when you try to change the value in a cell. For example: = 42 is accepted as a value but of course the system cannot operate with anything different from an integer.

Everybody expressed a desire for an UNDO function.

---------- * ----------