

Designing a Programming System for Children with a Focus on Usability

John F. Pane

Computer Science Department

Carnegie Mellon University

Pittsburgh, PA 15213 USA

+1 412 268 8078

pane+chi98@cs.cmu.edu

ABSTRACT

This research proposes the design of a new programming language and environment for children. Emphasis throughout the design will be on usability. I will apply prior results from empirical studies of programmers and the psychology of programming, as well as new empirical studies that investigate areas that have not yet been studied completely. My thesis is that this focus on usability will produce a system that is easier for children to learn and use than existing systems. I will evaluate this thesis through user studies comparing the new system to other programming systems for beginners.

Keywords

Children, end-user programming, programming environments, psychology of programming.

INTRODUCTION

Most children who use computers are not programmers. Yet many would like to create their own software, and to customize the software that they have. Typically, they would like to create programs that are similar to the ones that they use, such as games and educational simulations, which are highly-interactive and graphically-rich. But for this kind of programming task there are no suitable tools that are easy enough for beginners to pick up and use. The powerful programming environments used by commercial software developers are clearly inappropriate for beginners because they are too complex and assume that the user has extensive programming expertise. Other tools that were designed for beginners are effective only in limited domains, such as Visual Basic for creating applications that are based on forms and dialog boxes, spreadsheets for tabular numeric applications, or Cocoa for some types of simulations and games [4]. More general languages for beginners, such as Logo, are easier to learn than languages for professionals, but are not powerful enough to create sophisticated programs that rival commercial software. This gap, between the quality and sophistication of commercial software and the programs that children can create themselves, leads to disappointment and frustration.

There is opportunity to narrow this gap, and my research will use three complementary strategies:

1. Develop a card game metaphor as a new way of thinking about programming. This familiar real-world activity will provide concrete analogs for many of the essential concepts of programming that are currently abstract and difficult to learn.
2. Apply results from research on the psychology of programming and human-computer interaction, supplemented by my own empirical studies, to influence the design of a programming language and environment so that it will be easier to learn and use.
3. Use modern software technology to simplify the programming process, including direct manipulation and demonstrational techniques, programming environment technologies like structure editors, and mechanisms for sharing software components and integrating them into programs.

The result will be a programming language and environment based on the card game metaphor, targeted at the needs and abilities of children and other non-programmers.

METAPHOR

An appropriate concrete model can have a strong positive effect on the usability of a programming language [1]. Many programming languages for professional programmers are based on the very general computational model of the von Neumann machine, which has no familiar physical world counterpart. This research will introduce a new computational model that is amenable to general purpose computation, and yet has a strong metaphorical tie to a familiar real-world system.

I am developing a metaphor for programming around the general concept of a card game because it is familiar to children, and it provides a framework that has concrete representations for the essential concepts of programming. For example, in this metaphor, a program consists of players, representing processes or modules, sitting at a table with a game board on it. Each player holds a hand of active cards, and also has a pile of inactive cards. The player's cards are private, providing information hiding. Cards that are on the table are visible to all players, providing global or shared memory. Cards of different suits hold different types of information, such

as pieces of programs or data. Players take turns executing the rule cards they hold in their hands, a form of cooperative multitasking.

HUMAN-CENTERED APPROACH

More than a decade ago, Allen Newell pointed out:

Millions for compilers but hardly a penny for understanding human programming language use. Now, programming languages are obviously symmetrical, the computer on one side, the programmer on the other. In an appropriate science of computer languages, one would expect that half the effort would be on the computer side, understanding how to translate the languages into executable form, and half on the human side, understanding how to design languages that are easy or productive to use. ... [But] the human and computer parts of programming languages have developed in radical asymmetry. [2, p 212-3]

In this research I acknowledge the importance of the human aspect of programming, and focus on usability throughout the design process. I will utilize studies of the human side of the programming process, as well as general Human Computer Interaction (HCI) principles. I have summarized research about beginner programmers, from the fields of *The Psychology of Programming* and *Empirical Studies of Programmers (ESP)* [3]. Surprisingly, these results do not seem to have influenced the design of popular new languages such as Java. In contrast, the design of my system will be heavily influenced by the prior work, and my research will contribute additional empirical studies to the field.

For example, I have already begun to use empirical studies to investigate how children who have never programmed express their solutions to a set of programming tasks. Some of my preliminary findings are:

- Most program control was expressed in an “event language” or “production system” style, with rules controlling behaviors (e.g. “If PacMan loses all of his lives, it’s game over”).
- Often the general case was elaborated first, and then exceptions were listed afterwards (e.g. “When you encounter a ghost the ghost should kill you. But if you get a little pill you can eat them”).
- Operations such as counting, which are traditionally implemented with iteration, were expressed implicitly by operating on sets of objects (e.g. “When PacMan eats all of the yellow balls he goes to the next level”).
- Continuous motion of objects was expected, and commands were issued only to change the motion (e.g. “If PacMan hits a wall, he stops”).

These tendencies, and any others that I discover, will influence the design of my system.

TECHNOLOGY

This research will apply a broad set of technologies to make the system easy to use. For example, the system will

take advantage of user-interface technologies such as direct manipulation and demonstrational techniques, as well as programming environment technologies such as structure editors, to reduce tedious operations and make it easier to build error-free programs. The system will also include software engineering technologies such as component integration mechanisms to enable users to extend their programs by harnessing and reusing existing resources (e.g. other programs or data on the worldwide web).

DOMAIN

My system will be capable of producing a full range of programs. However, the most successful end-user programming systems are domain-specific, so my system will focus on the class of software that children would like to create, such as games and educational software. These are highly-interactive and graphical, and contain animations, simulations and multimedia. The card game metaphor, along with the graphics primitives that I build into the system, will facilitate the creation of programs within this domain without sacrificing the general capabilities of the underlying programming language. This is similar to the way that the turtle metaphor in Logo transformed a general purpose language into one that facilitates the building of programs based on “turtle” graphics.

THESIS STATEMENT

My thesis is that a programming system that uses a card game metaphor for computation, and embodies principles from the psychology of programming and human-computer interaction, will enable children and other non-programmers to create sophisticated programs containing animations, simulations and multimedia. The new metaphor and language, with the supporting technology in the environment, will make learning to program easier for children, and yet will be more powerful than existing systems that were designed for this audience.

REFERENCES

1. Mayer, R. E. *The Psychology of How Novices Learn Computer Programming*. In *Studying the Novice Programmer*, E. Soloway and J. C. Spohrer, Eds. (Hillsdale, NJ, 1989), Lawrence Erlbaum Associates, 129-159.
2. Newell, A. and Card, S. K. *The Prospects for Psychological Science in Human-Computer Interaction*. *Human-Computer Interaction, 1 (3)*, (1985), 209-242.
3. Pane, J. F. and Myers, B. A. *Usability Issues in the Design of Novice Programming Systems*. Carnegie Mellon University, School of Computer Science Technical Report CMU-CS-96-132, (Pittsburgh, PA, August 1996).
4. Smith, D. C., Cypher, A., and Schmucker, K. *Making Programming Easier for Children*. *interactions, 3 (5)*, (September/October 1996), 59-67.