Presented at *Children's Programming Odyssey*, HCC 2001 IEEE Symposia on Human-Centric Computing Languages and Environments, Stresa, Italy, September 5-7 2001.

Human-Centered Design of a Programming System for Children

John F. Pane
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+1 412 268 8078
pane+hcc01@cs.cmu.edu

Abstract

HANDS is a programming system for children that is designed to be easier for beginners to learn and use than other existing systems. Throughout the design and development of HANDS, usability has been a primary focus, guiding the generation of ideas for the system's features and the selection among various choices. This document presents an overview of some of the prominent features of HANDS along with the research that motivated their inclusion.

1. Sources of Design Guidance

A review of prior research in the areas of Psychology of Programming (PoP) and Empirical Studies of Programmers (ESP) has yielded a wealth of information about the problems that beginners have with programming [1]. In addition, the field of Human Computer Interaction (HCI) offers general principles to be applied during the design process. The HANDS design was guided by awareness and application of this prior work.

In several areas where the prior work left open questions about design issues, we have conducted new studies to examine the questions. For example, researchers have established that existing programming languages do not speak the user's language, and do not provide a close mapping between the programmer's mental plan and the language's statements and operators. Two new studies examined the language and structure of non-programmer's solutions to problem solving tasks [2], in order to design HANDS to match these natural expressions, untainted by exposure to programming. Later, when it became clear that queries would be an important aspect of the HANDS system, another study was conducted to examine methods for expressing queries [3], because the prior work clearly established that the boolean operators (AND, OR, NOT) are problematic.

2. The HANDS Design

This human-centered approach to programming system design has impacted almost every aspect of the HANDS system. This paper discusses five important features of the design, along with a brief summary of the motivating research.

2.1. Model of Computation

Researchers have noted that the *von Neumann* computational model is difficult for beginners because it is unfamiliar outside the domain of programming, and it has no real-world counterpart. It is hard to visualize the program's data because it has no concrete representation. Often it is not evident whether a data element is allocated at some particular time or accessible from some particular place in the program. HANDS attempts to relieve these problems by providing a familiar concrete representation of the program: an *agent* at a *table*, reacting to events and manipulating information on *cards*. All of the program's data is stored on cards, which are global, persistent and visible on the table. Each card must have a unique name, which is not case sensitive. There are no local variables. The front of each card contains a list of name-value pairs called properties. Several properties are always present: the *cardname* property holds the card's name, and the *x* and *y* properties contain the card's position coordinates. The programmer can add more properties as needed, so cards are similar to records (or structs) in other languages. Each property on a card has a unique name, which is not case-sensitive. The programmer refers to the nectar property of a card named flower with *nectar of flower* or *flower's nectar.*

2.2. Event-Based Style

The prior research offered few recommendations about which style of programming to select, except that full-fledged object-oriented programming is very difficult. So this question was examined in the first study of how non-programmers naturally express their problem solutions. This study identified *event-based* programming as the most prevalent style in the participants' solutions, so HANDS adopted this style. A program in HANDS is a collection of event handlers that are automatically called by the system when a matching event occurs. Inside an event handler, the programmer inserts one or more imperative statements to execute in response to the event. After these statements are

executed, control returns to the system, where the next event is dispatched. The system is not object-oriented. There is no inheritance, and all of the code is centrally located and is not encapsulated with the program's data.

2.3. Queries for Data Access

In our studies, we observed that users do not maintain and traverse data structures. Instead they perform queries to assemble a list of objects on demand. For example, they say "all of the blue monsters." HANDS provides a query mechanism to support this. The query mechanism searches all of the cards for the ones matching the programmer's criteria. Queries begin with the word *all*. If a query doesn't specify to look in a particular property, the query returns all of the cards that have that value in any property.

Queries depend on the accurate specification of boolean expressions, which is an area that is very difficult and has been studied extensively. The common uses of the words AND and OR in natural language lead to errors when these words are used to name the boolean operators in queries, and the intended scope of the NOT operator is ambiguous. To address these problems, we examined different keywords and structures for specify boolean expressions, as well as a tabular alternative called *match forms* that we designed to be similar to cards. On a match form, all of the listed values implicitly form a conjunction. Negation is specified by prefacing a value with the NOT operator. Disjunction is specified by including an additional match form adjacent to the first one. In our study, match forms were more effective than the other methods we tested for specifying queries, including textual boolean expressions. So, HANDS will provide match forms for specification of queries.

2.4. Aggregate Operations

Loops have been repeatedly identified as troublesome for beginners. Researchers have identified ways to improve performance by redesigning loops, yet these solutions have not been adopted by most languages. More importantly, many languages force users to use iteration in situations where aggregate operations could accomplish the task more easily. In our studies, the participants used aggregate operators, operating on whole sets of objects in one statement rather than iterating and acting on them individually. HANDS supports aggregate operations. All operators accept lists as well as singletons for their operands, or even mixture of lists and singletons.

2.5. Domain-Specific Support

The programmer must translate a mental plan into language-level operations. When the language does not provide a direct mapping for a particular mental operator, the programmer has to compose lower-level language primitives to achieve the goal. This synthesis has been identified by researchers as one of the greatest cognitive barriers to programming. One example of this barrier was mentioned above: some languages require the use of loops to synthesize operations that the programmer may otherwise consider to be atomic. This issue also motivates the inclusion of domain-specific support into a language.

HANDS provides domain-specific features that enable the programmer to easily create highly-interactive graphical programs. The *back* property of a card, if present, is automatically displayed on the back of the card when it is face-down. If the value of this property is the name of a file containing an image, the image is displayed on the back of the card. Otherwise, the value in the property is displayed literally on the back of the card. Any card that contains slots named *speed* and *direction* are automatically animated by the system without any programming. Direction specifies an angle expressed in degrees, using the convention that zero points to the right and increasing angles go counter-clockwise. Since some users may not be familiar with this convention, an image of a compass is available in HANDS for the user to refer to when working with directions. Speed is a relative value, and can be positive or negative. Also, HANDS uses events to make it easy for the program to react to user input via the keyboard and mouse.

3. Summary

These five features of the HANDS design are only the most prominent of many factors in the human-centered approach that was taken during its design. It is expected that this approach will lead to a system that is easier for children to learn and use than any other existing programming system. We will conduct user-studies to evaluate this.

Acknowledgments

The author would like to thank Brad Myers and Albert Corbett, and the many others who have contributed to this work. This research is funded in part by the National Science Foundation under Grant No. IRI-9900452. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

[1] J. F. Pane and B. A. Myers, "Usability Issues in the Design of Novice Programming Systems," Carnegie Mellon University, Pittsburgh, PA, School of Computer Science Technical Report CMU-CS-96-132, August 1996.

[2] J. F. Pane, C. A. Ratanamahatana, and B. A. Myers, "Studying the Language and Structure in Non-Programmers' Solutions to Programming Problems," *International Journal of Human-Computer Studies*, vol. 54, pp. 237-264, 2001.

[3] J. F. Pane and B. A. Myers, "Tabular and Textual Methods for Selecting Objects from a Group," in *Proceedings of VL 2000: IEEE International Symposium on Visual Languages*. Seattle, WA: IEEE Computer Society, 2000, pp. 157-164.