A Usability Study of End-User Construction of
Direct Manipulation User Interfaces

T. Paul McCartney

WUCS-96-26

October 1996

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

A Usability Study of End-User Construction of Direct Manipulation User Interfaces

T. Paul McCartney
Department of Computer Science
Washington University
St. Louis, Missouri 63130, USA
paul@cs.wustl.edu
http://www.cs.wustl.edu/cs/playground/euphoria/

Abstract

This paper describes an empirical study of end-users that tested the usability of *The Programmers' Playground* graphical environment. The Programmers' Playground is a software library and run-time system for constructing distributed multimedia applications. Playground's graphical environment enables end-users to create direct manipulation graphical user interfaces (GUIs) and to dynamically configure communication among distributed application components. In this study, 28 end-users with no prior experience in distributed computing or user interface construction were timed and evaluated on several tasks using our graphical environment. Tasks included the use of direct and indirect constraint relationships, visual configuration of distributed applications, and graphical user interface construction. The results show that a wide variety of end-users (i.e., not just programmers) can learn and apply these concepts, utilizing our graphical environment to construct distributed multimedia applications.

1. Introduction

This paper describes an empirical study of end-users that tested the usability of the Programmers' Playground graphical environment. The Programmers' Playground [2], [3] is a distributed programming environment that aids both programmers and end-users in the construction of distributed multimedia applications. Here, we focus on the evaluation of end-user mechanisms available, including a graphical tool for assembling distributed multimedia applications through graphical configuration of software components (Section 2.1) and interactive construction of direct manipulation graphical user interfaces (Section 2.2).

This study, like other similarly designed usability studies [1], [4], [5], tests the usability of our system through a variety of end-user tasks. The tasks were designed to be representative of the tasks encountered during distributed multimedia application construction with our graphical environment. This included the use of direct and indirect *constraint relationships* for establishing behavior among GUI components, *visual configuration* for specifying a distributed application's communication structure dynamically at run-time, and *application-GUI integration* for associating an end-user constructed GUI with an underlying application.

In addition to testing the overall usability of the system, we were also interested in comparing the performance of end-users with different levels of computer science experience. We conjectured that to be truly usable, the graphical environment should not be biased toward users with a higher degree of technical training. That is, novice end-users who have little or no programming experience should be able to use our system as effectively as expert end-users who have extensive programming experience.

2. Background

Distributed multimedia applications supported by a global electronic infrastructure have tremendous potential for providing users with customized communication and computation environments. Applications include remote collaboration, information and resource sharing, and access to broadcast media. Future users of the infrastructure will vary greatly in technical skills, ranging from novice users to sophisticated expert users and programmers. Since communication and computation requirements vary by context and change dynamically, it is unlikely that off-the-shelf applications will anticipate the needs of all users. Therefore, empowering end-users to create their own customized applications for both communication and computation is an important challenge. Support for end-user construction of distributed applications means not only that users can combine software components, but also that they can construct graphical user interfaces for interaction with these custom applications.

The Programmers' Playground is a software library and run-time system for creating distributed multimedia applications. Playground is based on *I/O abstraction*, a new programming model for distributed systems that provides a separation of communication and computation. A distributed application consists of a set of communicating *modules*, written as C++ programs, using special data types from the Playground library. Variables declared using these data types can be *published*, making information available to external modules. The communication structure among the modules of a Playground application is defined by a set of *logical connections* among published variables of the modules. Communication among modules of a distributed system occurs implicitly; when a published variable is modified, the new value of the variable is automatically sent to its connected variables. In this way, each module can be created independently of the modules with which it communicates.

2.1 Visual Configuration

Active software modules of a distributed system are viewed through the use of a visual configuration language. Modules are represented as boxes with each published variable represented as a color-coded "plug." End-users can assemble a distributed application by configuring communication among collections of modules run-time. This is accomplished by simply drawing connection arrows among the modules' published variables.

For example, Figure 1 shows three modules, Earth, Moon, and EUPHORIA, comprising a planetary orbits simulation. The Earth module simulates the orbit of the Earth over time, and publishes simulation information through two variables `Ep` and `Ed` (position and diameter). The Moon module simulates the orbit of the Moon in relation to the Earth's position publishes three variables, `Ep`, `Mp`, and `Md`. `Ep` is used to receive the Earth's position from the Earth module; `Mp` and `Md` are the simulated position and diameter of the Moon. The EUPHORIA module represents an animated graphical display of the planetary orbits simulation. Each of the Earth and Moon's published variables are configured to the EUPHORIA module through the use of logical connections. Changes to the Earth and Moon's published variables over time are communicated to EUPHORIA, resulting in a real-time distributed animation.

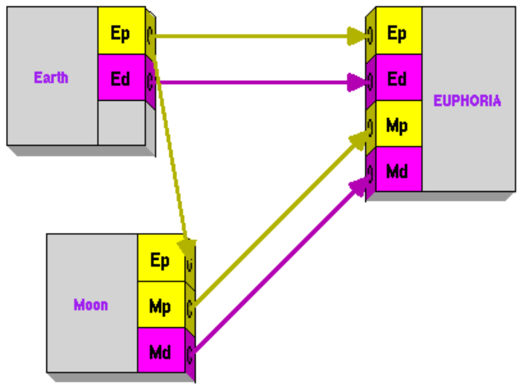

Figure 1: Configured modules of an planetary orbits simulation.

2.2 EUPHORIA UIMS Module

EUPHORIA, Playground's user interface management system [2], [6], [9], is a specialized module for creating customized direct manipulation GUIs without the need to write user interface source code. In EUPHORIA, end-users simply draw GUIs using a graphics editor (Figure 2). Attributes of a user GUI are selectively published, exposing their state to the external environment. This allows other modules of a distributed application to selectively view or control the state of EUPHORIA's display. For example, in Figure 2 the center position of the Earth's picture is published as variable `Ep`. When an external change is received for `Ep` (i.e., communicated through a logical connection), the system responds by animating the Earth's picture to the appropriate location. Similarly, changing graphics items associated with published variables within EUPHORIA results in the communication of the new state information to external modules.

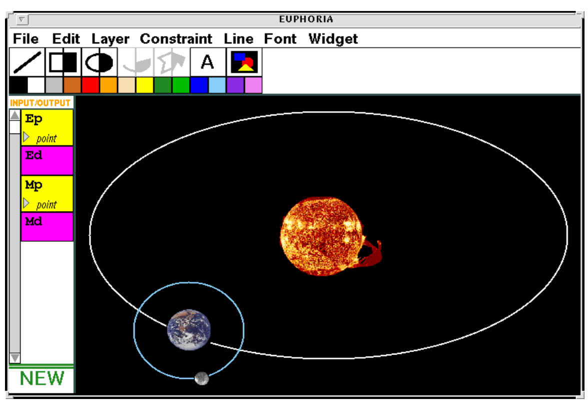

Figure 2: EUPHORIA display of the planetary orbits simulation.

EUPHORIA supports a number of mechanisms for interactive construction of GUIs. A *constraint* is a persistent relationship to be maintained among graphics object attributes. In EUPHORIA, constraints can defined graphically by the end-user using each graphics object's handles. For example, an oval can be inscribed within a rectangle through the use of two constraints (see Figure 3). A constraint is formed between the top-left handles of the oval and rectangle, causing the shapes to "snap together" (established constraints are shown here as lines). A second constraint is formed between the bottom-right handles, resulting in the oval changing shape to fit into the rectangle. Since these relationships are persistent, resizing or moving either of the shapes causes the other to also change. The details of EUPHORIA's underlying constraint solver can be found elsewhere [7].

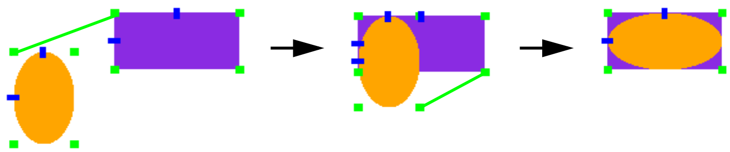

Figure 3: Inscribing an oval within a rectangle.

In EUPHORIA, constraints can be selectively viewed, revealing the established relationships among graphics objects. Equality constraints are shown as either arrows between graphics object handles (if handles are sufficiently far apart) or circles (if handles overlap); anchor constraints, or fixing an attribute to be constant, are shown as squares over handles (see Figure 4).

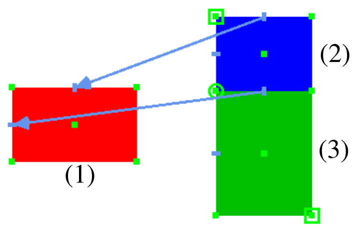

Figure 4: Constraint visualization.

Constraints are an intuitive way of establishing *direct* relationships (e.g., equality) among graphical attributes. However, many times it is necessary to create *indirect* relationships among graphics attributes where there are one or more intermediate computations involved in relating the attributes. EUPHORIA's *imaginary objects* mechanism allows end-users to define relationships among graphics objects through the use of intermediate graphics objects, serving as an abstraction for defining indirect constraint relationships.

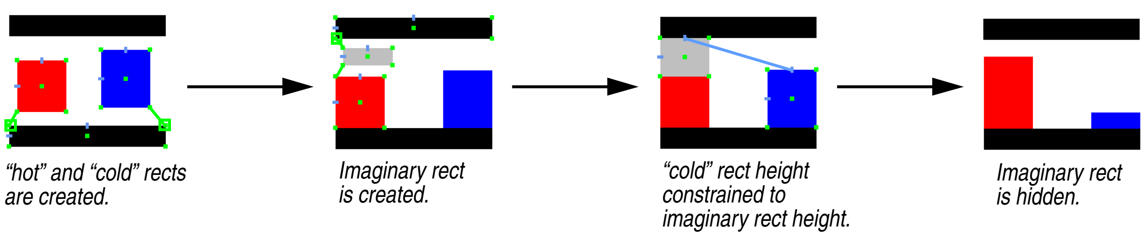

Figure 5: Using imaginary objects to construct a temperature controller.

For example, an imaginary object can be used to create a temperature controller (Figure 5) that displays the proportion of "hot" (shown as a red rectangle, left) to "cold" (shown as a blue rectangle, right). The desired behavior is that the red rectangle height should always be inversely proportional to the blue rectangle's height (e.g., dragging the red rectangle taller should result in the blue rectangle becoming shorter). Figure 5 shows the stages in creating the controller. First, the red and blue rectangles are created and constrained to be attached to the bottom portion of the controller (established equality constraints are shown here as lines, “anchors” are shown as squares). Second, an imaginary rectangle is created and is constrained to be positioned between the red rectangle’s top and the controller’s top. Third, the height of the blue rectangle is constrained to be equal to the imaginary rectangle’s height. Finally, the imaginary rectangle is hidden. Manipulating either the red or blue rectangles’ height has the effect of changing the other’s height through the imaginary object.

In addition, EUPHORIA supports a number of other features not described here such as end-user defined widgets and aggregate mappings. A more detailed discussion of EUPHORIA’s functionality can be found elsewhere [6], [9].

3. Design

For the purposes of this paper, an *end-user* is defined as a person who can operate a computer and use WYSIWYG applications such as a graphics editor, but cannot necessarily write textual computer programs. The usability Playground’s graphical environment was evaluated through an empirical study of end-users.

<u>*Primary Hypothesis:*</u> Through the use of the EUPHORIA user interface management system and the visual configuration language, first-time end-users of our system can learn and effectively apply the techniques of: (1) constraints, (2) constraint visualization, (3) imaginary objects, (4) configuration of distributed applications, and (5) distributed multimedia application construction.

<u>*Secondary Hypothesis:*</u> Novice end-users (i.e., end-users with little or no computer science training) can utilize the above application construction techniques as rapidly and accurately as expert end-users (e.g., computer science graduate students).

The study consisted of five representative tasks associated with the construction of distributed multimedia applications. The series of tasks were designed with a target time of 75 minutes for the experimental session. Each end-user was given a booklet of instructions that introduced all of the necessary concepts and described the tasks to be performed (see [6], Appendix E). The instructions were organized as a series of five sections. Each section consisted of a number of practice exercises, designed to provide a hands-on introduction of the associated application construction techniques, followed by a task to be performed. The following subsections describe these practice exercises and tasks.

3.1 Constraints

The practice part of this section consisted of exercises introducing basic drawing techniques (e.g., drawing a rectangle) and end-user defined constraint relationships.

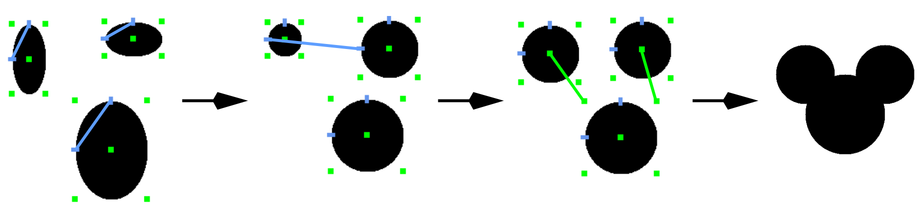

Figure 6: Constraints task solution.

In the task part of this section, subjects were asked to create a drawing of “Mickey Mouse’s” head. Subjects were instructed to make the ears and the head strictly circular, make the size of the ears equal, and attach the ears to the head (i.e., when any shape is manipulated, all of these properties should be maintained). These requirements necessitated the use of constraints. Figure 6 shows the series of steps leading to one possible solution of this task (established constraints are shown as lines).

3.2 Constraint Visualization

The practice part of this section consisted of exercises introducing constraint visualization and deletion of constraints. In the task part of this section, subjects were given a constrained drawing (Figure 4) and were asked to use constraint visualization to understand its interaction behavior. The drawing consists of three rectangles. Rectangle 1 changes width and height whenever rectangle 2 or rectangle 3 is manipulated. Rectangle 2 and rectangle 3 change heights when rectangle 1 is resized. Subjects had to describe the constraint relationships among these rectangles. That is, the width and height of rectangle 1 is constrained to the heights of rectangle 2 and rectangle 3; the total height of rectangle 2 and rectangle 3 is constant because of anchor constraints on their corners.

3.3 Imaginary Objects

The practice part of this section consisted of exercises introducing imaginary objects. In the task part of this section, subjects were asked to create a “temperature controller” (see Section 2.2), requiring the use of an imaginary object. In the instructions, subjects were informed of the controller’s desired behavior and were shown examples of valid proportions of “hot” to “cold.” Figure 5 shows the series of steps leading to one possible solution of this task.

3.4 Module Configuration

The practice part of this section consisted of exercises introducing modules, published variables, and configuration. In the task part of this section, subjects were given three running modules with descriptions, including a completed GUI constructed within EUPHORIA. To establish the behavior of the application and its GUI, subjects had to configure the communication among the modules using the visual configuration language.

The application was a planetary orbits simulation animating the path of the Earth and Moon around the Sun (see Section 2.1 and Section 2.2). An *Earth* module was supplied that generates the position and diameter of the Earth. A *Moon* module was supplied that computes the position of the Moon, relative to the Earth, and generates the position and diameter of the Moon. Subjects had to correctly configure logical connections among the Earth, Moon, and EUPHORIA display (Figure 1).

3.5 Distributed Multimedia Application Construction

The practice part of this section consisted of exercises introducing publishing variables and using text and image graphics objects in EUPHORIA.

In the task part of this section, subjects were given three running modules with descriptions and were asked to create a distributed multimedia application. The application to be constructed was a *multimedia*

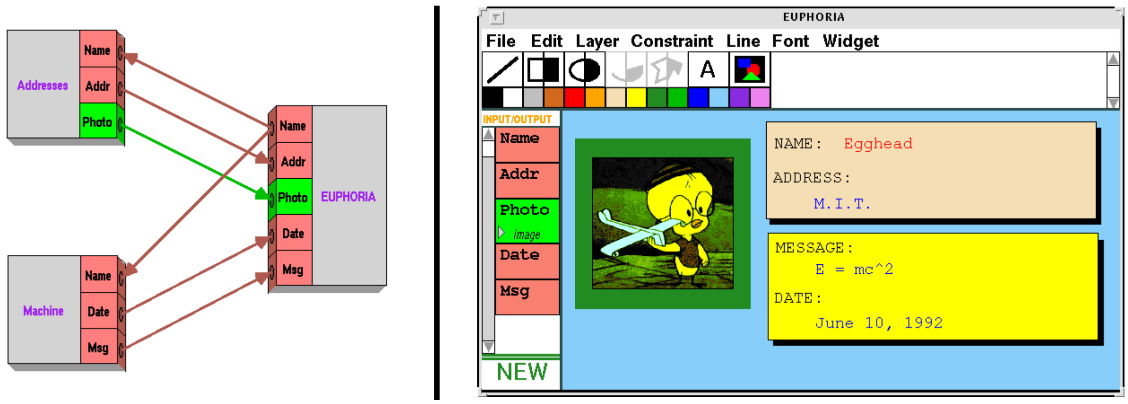

Figure 7: Application construction task solution.

message system; given the name of a person, entered graphically by application's user, the application displays their address, photograph, and current status. An *address book* module is supplied to retrieve addresses and photographs for individuals given their name. An *answering machine* module is supplied to retrieve a status and date for individuals given their name.

Subjects were asked to create a graphical user interface within EUPHORIA, to configure the communication among the three modules, and to interact with the interface. This task required an understanding of published attributes in EUPHORIA and their relationship to a distributed application's configuration.

4. Subjects

Subjects were Washington University undergraduate and graduate student volunteers ranging from 18 to 30 years of age. The study was advertised by posting flyers around the Washington University campus. Subjects were compensated $10 for their participation.

A total of 31 subjects participated in the study. The prerequisite for participation was experience using commercial graphics editors (e.g., MacDraw), but no prior exposure to our software. Three subjects were eliminated from the statistical analysis because they had never used a graphics editor. The remaining 28 subjects represented a wide variety of disciplines: Biological Engineering, Business, Chemical Engineering, Computer Engineering, Computer Science, Electrical Engineering, English Literature, German, Graphic Design, Jewish & Near Eastern Studies, Math, Physics, Psychology, Political Science, and Systems Science.

5. Testing Procedure

Each subject performed the various tasks using our software on a Sparc workstation (a Sparc 5, Sparc 20, or UltraSparc 1). The booklet of instructions provided each subject with tutorials and descriptions of the tasks to be performed. Subjects had no previous exposure to our system and no additional instruction was provided.

An evaluation form was also given to each subject which included a brief questionnaire and space to write comments. Subjects were responsible for recording the times associated with each section (a clock was provided on the computer screen) and saving the task results (i.e., drawings and configurations) for later evaluation. The evaluation form also included a confidentially agreement, signed by each subject, stating that they would not discuss the details of the study with future participants.

Scoring

For each task, an accuracy was computed for each subject based on the percentage of the task completed. In summarizing and comparing the subject times on a task, only the times of the subjects who completed the task with 100% accuracy were used. The reason for excluding the less accurate subjects is to get a more reliable measure of the task time, avoiding subjects who finished prematurely. All times for the practice and task portions include the time for the subject to read the associated instructions.

6. Primary Hypothesis Results

The primary hypothesis is concerned with the overall usability of our graphical environment. For all sections, the majority of the subjects were able to complete the associated tasks with in a reasonable amount of time. Table 1 shows the overall subject accuracy and times.

Table 1: Overall mean times and accuracy.

Section	Practice Time (minutes)	Task Accuracy	% of Subjects Completely Accurate	Task Time for Subjects Scoring 100%
Constraints	4.9	86	64	7.9
Constraint Visualization	4.5	94	82	5.7
Imaginary Objects	3.1	84	75	11.3
Module Configuration	3.4	100	100	3.5
Application Construction	7.2	94	89	14.3

Table 1's "Practice Time" column represents the mean practice time for all subjects. "Task Accuracy" is the mean accuracy of all subjects, including partial credit for incomplete tasks, as described in Section . The "% of Subjects Completely Accurate" column represents the percentage of subjects who completed each task with 100% accuracy. The mean task times for the 100% accurate subjects are listed in the "Task Time" column.

6.1 Constraints

The accuracy for the constraints task (Section 3.1) was a bit lower than we had anticipated. Ten subjects were unable to successfully complete this task. We conjecture that this is primarily due to the fact that it was the first part that involved "problem solving" rather than just following step-by-step instructions. Subjects only spent an average of 4.9 minutes learning the basic concepts and practicing before starting this task; further practice exercises probably would have resulted in a higher task accuracy rate. Also, it is clear that several subjects just did not fully comprehend the consequences of the instruction requirements. These subjects claimed to have successfully completed the task, drawing an accurate *looking* picture. However, not all of the required constraint relationships were established in the drawing. This seems to indicate that subjects are accustomed to static drawings that do not change and are not used to dynamic drawings that can be manipulated.

6.2 Constraint Visualization

The time and accuracy ratings for the constraint visualization task (Section 3.2) were very good. Subjects found EUPHORIA's constraint visualization mechanism to be very intuitive. Several people even complained that they would have done better in the first task if constraint visualization had been introduced earlier.

6.3 Imaginary Objects

The imaginary objects task (Section 3.3) was the most difficult task of the study, requiring the most problem solving skill. Even though more subjects successfully completed this task than the constraints task, the people who did not complete this task did not even attempt to use imaginary objects (a requirement to solve this task). Subjects only spent an average of 3.1 minutes practicing before starting this task; further practice exercises may have resulted in a higher task accuracy rate. Unlike the constraints task, the wording of the instructions was probably not a factor in the accuracy ratings.

6.4 Module Configuration

The module configuration task (Section 3.4) was completed with 100% accuracy by all subjects fairly rapidly. This seems to indicate that our visual configuration mechanism is especially effective.

6.5 Distributed Multimedia Application Construction

The application construction task (Section 3.5) was, perhaps, the most important part of this study. This section required the use of concepts from other sections, combining user interface construction techniques with distributed application configuration. The results are encouraging: subjects were able to construct and configure a complete distributed multimedia application in an average of 14.3 minutes. The mean accuracy score for this task was also excellent.

7. Secondary Hypothesis Results

The secondary hypothesis is concerned with comparing the performance of end-users with different levels of computer experience. For this hypothesis, the subjects were divided into three categories (Table 2) according to their amount of formal computer science training.

Table 3 lists the accuracy rates of the subject categories for the various tasks. For each task, the accuracy of the novice subjects is nearly as high as the accuracy of the expert subjects. The only notable exception is in the application construction task. All of the expert and intermediate subjects completed this task with 100% accuracy, but three novice subjects were unable to complete the task, resulting in a lower accuracy

Table 2: Secondary hypothesis subject categories.

Category	n	Description
Novice	8	People who have never taken a computer science course.
Intermediate	10	People who have taken 1 or 2 computer science courses.
Expert	10	People who have taken more than 2 computer science courses.

rating. Also note that there is not a strong trend among the differences between the three groups. Sometimes the novice subjects out-performed the intermediate subjects or the intermediate subjects out-performed the expert subjects.

Table 3: Category-based mean accuracy scores (percent correct).

Task	Novice	Intermediate	Expert
Constraints	81	88	83
Constraint Visualization	91	81	100
Imaginary Objects	75	90	88
Module Configuration	100	100	100
Application Construction	79	100	100

The following subsections compare the times of the three categories on the various tasks using a one-way between-subjects analysis of variance test. As in Section 6, for each task, only the times of the 100% accurate subjects are used.

7.1 Constraints Task

The mean times for the three end-user categories in the constraints task were: novice, 10.6 minutes; intermediate, 8.7 minutes; expert, 6.8 minutes. Analysis of variance performed on the data indicated that there were *no significant differences* among the means, $F(2, 15) = 1.18$, $MS_e = 16.50$.

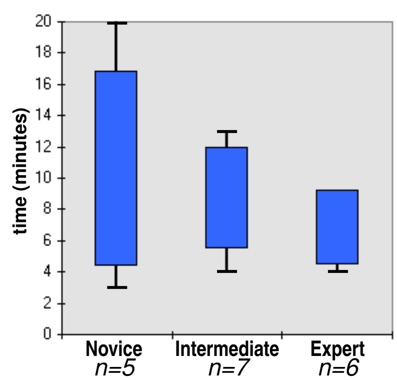

Figure 8: Constraints task results.

Figure 8 shows the times of the three categories for the constraints task. In this chart, thin bars are used to show the time range and thick bars are used to show one standard deviation above and below the mean time. Nearly an equal number of subjects in each category were less than 100% accurate (thus, were eliminated from the analysis and this chart). Although it appears that the times of novice category are much higher in comparison to the other categories, this is due to a single outlier value (20 minutes) that is nearly double of the next highest time. The standard deviation is also broad because of an unusually fast time in this category (3 minutes), which is actually the fastest time of all the subjects. We did not remove outlier values from our data since the sizes of the categories are relatively small.

7.2 Constraint Visualization Task

The mean times for the three end-user categories in the constraint visualization task were: novice, 5.7 minutes; intermediate, 5.6 minutes; expert, 5.8 minutes. Analysis of variance performed on the data indicated that there were *no significant differences* among the means: $F(2, 20) = 0.02$, $MS_e = 6.43$.

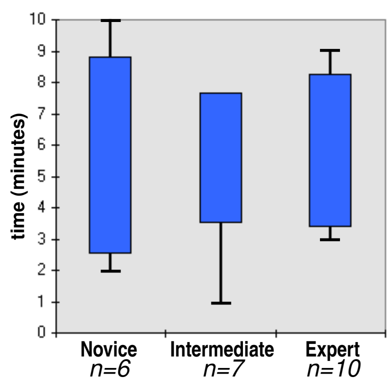

Figure 9: Constraint Visualization task results.

Figure 9 shows the times of the three categories for the constraint visualization task. Nearly an equal number of novice and intermediate subjects were less than 100% accurate (thus, were eliminated from the analysis and this chart); the expert subjects all completed this task with 100% accuracy. It is clear that there is almost no difference among the three groups on this task.

7.3 Imaginary Objects Task

The mean times for the three end-user categories in the imaginary objects task were: novice, 9.8 minutes; intermediate, 13.1 minutes; expert, 12.9 minutes. Analysis of variance performed on the data indicated that there were *no significant differences* among the means: $F(2, 18) = 0.395$, $MS_e = 49.70$.

Figure 10 shows the times of the three categories for the imaginary objects task. Nearly an equal number of subjects from each category were less than 100% accurate (thus, were eliminated from the analysis and this chart). Although it appears that the expert subjects' times are actually much higher than the other categories, this is due to a single outlier value (31 minutes) of a very persistent computer science graduate student. Also, this category contains the fastest two times (a tie, 5 minutes) resulting in a fairly broad standard deviation.

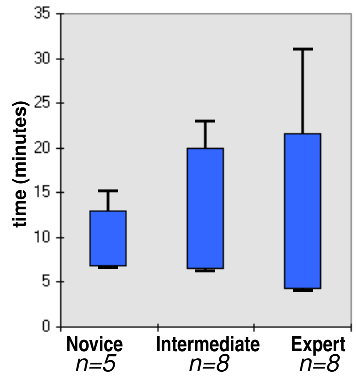

Figure 10: Imaginary Objects task results.

7.4 Module Configuration Task

The mean times for the three end-user categories in the module configuration task were: novice, 6.3 minutes; intermediate, 3.5 minutes; expert, 2.8 minutes. Analysis of variance performed on the data indicated that there were *significant differences* among the means: $F(2, 25) = 7.21$, $p < .01$, $MS_e = 3.98$.

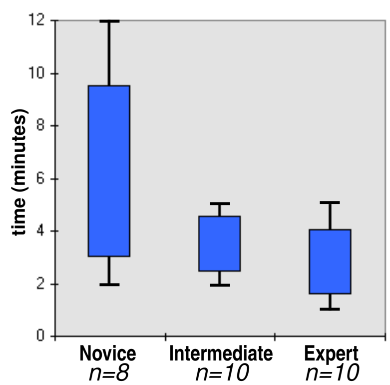

Figure 11: Module Configuration task results.

Figure 11 shows the times of the three end-user categories in the module configuration task. Clearly, the times for the novice subjects are greater than the times of the intermediate and expert categories. The times of the intermediate and expert categories, however, are not significantly different. The intermediate and expert categories consisted largely of computer science and electrical engineering majors; the novice category consisted mainly of non-technical majors. The difference is probably because people in computer science and electrical engineering routinely deal with component-based computation abstractions and people in non-technical majors do not have this exposure. But it should also be noted that all subjects successfully completed the task. Also, one person in the novice category solved this task in only 2 minutes and was in a fairly non-technical area of study (German).

We believe that with additional visual configuration experience, novice end-users could configure distributed applications as quickly as expert end-users. After all, the novice end-users only spent an average of 3.5 minutes practicing for the module configuration task.

7.5 Distributed Multimedia Application Construction Task

The mean times for the three end-user categories in the distributed multimedia application construction task were: novice, 14.6 minutes; intermediate, 19.1 minutes; expert, 12.3 minutes. Analysis of variance performed on these data indicated that there were *no significant differences* among the means: $F(2, 22) = 3.37$, $MS_e = 35.01$.

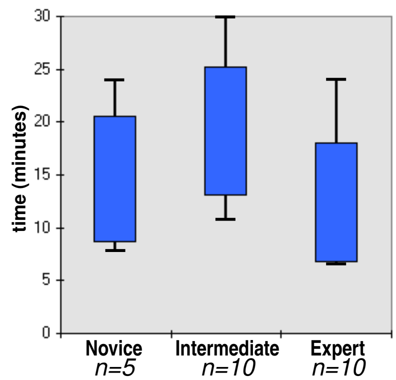

Figure 12: Application Construction task results.

Three subjects in the novice category were less than 100% accurate (thus, were eliminated from the analysis and this chart). While one of these subjects only made a minor mistake, the other two did not successfully construct the application. We believe that phrasing of the instructions may have intimidated these individuals (e.g., “Create a user interface”). This, combined with fatigue, resulted in these subjects quitting prematurely.

Figure 12 shows the times of the three end-user categories in the distributed multimedia application task. It is clear that there is little difference between times of the novice, intermediate, and expert subjects.

8. User Feedback

In general, the subjects of the study enjoyed using EUPHORIA and the visual configuration language. The main complaint was that EUPHORIA does not have an “Undo” mechanism. The following are some end-user quotes taken from the evaluation form comments:

> “I hadn’t seen constraint-oriented graphics before and it was really easy to grasp.”

> “...when a handle is being pointed to, it should become shadowed or something. I often missed when I tried to select a handle, and since I didn’t know if I had succeeded or not, I had to backtrack, checking my constraints until I found the one I missed.”

"The most difficult part was getting the look right. The actual configuration part was very easy. Imaginary objects are very helpful."

"My brain hurt for the first minute thinking about it [temperature controller], but once I anchored the black bars it was easy... I started with only one imaginary object and then putzed around and saw the problem. It was never frustrating, though."

"Cool! Really neat graphics."

"Constraint-based interface is easy to use but it's hard to imagine beforehand what a particular constraint will do."

"Interesting. This is a cool concept" [imaginary objects]

"The 'show constraints' idea seems very helpful, but it could become confusing with complicated drawings."

"It's a good system. I especially like the interaction between EUPHORIA and the Connection Manager."

"I like EUPHORIA. It has a lot of neat features I've never seen before."

"I was having far too much fun playing with this one!" [application construction]

9. Summary

To test the effectiveness of the The Programmers' Playground graphical environment, we conducted a usability study in which end-users performed tasks representative of constructing distributed multimedia applications. A total of 28 undergraduate and graduate Washington University students from a variety of disciplines participated in this study. The results showed that end-users were able to construct and configure complete distributed multimedia applications using the techniques of constraints, constraint visualization, imaginary objects, and visual configuration. Further, for all of this study's user interface construction tasks, end-users with no formal computer science background were able to perform as well as experienced computer programmers.

Acknowledgments

Ken Goldman provided guidance and helpful suggestions during the design of this study. Karl Schmidt, Jyoti Parwatikar, and Eileen Kraemer "pre-tested" this usability study, helping to make the testing of the actual subjects relatively problem-free. This research was supported in part by National Science Foundation grants CCR-91-10029, CCR-94-12711, and ARPA contract DABT63-95-C-0083.

References

[1] Byrne, M, et al. The Role of Student Tasks in Accessing Cognitive Media Types. To appear in the Second International Conference on the Learning Sciences, 1996.

[2] Goldman KJ, et al. The Programmers' Playground: A Demonstration. In *Proceedings of the Third ACM International Multimedia Conference*, San Francisco, CA, November 1995, pages 317-318. See also the conference CD-ROM proceedings for a longer version.

[3] Goldman KJ, et al. The Programmers' Playground: I/O Abstraction for User-Configurable Distributed Applications. *IEEE Transactions on Software Engineering*, 21(9):735-746, September 1995.

[4] Gutwin C, Roseman M, Greenberg S. A Usability Study of Awareness Widgets in a Shared Workspace Groupware System. In *Proceedings of ACM CSCW'96 Conference on Computer Supported Cooperative Work*, (in press) 1996.

[5] Kamba T, et al. Using small screen space more efficiently. In Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems, April 1996, pages 383-390.

[6] McCartney TP. End-user Construction and Configuration of Distributed Multimedia Applications. D.Sc. Thesis, Washington University 1996. Also appears as Washington University Department of Computer Science technical report WUCS-96-24, September 1996.

[7] McCartney TP. User Interface Applications of a Multi-way Constraint Solver. Washington University Department of Computer Science technical report WUCS-95-22, August 1995.

[8] McCartney TP, Goldman KJ. Visual Specification of Interprocess and Intraprocess Communication. In *Proceedings of the 10th International Symposium on Visual Languages*, October 1994, pages 80-87.

[9] McCartney TP, Goldman KJ, Saff DE. "EUPHORIA: End-User Construction of Direct Manipulation User Interfaces for Distributed Applications," *Software-Concepts and Tools,* 16(4):147-159, December 1995.