RAPIDS: A SIMULATION-BASED INSTRUCTIONAL
AUTHORING SYSTEM FOR TECHNICAL TRAINING

Douglas M. Towne
Allen Munro
Behavioral Technology Laboratories
University of Southern California
1845 South Elena Avenue, Fourth Floor
Redondo Beach, California 90277

Training Systems Division
Brooks Air Force Base, Texas 78235-5601

January 1991

Approved for public release; distribution is unlimited.
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

J. WESLEY REGIAN
Contract Monitor

HENDRICK W. RUCK, Technical Advisor
Training Systems Division

HAROLD G. JENSEN, Colonel, USAF
Commander
RAPIDS: A Simulation-Based Instructional Authoring System for Technical Training

Douglas M. Towne
Allen Munro

Behavioral Technology Laboratories
University of Southern California
1845 South Elena Avenue, Fourth Floor
Redondo Beach, California 90277

Training Systems Division
Air Force Human Resources Laboratory
Brooks Air Force Base, Texas 78235-5601

Approved for public release; distribution is unlimited.

There are very few applications of intelligent tutoring systems in use today that were developed outside of the research environment. The existing systems demonstrate very promising instructional power, but were typically developed at extremely high cost by unusually talented researchers and programmers. Instructional developers with those skills are not generally available in the training community. The research described in this report was undertaken to allow a larger user community to employ the Intelligent Maintenance Training System (IMTS) simulation authoring and simulation execution resources. Users should be able to produce highly tailored instructional systems much more rapidly, employing available subject-matter expertise. The resulting system is intended to address both research requirements and operational training requirements. It is to serve as a tool for studying the effectiveness of competing approaches to training, as well as providing a set of tools for the production of simulation-based courses.
SUMMARY

RAPIDS (the Rapid Prototype ITS Development System) is a system for creating and delivering training based on interactive graphical simulation. RAPIDS is based on the foundation of IMTS (the Intelligent Maintenance Training System), which includes tools that enable simulations to be constructed by direct manipulation. Using RAPIDS, an expert creates computer-based instruction directly, by carrying out the tasks to be taught. Like frame-based authoring systems, RAPIDS facilitates production of quality course materials by developers who do not have to be computer programmers. Instead of being based on the concept of the frame, however, RAPIDS is based on the concept of actions performed in a simulation.
PREFACE

The work described here was supported by the Office of Naval Research, the Navy Personnel Research and Development Center, and the Air Force Human Resources Laboratory, under ONR Contract No. N00014-87-C-0489. J. Wesley Regian of AFHRL served as scientific officer for the RAPIDS portion of this research contract.

Our colleagues Lee D. Coller, Quentin A. Pizzini, David S. Surmon, and James Wogulis assisted in the design and carried out the implementation of RAPIDS.
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>I. INTRODUCTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Problem</td>
<td>1</td>
</tr>
<tr>
<td>Organization of the Report</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>II. IMTS: THE SIMULATION FOUNDATION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparing IMTS to Other ICAI Simulation Training Systems</td>
<td>3</td>
</tr>
<tr>
<td>SOPHIE</td>
<td>3</td>
</tr>
<tr>
<td>STEAMER</td>
<td>4</td>
</tr>
<tr>
<td>QUEST</td>
<td>4</td>
</tr>
<tr>
<td>SHERLOCK</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>III. RAPIDS COURSE STRUCTURE AND DELIVERY</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Production Overview</td>
<td>5</td>
</tr>
<tr>
<td>Natural Instructional Processes</td>
<td>6</td>
</tr>
<tr>
<td>Course Structure</td>
<td>7</td>
</tr>
<tr>
<td>Organizational Units</td>
<td>9</td>
</tr>
<tr>
<td>Content Units</td>
<td>9</td>
</tr>
<tr>
<td>Expositions</td>
<td>10</td>
</tr>
<tr>
<td>Items</td>
<td>11</td>
</tr>
<tr>
<td>Actions</td>
<td>11</td>
</tr>
<tr>
<td>Instructional Modes</td>
<td>13</td>
</tr>
<tr>
<td>Scoring</td>
<td>14</td>
</tr>
<tr>
<td>Managing Time and Content Presentation</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IV. AUTHORING AND INSTRUCTOR FUNCTIONS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternative Time Allocation Strategies</td>
<td>16</td>
</tr>
<tr>
<td>The RAPIDS Authoring Tools</td>
<td>17</td>
</tr>
<tr>
<td>Authoring Content Units</td>
<td>17</td>
</tr>
<tr>
<td>Authoring Expositions</td>
<td>19</td>
</tr>
<tr>
<td>Authoring Content Items</td>
<td>19</td>
</tr>
<tr>
<td>Authoring the Instructional Plan</td>
<td>20</td>
</tr>
<tr>
<td>Monitoring Student Progress</td>
<td>22</td>
</tr>
</tbody>
</table>
## TABLE OF CONTENTS (CONCLUDED)

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>V. EXPLORATION MODES IN RAPIDS........................................22</td>
</tr>
<tr>
<td>Free Play Simulation Sequences...........................................23</td>
</tr>
<tr>
<td>System Configuration.....................................................23</td>
</tr>
<tr>
<td>Fault Diagnosis..................................................................23</td>
</tr>
<tr>
<td>Device Behavior Exploration.............................................24</td>
</tr>
<tr>
<td>VI. SUMMARY AND CONCLUSIONS.............................................24</td>
</tr>
<tr>
<td>Comparing RAPIDS with Other CBI Tools..................................24</td>
</tr>
<tr>
<td>Ease of Authoring............................................................25</td>
</tr>
<tr>
<td>Probability of Bugs..........................................................26</td>
</tr>
<tr>
<td>Effective Tools for Course-Level Planning..............................27</td>
</tr>
<tr>
<td>Automatic Handling of Low-Level Student Interactions...............28</td>
</tr>
<tr>
<td>Breadth of the Domain of Application of the Tool....................28</td>
</tr>
<tr>
<td>Future Developments.........................................................29</td>
</tr>
<tr>
<td>REFERENCES......................................................................30</td>
</tr>
</tbody>
</table>
RAPIDS: A Simulation-Based Instructional Authoring System for Technical Training

I. INTRODUCTION

RAPIDS (Rapid Prototype ITS [Intelligent Tutoring System] Development System) is a system for creating and delivering training based on interactive graphical simulation. Using RAPIDS, an expert creates computer-based instruction directly, by doing the tasks to be taught. Like frame-based authoring systems, RAPIDS facilitates production of quality course materials by developers who do not have to be computer programmers. Instead of being based on the concept of the frame, however, RAPIDS is based on the concept of actions performed in a simulation.

The Problem

There are very few applications of intelligent tutoring systems in use today that were developed outside of the research environment. The existing systems demonstrate very promising instructional power, but were typically developed at extremely high cost by unusually talented researchers and programmers. Instructional developers with those skills are not generally available in the training community.

Some of the existing systems were especially developed to instruct in the operation or diagnosis of one particular device, and thus offer little more than design guidance to developers of instructional systems for other devices. Other systems take a generic approach in which the device is described separately from the instructional processes. In either case, most of the intelligent instructional systems that actually function outside of the laboratory apply an instructional strategy that is embedded deeply in the system.

A significant portion of the development costs of these applications is attributable to research endeavors that would not necessarily be repeated to produce additional applications. Still, it is safe to say that the cost, time, and skill requirements of producing intelligent training applications today are very high, and the options for
tailoring instructional strategy are limited in most intelligent computer-aided instruction (ICAI) systems.

Under certain conditions, simulation-based training systems may provide training hours at a rate comparable to conventional frame-based instruction systems. The relatively fixed cost of developing a simulation can serve as the foundation for many hours of instruction for each student, with negligible incremental cost for producing additional training hours. A simulation can support the automatic or nearly automatic production of a wide variety of training scenarios, such as practice, debriefings, demonstrations, and free exploration. Many different tasks, such as different troubleshooting problems or operational exercises, can be generated from one simulation.

The research described in this report was undertaken to allow a larger user community to employ the Intelligent Maintenance Training System (IMTS) simulation authoring and simulation execution resources. Users should be able to produce highly tailored instructional systems much more rapidly, employing available subject-matter expertise. The resulting system is intended to address both research requirements and operational training requirements. It is to serve as a tool for studying the effectiveness of competing approaches to training, as well as providing a set of tools for the production of simulation-based courses.

Organization of the Report

Section II of this report discusses the role of IMTS in RAPIDS instruction, and its relationship to other ICAI simulation training systems. Section III describes the structure and delivery of RAPIDS courses from the student's point of view. Section IV describes the process for authoring RAPIDS courseware. Section V covers the exploration modes now presented under RAPIDS course control. Section VI presents conclusions.

II. IMTS: THE SIMULATION FOUNDATION

The Intelligent Maintenance Training System (IMTS) was developed to exploit the potential to automatically generate instruction for training troubleshooting skills, based upon a deep simulation model of the device (Towne & Munro, 1988; Towne & Munro, 1989; Towne, Munro, Pizzini, & Surmon, 1987; Towne, Munro, Pizzini, Surmon, & Johnson, 1985; Towne, Munro, Pizzini, Surmon, & Wogulis, 1988; ). IMTS offers tools that allow a subject-matter expert to construct a simulation that can then be presented to a learner in a variety of ways, depending upon the demonstrated proficiency of the learner. The instructional delivery functions in IMTS take care of interacting with the student, selecting problems, guiding the student when diagnostic progress stalls, and recording performance data. Intelligent instruction can be produced by a non-programming subject-matter expert, without the labor associated with the
implementation of an instructional strategy, student-computer interactions, and adaptation to individual student proficiency.

This automaticity of instructional delivery comes at the price of a limited instructional range. IMTS follows a built-in strategy for delivering and tailoring instruction. It invokes a built-in, and effectively non-modifiable, diagnostic expert to evaluate student work and to generate recommendations. This expert can generate instructional scenarios in any of four modes (free-play practice with guidance as necessary, expert demonstration, debriefing of student performance, and free exploration). The student can select from these scenarios, or IMTS can choose among them automatically.

Although the diagnostic expertise that IMTS provides is of high quality and the instructional scenarios are quite effective for providing practice in diagnostic activities, the system is of limited use in non-diagnostic training. By itself, IMTS cannot be used to deliver a front panel operation drill or a symptom assessment drill. It cannot instruct the student on theory of operation or on the performance of procedures. Nor can instructional developers make any changes to the instructional strategy or to the diagnostic expert. These limitations on the range of instructional presentation are especially unfortunate in that the simulation-building tools, the simulation execution functions, and the student-computer interactions within IMTS are so generic.

Comparing IMTS to Other ICAI Simulation Training Systems

In the last 15 years, a large number of interesting research projects have resulted in ICAI systems. Many of these systems have produced courses for teaching equipment operation or troubleshooting, the focus domains of RAPIDS.

Immediately below, a number of the features of four such systems are compared with specific features of IMTS. The research embodied in these systems influenced the development of IMTS. An important difference between IMTS and these experimental systems should be stressed at the outset, however. Most of the earlier ICAI systems have not resulted in new development tools for the production of computer-based training courses. Instead, they have developed and applied Artificial Intelligence (AI) methodologies to particular training problems, such as teaching how to maintain a single complex device. In effect, programming languages and general-purpose artificial intelligence tools were applied to particular training problems that resulted in particular solutions. In many cases, these projects illustrated general principles or techniques that could be of use in future systems, but most did not result in the development of new tools for producing intelligent instruction.

SOPHIE

SOPHIE is an intelligent simulation training system that produces simulation-based inferences during interactive training (Brown & Burton, 1975; Brown, Burton, & Bell, 1975). SOPHIE generates textual responses to student queries and answers. It is
not a graphical system; students inquire about indications and test results by typing, not by directly manipulating and observing indications on the screen. Its language generation system is more sophisticated than that of IMTS/RAPIDS. In SOPHIE-III (Brown, Burton, & deKleer, 1982), the reactive style of earlier versions of SOPHIE is supplemented with a pedagogically active component that estimates and responds to student needs.

Like IMTS, SOPHIE bases its interactions with students on simulated responses to student actions. In IMTS, the student's actions are physical (actuating the mouse); in SOPHIE, they are textual. IMTS provides generic tools for the development of new simulations by non-programmers; SOPHIE does not.

STEAMER

STEAMER (Hollan, Hutchins, & Weitzman, 1984; Williams, Hollan, & Stevens, 1981) is an interactive, inspectable, graphical simulation of a steam propulsion system used in nuclear submarines. Like SOPHIE and IMTS, STEAMER can generate responses to student actions. Like IMTS, STEAMER has generic tools for building the interactive graphic components of a simulation. Unlike IMTS, STEAMER calls for the development of an underlying abstract model with which these graphic components will be associated. In most applications of the STEAMER methodology, a computer program is written to produce this underlying model.

A STEAMER module called the feedback minilab (Forbus, 1984) makes it possible to build small simulation systems from predefined components graphically. The minilab then automatically generates code to run the simulation. This module can be seen as a forerunner of the IMTS scene editor. IMTS extends this authorability concept in two ways. First, it permits the graphic composition of arbitrarily large simulations in the scene editor. Second, IMTS includes a generic object editor that is used not only to draw new components, but also to define their behaviors.

QUEST

QUEST (White & Frederiksen, 1984, 1985, 1987) represents an attempt to study the effectiveness of intelligent training in the domain of electricity, using a succession of qualitative models. Like IMTS, QUEST makes use of both circuit-independent and circuit-dependent elements in a simulation model. White and Frederiksen propose that conceptual training in electricity can be carried out by using an appropriate progression of successively more elaborate qualitative models. The IMTS/RAPIDS concept of using simplified simulations consisting of yoked objects was significantly influenced by this research.

QUEST is very much a tutor for a single subject matter (electricity), using many different circuit examples. As a result, it takes a more programmatic approach to model progression than is enforced in IMTS. IMTS/RAPIDS is a tool for building interactive training environments for a variety of devices, but without a unifying conceptual
subject matter. QUEST does not include general-purpose tools for building training environments.

SHERLOCK

SHERLOCK (Lesgold, Lajoie, Bunzo, & Eggan, 1988) is a practice environment, with coaching, for troubleshooting one particular complex device, the F-15 manual avionics test station. It emphasizes the development and refinement of mental models, including the analysis of inferred student models with respect to expert models. Some student actions, such as making simulated tests, are conducted graphically, but SHERLOCK places less emphasis on free play simulation of complex devices than does IMTS/RAPIDS.

Although SHERLOCK takes an object-oriented approach that should facilitate adaptation to other training domains, it is not a general-purpose authoring tool set for creating practice environments.

III. RAPIDS COURSE STRUCTURE AND DELIVERY

Unlike IMTS, which automatically generates almost all student-computer interactions, a RAPIDS course involves an identifiable courseware development step. RAPIDS courseware development differs in three important ways, however, from that using conventional CAI languages and authoring environments:

1. The bulk of the instructional material is produced by performing tasks on a simulation rather than by describing them.
2. Authoring of technical content is entirely separate from the specification of the instructional strategy.
3. The RAPIDS author does not deal with low-level interactions between the learner and RAPIDS. Such interactions are all produced automatically as the situation warrants.

Thus, RAPIDS retains the features in IMTS that minimize development effort, and adds capabilities for expanding the range of instruction.

Course Production Overview

There are three major functions involved in producing a RAPIDS course:

1. constructing the device simulation,
2. specifying the instructional strategy, and
3. producing the technical content of the instruction.
These functions may be carried out by one person or by several specialists, none of whom need have programming skills.

The simulation is produced first by an expert in the behavior of the device. (The simulation authoring process is described in Munro & Towne, 1989a; Towne & Munro, 1988, 1989). The instructional plan and the instructional content can then be developed in parallel; or alternatively, the plan can be completely specified before producing the technical content. In either case, it is likely that the production of the content will have some impact on the organization of the course. When completed, the instructional plan specifies how RAPIDS should deliver the technical content, and it serves as the interactive medium for accessing data on progress of individual students.

Natural Instructional Processes

When a device expert teaches someone to operate a device that is physically present, he or she might start demonstrating some basic tasks, explaining things to notice or remember as the steps are performed. At some point, the expert might show some diagrams that will help in understanding why the device responded as it did or why certain steps were necessary. Alternatively, the expert might begin by acquainting the student with the layout of the device, possibly having him or her practice locating and identifying important elements before moving on to procedures. In other cases, the expert might choose to present some basic theory of operation, followed by some exercises on the real device. For diagnostic training, the expert might elect to instruct on the major functions of the device, the indications of normal and abnormal functions, and the testing strategy that will effectively isolate failures.

Clearly, individual instructors have personal preferences and differences about how to instruct, and each may vary the approach, depending upon the subject matter and the student. Whatever the particular selection and sequence of technical content, human subject-matter experts typically convey their knowledge remarkably efficiently if

1. they have the device at hand, dedicated to instructional use;
2. they are instructing an individual learner who can practice the operations and exercises under the expert's guidance;
3. they have access to, and facility with, graphic aids such as schematic diagrams, ad hoc sketches, and other diagrammatic displays; and
4. they have the time, facilities, and motivation to convey their expertise to others.

This is the kind of instructional environment we have attempted to create in RAPIDS. The objectives are to have the content expert show, tell, and explain the technical information with which he or she has such great facility, and to have the student progress as follows:
from observing, listening, and reading

to performing with assistance

to performing without assistance.

Building instruction around an active simulation allows the expert to focus on the
device or on graphic representations of the device and on the tasks to be taught. Instead
of writing out how a procedure is performed, the expert can demonstrate the
operation. Instead of writing out what effects would result from an action, the expert
can demonstrate the action and the results and show why the effects occurred. Instead of
planning and implementing detailed procedures for dealing with student errors, the
expert can leave those details to RAPIDS.

In RAPIDS, the primary tasks of the instructional planner are

- to specify the major instructional elements of a course (such as front
  panel drills, symptom effect drills, procedure demonstration, theory of
  operation);
- to organize the instructional units into a cohesive course;
- to allocate available time to the various course units; and
- to specify criteria for success.

It is the job of RAPIDS to manage all the micro-interactions with the student,
including

- score-keeping,
- time-monitoring,
- maintaining the simulation,
- remediating the learner, and
- following the instructional plan.

Figure 1 illustrates RAPIDS informing a learner about an error, giving a second
chance, and then showing the correct answer. The details of these interactions were not
prescribed by the author of the lesson, but were generated by the RAPIDS run-time
software. (The simulation of a dual-engine aircraft was adapted from Kieras, 1988.)

Course Structure

A RAPIDS course consists of a number of units, each of which either specifies a
body of instruction or is a collection of other units. Units that are collections of other
units are termed "organizational units." Units that specify instruction are termed
"content units." The author has control not only over the structure and content of the
units but also, via parameters, over the time available in the units, the proficiency
required for successful completion, and (optionally) the conditions under which a unit is
presented or not presented. These controls apply to both content units and
organization. units.
A lengthy course may be conveniently structured as a manageable number of major units, each of which may be delivered according to its own unique specifications. The major units may also be structured of smaller ones, and so on until content units are reached. The terminal nodes of all instructional plans are content units, and all higher level units are organizational units.

Figure 2 illustrates the structure of a course to train the procedures for starting a dual-engine aircraft. The terminal units, the content units, are shown in dashed boxes, while the organizational units are shown in solid boxes. The highlighted unit, Elementary Drill, is an organizational unit consisting of two content units, Start Left Drill and Start Right Drill. This organizational unit belongs to a higher level organizational unit, Operation Drill, which belongs to the top organizational unit for the course.
Organizational Units

Organizational units are simply groupings of other units (organizational units or content units). For each unit contained in an organizational unit, the course planner specifies the time available to cover it; the accuracy and speed scores necessary to successfully pass it; the importance (weight) of it in relation to the other units (used for scoring); and, optionally, the condition under which the unit should be presented or skipped. Also, *if the sub-unit is a content unit, its mode of presentation is specified.*

The specification of the Elementary Drill unit is shown in Figure 3. This unit presents each of the two content units in Drill mode at least one time. The learner will work through the Start Left Drill until he or she can do the operation perfectly in under 5 minutes. If the learner cannot do the drill perfectly after three tries or after 20 minutes, then the unit ends in a failed condition. The Right Drill is similar, except that the learner is expected to master this similar operation much faster.

Unit: Elementary Drill

<table>
<thead>
<tr>
<th>Unit</th>
<th>Weight</th>
<th>Mode</th>
<th>Condition</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Limit</th>
<th>Accuracy</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Left</td>
<td>1</td>
<td>Drill</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>20</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>Start Right</td>
<td>1</td>
<td>Drill</td>
<td>-</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3. Elementary Drill Unit.

Content Units

While organizational units contain all the course structure information, content units contain all technical content to be presented to the learner. Because content units contain no instructional planning information, they can be used multiple times in a course, and they may be used in multiple courses. A unit that instructs use of some general-purpose test equipment, for example, could be referenced by any diagnostic course and could be invoked for any student whose performance warrants. For
instructional research, this capability allows the RAPIDS user to easily build multiple alternative instructional plans, all utilizing the same content.

A content unit may begin with an instructional presentation, called an "exposition," that combines text, graphic, and video presentations to explain or describe new information. This opening exposition might also inform the learner about the objectives or content of the unit.

The content unit then presents a series of instructional items. Each item involves an action to be performed either on the simulation or on menus that list alternative answers to questions. The items in a content unit can be presented either in a fixed sequence or in random order. After all the items have been completed, a content unit may present a closing exposition. A closing exposition might draw the learner's attention to results of the actions and it might summarize what has been learned from the unit.

Expositions

Expositions can range from very simple textual statements to long and complex sequences of text, graphics, videodisc segments, and learner responses. The following events can be presented in an exposition:

- present text in the message window
- present text overlaid on the simulation graphics in a custom window
- clear the message window
- wait for a student response (mouse click)
- wait for a specified amount of time
- play a videodisc segment
- highlight or unhighlight an object in the simulation window
- highlight or unhighlight a region in the simulation window
- change the simulation scene displayed

An exposition might simply present text explaining some new concept, or it might highlight one or more areas of the simulation and explain something about the highlighted elements. Or an exposition might show a segment of a videodisc presentation, then wait for the student to click the mouse, and then present some text overlaid on the simulation window, next to some important object. There are no limitations on the number of basic exposition events that can be called for in an exposition.

When desired, the expert may position a text field on the graphic simulation to draw the learner's attention to a particular area of the simulation, as shown in Figure 4.
Every source of electrical power must be routed through the cross-start relay.

![Diagram](image)

**Figure 4.** Text Field on Graphic Simulation.

**Items**

The items of a content unit may be presented either in a fixed sequence or in random order. Like a content unit, each item may have an opening exposition and a closing exposition. The opening exposition might explain the purpose or manner of performing the action, or it might present some bit of theory. After the opening exposition, *RAPIDS* either demonstrates some action or prompts the student to perform some action, depending upon the mode in which the item's unit is being presented.

The action is performed by clicking the mouse on a particular graphic object or area in response to some textual prompt (that is, the student performs the same action as the expert who created the unit). Thus, the learner is either doing something on the simulated device or responding to some type of query. Typical actions include the following:

- finding some element of the simulated device
- making some change to the simulated device, such as setting a switch
- answering a question (by clicking-on the correct displayed answer)
- indicating that the learner is ready to proceed to the next item

**Actions**

The action part of an item is the correct response to some stated objective. Often, the action is a single response such as finding a switch, doing the next step of a procedure, or responding to a question. The action might also be a small set of responses to a single prompt, such as *Find all the Frequency Mode Controls* or *Select three failures that could cause an abnormal output power*. The following actions are possible (all performed with the mouse):

- selecting one or more objects in the simulation
- selecting a region of the simulation
- selecting one or more items from a menu
- setting a switch to a specified state
- replacing a simulated object
- performing a test using a simulated front panel indicator
- performing a test using simulated test equipment
Several types of multiple-choice and multiple-answer questions are supported. The answers could be directly related to the current simulation, as in questions like

*Is the Power Meter normal for this mode?*

to which the student would select from Yes and No in a menu.

The question might require the learner to apply knowledge learned over time, such as

*What modes of operation involve the frequency divider?*

to which the student would select from a menu of choices. Figure 5 illustrates this type of multiple-answer question.

![Sample Multiple-Answer Question](image)

Figure 5. Sample Multiple-Answer Question.

Another version of the multiple-answer question is one that asks the learner to make multiple selections on the graphic display, such as

*Indicate all the modules that affect the Oscillation Frequency*

To this challenge, the learner makes responses directly on the simulation graphics, rather than on a list of possible answers.
Instructional Modes

When the instructional planner selects a particular content unit to be included in a course, he or she also assigns a mode in which all the items will be presented. The three possible modes are as follows:

1. instruct mode, in which the learner studies the expositions and observes the demonstrations;
2. drill mode, in which the learner attempts to perform the actions with the expositions as a guide; and
3. test mode, in which the learner attempts to perform without the guidance of the expositions.

For example, an expert could produce a front panel familiarization unit by selecting and naming each element on a panel. Any RAPIDS course could then call for that unit in instruct mode, in drill mode, and in test mode.

In Instruct mode, the opening exposition of each item is presented, the action is automatically performed, the device simulation is automatically updated, and the closing exposition is presented. The learner sees both the action of the expert and the reaction of the device. The job of the student is to observe, read, listen, and pace the rate of progress through the unit.

In Drill mode, RAPIDS presents the same expositions as in Instruct mode; but it prompts the student to perform the actions, rather than to observe them. If the student performs the wrong action, RAPIDS automatically tells the student what he or she did, and then allows the student to try one more time. If the student cannot do the action correctly, the system performs the action as if in Instruct mode. Accuracy and speed scores are automatically computed and recorded. These proficiency scores are used to determine when the student can progress to the next unit, and they may be used in conditions that determine whether or not other units should be presented.

Test mode is similar to Drill mode, except that the expositions are not presented. Thus, the student must perform the tasks without guidance. If the unit is covering front panel familiarization, for example, the prompt might be Now find the Frequency Modulation Control. For a unit covering a procedure, the prompt in Test mode would be Now do the next action, whereas in Instruct mode the prompt might have been Now set the Power Switch to Standby. All this micro-management of what to say to the learner and how to deal with student errors is handled automatically by RAPIDS.

When test units are produced, the expert indicates whether or not student errors should be corrected. In some cases, correcting the student's errors may supply information that detracts from the validity of the later test items. In other cases, student errors must be corrected for later items to be valid. This is usually true if the test is
covering the performance of a procedure, because one student error could put the device into an incorrect state.

Test mode can be used in several possible ways. The obvious use of Test mode is to assess student proficiency. In this case, the test is administered one time, errors are corrected only if necessary to ensure control of the simulation, and the automatically computed performance scores are recorded. Alternatively, test mode might be used to increase student proficiency, by repeating the unguided exercise until either the proficiency criteria have been attained or the time allocated to the test runs out. If desired, a final test could follow such a practice phase.

Scoring

RAPIDS maintains performance measures for each student on each unit of instruction. These scores determine when a student can move on to new topics, and they can be used to control the presentation or skipping of other units. The performance scores of each student can be reviewed by an instructor.

RAPIDS computes a speed score and an accuracy score for each item presented. The speed score is the time required for the student to complete the item, whether right or wrong, including any time devoted to studying expositions. The accuracy score for an item is 1 if the student performed the action correctly on the first try and 0.5 if the student performed the action correctly on the second try; otherwise it is 0.

The speed score for a content unit is the total time spent in the unit, including the time spent observing expositions. The accuracy score for a content unit is the percent of the items in the unit that were correct. Items not attempted due to the expiration of time limits count as incorrect.

The accuracy score for an organizational unit is the weighted average of the accuracy scores of the constituent units, whether they are organizational units or content units. The speed score for an organizational unit is the total of the speed scores for the constituent units.

Managing Time and Content Presentation

For each unit in a course, the instructional plan specifies the maximum time allowed, the minimum and maximum repetitions allowed, and the speed and accuracy scores required to successfully pass. The maximum time parameter limits the duration of a unit. The accuracy and speed scores specify the criteria for success on a unit.

RAPIDS delivers a unit in the specified mode the minimum number of times, unless time runs out first. If the minimum presentations are completed, it determines if the student has achieved satisfactory performance. Performance on a unit is satisfactory if
the accuracy score meets or exceeds the criterion, and
* the speed score is equal to or less than the criterion.

If these rules are satisfied, then *RAPIDS* moves on to the next unit. If not, *RAPIDS* remains in the unit until

1. the student performs the unit satisfactorily, or
2. the maximum number of repetitions is reached, or
3. the time for the unit expires.

*RAPIDS* applies one other important rule when an organizational unit ends: *RAPIDS* examines the time allocated to that unit and the time expended by the student. If there is time available and if some of the member units were not passed successfully because time ran out for them, then *RAPIDS* repeats those failed units until they are all completed successfully or until time on the parent unit is exhausted. In this manner, *RAPIDS* attempts to selectively apply unused time to those sub-units needing more attention.

Figure 6 shows an example specification for an organizational unit called *Troubleshooting*.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Weight</th>
<th>Mode</th>
<th>Condition</th>
<th>Maximum</th>
<th>Minimum</th>
<th>Limit</th>
<th>Accuracy</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptoms</td>
<td>-</td>
<td>Instruct</td>
<td>-</td>
<td>2</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Symptoms</td>
<td>1</td>
<td>Drill</td>
<td>-</td>
<td>3</td>
<td>1</td>
<td>15</td>
<td>85</td>
<td>7</td>
</tr>
<tr>
<td>Strategies</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>80</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Figure 6. Troubleshooting Unit**

This unit first presents Unit Symptoms in Instruct (I) mode. The student has up to 20 minutes to work through the unit twice (minimum number of repetitions is 2). The only criterion for success in Instruct mode is that the learner steps through the instruction in the time allowed; thus, no speed and accuracy scores are specified.

The same unit is then presented in Drill (D) mode. When the student can get 85% of the items correct, completing the unit in under 7 minutes, the exercise is not repeated further. If the student cannot attain this proficiency in 15 minutes or after three repetitions, then the Drill is ended. Thus, this particular instructional plan ensures that a learner will not expend all available time on one unit that is particularly troublesome.

Finally, a large unit on Strategies is presented. It is an organizational unit, presented just once over the course of 80 minutes. It might consist of several content units in various instructional modes, or even more organizational units. If the student can learn the material and get through any drills or tests presented within the 80 minutes, then the Strategies unit ends successfully.
The student's accuracy score on this Troubleshooting unit is a weighted average of the score on Symptoms (weight 1) and on Strategies (weight 2). Thus, a score of 75% on Symptoms and a score of 90% on Strategies yields a composite score of 85% for this Troubleshooting unit.

IV. AUTHORING AND INSTRUCTOR FUNCTIONS

The authoring functions in RAPIDS are designed to maintain the generic power of IMTS; i.e., an author can construct a simulation of choice and produce instruction for that device without involving domain-specific programming.

Alternative Time Allocation Strategies

The way in which time and repetitions are allocated to units heavily influences the character of the instruction. One possible approach is to set the minimum and maximum repetitions for each unit to closely bracket a value felt to be ideal for it. Thus, the minimum and maximum repetitions for a short front panel drill might be 2 and 3, thereby assuring high uniformity of exposure for every student.

The drawback of this strategy, of course, is that fast learners or students with prior knowledge will be forced to repeat the unit unnecessarily, but at least they would do so quickly. Thus, the minimum repetitions will often be set to a low value, such as 1; however instructional policy and the nature of some units may argue for a higher minimum.

The maximum time to allocate to a unit has even more interesting implications. In some instruction, the student should remain on a subject until it is mastered; to proceed to dependent subjects would be unproductive if the critical prerequisite subject were not mastered. In this case, the maximum time for each such critical unit should be set to a high number, such as the available course time, so that time never runs out on any individual unit unless there is no more time for the course.

In this case, a learner might expend a substantial amount of the available instruction time in one troublesome unit. Following such a problem area, the student would have to move through the remaining units more quickly to complete the course. In an extreme case, a student would expend the entire remaining course time in the problem unit.

In other situations, the desire is to ensure that every student is exposed to all topics, something like the way in which conventional classrooms are typically run. In this case, the RAPIDS plan would specify maximum times on member units that sum to the time available in the parent unit. Using the time allocations shown above, a student having trouble with the Symptom drill would exit it after 15 minutes (the time limit) and move on to the Strategies unit. If the student completes the Strategies unit
successfully, then he or she would be returned to the Symptoms unit if the time expended on the Troubleshooting unit has not exceeded its time limit.

Thus, a relatively intelligent scheduling process is achieved in which all students are sure to have a nominal amount of time on each subject yet available time can be dedicated to problem areas for individual learners.

The RAPIDS Authoring Tools

RAPIDS authoring is performed using two editors. The content editor is used to create content units, including action items and expositions. The instructional plan editor is used to build organizational units that form a complete course composed of invocations of the content units.

Authoring Content Units

The content expert creates instructional units by performing operations on the simulation graphics and by entering explanations and graphic aids, as appropriate. This process will create front panel drills, drills for learning functions and organizations of system components, and various types of diagnostic part-task exercises such as identifying indicators affected by various failures. Naming drills, for example, are created by selecting objects or areas of the graphic simulation and keying-in related names.

The authoring environment is shown in Figure 7. The expert sees a list of existing Content units in the left-hand text window. Any of these may be edited or deleted, or a new unit can be constructed.

To build a new unit, the expert selects NEW and enters a name for the unit. A short comment may be entered to document the unit for course developers using it. If necessary, the expert then selects an existing system configuration or creates a new one by setting switches to produce the mode desired. Then the expert produces the following elements: the exposition to introduce the unit (see below), all the action items, and the closing exposition. Finally, the expert indicates whether the items should be delivered in fixed or random order, and whether the names of the items (called identifying text) should be shown in Test mode.¹

¹In a performance drill, RAPIDS presents the identifying text of each step, such as Now set the Power Switch to On; however, in Test mode, if the author has specified that identifying text not be presented, RAPIDS simply prompts the learner with Now do the next step.
Figure 7. RAPIDS Authoring Environment

Figure 8 shows the status table for a unit under construction. This table indicates that a closing exposition has not been created, and that the items will be delivered in random order.

Figure 8. Status Table for Unit Under Construction.
Authoring Expositions

Each part of an exposition is authored by selecting the exposition type from the menu shown in Figure 9 and then, for some of the types, supplying information required. For a Text type, the expert keys-in any amount of text. For a highlighting type, the expert selects objects or areas to be highlighted.

![Exposition Type Menu](image)

*Figure 9. Exposition Type Menu.*

For a video type, the expert uses the menu shown in Figure 10 to control the videodisc player and to select single frames or ranges.

![Video Exposition Menu](image)

*Figure 10. Video Exposition Menu.*

Any number of exposition types can be used to form a complete presentation. Typically the expert will call for a student response (clicking the mouse) between major sections, so that the student can pace the presentation.

Authoring Content Items

Content units consist of sets of content items, each of which requires some student action. When an author creates a *New* content item, an item editor window appears (See Figure 11).

Student actions are typically authored in a very direct fashion. The author clicks on the word *Action* in the editor window, and then simply performs the task that this item will require of the student. If the author changes a switch in the simulation, then the student action required by this content item will be that the same switch be put into the new setting. The *identifying text* is automatically produced by the editor, based on the author's action. In Figure 12, the identifying text was automatically composed when the expert set the Left Start Button. *RAPIDS* will present this text to the student to prompt this action.

19
Like content units, content items may have their own pre- and post-expositions. Thus, unit expositions can provide overviews, general explanations, and summaries; item expositions, however, can provide very detailed information about each step of a procedure.

Other student action types, such as object identification (clicking-on a named object, for example), replacing a faulty object, performing a test, or choosing from a menu of textual choices, are also supported in the content item editor. A menu of these options is presented when the author chooses to specify a student action. In Figure 13, the author has specified that a replacement is called for in a content unit. The editor directs the author to click-on the object that the student should replace.

Direct manipulation techniques are also supported for authoring the other types of student actions, such as menu selection and object identification.

Authoring the Instructional Plan

The instructional plan is authored by creating a graphic diagram that represents the composition of the course, and by specifying the manner in which each unit will be
instructed. The units appear as boxes in this graphic representation, as shown in Figure 14.

![Diagram showing units and options]

**Figure 13.** Item Editor Options Menu.

Here, the expert has selected the Elementary Drill unit and is seeing the default values set for Drill mode. These default values can be changed as desired.
The names of the units in this representation are assigned by the course planner as the plan diagram is produced. Whenever the content units have been produced, they are associated with each of the content units in the plan. Thus, the plan can be fully developed prior to the production of any of the content units, or it can be entered after the content units have been constructed.

**Monitoring Student Progress**

The instructional plan is also used as the medium for determining the progress of a particular student. The user enters a student’s number and then sees the instructional plan. Upon selecting a unit of instruction at any level, the user sees the performance data for that student on that unit.

In Figure 15, the user has selected the content unit Diverter Valve Intro and sees the performance data for the student.

![Sample Student Performance Data for Content Unit](image)

In this manner, the instructor can determine what units have been attempted, what units have been successfully passed, and how well the student is doing at each level. Upon completion of a course, the data for the top-level node reflects the student’s overall performance on the course.

**V. EXPLORATION MODES IN RAPIDS**

In RAPIDS, the author can constrain the sequence of actions that the student is to perform in the simulation, requiring a particular sequence of operations. This manner of instructing has been described in the section above. There are times, however, when
authors would like to present problems to students and allow them to work more freely and naturally to achieve some goal. RAPIDS therefore includes a student action type called "Free Play."

**Free Play Simulation Sequences**

*Free Play* student actions have one of the following goals:

- Establish a particular device mode (system configuration)
- Isolate and repair a simulated malfunction condition
- Explore the device behavior without system intervention

**System Configuration**

The first option makes it possible to create a task that instructs the student to "Set up the low-power transmit mode," and to recognize the specified mode when it is attained. Using the Free Play option, authors do not have to require that a specific sequence of control manipulations be used to achieve the mode. The author defines a system configuration objective by directly manipulating the simulation. Students work until they believe they have achieved the objective configuration; then they select an *End Problem* menu item.

If the mode was not achieved, the student is informed of the error and given another chance. If the unit is being performed in drill mode and the student errs again, then RAPIDS will show the student every action required to achieve the mode. Any scene changes necessary to demonstrate the sequence of object manipulations will be performed automatically.

**Fault Diagnosis**

Two types of diagnostic exercises can be presented: (a) problems in which no diagnostic guidance or advice is available, and (b) problems for which guidance is available, provided by the Profile (Towne, 1984, 1986; Towne & Johnson, 1987) model of diagnosis.

*Without AI Guidance*. This option presents troubleshooting problems to be worked without intervention or support by the instructional system. Any single or multiple set of malfunctions can be selected by the author and presented by RAPIDS for Free Play diagnostic exercise. The student works to isolate and replace the failed components, then calls for the end of the problem by selecting *End Problem*. If the student has not replaced all the failed components (in the correct order if there is a cascading effect), then RAPIDS gives the student a second chance. After a second chance, RAPIDS identifies the failed component(s) and continues to the next item in the content unit.
With AI Guidance. At the discretion of the instructional author, diagnostic problems can be presented with guidance available. To do this, symptom effect data must be generated using a failure simulation and analysis routine supplied with IMTS. This enables the Profile model to evaluate the student's diagnostic skill, and to provide assistance whenever required. If a student cannot solve a problem in this mode, the diagnostic performance of an expert (Profile) is demonstrated for the problem.

This mode of instruction is identical to the standard IMTS diagnostic instruction, but it now operates within a RAPIDS course. Using RAPIDS, the time invested in a problem can be controlled, problems can be selected depending upon student performance on any RAPIDS drills, and remedial RAPIDS-authored instruction can be provided, as appropriate.

Device Behavior Exploration

The third option is the simplest one; students can freely explore the simulation environment until they choose to leave it or until time expires. Students can manipulate controls in order to observe indicator values in a variety of different modes. They can use test equipment to get test point readings under conditions of their choosing. They can even elect to insert failures of their choosing into the simulated equipment and then observe the effects on indicator values and test equipment readings.

VI. SUMMARY AND CONCLUSIONS

RAPIDS constitutes a new kind of tool for building courseware in which the course is built on the foundation of a simulation. We first compare RAPIDS with other tools for creating computer-based instruction. The report ends with a short discussion of future plans.

Comparing RAPIDS with Other CBI Tools

Authors of computer-based instruction (CBI) courses make use of a variety of tools to develop course materials. These tools include:

- conventional programming languages,
- CBI programming languages,
- graphical programming environments (e.g., Hypercard), and
- frame-based CBI authoring systems.

There are a number of features worth examining for each of these environments. Consider the following questions:

How easy is it for a domain expert to author instructional content?
What is the probability of bugs--unanticipated breakdowns in the course?
How effective is the support for organizing authored content (e.g., lessons) into complete courses?

Are low-level instructional interactions with the student handled automatically in the system?

How wide is the domain of subject matters that can be instructed using the system?

Ease of Authoring

Conventional programming languages can be difficult for subject-matter experts to use. Computer programming is a complex skill that is rarely mastered by subject-matter experts. Teams of professional programmers, instructional designers, and content experts have sometimes created successful courses working together. Bork and his associates, for example, have had good results using the APL or Pascal programming languages as a courseware development tool (Bork, 1981). This approach seems to work well for large courses that can be designed and developed in a top-down fashion. It is not particularly well suited to prototyping, however. In addition, the maintenance of such courses can be difficult when the original programmers who developed a course are no longer part of the team.

CAI programming languages are intended to promote programming by subject-matter experts either instead of or in collaboration with professional programmers. Some formerly non-programming content experts have found it possible to master a subset of a CAI programming language (such as TUTOR; see Bitzer, Sherwood, & Tenczar, 1973; Sherwood, 1973) that makes it possible for them to develop course materials of reasonable quality. Few content authors find the transition to programmer/author a comfortable one, however.

Expert instructors are good at explaining complex phenomena if they have means for conveniently demonstrating the effects and for focusing the learner's attention on critical events. In many conventional CAI programming languages, however, the author is burdened with programming many low-level interactions between learner and computer, dealing with student errors, providing and manipulating the graphic displays that represent the real world, and collecting and summarizing student data. The RAPIDS technique attempts to partition course development into two specialties -- instructional planning and subject-matter exposition -- and to automatically take over the low-level details concerned with delivering the instruction according to the plan.

Graphical programming environments like Hypercard are, in some respects, very easy to use to build interactive training applications. Hypercard makes it especially easy to develop training packages that have a strongly user-controlled browsing style. Authors can easily make certain "hot spots" on one card serve as links to other cards, for example. Other aspects of instructional development (such as answer-judging, scoring, and student record-keeping), however, are no better supported than in conventional programming languages; that is, not at all. Development of such features
requires programming development (scripting) that is as difficult as in ordinary programming languages.

Like RAPIDS, frame-based CBI authoring systems make authoring very easy. Frame-based systems and RAPIDS have complementary ranges of application. Frame-based authoring is appropriate for essentially expository instruction, whereas RAPIDS is better suited for instruction that relies on interactive simulation.

Probability of Bugs

Instructional development in a conventional programming language such as C, Pascal, or Lisp offers complete control of the computer-based environment to the author. This control comes at the cost of exposure to programming bugs (not to be confused with errors in technical content) that may result in instructional catastrophes.

The use of CAI programming languages somewhat reduces the probability of introducing bugs in a course, especially for courses that make use of standard, well-debugged CAI programming constructs. Bugs are most likely to appear in the most innovative parts of lessons built with CAI languages, such as in programmed interactive simulations.

Graphical programming environments such as Hypercard -- like frame-based authoring systems -- reduce the chance of creating system-crashing bugs. On the other hand, authors can certainly build Hypercard routines that put students into situations that prevent further activity without the intervention of someone with some expertise in the Hypercard environment. (For example, authors can create dead-end cards. If the student doesn’t know any of the "magic" Hypercard keyboard commands for changing the displayed card, he or she will be stuck there until help arrives from someone more knowledgeable.)

This type of programming error is unlikely to be produced using a frame-based authoring system. Their limited set of constructs typically constrains the events of a training session to a sequence of frame presentations and student responses. Though content errors can be authored, it is usually not possible to create a way for the course to break down completely during use.

RAPIDS, like frame-based authoring systems, makes it difficult or impossible for lesson and course authors to produce materials with structural bugs. Because the simulation on which the course is based has already been developed and tested, the correctness of many of the author's actions is assured. The RAPIDS authoring environment constrains the author to actions that are possible in the IMTS simulation on which a RAPIDS course is based.
Effective Tools for Course-Level Planning

Conventional programming languages offer no built-in support for instructional planning. Such support must be developed by the programmer. Where a programming team frequently develops courses with instructional planning requirements, the team may develop special code units that can be reused in different courses to support the development and delivery of an instructional plan or organization.

Some CAI languages, such as the PLATO system's TUTOR, offer tools in support of computer-managed instruction, including the organization of lessons into courses.

Graphical programming languages, such as Hypercard, do not offer instructional planning tools, although -- as with programming languages -- the developers of a course could produce such tools using Hypercard.

Some frame-based CAI authoring systems -- like some CAI languages -- offer CMI features and utilities that can assist in organizing lessons into courses and in determining the control of lesson presentation to particular students.

In RAPIDS, the technical content is produced by operating the simulated device, while the instructional plan is produced by creating a graphical model of the course and by specifying parameters that control the presentation. Because instructional content is packaged into stand-alone units that contain no instructional strategy information, any unit can be used multiple times in a course and in different modes of instruction. Furthermore, a unit of instructional content can be used in different instructional plans. This makes it easy to experiment with different approaches to instructional organization without changing the details of the technical content.

Unlike CAI programming languages, RAPIDS provides no true branching capability, although any unit can be skipped. There are, however, significant ways in which the training is individualized. First, the learner paces the rate of progressing through instruction of declarative information and procedural demonstrations. Second, the duration and repetitions of exercises are determined by the learner's ability; skilled students will move rapidly through easier portions while others will repeat units numerous times if necessary (and specified in the plan) to meet the proficiency criteria. Third, the extent of guidance provided in an instructional item is determined by the student's ability. If the student can do an action without assistance, then he or she gains credit for that demonstrated proficiency, and thereby moves closer to completion of the unit. Otherwise, the student is assisted. Finally, the presentation of any unit can be skipped if the individual student has demonstrated sufficient proficiency on related units.
Automatic Handling of Low-Level Student Interactions

Ordinary programming languages offer no built-in support for low-level student interactions, such as answer-judging, scoring, and item feedback. Neither does Hypercard.

CAI languages and frame-oriented authoring systems ordinarily do offer some support for these features, and thereby improve the productivity of developers and the average quality of student interaction in lessons. With CAI languages, in particular, there are often many interaction options available to authors. As a result, authors are sometimes encouraged to take a very interaction-level orientation to authoring lessons, rather than a more global content approach.

At the lowest level, the item level, there are significant differences between the way RAPIDS interacts with different learners. A learner who cannot respond to an item, or who responds incorrectly, will be handled differently than one who responds correctly. These low-level remediations are handled automatically by RAPIDS. Thus, the RAPIDS author is more concerned with high-level strategic and content issues than with low-level interactive issues -- somewhat in contrast to conventional CAI authoring.

Breadth of the Domain of Application of the Tool

Conventional programming languages can be used to create CBI courses for any domain. Some of the experimental intelligent computer-aided instruction projects are monuments to what can be done using conventional programming languages (such as Lisp) to create innovative instruction. The inverse of breadth of applicability, of course, is the expense of developing quality courses without special features in support of instruction.

CAI programming languages differ in their power, but some have as much range of expression as do conventional programming languages. These development tools can be applied to a very wide range of training domains and can be used to implement many different styles of training. Needless to say, some types of courses (such as those that employ complex interactive simulations) will be very costly to develop using CAI programming methodology.

The Hypercard programming environment does not offer the full range of expression of a modern programming language. In particular, the development of complex data structures is not well supported. This limits (or at least makes more complicated) certain types of instructional development. In addition, Hypercard passes through to the end user (the student) more power and flexibility than may be desired in some situations. Students aware of these features may be able to subvert the author's control of the instructional process. Hypercard offers a somewhat smaller domain of application than do programming languages.
Frame-based authoring systems have a natural domain that includes topics that can be taught in a largely expository fashion. This is a wide domain, but it is less wide than the programming approaches to instruction and it does not include inherently simulation-based training.

The RAPIDS tools are most useful in instructional domains in which there is a complex system that can be simulated. The range of effective applications includes such topics as biology, manufacturing processes, economics, troubleshooting, and traffic flow analysis, as well as equipment operation. Courses on subjects such as history, language, philosophy, and sports would seem to offer few opportunities for implementation using the simulation-based approach of RAPIDS.

**Future Developments**

In the relatively near future, the simulation power of IMTS will be significantly expanded to allow for continuous-change simulations and time-dependent event simulation. When this work is completed, a simulation object may exhibit a state appearance that is computed from the states of other objects, or from the values of object attributes, or from the current time. When the student performs an operation in this version, the simulation graphics will change continuously over time, in a manner that more accurately reflects the responses of many real devices.

Not only will the simulation of some devices be far more realistic, but the range of simulations possible for development will be increased dramatically. The RAPIDS functions described here will operate almost identically under the new simulation process. In addition to the real-time simulation mode, there will be options for speeding up processes that occur slowly in real devices, and for slowing down reactions that happen too fast for effective viewing.

Other, more minor, enhancements will be added to RAPIDS. One such planned development is the incorporation of a general-purpose voice output capability. Currently, RAPIDS expositions can play videodisc segments with sound. The exposition possibilities will be expanded to include use of digital voice output, allowing an author to guide learners by voice while the learners concentrate on the graphic screen.
REFERENCES


