

ONR Final Report

**Intelligent Instruction of
Dynamic Decision Making Tasks**

Douglas M. Towne

**Behavioral Technology Laboratories
University of Southern California**

**Developed under funding by:
Office of Naval Research**

Under ONR Contract No. N00014-98-1-0214

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited



Approved for Public Release: Distribution Unlimited
Reproduction in Whole or in Part is permitted for any purpose of the United States Government

20000317 035

**Final Report
Contract N00014-98-1-0214**

**Intelligent Instruction of
Dynamic Decision Making Tasks**

Douglas M. Towne

Technical Report No. 121

**Behavioral Technology Laboratories
University of Southern California
1120 Pope St., Suite 201C
St. Helena, CA 94574**

**(707) 963-3060
dtowne@usc.edu
<http://www.fcs.net/dtowne/dmt.htm>**

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March, 2000	3. REPORT TYPE AND DATES COVERED Final Technical (4/1/98 - 3/31/00)
----------------------------------	-------------------------------	--

4. TITLE AND SUBTITLE Intelligent Instruction of Dynamic Decision Making Tasks	5. FUNDING NUMBERS N00014-98-1-0214
---	--

6. AUTHOR(S) Douglas M. Towne

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Southern California Behavioral Technology Laboratories 1120 Pope St., Suite 201C St. Helena, CA 94574	8. PERFORMING ORGANIZATION REPORT NUMBER Technical Report No. 121
--	---

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
---	--

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public Release: Distribution Unlimited	12b. DISTRIBUTION CODE
--	------------------------

13. ABSTRACT (Maximum 200 words)

Very many of the high-level decision tasks in both the military and private sector involve a complex problem situation changing over time and at least partially in response to the decision maker's actions. Gaining proficiency in such tasks as tactical decision making, response to natural disasters, and hostage negotiation typically requires many years of training and experience.

A simulation-based system, D³M (Dynamic Discrete Decision Making), provides a relatively realistic environment in which to learn and practice managing such operations. The instructional system has the capacities 1) to demonstrate exercises previously worked by experts and issue those experts' rationale as the exercise unfolds, 2) to adaptively produce practice exercises at a difficulty level appropriate to the individual learner, 3) to monitor the learner's actions in relation to critical-condition specifications defined by task experts, and 4) to maintain the status of learning objectives achieved by the learner.

14. SUBJECT TERMS Scenario-based instruction, Intelligent tutoring, Adaptive instruction, Simulation-based instruction	15. NUMBER OF PAGES 72
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL
---	--	---	----------------------------------

ABSTRACT

Very many of the high-level decision tasks in both the military and private sector involve a complex problem situation changing over time and at least partially in response to the decision maker's actions. Gaining proficiency in such tasks as tactical decision making, response to natural disasters, and hostage negotiation typically requires many years of training and experience.

A simulation-based system, D³M (Dynamic Discrete Decision Making), provides a relatively realistic environment in which to learn and practice managing such operations. The instructional system has the capacities 1) to demonstrate exercises previously worked by experts and issue those experts' rationale as the exercise unfolds, 2) to adaptively produce practice exercises at a difficulty level appropriate to the individual learner, 3) to monitor the learner's actions in relation to critical-condition specifications defined by task experts, and 4) to maintain the status of learning objectives achieved by the learner.

ACKNOWLEDGEMENTS

This work was supported by ONR funding, Susan Chipman Scientific Officer.

The system described here was produced in two forms, one for delivery on Linux platforms and one targeting Windows/NT platforms. The Linux-based system relies upon the RIFES simulation authoring and delivery system, developed under ongoing ONR and AF funding, Susan Chipman and Wes Regian Scientific Officers, respectively.

The high-rise fire fighting application described in Part IV was made possible by the participation and contributions of the following three individuals.

- Deputy Chief James Olson (retired), San Francisco Fire Department.
- Mr. Karl Phillips, Chief Engineer, Del Amo Financial Center, Torrance, CA.
- Battalion Chief Louis A. Roupoli, Los Angeles City Fire Department.

Chief Olson and Battalion Chief Roupoli provided invaluable expertise in formulating the fire fighting command process. Mr. Phillips was most generous with his time in providing access to the internal resources of a high rise office building.

Dr. Quentin Pizzini of our organization contributed significantly to the analysis and specification of a typical high rise office building, he conducted liaison with the Los Angeles City Fire Department, and he provided able support in producing video media.

CONTENTS

I. Overview	1
Task Types	2
Application Development	5
Instruction	8
II. Instructional Functions	10
Instructional Strategy	10
Proficiency Assessment	12
Scenario Tailoring	14
Exercise Scoring	19
Between-exercise Functions	20
III. Application Development Guide	21
Task Representation	21
Simulation Objects	22
Simulation Control	24
Producing Scenarios	26
Scheduled Events	29
Theaters of Operation	31
Instructional Specifications	31
Learning Objectives	32
Exercise Specification	33
Scoring	34
Explanations	35
Instructional Strategy	38
User Progress	39
Options	40
Application Evaluation and Testing	42
IV. Sample Application	43
Use of Video Media	43
Use of Text to Speech Media	43
The Internal Simulation	44
Representing the Task Environment	46
Acting Upon the Simulated World	48
Instructional Content	57
Summary	59
V. Conclusions	60
Instructional Issues	60
Development Issues	61
Adaptation	62

References	64
Appendix A. Detailed Student Performance File	65
Appendix B. Student Status and Progress File	66
Appendix C. Applicator Routines	67
Appendix D. Help Text	68
Appendix E. Learning Objectives	70

I. Overview

Many critical tasks in both the military and private sectors require a stream of decisions dedicated to resolving or managing some critical and dynamic situation. The observable result of this cognitive activity is the performance of various discrete actions, often separated with periods in which no actions are taken. The actions performed may have the objective of directly remedying a situation, such as negotiating with a hostage taker; they may direct others to take action, such as calling in water bombers over a forest fire; or they may be conducted to acquire additional information, such as changing location or calling for a report from the field.

Endeavors that may involve such discrete-action performance include the following (Towne, 1998):

Military	tactical command and control
	damage control
	strategic planning
	logistics command and control
Civilian	air traffic control
	medical emergency treatment
	control of communicable diseases
	police operations
	fire fighting
	911 dispatching
	natural disaster or rescue
	toxic/industrial accident or disaster
Terrorism prevention or response	hostage negotiation
	explosive threats
	biological/chemical/nuclear

This report describes a system, D³M, (Dynamic Discrete-action Decision Making) for developing and delivering scenario-based instruction for such task domains. The system calls upon a domain expert to specify the task environment, to identify a set of prototype scenarios of instructional significance, to demonstrate and explain proficient decision making performance in the task environment, and to specify domain-specific conditions that reflect upon the learner's proficiency.

It is the role of the instructional system to adapt expert-defined practice scenarios to the learner, to present authored instructional dialogs, and to maintain proficiency evaluations as the practice scenarios unfold. D³M is intended to support a learner as he or she practices performing the task in a simulation environment that may change in response to

Overview

the learner's actions as well as to simulated factors beyond the learner's control. It is not intended to instruct general decision making skills isolated from the domain.

The remainder of part I of this report presents an overview of the kinds of tasks addressed by the instructional system, the steps involved in developing an instructional application, and the instructional processes that are executed. Part II describes the instructional processes that are carried out by the system; Part III presents the application development process; and Part IV provides a complete example application, the command of high rise fire fighting operations. Part III, the application guide, could be skipped by those not intending to develop instructional applications.

Task Types

This work addresses a task type termed *dynamic discrete-action* (DDA), signifying that 1) the problem environment changes over the time during which the decision maker deals with it, and 2) the observable actions are relatively discrete. The DDA task type may be distinguished from those involving a sequence of decisions to solve some static problem, such as fault diagnosis, and from those involving continuous motor-perceptual actions, such as tracking. The following table lists examples of the types of tasks in each possible category.

PROBLEM ENVIRONMENT	OBSERVABLE ACTION TYPE	
	Discrete	Continuous
Static	Troubleshooting Assembly Stock portfolio assessment	Not defined
Dynamic	Air traffic control Fire fighting command Tactical decision making	Space shuttle docking Aircraft landing Tank maneuvering

Of course any partitioning of problem spaces is imperfect. We know that sometimes a system can change while one is troubleshooting it, but typically the failure state of the target system is fixed during the troubleshooting phase, unless the diagnostician introduces additional faults. Likewise, a space shuttle docking task probably also involves decision making operations in addition to those managing the motor-perceptual performance. Nevertheless, the point is that we are not attempting here to deal with such motor-perceptual operations, or with problem solving tasks in which the environment does not react to at least some of the actions taken by the decision maker.

The Family of DDA Tasks

A significant portion of the research on dynamic discrete-action decision making tasks has been devoted to one particular case, termed tactical decision making (TDM). Tannenbaum, Beard, and Salas (1992) offer the following list to characterize the attributes of TDM:

Overview

- rapidly evolving scenarios
- time compression
- threat
- adverse physical conditions
- auditory overload/interference
- high workload
- ambiguity
- command pressure

This characterization applies very well to the more general class of DDA tasks, as illustrated by the following table. This displays some subjective judgments about the extent to which various tasks involve the characteristics enumerated for TDM.

Characteristic	tactical decision making	air traffic control	fire fighting	emergency medical care	disaster response	mission control
rapidly evolving scenarios	+++	+++	+++	+++	+++	+++
time compression	+++	+++	+++	+++	+++	+++
threat (to life or property)	+++	+++	+++	+++	+++	+++
adverse physical conditions	varies	+	+++	++	+++	+
auditory overload/interference	+++	+++	++	+++	+	++
high workload	+++	+++	++	+++	+++	+++
ambiguity	+++	+	+++	+++	++	+
command pressure	+++	+++	++	++	++	++

+++ high involvement ++ medium involvement + low involvement

The Decision Making Processes

While the observable actions in such tasks are relatively discrete, as opposed to tracking tasks for example, the decision making itself is apparently ongoing. It is doubtful that the observable actions of the decision maker would map back in any orderly manner to the cognitive processes which generated them, i.e., we doubt that for each action taken the decision maker made one conscious and explicit consideration of that action.

Zachary and Ryder (1997) provide a thorough discussion of modern decision theories, and aspects of the decision making process. The work described here appears not incompatible with the tenets of the naturalistic decision theory they review. Within the account of this theory, Zachary and Ryder note:

“ ... human experts do not make decisions in an analytical manner, indeed or even in a conscious manner, but rather apply their accumulated experience and knowledge (collectively called their expertise) to identify and effect the most appropriate action.”

They further cite the tenet of that theory:

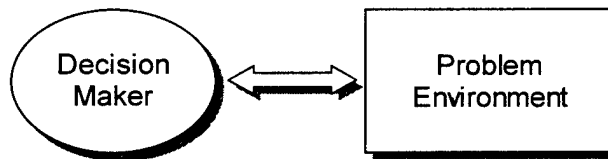
“There is no decision event, but rather a larger dynamic task in which decision makers take actions.”

Additionally, part of expertise also includes inaction, with performers electing instead to await a change in the problem environment or for additional information. The point here is that we make no assumptions about the nature of the decision making processes that drive the performance of DDA tasks, except for the belief that an individual’s proficiency in performing such tasks can be significantly improved via guided practice in simulated environments. This is not to say that all aspects of the kinds of tasks listed above yield easily to computer based representation and simulation, as discussed next.

Problem Representation and Simulation Issues

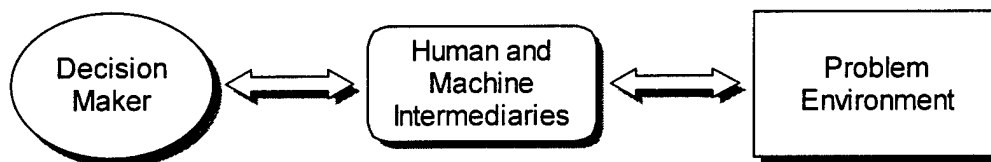
The prospects for representing DDA task domains with sufficient realism to support learning vary greatly from one domain to another. The key distinction is whether or not the decision maker experiences and acts upon the problem environment directly or indirectly.

In those tasks in which the decision maker experiences and operates directly upon the environment, as shown here, that environment must be represented with relatively high fidelity, possibly including representing spatial relationships as well as sensory cues such as sounds and temperatures.



Examples of such tasks are: 1) apprehension of suspects in a building, or 2) leading disaster victims to safety. While virtual reality technology may ultimately progress sufficiently to cope with these requirements, this remains an expensive and difficult technology to apply today.

Fortunately, many DDA tasks are accomplished indirectly, i.e., the decision maker neither senses the problem environment directly, nor does he or she operate directly upon it. Instead the command function is often done at a remote site in which information is received via voice or data lines, and responses to the problem are carried out by those under the command of the decision maker. As shown here, both human and machine intermediaries may serve between the decision maker and the real world.



In these cases, the problem *representation* becomes highly tractable, as it is limited to emulating what the decision maker would see and hear at the command post, and the

Overview

problem *simulation* centers upon defining how the problem environment would change in response to passage of time and to actions of the forces under the decision maker's command. We will refer again to the functions of representing the problem interface and simulating the problem environment when discussing the development of training applications.

Learning Issues

Regardless of the manner in which the environment responds to the decision maker, learning what to do and when to do it is typically more difficult than learning to perform the actions themselves. Even when time, threat, and uncertainty are not factors of a problem environment, we find this to be true. Youngsters often learn to add and subtract numbers relatively easily, but have much more difficulty completing the dreaded "word problems", which require that they know what values to deal with, what operations to perform on those values, and the order in which to carry out the operations. That is, real world problems require generating solutions as well as performing the operations that implement those solutions.

In critical time-dependent environments the decision maker must also deal with a changing problem space, one typically obscured with unknowns and often charged with high risks. Sometimes planning, sometimes reacting, and sometimes just waiting for more information or for a change in conditions, the proficient decision maker is constantly evaluating the situation and considering alternatives.

As a consequence, instruction of DDA tasks must provide an environment in which the learner can exercise both the conduct of these decision making elements as well as managing his or her focus of attention. This latter aspect of performance is virtually impossible to address via conventional CBT methods in which a learner and a machine carry on a dialog about the task. The mere act of asking a learner to evaluate a particular facet of a situation artificially directs the learner's attention to that aspect. While this may be both necessary and desirable when instructing novices, it strips away from the practice environment a critical aspect of performance that must be experienced to achieve true expertise.

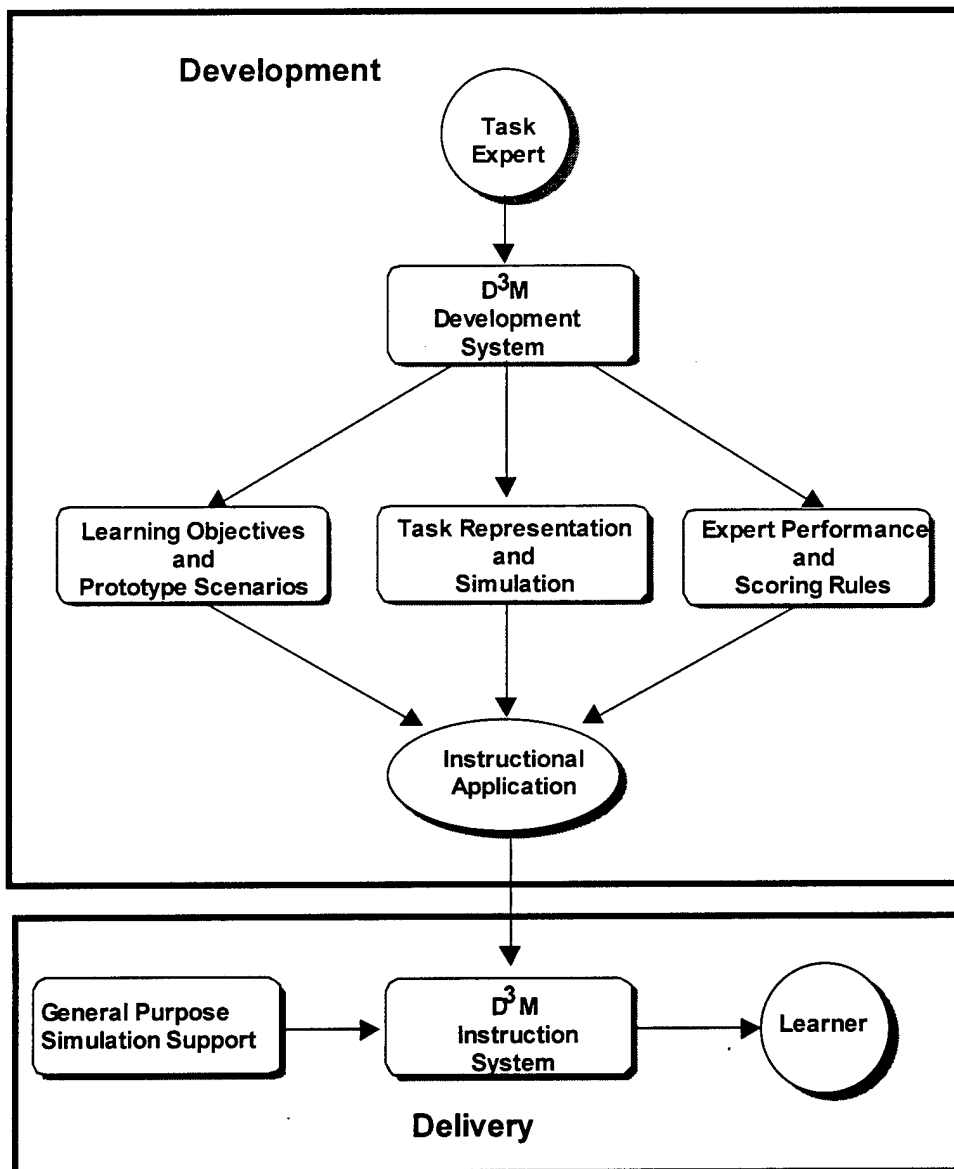
Application Development

The following diagram reflects the basic elements of a D³M instructional application. Working with the application development system, the applicator 1) enumerates the *learning objectives* to be addressed by the application and designs a number of prototype scenarios to exercise those objectives, 2) specifies *the task environment* to be represented in terms of the kinds of actions that may be taken by the learner and the way the real world will be presented to the learner, and 3) defines task-specific scoring conditions that reflect good or poor performance and the relative import of each.

Upon development of an internal *simulation* of the task environment, task experts perform the prototype scenarios to the criterion level of proficiency desired in the

Overview

trainees, and the scores achieved are recorded for use in assessing learners. Details for producing these scenarios will be provided first, following by an overview of scoring and provision of instructional dialogs.



Prototype Scenarios

Various alternatives have been explored for producing practice scenarios that are adapted to the individual learner. Stretton and Lackie (1996) employ learning objectives as the basis, and produce scenarios from those objectives. The approach described here (Towne, 1999) calls upon the task expert to define a number of *prototype scenarios* each of which presents a situation of instructional value. For example, in training air traffic control, one prototype scenario might involve a commercial aircraft that has lost radio reception but can transmit, and is running low on fuel near an airport with poor landing conditions.

Overview

Each prototype scenario establishes the conditions that exist at the start of an exercise, including characteristics of simulated entities that may rule their behavior during the exercise. For example, a tactical decision making scenario might define the positions, headings, and speeds of ships and aircraft in a particular theater of operation, as well as their level of hostility and their intentions. Then, as the learner issues commands, the simulated entities respond according to their capabilities and intentions.

For each prototype scenario, the expert defines two instances: 1) an extremely easy instance, that presents the simplest set of conditions while still retaining the essence of the prototype scenario, and 2) an extremely difficult instance, which presents the same situation under the worst possible conditions, for a decision maker. The scenario conditions are established by specifying values of variables that define the entities in the scenario.

For example, an extremely difficult tactical decision making scenario might involve many hostile aircraft, coming toward the ship or fleet being defended, at low altitude and high speed. The easy version might involve few aircraft, with less threatening headings, at lower speeds and higher altitudes. Both instances might have in common the time of day or night, the weather conditions, the status of the ship's radar, and other matters that establish a particular learning situation.

As outlined below, it is the responsibility of the instructional system 1) to select the prototype scenario to be presented next to an individual learner, depending upon the instructional strategy that has been selected and the progress of the learner, and 2) to customize that prototype scenario so that its difficulty is appropriate for the learner.

Scoring Rules and Instructional Commentary

The task expert supplies a set of performance rules that specify conditions that reflect upon the proficiency of the learner, either positively or negatively. Associated with each rule is a value, positive or negative, that is added to the current score. Full details for this specification are provided in Section II. To permit the system to make valid proficiency assessments, and corresponding adjustments in instruction, there must be some way to interpret an individual's score in relation to the instructional requirements, or the objective proficiency. This is accomplished by having a task expert perform each of the prototype scenarios, as described next, to determine the scores they achieve when performing to the level required by the course.

Expert Performance

Finally, one or more experts perform both the easy and difficult instance of each prototype scenario, producing for each scenario: 1) a complete record of the expert's performance, in a form sufficient to present a real time demonstration, and 2) a performance score corresponding to expert proficiency on the scenario. Optionally, less proficient individuals might be asked to perform the prototype scenarios, yielding a measure of the sensitivity of the scoring system to proficiency. These data are then used to establish proficiency objectives that are known to represent a desired level of ability.

Instruction

D³M operates in two instructional modes: 1) *demonstration* mode, in which expert performances are demonstrated and explained, and 2) *practice* mode, in which exercises are presented for the learner to perform.

Demonstration Mode and Instructional Commentary

Demonstrations of expert performance are produced by having a highly experienced operator perform each of the scenarios. Since D³M automatically records all performances of exercises, the resulting data files are available for replay at any time. At any time during the expert's performance, he or she may select an **Explain Now** button that pauses the progress of the scenario, permitting the entry of a textual commentary that explains an action or passes on the thinking of the expert at that time. As a labor saving device, D³M also permits the developer to pre-author a body of comments so that the expert may select from this set when an appropriate explanation is available. Upon entry or selection of an explanation, D³M proceeds with the task simulation.

Replay of the Expert's Actions. When an expert performance is presented in this mode, the learner observes the scenario progressing just as the expert saw it. In addition, all of the expert's actions are shown. For each action, the simulation pauses momentarily while the mouse cursor moves to the position the expert moved it, a mouse action (down and up) is shown, and then the simulation resumes. The effects produced by the reproduced action are then shown as a normal part of the D³M simulation process.

Presentation of Explanations. During the replay, the task simulation pauses when it encounters an explanation, and that explanation is provided to the learner. When the learner has digested the information, he or she selects a **Continue** button, and the task simulation resumes. This process continues until the exercise ends just as it did when the expert performed it.

As explained below, the replay function of D³M can also permit a learner or an instructor to review the learner's work. It should also be noted that experts can use this facility to demonstrate poor performance and the consequences thereof.

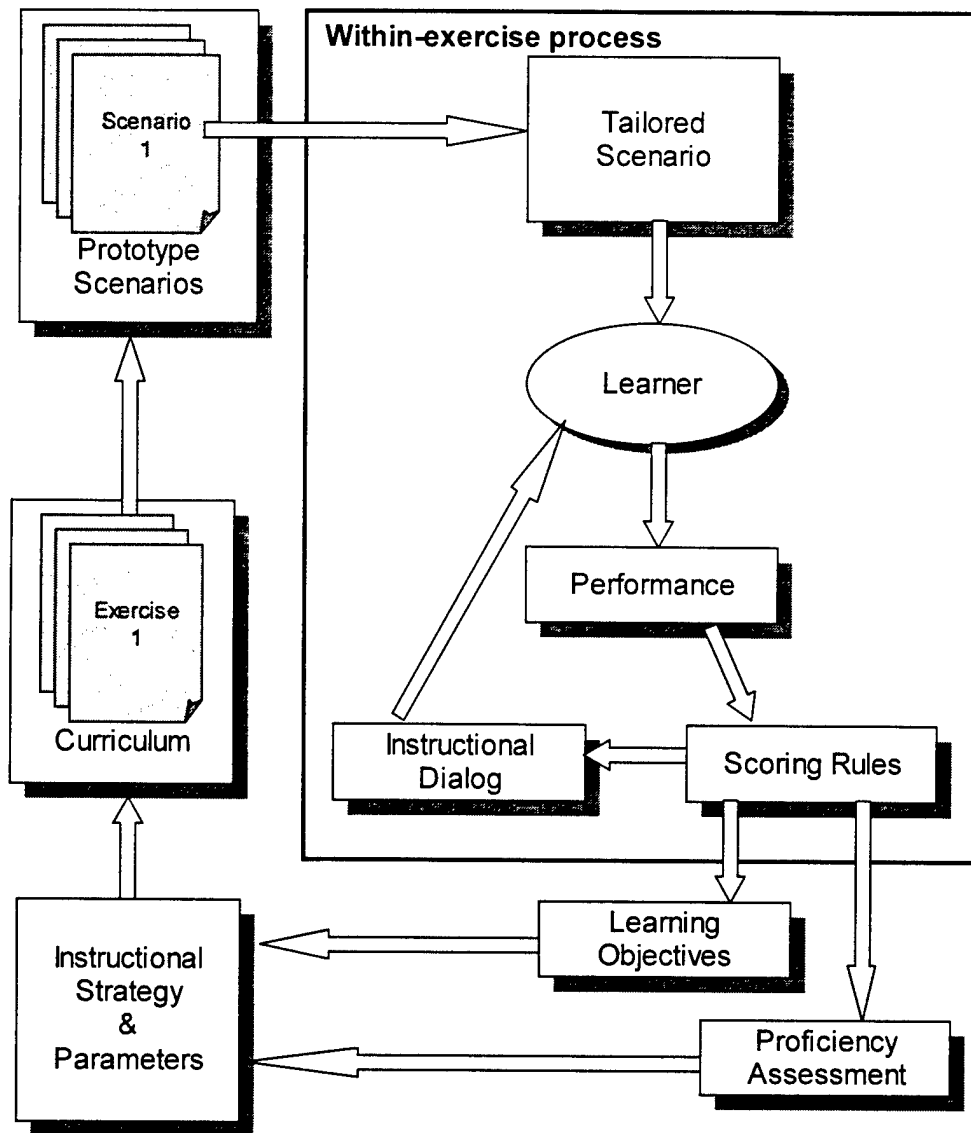
Practice Mode

When a learner works an exercise in practice mode, D³M:

- selects the next exercise for the learner to work, by referring to the *instructional plan*;
- tailors the scenario associated with the selected exercise, to the desired difficulty;
- presents and maintains the scenario as the learner performs
- records the learner's actions, permitting later replay of the performance;
- maintains records of the learner's proficiency and mastered learning objectives.

The instructional process is illustrated in the following diagram.

Overview



The process begins with the automatic selection of an exercise, according to the Instructional Strategy and the individual's status. The selected exercise calls for presentation of a particular prototype scenario, which is then tailored to provide a desired level of difficulty and presented to the learner. As the learner carries out his or her decisions via actions upon the simulation, the instructional system evaluates the proficiency of the performance, and may, depending upon the mode of instruction, intervene with suggestions, positive remarks, or warnings. Whether the system communicates its assessments with the learner or not, it maintains records of the proficiency of the performance on that exercise as well as detailed records of the actions performed and of any learning objectives newly learned or found to be not learned.

The next section will fully describe the functions carried out in this practice mode.

II. Instructional Functions

The instructional functions carried out in practice mode include these:

- producing customized scenarios to meet the individual learner's ability;
- monitoring the learner's work during exercises, providing available commentary;
- maintaining the simulation as the learner acts upon it;
- measuring and recording learner progress and proficiency;

This section will describe these functions in detail, starting with the selection of a suitable exercise, which leads to the production of an appropriately difficult practice scenario, progressing through the interactions within a practice exercise, and concluding with those functions that occur between the completion of one exercise and the initiation of another.

Instructional Strategy

The instructional developer makes selections in the Instructional Strategy to specify how D³M shall manage instruction. This establishes 1) the exercise selection and presentation scheme, 2) a key parameter for maintaining the proficiency measure, 3) the scenario difficulty adjustment parameters, and 4) the total time allowed for all instruction.

Exercise Selection and Presentation Scheme¹

The basic unit of instruction is an *exercise*, which specifies:

- a particular scenario to present;
- a particular theater of operation in which to present the scenario;
- predefined events that will occur during the scenario;
- starting and ending difficulty levels for the exercise;
- minimum and maximum number of trials to present; and
- learning objectives associated with the exercise.

When and if the learner completes an exercise at or above the target difficulty level, that exercise is marked as mastered, and all learning objectives associated with that exercise are marked as achieved. Exercises also call out the minimum and maximum number of trials (repetitions) allowed for an exercise, as well as a time limit. If the learner exceeds the time limit or maximum trials, and has not reached the target level of difficulty, the exercise is marked as failed. The instructional applicator may eliminate either or both of these constraints by setting the number of trials and the exercise time limit to very large numbers.

There are four alternative exercise selection schemes, as follows:

¹ None of these measures refer to individual training sessions; the presentation parameters apply across any number of different instructional sessions.

Instructional Functions

Step. Step through the exercise list in sequence, presenting each exercise one time at its initial difficulty level. Repeat the process, presenting each exercise at a difficulty level determined by the individual's performance. Drop an exercise from the pool when the individual masters it or when the exercise has been presented the maximum times allowed.

Repeat. Present the first exercise in the list, starting at its initial difficulty level. Continue to present the exercise until it is mastered, or until it is presented the maximum specified times. Repeat the process for exercise 2, then 3, and so on.

Random. Select an exercise randomly, and present it at its current difficulty level, which initially is the initial difficulty level for the exercise. Repeat the process, presenting each exercise at the difficulty level determined by the individual's performance. Drop an exercise from the pool when it is mastered, or when it has been presented the maximum times allowed.

Learning. Present the exercise involving the fewest unmastered learning objectives, at its current difficulty level. Note that this could present a previously mastered exercise, one that involves a learning objective that has been set back to *unmastered* due to some performance error committed on a subsequent exercise. This provides a very powerful way to adaptively select exercises to remediate performance deficits.

Example Exercise Presentation Sequences

Suppose there are five exercises, denoted E1 through E5, and suppose that all five exercises call for at least two presentations and a maximum of four. The following table displays possible presentation sequences for the Step, Repeat, and Random schemes (the Learning scheme could produce any sequence).

Presentation Scheme		
Step	Repeat	Random
E1	E1	E3
E2	E1	E1
E3	E1	E5
E4	E1*	E1
E5	E2	E2
E1	E2	E5
E2*	E2*	E2*
E3	E3	E4
E4	E3	E3
E5*	E3	E4*
E1	E3x	E1*
E3*	E4	E3
E4	E4*	E5*
E1x	E5	E3x
E4*	E5	
	E5*	

* indicates the exercise was completed successfully at the target difficulty level.

x indicates that the exercise was performed the maximum number of times allowed (4).

Instructional Functions

The choice of exercise selection scheme depends primarily upon the nature of the task and the instructional environment in which training is conducted. Some kinds of tasks involve unknowns that materially affect the scenarios, consequently the Repeat selection scheme cannot be used for such tasks. This does not necessarily mean that such exercises can only be presented one time, but rather that repetitions must be separated by presentation of other scenarios. Other kinds of tasks involve no unknowns or discoveries, and can be repeated until mastery is attained.

Use of the Random selection scheme would seem to be appropriate only when there are no prerequisite relationships among the scenarios involved in the exercises. Finally, use of the Learning scheme would seem to be effective for virtually any task type, although one might wish to employ a Step process if it is desired to maintain correspondence between automated instruction and classroom presentations.

Proficiency Assessment

During performance of a scenario, a score is maintained via authored rules that detect conditions produced by, or allowed to exist by, the learner. The details for scoring within an exercise are provided later.

At the conclusion of the first exercise, the instructional system sets the learner's proficiency to his or her normalized score, which is simply the score earned on the scenario divided by the score achieved by, or *projected to be achieved by*, an expert performing the same exercise at the same level of difficulty. If possible, actual expert scores for each difficulty can be obtained and used as the baseline for normalizing student scores. If this is not possible, an interpolated score can be computed from the expert's scores on the two extreme prototype instances.

At the conclusion of each successive exercise, the system recomputes the learner's proficiency by:

1. normalizing the learner's score on the most recent exercise, as described above, and
2. computing the learner's average proficiency, P_A' , as:

$$P_A' = W_r P_1 + (1 - W_r) P_A$$

where:

W_r is the *recency* weight, a value from 0 to 1 expressing the weighting of the most recent exercise proficiency;

P_1 is the proficiency computed for the student's most recent exercise performance;

P_A is (prior) average proficiency

The key parameter here is the recency weight, which is set by the instructional developer to reflect the importance given to recent performance versus prior performance.

Adjusting Scenario Difficulty

The difficulty of the first scenario presented to an individual is the minimum difficulty established for the selected exercise. At the completion of each exercise, the system may adjust the difficulty level for that exercise in response to the average proficiency demonstrated by the learner. The Instructional Strategy offers three possible difficulty level changes that *may be* executed at the conclusion of each exercise, depending upon the learner's average proficiency. These three changes are displayed to the instructional developer as follows:

Decrease the difficulty level by 2, if average proficiency $< p_1$
Decrease the difficulty level by 1, if average proficiency $< p_2$
Increase the difficulty level by 1, if average proficiency $> p_3$

The developer supplies the values of p_1 , p_2 , and p_3 . Upon completion of this step, the difficulty level change section might appear as follows (recall that average proficiency, a value relative to the expert's performance, normally ranges between 0 and 1, although scores exceeding 1 are theoretically possible):

Decrease the difficulty level by 2, if average proficiency < 0.4
Decrease the difficulty level by 1, if average proficiency < 0.5
Increase the difficulty level by 1, if average proficiency > 0.6

Using these parameters, if a learner's average proficiency

- is below 0.4, the difficulty on the next exercise will be reduced by 2;
- ranges from 0.4 up to 0.5, difficulty on the next exercise will be reduced by 1;
- ranges from 0.5 to 0.6, the difficulty on the next exercise will be unchanged; and
- is greater than 0.6, difficulty on the next exercise will be increased by 1.

With this kind of process exercise difficulty can be quickly reduced if the learner is doing exceedingly poorly, but is not increased quite as rapidly for excellent performance, to ensure that difficulty does not become inappropriate due to an isolated excellent result. Of course the recency weight described above comes into play here, as well, since it determines how quickly average proficiency reflects performance on the latest exercise.

Total Time Constraints

The last instructional variable established in the Instructional Strategy is the total time allowed for all instruction, over all sessions (there is no time limit on a single session other than the total instruction time limit). The total time consumed by a learner is maintained over sessions, and, at the conclusion of each exercise, is compared to the total time allowed. If the time allowed for all instruction has been consumed, the status of that exercise is marked as usual (as either "failed", "in progress", or "mastered") and the learner is advised that time has elapsed. As with time limits on individual exercises, the

Instructional Functions

instructional applicator may enter a very large number for this parameter to provide unlimited time.

We turn now to an elaboration of two processes, first the production of tailored scenarios, and finally the within-exercise scoring process.

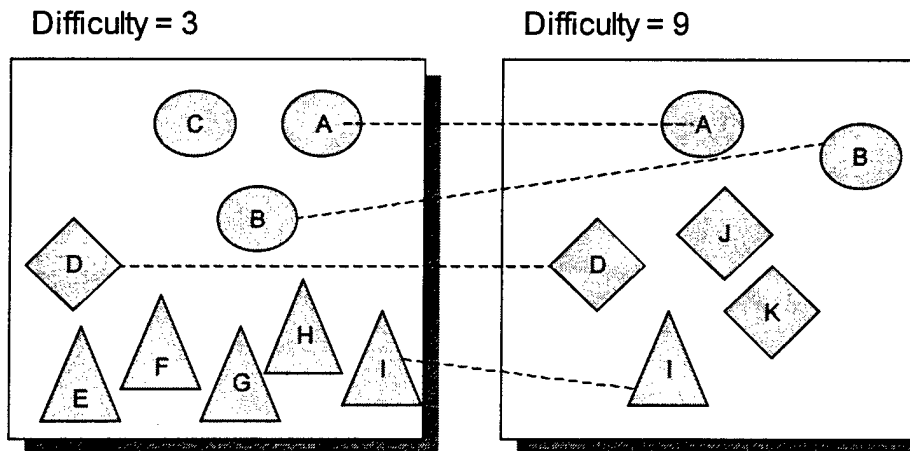
Scenario Tailoring

Upon selecting an exercise for presentation, the system tailors the associated prototype scenario to provide the level of difficulty appropriate to the learner at that time. This section will describe that tailoring process in detail.

A scenario consists of a set of *objects*, each carrying one or more properties. In an air traffic control setting, the objects would include some number of aircraft, some ground personnel and equipment, and possibly some weather-related elements. In an emergency medical application the objects might include the patient, the vehicles available for transport, and other types of equipment and devices. Each of those objects would carry some properties that determine how they behave in the simulation environment. A highly hostile aircraft, for example, might act in a way that disregards its own safety in order to inflict some damage upon its adversary, while a moderately hostile craft might only attack when there is a higher probability of surviving the attack.

Recall that a prototype scenario is represented with two instances, one of which is extremely easy and one of which is extremely difficult. Any such pair will exhibit some commonality, which are the aspects that characterize that prototype, and some differences. The tailored scenario will maintain exactly the same objects and properties that are *common* between the two extreme instances, and it will produce a computed compromise between all differences, in order to achieve the desired level of difficulty.

To continue this discussion, let us consider the following simple example, involving the easy instance on the left and the difficult one on the right, rated at difficulty levels 3 and 9 respectively. (Here, the spatial positions of the objects have no meaning, as these diagrams are only meant to depict the constituency of each instance).



Suppose we wish to produce a tailored scenario from this prototype at difficulty level 7. This is done in three steps: 1) property interpolation of matching objects, 2) apportioning unmatched objects from the difficult instance, and 3) apportioning unmatched objects from the easy instance.

1. Property Interpolation

Two of the ellipses in the easy scenario also exist in the difficult one, as indicated by the lines linking them (their names are identical). Likewise, one diamond occurs in both, and one triangle. For these four *matching* objects, the system interpolates the values of all their properties

The formula for adjusting a property, a , in terms of

- 1) its values in the easier and more difficult scenario instances,
- 2) the difficulty levels of the easier and more difficult scenario instances, and
- 3) the desired difficulty of the adapted scenario.

is

$$a' = a_e + (a_d - a_e) (D - i_e) / (i_d - i_e)$$

where

- a' is the adjusted value of the attribute
- a_e is the value of the attribute in the easier instance
- a_d is the value of the attribute in the more difficult instance
- D is the difficulty level of the scenario being produced
- i_d is the difficulty level of the more difficult scenario instance
- i_e is the difficulty level of the easier scenario instance

For ellipse A, suppose it carries some property X with the value of 50 in the easy instance, and a value of 20 in the difficult one. Its value in the adapted instance, a' , would be computed as follows:

$$a' = 50 + (20 - 50) (7 - 3) / (9 - 3) = 50 + (-30)(4)/6 = 50 - 20 = 30$$

This process is executed for all properties of all the matching objects. Note that:

1. this operation preserves commonality, i.e., any properties that are identical in the two instances will remain as they are, thereby reflecting the essence of the prototype scenario, and
2. it is immaterial whether property values increase as difficulty increases or whether they decrease; the formula above adjusts values properly.

2. Apportion unmatched objects from the difficult instance

If some objects occur in the difficult scenario but not in the easy one, some portion of them may need to be placed in the scenario being produced. The number of such objects is computed similarly to property values, using the number of unmatched objects as the

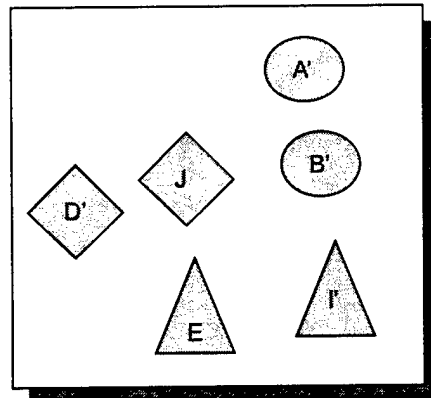
difference to be interpolated. Here, the number of unmatched diamonds in the difficult instance is 2, thus we should bring in $2 \times (7 - 3) / (9 - 3)$, or 1.33 of them to the scenario, thus we bring in one of them.

3. Apportion unmatched objects from the easy instance

The apportionment of unmatched objects from the easy instance is very similar to that of step 2, except that the ratio is reversed. Of the four unmatched triangles in the easy instance, we should bring in $4 \times (9 - 7) / (9 - 3)$, or 1.33 of them, thus we bring in just one triangle. Note that triangles apparently make the scenario easier, as there are many more in the easy instance than in the difficult one. But, since we are producing a relatively difficult instance, we only get a total of two triangles. Also, there is one unmatched ellipse in the easy instance, but since we only get 1/3 of it, it does not make it into the scenario at this difficulty level.

The following figure shows the final constituency of the level 7 scenario. There are six objects, four of which (A, B, D, and I) have had their properties interpolated to produce the desired level of difficulty, one of them (E) was copied from the easy instance, and one (J) was copied from the difficult instance.

Difficulty = 7



Thus, if ellipses represented hostile aircraft, we would have two of them in this scenario, and all of their properties would have been established to a level 7 difficulty. These properties could represent speeds, altitudes, positions, intentions, radar capabilities, or any other factors we recognize in the simulation.

Dealing with Non-Linear and Non-Continuous Variables

In the example above, the properties of the objects were considered to vary linearly with difficulty, thus we employed a linear interpolation formula. Sometimes, however, object properties will vary nonlinearly with difficulty, or they may even vary in a non-continuous manner. In these cases, the values are set within a built-in routine named `application_startExercise` that is provided for the developer's use. This routine executes

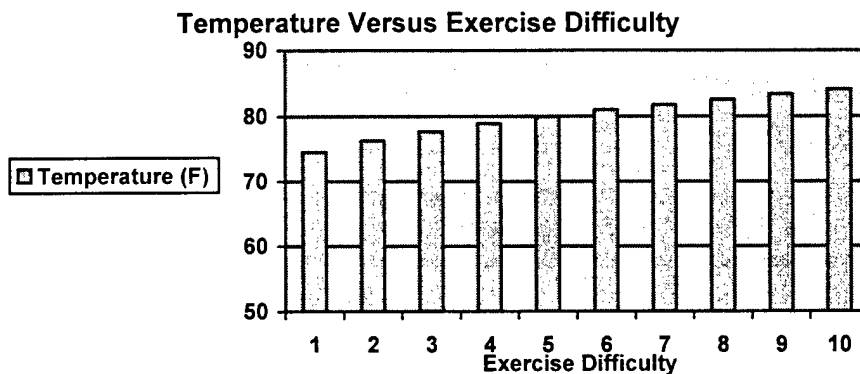
Instructional Functions

automatically at the start of each exercise, and importantly, after the exercise difficulty has been established by the instructional system.

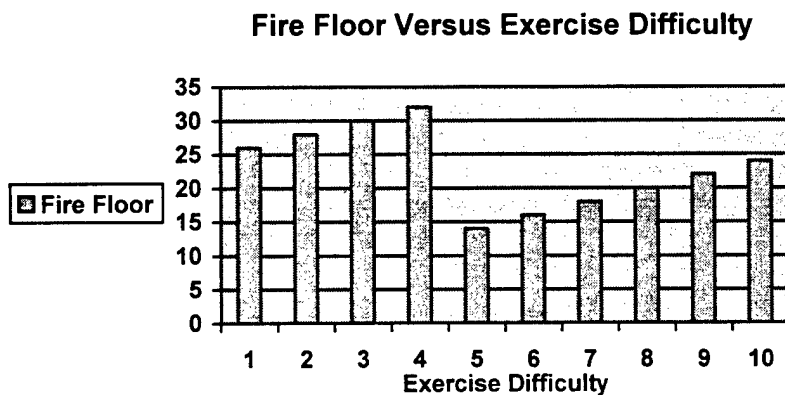
Suppose that ambient temperature impacts the difficulty of combating a forest fire, and that the relationship is non-linear. If this were true, the developer might insert a statement like the following into the `application_startExercise` routine:

$$\text{AmbientTemp} = 70 + (20 * \text{Difficulty}) ^ 0.5$$

As shown in the following chart, temperature increases diminish with exercise difficulty.



An example of a non-continuous property is the number of the floor upon which a high rise building fire exists. In general, more difficult exercises involve a higher 'fire floor', primarily due to the effort involved in climbing stairs, however safety policy allows fire fighters to use elevators if the elevator terminates at least five floors below the fire floor. Thus, getting to a fire on floor 28 is actually easier than on floor 15, *if there is an elevator that terminates at floor 20*. The following chart illustrates how difficulty and fire floor (even numbers) are related in such a building.

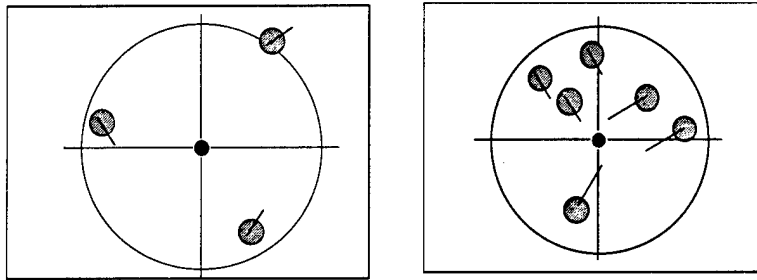


Instructional Functions

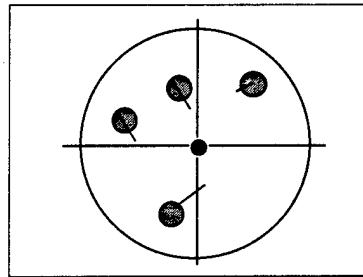
The applicator would enter a statement to application_startExercise that selects the fire floor from an array of values such as {26, 28, 30, 32, 14, 16, 18, 20, 22, 24}, using exercise difficulty as the selection index. If the array were named floorList, the statement would be **fireFloor = floorList (difficulty)**.

Examples

More concrete examples (Towne, 1999) may help to illustrate the scenario adaptation process. The following figure displays two versions of a simplified radar screen in which the small circular figures represent aircraft traveling in the headings indicated by their velocity leaders, at speeds indicated by the length of those lines.



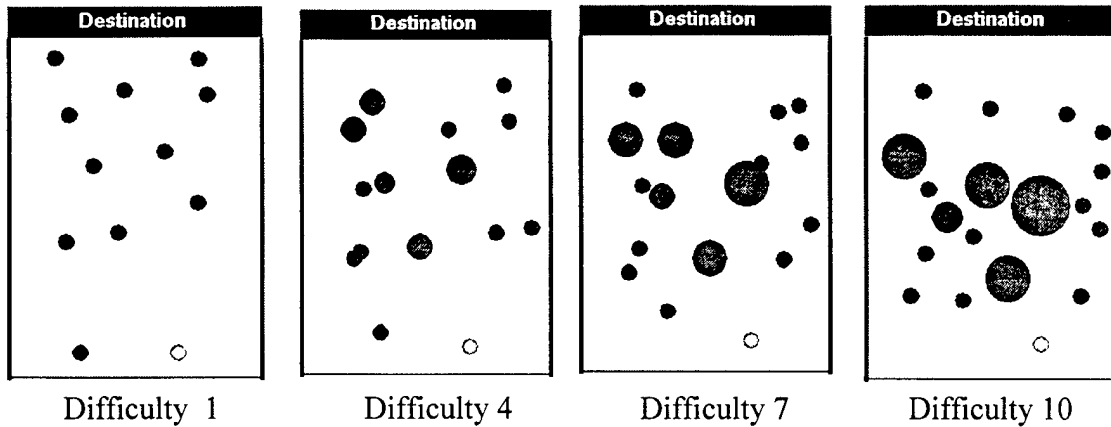
The situation on the left is extremely easy because there are few aircraft to manage, they are travelling slowly, and they are not heading toward the center of the display, representing own ship. The situation on the right is difficult owing to more aircraft travelling faster and heading toward own ship. In addition, but not seen graphically, the aircraft carry altitude properties that also are easy on the left (high altitude) and difficult on the right (low altitude). If the difficulty of the easy scenario on the left is set to 1 and that of the difficult one is 10, a scenario of difficulty 4 would be generated as shown here.



In a second example, shown below, we have an overhead view of a mined area of sea. The task of the decision maker is to guide the craft shown at the bottom right through the mine field as quickly as possible, using the slower moving mine sweeper craft at the bottom on the left as desired. This test task was patterned after a task described by Gordon *et al* (1994).

As shown in the figure, as difficulty increases the mines become more numerous, more sizeable (or sensitive), and more proximate to the craft being guided.

Instructional Functions



Exercise Scoring

The basis for evaluating the individual's proficiency on an exercise is a set of authored *scoring rules*, each of which specifies some condition that the learner might produce or allow to occur in the scenario via his or her performance. These rules, or conditions, are evaluated continually, and they may express either desirable or undesirable conditions. When a scoring condition is found to be true, the system marks associated learning objectives as learned or not learned, it updates the measures of the learner's proficiency, either positively or negatively, and it may present an instructional remark, depending upon availability. The status of the learning objectives and the measure of learner proficiency then influence the process for selecting the next exercise.

The instructional developer devises a general scoring scheme for the domain that awards some fixed score value, including zero, at the start of each exercise, and then updates the score as particular conditions occur during scenario performance. Each defined scoring condition specifies:

- a *condition* which, when it becomes true, causes some value to be added to the current score for the exercise;
- a *value*, either positive or negative, that is added to the current score;
- a *reset condition*, that specifies when evaluation of this condition is resumed after having been found to be true;
- an optional list of learning objectives that should be marked as learned or not learned when this condition becomes true; and
- an optional reference to a particular instructional remark to be issued when the condition becomes true.

As an example, one scoring scheme is to initialize the score at 100 at the start of each exercise, and then subtract various values from that score when undesirable conditions become true. Alternatively, one could initialize the score to 0 and award positive points for desirable outcomes, or one could initialize to some intermediate value, like 50, and award both positive and negative points as the scenario proceeds.

Instructional Functions

An example scoring condition for an air traffic control task might be as follows:

Rule Component	Assigned Value
Condition	Any two aircraft have < 1000 feet vertical separation
Value to add to score	-40
Resume evaluation	when condition once again becomes false
Learning objectives	3, 4, 9
Instructional remark	"You have allowed two aircraft to be separated by less than 1000 vertical feet. You must immediately advise ..."

Since the value to add to the score is negative here, learning objects 3, 4, and 9 would be marked as unlearned when this condition becomes true. This within-exercise condition evaluation is one of two ways to maintain the status of learning objectives. The exercise specification can also list learning objectives to be marked as mastered when the learner masters the exercise (however, failure to master a particular exercise does not cause the associated learning objectives to be marked as unlearned as only some of them may have led to the failure).

The values associated with the conditions are established by the task expert to reflect the relative import of the conditions, according to his or her best judgement. Since the resulting raw scores are not normalized, learner scores are then compared to expert scores to obtain calibrated values.

Between-Exercise Functions

At the conclusion of an exercise, the learner may request to view a replay of his or her performance, at which time the instructional system presents the replay in real time, with any provided instructional commentary. This replay is recreated from a data file written as the learner worked (Appendix A). During a replay, the learner may pause the simulation in order to study any external documentation or instructions or to confer with others about the situation. Following the replay the instructional system selects and tailors the next scenario based upon the learner's current level of proficiency, according to the chosen exercise selection scheme. Format of the data file that records the individual's status and progress is presented in Appendix B.

III. Application Development Guide

This section describes the detailed steps involved in producing a new D³M application. It employs some examples from the fire fighting application described fully in section IV, which should also be reviewed by any prospective applicator. We look first at how a task environment is represented to the user, then how it is simulated internally, and finally how instructional content is produced for that environment.

Task Representation

The first step in implementing a D³M application is deciding how the task will be represented, externally via a graphical interface upon which the learner operates, and internally via unseen functions that model the behavior of the simulated world. Typically, this decision is facilitated by first defining exactly what parts of the real world will be presented and what actions the learner may perform on that presentation. Then, the necessary internal processes become evident, and program statements may be entered that cause the learner's actions to affect the world model.

The problem representation provides the means by which the learner both experiences the world and acts upon it. Either of these may also require the decision maker to 'move' to another location in the simulated environment. The content and configuration of the representation is largely determined by the instructional objectives of the application. The learner's knowledge of the simulated world and the opportunities for changing that world will be entirely determined by the elements provided.

Very often the information about the world comes to the decision maker via 1) verbal reports from the field, either by telephone or radio, and 2) computer-driven displays of some type. Likewise, high level decision makers tend to act on the world either by issuing verbal orders or by acting upon some graphical display.

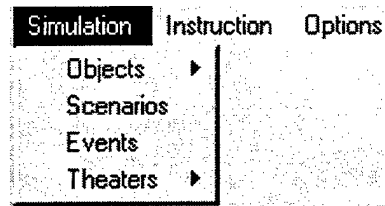
If graphical displays are involved, they would be drawn using the graphical editor within D³M. Those parts of the display that change would be assigned unique names, so that the simulation can maintain them. The graphical displays maintained in the high-rise fire fighting application include: 1) an alarm enunciator panel that displays the various abnormalities that have been sensed within a building, 2) a status board that depicts current crew assignments and the status of units in transit, and 3) elevator controls and displays.

Incoming verbal reports, as generated by the simulation, can be displayed textually or can be spoken via the text to speech capability within D³M (Windows version). Alternatively, spoken reports can be pre-recorded, however this can lead to serious problems in accommodating all the possible situations that might need to be voiced.

Usually, some means for issuing verbal orders will be developed. By far the easiest manner of accomplishing this is to provide some sort of dialog box by which the decision

maker composes his or her statements. In some cases – those involving limited vocabularies and forms of statements – use of speech recognition might be feasible.

The simulation development process employs the following menu, which provides access to object retrieval functions, scenario development resources, specification of scheduled events, and the creation of theaters of operation.

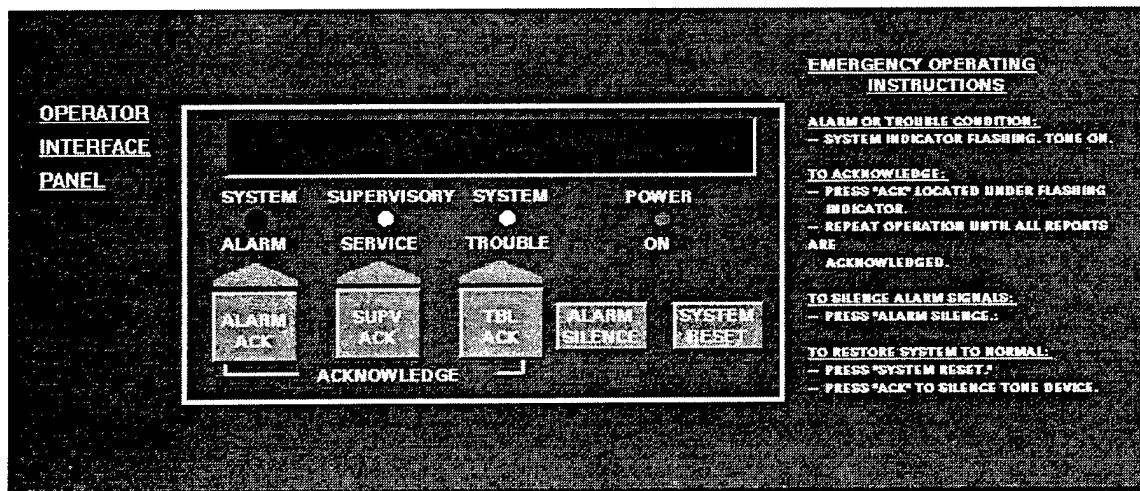


Each of these four functions will now be described.

Simulation Objects

Upon determining what elements will be provided for the learner to observe and to act upon, the developer creates graphical objects using the low-level graphical tools. The observable objects to be simulated are drawn using a built-in graphical editor and then given any required behavior rules that allows them to respond to the user or to changes in other objects. Instructions in the use of the graphical tools and behavior rule editing will not be repeated here, as these application details are documented elsewhere.

Here, for example, is an enunciator panel found in a building's fire control room. It is constructed of four copies of an indicator light object, three instances of the square button object, two non-square buttons, along with some fixed text labels and some variable text messages (in the center box). The content of the message text and the status of the indicator lights all depend upon values of scenario variables.



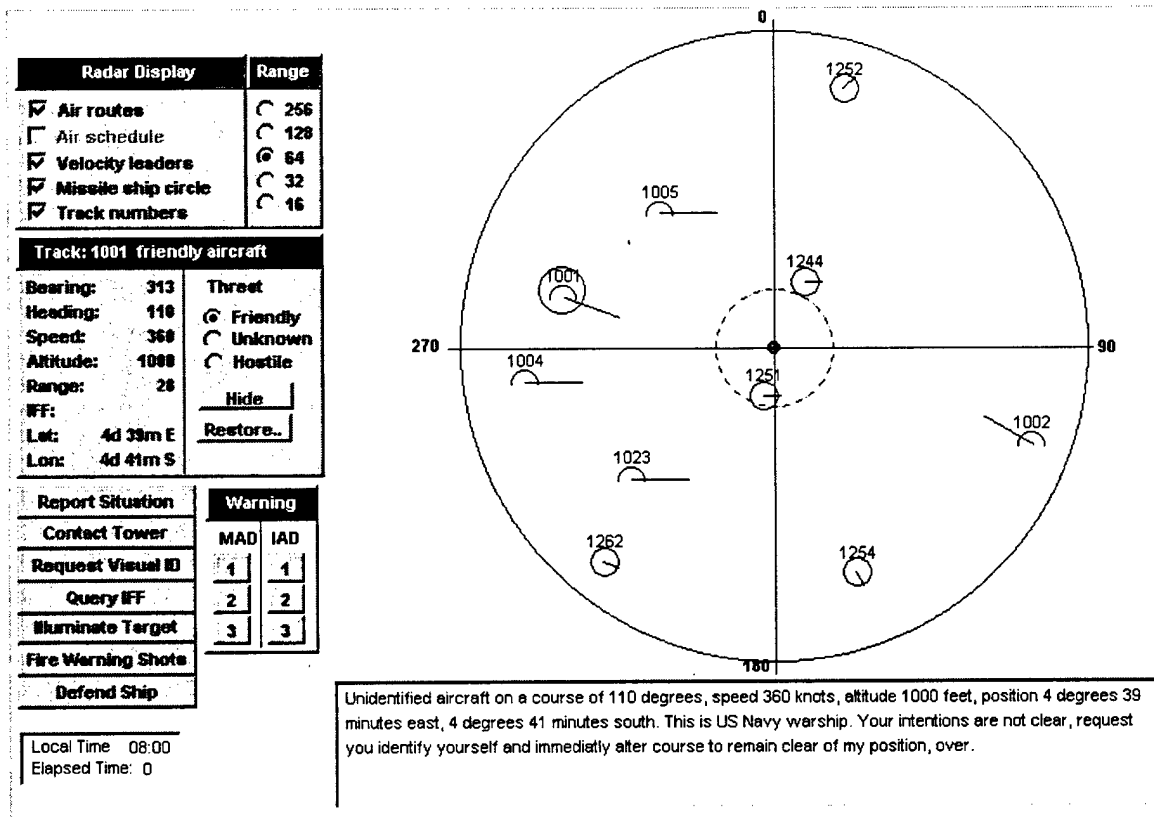
Object Prototypes

The applicator produces a graphical prototype of each scenario-variable object to be included in scenarios, and names each obj_<name>, where *name* is any sequence of alphanumeric characters, not including space or special characters. Example prototype object names are obj_helicopter, obj_fire, and obj_gorilla. The prototype objects form an application-specific library of objects that will be used to construct scenarios.

The applicator adds the text property *nameList* to each prototype object, and enters into this property a space-delimited list of property names to be saved when scenarios are created. For example, for a helicopter, we might wish to save the initial position, heading, speed, and altitude of the helicopter when a scenario is created (saved). It is not necessary to list the properties location and visibility (whether or not the object can be seen), as D³M saves these for all scenario-variable objects.

Example

The following is an example of a single-screen representation of a shipboard radar screen (Towne, 1995).



Here, the figures within the large circular radar display represent ships and aircraft, and the selection boxes on the left side permit the operator to control the display. Each of the ship and aircraft objects carry properties that govern its behavior, including:

- type of craft (private, commercial, or military)
- intentions (hostile, friendly/cooperative, indifferent);
- readiness of radio transmitter and receiver
- IFF mode and code, if any
- radio channels being monitored
- speed, heading, altitude

After producing the graphical representation, the applicator enters program statements that accomplish the simulation in response to actions of a decision maker and external forces that affect the problem environment over time. These statements may be embedded within the various objects, so that each object takes care of its own behavior and graphical rendering, they may be placed within the main timing loop, outlined below, or they may be distributed to both. This choice depends very much upon the specific nature of the application. Typically, some statements within the individual objects take care of graphical rendering of the object, while statements within the main timing loop update the condition of each object.

In the ship radar example shown above, each ship and aircraft symbol includes a straight line, called the *velocity leader*, whose angle signifies the heading of the craft, and whose length is proportional (non-linearly) to speed. The statements to establish this line, based upon speed and heading, were placed within the main timing loop, which updates the appearance of each craft every second. This was found to be the most straightforward way of maintaining the radar display.

Simulation Control

The primary task of the simulation developer is to specify how the world changes over time, and in response to conditions that either came about at the start of the scenario or were created by the decision maker. For ease of application, D³M provides all the necessary timing loops and maintenance of time-related data. Thus, when D³M starts an exercise, it automatically initializes the elapsed time variable *elapsedTime* to zero, and it automatically updates this variable, in seconds, as the exercise progresses. Further, D³M automatically calls the update routine, *application_updateSim* as often as possible, executing the developer's statements placed there to specify how the world changes over time. If the instructional developer elects to show the Pause/Resume button, built in to D³M, the system automatically stops and resumes the simulation in response to use of this function.

In general, the developer would place statements in *application_updateSim* that update all objects directly affected by the passage of time, such as aircraft flying a particular route, a forest fire burning, or a medical patient's loss of blood. Almost always, such updating processes involve not only the time that has passed since some condition began, but also related variables, such as the speed of an aircraft, the tree density in a forest, or the gravity of a patient's wounds. Since the actions of the decision maker may affect the situation, the developer must ensure that such actions set variables that are referred to in the update computations.

Since D³M automatically starts, pauses, resumes, stops, and replays exercises, a way must be provided for the simulation developer to provide domain-specific control statements that will be executed at the proper times. For example the simulation of a toxic spill might require that, at the start of each exercise, all emergency resources be initialized at their normal sites.

Five special routines are therefore included in D³M, which are called as shown in the following table (see Appendix C for further details):

Routine Name	Purpose	Executed
application_vals	expose domain-specific variables to D ³ M	at start of session
application_startExercise	initialize the domain simulation	at start of each exercise
application_events	execute special domain-specific events	when an event is ready to occur
application_updateSim	update the domain simulation over time	as often as possible
application_terminateExercise	perform any special end of exercise operations	at conclusion of each exercise

To simulate the flow of a toxic spill down a river, one could insert statements into the routine *application_updateSim*, to compute the location of the toxins according to the elapsed time since the spill, flow rate of the river, and any other pertinent variables.

The Simulation Engine

Updating of most other objects within a D³M simulation is accomplished automatically, that is, without the necessity to enter additional coding beyond that already embedded within the objects. This happens because an embedded *simulation engine* automatically updates the conditions of all objects that refer to objects that have been explicitly changed. So, when the location of an aircraft is changed within the timing loop, a radar blip of that craft will automatically change, as long as the blip object expresses its appearance in terms of the location of the aircraft.

If it is necessary to temporarily stop the automatic updating of time, so that the error can be corrected, the developer should return to the first screen of the application and enter "author" as the user name. Now, time updating is stopped, and corrections may be made.

Authored Buttons

The application may include buttons that contain `buttonDown` handlers which respond to the user's mouse. These handlers must include the statement

send writeDecision self

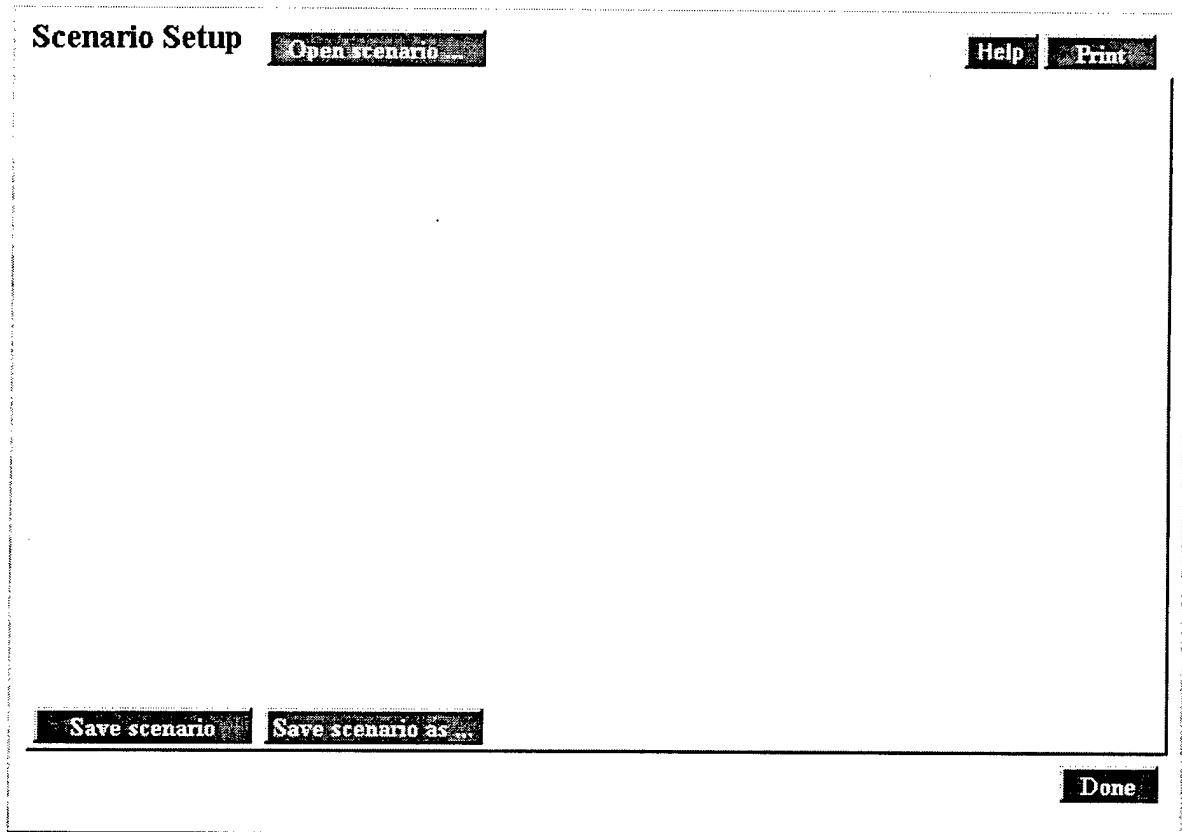
so that D³M will record the use of these buttons for replay.

Producing Scenarios

As described in Section II, scenarios are specified entirely in terms of objects with properties, and D³M generates modified scenarios by comparing the constituency and properties of those objects in two extreme, prototype, cases. If desired, the applicator can represent all relevant task variables using explicitly defined objects, however this often involves defining a number of invisible objects to represent such factors as weather or traffic density. As a convenience, D³M invisibly includes in each scenario specification one invisible object that can contain all such variables. The applicator may establish the properties carried by this special object, and all its property values, by constructing a domain-specific dialog box, as described next.

The Scenario Setup Dialog Box (optional)

The applicator determines what variables affect the difficulty of the task but are not represented within existing object properties, and then produces a domain-specific dialog box to specify those variables. D³M provides the template for the dialog box, including the buttons that save and retrieve scenarios, as shown here.



The buttons provided perform the following functions:

Button Name	Function
Open scenario ...	Select an existing scenario to read into the dialog box for editing
Save scenario	Save the current dialog box settings under the current file name
Save as scenario ...	Save the current dialog box settings under a new file name
Help	Assist the applicator in using the dialog box
Print	Print out the current scenario as specified in the dialog box
Done	Leave this screen and return to the main development screen

Into this dialog box the applicator inserts labels, selection buttons, and fields for entering numerical values. The following is an example of a completed dialog box, for specifying scenarios in high rise fire fighting (Appendix D provides the Help for this and all other dialog boxes used).

Scenario Setup Scenario: ESIC_d

Fire starts at am pm in room (Enter 0 if false alarm.)

Number of alarms at annunciator panel: 1 2 3 4
 Describe the alarms below, in chronological order.

manual smoke heat water reported minutes after fire starts, on fire floor +

manual smoke heat water reported minutes after fire starts, on fire floor +

manual smoke heat water reported minutes after fire starts, on fire floor +

manual smoke heat water reported minutes after fire starts, on fire floor +

Transit time E-1: Minutes for E-1 (user) to travel from station to scene of reported fire.

Transit time E-5: Minutes for E-5 to travel from station to scene of reported fire.

Appearance on arrival: no fire or smoke showing fire or smoke is showing

Bldg. engineer arrival: Minutes after company arrives that engineer arrives (enter 0 if engineer is present on arrival).

Building floors: Total floors* in building.

Building height: Feet, from roof to ground.

Building occupancy: Number of people in building at time of alarm.

Fire floor occupancy: Number of people on true fire floor at time of alarm.

The elements that may be inserted by the applicator are these:

- Radio button sets for selecting one option from multiple alternatives;
- Check boxes for setting an attribute to true/false or yes/no;
- Numeric entry boxes, for setting values; and
- Text that identifies the data entry elements.

All of these data entry objects are provided within D³M, thus one simply copies, pastes, relabels, and renames as many of each as is required. The renaming is critical, as it establishes the names of the variables saved, as follows:

Naming Radio Button sets. The name of the set, the group of buttons, will be the name of the attribute saved. Name each of the individual buttons D1, D2, etc. For example, the radio button set next to the label *Appearance on Arrival* is named *fireSmokeShowing*, and this attribute will be recorded as either 1 or 2, depending upon the selection of button D1 (captioned *no fire or smoke showing*) or D2 (captioned *fire or smoke is showing*).

Naming Check Boxes. The names of check boxes are set to the names of the variables they represent. There happen to be no check boxes in the example shown above. If a check box named *toxicProblem* were provided, the attribute named *toxicProblem* would be recorded as true if the box were checked, else it would be recorded as false.

Naming Numeric Entry Boxes. Like check boxes, the names of numeric entry boxes determine the name of the variable that will be recorded for the scenario. The value entered will be the value saved for that attribute name. For example, the entry element above containing the value 18, for building floors, is named *bldgFloors*, thus when this prototype scenario is saved, the attribute *bldgFloors* is recorded as 18.

Following this one-time development step, the author creates as many prototype scenarios as desired, by entering values and making selections in the domain-specific dialog box, as outlined next.

The Process

The developer establishes objects, including that object described in the scenario setup dialog box, if any, to specify an extremely easy instance and an extremely difficult instance of each situation to be instructed. The variables that are unchanged between these two extremes characterize the situation.

As an example, a scenario in high rise fire fighting might involve a fire in a very old industrial building with no sprinklers and a failed alarm system. An easy instance of this scenario might have that fire occurring during the early evening, when the building is virtually empty and traffic is light, and it might specify little combustible material on the fire floor, to name just a few of the possible variables that could be manipulated. The difficult version of this scenario might occur at midday, with many occupants, a higher fire floor, considerable combustibles, and even heavy automobile traffic from an event at a nearby arena. Both instances would carry the attributes of having no sprinkler system, no alarm being sounded or transmitted to the fire dispatch center, and other characteristics consistent with an older building.

To build a new scenario, the applicator does this:

1. select **New scenario**, or open an existing scenario that is similar to the one to be built;
2. add objects to the scenario, by selecting the menu items **Object** → **Add** then clicking on the object prototypes desired. Each click on a prototype adds one copy of that object type to the scenario.
3. If necessary, remove any objects not wanted by Control-clicking each (selecting the objects with the mouse while the Control key (Ctrl) is down).

4. drag the objects into position, and set any other properties desired for that scenario
5. optionally, enter values to the Scenario setup dialog box
6. save the scenario under a new name by selecting menu items **Scenario** → **Save as**, then entering the name you wish to assign to the scenario.

Easy instances of prototype scenarios are saved under any name followed by ‘_e’, and difficult instances carry the same name, but ending in ‘_d’.

Thus, the two names for a prototype scenario involving a false alarm might be “falseAlarm_e” and falseAlarm_d”.

Scheduled Events

A scheduled event is some action the application developer wishes to inject into an exercise that is *not* produced as a result of the task simulation (the objects and the scenario specifications). A change in weather or an equipment breakdown, for example, could be produced via a scheduled event, unless those occurrences were already being modeled in the simulation. The developer defines a set of scheduled events by selecting *Events* under the *Simulation* menu, then completing the following dialog box.

The author may start specifying anew, or may Open an existing event data file for modification. If an existing file is opened, the first event in that file is shown, as in the following figure.

Scheduled Events

Current Event File: NONE **Create new event** **Help** **Print**

At seconds into exercise ◀ ▶ Event 0 of 0

change to

instantly

over seconds

do function
with parameters e.g., 10, 0, -125 (no spaces)

display text:

when exercise difficulty is

Delete this event

Open **Save** **Save as** **Revert** **Done**

Three different types of events may be specified here:

1. *change* a simulation attribute, either instantaneously or over a period of time. or
2. *do* (execute) a developer's function; or
3. display some *text message* to the user

Change to Simulation Attribute. The developer enters the time at which the event should commence, selects the Change button, and enters the attribute to change and the value the attribute should acquire. The developer then either selects the "instantly" button, to make the change happen instantly at the specified time, or selects the "over" button and enters the time period over which the change should occur. The event shown above calls for changing a simulation variable windSpeed to 65 over a 300 second interval, starting ten minutes into the scenario.

Do Developer's Function. The author enters the time at which the event should commence, selects the "Do function" button, and enters the function name and any required parameters.

Display Text Message. The author enters the time at which the message should appear, and the content of the message. At the specified time, the message will be displayed. This option provides all kinds of ways to introduce complexities into a running exercise.

Controlling the Difficulty of Events

For any of the three event types, the author may specify 1) the difficulty levels at which exercises will include the event, and 2) ranges of values that will be computed depending upon the difficulty of the exercise.

Specifying Difficulty Levels. In the example above, at the bottom of the dialog box, the author has selected the ">" and entered 4. This tells the system to only execute this event if the current difficulty level exceeds 4. The drop down list provides all the necessary equality and inequality symbols.

Specifying Difficulty-determined Intervals. Any of the numerical values in an event specification may be entered as intervals rather than single values. For example an author could enter "60-120" for the time at which the event above starts, or "40-20" for the value to which "occupancy" should be set.

When D³M executes an event it computes the values to use for any specified intervals by interpolating on the interval according to the current difficulty level of the scenario. By convention, the first number corresponds to the easiest case and the second to the most difficult, thus one can specify a condition in which higher values correspond to easier scenarios by reversing the order.

Because of the flexibility of the scheduled event language, some applicators may find that this feature alone is sufficient to support most of the time-related simulation required in an application.

Theaters of Operation

A theater of operation is simply an optional fixed graphical background upon which scenarios are simulated. In general, theaters of operation are employed to vary those graphical factors of a task that cannot be manipulated automatically from values of scenario specifications.

Since a theater of operation is specified for each exercise, one can partially control the nature of the problem via this selection. For example, in an air traffic control task one could represent the landing strip configurations of different airports via different theaters of operation, then vary the scenario difficulty within each airport by manipulating visibility, traffic density, and other factors. In the fire fighting application, different building types can be represented via theaters of operation.

The theater of operation does not change during an exercise, and typically multiple exercises will be presented in any single theater. The applicator creates as many theaters of operation as desired, and specifies which one, if any, is to be used in each exercise.

Creating and Modifying Theaters of Operation (TOPs)

To create a new TOP:

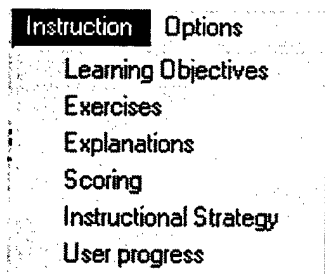
1. if necessary, clear any existing theater, by selecting the menu items **Theater** → **Clear**
2. draw the background graphics using the graphical editor, group the graphics into a single object, and name the grouped object "theater" plus a unique number, e.g., theater1, theater2, etc.

To modify an existing TOP:

1. open the TOP by selecting the menu items **Theater** → **Open**, then selecting the desired TOP
2. use the graphics editor to modify the TOP
3. Save the application, using the **File** → **Save** menu items.

Instructional Specifications

The second major phase of development involves use of dialog boxes to specify how the simulation is to be presented in instruction. This section will describe the use of each dialog box, in approximate chronological order. The instructional authoring functions are selected via the Instruction menu shown here.



Learning Objectives

The instructional system maintains records of those learning objectives attained by each individual. It may, if so commanded, select exercises based upon mastery of learning objectives, and it will update its records as the individual performs.

The developer goes to the Learning objectives editing screen by selecting Learning Objectives in the Instruction menu. D³M then displays the following screen which allows the author to create, modify, and delete objectives.

The developer enters a one line summary and a detailed description of each objective. The one line name is used when authors link objectives to exercises and proficiency conditions. The detailed descriptions document the learning objectives, and they are also shown to the learner if the instructor has so enabled such display. For some tasks, this display would reveal unknowns that are an essential part of the exercise, and should therefore not be shown to the learner during practice.

When a learner masters an exercise, the learning objectives associated with that exercise are marked as achieved. Failing an exercise does *not* cause the related learning objectives to be marked as failed, since we cannot automatically determine which of the critical learning issues led to the exercise failure. Instead, D³M looks to the authored scoring rules, described later, which specify particular domain-specific situations that do relate directly to specific objectives.

Exercise Specification

The curriculum is composed of a set of *exercises*, each of which is specified in terms of a prototype scenario, scheduled events, a theater of operation, and other variables, including the learning objectives that the exercise addresses. The developer selects the *Exercises* item under the *Instruction* menu, and D³M displays the following screen for adding, deleting, and modify exercises.

Exercises

Scenario: Theater: Exercise 2 of 4

Events: Time limit: seconds

Starting difficulty: Target difficulty:

Minimum trials: Maximum trials:

Critical learning objectives:

Highlighted objectives are considered critical to success on this exercise. Click an objective to change its highlighting.

FAO Correct choice of broadcast channel
 FAO assumes command
 FAO Call for additional alarm
 FAO Report size up upon arrival
 FAO assumes command
 FAO Correctly report life hazard in size up
 FAO Correctly select offensive or defensive strategy
 FAO Broadcast periodic status reports during ascent
 FAO Determine and report fire floor
 FAO Check door before opening
 FAO Close doors when leaving floors
 FAO Determine and report conditions above fire floor
 FAO Determine staging floor during ascent

The other parameters include:

- the maximum time allowed to complete the exercise;
- initial and target levels of difficulty to be presented;
- minimum and maximum number of trials allowed; and
- selected learning objectives, to be marked as mastered when the exercise is mastered.

Maximum time and trials can be set very large if it is not desired to impose limits on these aspects of the instruction. The maximum difficulty is the *target* difficulty for the exercise, the difficulty level at which one has considered to have mastered the exercise, given a satisfactory score. In the dialog box shown above, the first four learning objectives have been selected as being closely related to the exercise.

Upon selecting an exercise to present to a learner, D³M tailors the referenced prototype scenario so that it will present the desired level of difficulty. The process for doing this is detailed in a later section.

Scoring

The author sees the following screen upon selecting Scoring under the Instruction menu.

Scoring
Create new scoring rule
Help **Print**

Rule name:
◀ ▶ Rule 1 of 4

Condition:

Score Change: When the Condition becomes true, change user's score by this constant or function value:

Associated Learning Objectives: e.g., 12,4,9

Learning objectives:

- 1 Establish and communicate the staging floor
- 2 Call for additional alarm
- 3 Establish command center
- 4 Establish and communicate the staging floor
- 5 Assess and exploit the available information at
- 6 Identify and report the fire floor
- 7 Employ elevators safely and effectively
- 8 Ascend to fire floor

Explanation Number: Enter an explanation number, or 0.

Resume evaluating the Condition, during this exercise:

never

after the condition once again becomes false

after the following seconds have passed:

Delete this rule

temp
Done

Each rule contains four parts:

1. a *condition* that describes some desirable or undesirable task-specific situation;
2. a *value* that is added to the learner's current score on the exercise;
3. a set of *learning objectives* which are directly related to the task condition; and
4. an optional *explanation* that expresses the expert's assessment of the situation.

Condition

Here the author has entered a condition in terms of two domain-specific variables: 1) *timeToCall*, the time between arriving on a fire scene to the time an additional alarm is called in, and 2) *smokeShowing*, which is true if there is smoke showing upon arrival. According to this scoring rule, if the decision maker does not call in an additional alarm within 20 seconds of arriving on a scene with smoke showing, 10 points are deducted

from his or her score. Additionally, learning objectives 6 and 7 are set to unlearned, and this scoring rule will not be reevaluated again during the current exercise.

Value

During an exercise, D³M continually evaluates each defined proficiency condition. When a proficiency condition becomes true, D³M changes the learner's score on the current exercise by the amount specified. This may be a constant value (positive, negative, or zero) or it may be a function that reflects the degree to which the learner succeeded or failed to accomplish some goal or perform some subtask. As a simple example of a functionally defined score, the score value for a dispatching task might be $-timeToCall$, where *timeToCall* is the time from receiving a 911 call to the time a dispatcher issues an order to respond.

Learning Objectives

If the score change is positive, and the task is being performed at the criterion level of difficulty – the maximum difficulty specified for the exercise – then D³M also marks the associated learning objectives as being achieved. If the change in score is negative, D³M marks the associated learning objectives as not being achieved, regardless of exercise difficulty.

Explanation

In addition to updating the learner's proficiency score, D³M may present an *explanation* that has been prepared by the expert, as described in the next section. Note that one may present a comment without changing score by specifying scoring change to zero.

The Print command offers the choice of printing the current scoring rule or all scoring rules.

Explanations

The author goes to the Explanation editing screen by selecting Explanations under the Instruction menu. D³M then displays the following screen, on which the author may add, delete, and modify explanations to be presented when the learner replays expert performance.

Explanations

Item 3 of 5

Reference:

Explanation. Enter the rationale for this element of expert performance:

Because the stairways are locked from the outside, for security purposes, the warden at the staging floor must be informed that occupants of the fire floor are coming down the stairway to the staging floor. This permits the warden to open the locked door from the inside. The warden must also be told which stairway will be used.

These explanations may be presented by D³M when it detects a defined condition of instructional importance; and when it is replaying an expert's performance.

Scoring. As noted above, the domain expert may provide explanations to accompany scoring actions, so the learner understands why his or her proficiency has been noted. The remark might note a performance error, such as:

It has been more than 5 minutes since your last status report to dispatch. You should make such reports at least every 3 minutes.

or it might be positive:

Good. You called in the back up crew in time to prevent an explosion.

Non-scoring. The domain expert may specify any number of conditions that merit commentary, but do not reflect on the individual's proficiency. These are specified just as are Scoring conditions, but the scoring change and learning objectives are omitted. Typically, this type of explanation directs the learner's attention or performance, such as:

Because toxic agents have been detected, you must now carry out the County regulations concerning public disclosure.

Since the condition that triggers such explanation delivery can include the built-in attribute *difficulty*, the domain expert can author explanations that only appear during early trials of an exercise.

It is clear from analyzing just one complex decision making task – high rise fire fighting – that *absence* of action and *absence* of various conditions can be as critical to understanding task proficiency as are overt actions performed and actual conditions. Thus this explanation mechanism permits the expert to point out his or her thinking, rationale, and justification for acting or not acting at any time.

Replaying. As described in the next section, D³M offers a *presentation* mode, in which a learner can observe an expert's performance on any of the exercises which an expert has worked. As the expert performs an exercise, he or she may freely enter annotation that explains anything that should be observed or understood by the learner. These comments may include 1) rationale for why the expert performed a particular action, 2) rationale for why the expert did *not* perform some (or any) action at a particular time, 3) a condition or situation that the expert feels is important. For example:

Now, my first crew has been working under oxygen masks for over 15 minutes, and I need to get them some relief. I'll bring up the people from staging as soon as I hear that we have the line connected in the second stairway.

Note that commentary given during the learner's work is triggered by the existence of a condition which the expert specified, and is not keyed to any particular exercise or scenario. Thus, the domain expert need only specify a particular task situation one time, and D³M will watch for that situation over all exercises. Commentary given during replays, however, is simply embedded within the expert's performance of each exercise. To minimize repetitive effort, however, the domain expert may define any number of explanations and refer to them whenever he or she wishes to have that explanation rendered.

Notes Concerning Learning Objectives

It may now be noted that the scoring and learner status updating process used in D³M has the property that a learning objective recorded as being attained at one point in instruction can be marked as unattained at a later time, due to the learner's poor performance on a specific aspect of a later task. This is not inconsistent with what we know about human learning. People can demonstrate attainment of an objective when a subtask is embedded within a simple performance environment, yet fail to sustain that mastery when the performance environment becomes more complex. Thus, it is appropriate to change a learning objective back to unlearned, if performance so indicates.

Note also that there are two independent mechanisms available within D³M for specifying when a learning objective has been mastered: 1) when an *exercise* to which the objective has been linked is performed successfully, and 2) when a particular *proficiency*

condition to which the objective has been linked is achieved by the learner. The D³M applicator may choose to employ either or both of these mechanisms.

In some cases, it may be difficult to define specific conditions that indicate mastery of particular learning objectives, yet feasible to specify the learning objectives that are met when the learner succeeds on a particular exercise. Conversely, some tasks may offer straightforward opportunities for defining proficiency conditions, in which case the applicator may elect to exploit this means of expression alone, or may use a combination of the two.

Instructional Strategy

The author goes to the instructional strategy editing screen by selecting Instructional Strategy under the Instruction menu. D³M displays the following screen, permitting the author to establish the instructional plan.

Instructional Strategy		Print
Exercise presentation	Indicate here how exercises shall be selected from the exercise list ('completed' means either mastered or terminated due to maximum trials.)	
	<input type="radio"/> present each exercise in the curriculum list once, then repeat until all are completed. <input type="radio"/> repeatedly present one exercise until it is completed, then do next in curriculum list. <input type="radio"/> randomly select an exercise from those not yet completed. <input checked="" type="radio"/> present the exercise which involves the fewest unattained learning objectives.	
Recency Weight	Average proficiency is computed for the user following each exercise. How heavily should the just-completed exercise be weighted in computing average proficiency? <input type="text" value=".65"/>	
	(The proficiency achieved on all previous exercises will be weighted: .35)	
Exercise Difficulty Adjustment	The difficulty of each exercise is determined by the user's average proficiency.	
	Decrease difficulty of next exercise by 2 if average proficiency < <input type="text" value=".4"/>	
	Decrease difficulty of next exercise by 1 if average proficiency < <input type="text" value=".5"/>	
	Increase difficulty of next exercise by 1 if average proficiency > <input type="text" value=".6"/>	
Total Time Limit	Enter the maximum time allowed to complete instruction: <input type="text" value="300"/> minutes	
		Done

Here, the author has elected to:

- present as the next exercise that one which involves the fewest unlearned objectives;
- use a weight of 0.65 for the just-completed exercise, meaning that the average proficiency on all previous work is weighted at 0.35 when updating average proficiency;

- reduce exercise difficulty by 2 if average proficiency falls below 0.4, reduce by 1 if it falls below 0.5, and increase by 1 if it exceeds 0.6; and
- allow each user 300 minutes to complete all exercises.

User Progress

Selecting User Progress under the Instruction menu produces the following screen. Here, one may select any particular learner, and view and print that individual's status and progress.

User Progress

Select a user .

User: FRED Total time expended: 15.7 minutes **Print**

Ex	Trials Worked	Current Difficulty	Current Proficiency	Average Proficiency	Exercise Status
1	3	6	0.85	0.75	Failed
2	2	4	0.60	0.73	In progress
3	0	0	0.00	0.00	Not attempted

Learning Objectives	Mastered
Establish and communicate the staging floor	yes
Call for additional alarm	yes
Establish command center	yes
Establish and communicate the staging floor	no
Assess and exploit the available information at the fire control room	yes
Identify and report the fire floor	no
Employ elevators safely and effectively	no
Ascend to fire floor	no

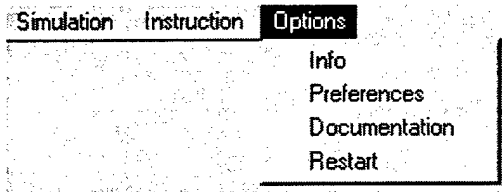
Done

The formatted data shown here are extracted from the student's progress file, as described in Appendix B.

The Print button offers the choice of printing the currently displayed student or a summary of all students.

Options

The applicator may select from four available options.

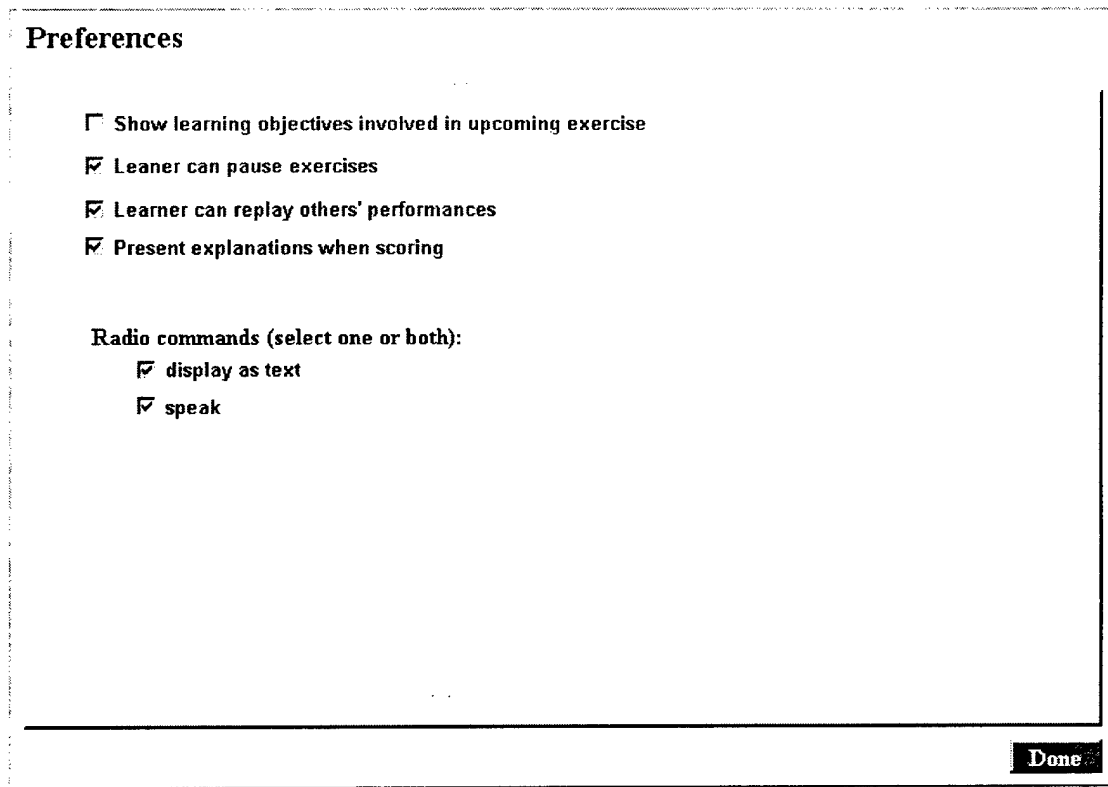


Info

The menu item produces a brief summary of the objects in the current scenario.

Preferences

This menu selection displays the preferences dialog box for review or change.



Documentation

This menu item displays a screen the applicator may use to document the application.

Documentation

Print

Task Description. Document the task to be instructed:

This application can support training of two positions:
1. the first arriving (also first ascending) officer (FAO), and
2. the incident commander (IC)
The task of the FAO is to arrive on scene, make an initial size up and report, gather information from all available sources, ascend toward the possible fire floor, identify and report the fire floor, staging floor, and conditions above the fire floor.
The task of the IC is to assess the situation as it exists upon arrival, including existing assignments, and to manage the resources and strategy from then on. This may include calling additional fire department resources, as well as calling in resources from agencies outside the fire department.

Author Notes. Enter anything that will be useful in documenting this application:

5. smoke showing upon arrival; no problems
6. smoke showing upon arrival; dangerous toxins encountered
7. no flames/smoke showing upon arrival; smoke alarm id's higher floor
8. nighttime fire; building engineer unavailable; janitor has heart attack
9. daytime fire; high occupancy; building occupants jam stairs and elevators
10. daytime fire; high occupancy; well-trained and directed building occupants
11. fire in top floor restaurant; danger of explosion
12. security encounters arson in progress
13. terrorists set fires
14. distraught employee takes fellow employee hostage; manual alarm is set off

Done

Restart

This menu item displays the initial sign in screen, allowing a change of user.

D³M Dynamic
Discrete
Decision Making

Key in user ID, then click on Proceed.

User:

Version 1.0
Fall 1999

About D3M

Application Evaluation and Testing

Three buttons are displayed to the author, providing full control over the running and testing of exercises.



The captions of the first two change to reflect the effect of the button in the current status.

1. **Begin/Stop**
Selecting Begin opens the current scenario file, starts the simulation clock, and changes the caption of the Begin button to Stop. As time passes, any events in the opened events file are executed. If this button is selected while the simulation is running (its caption is then Stop), the simulation is stopped.
2. **Pause/Resume**
When a scenario begins in author mode, the Pause button appears. If it is selected while the simulation is running, its caption changes to Resume. When Resume is selected, the simulation continues.
3. **Replay**
After a scenario ends, either when time expires or the author selects Stop, the Replay button appears. Selecting Replay causes the prior situation to run again, now showing all the actions of the author.

IV. Sample Application

This section will describe a D³M application pertaining to command of fire fighting operations in a high rise building. Portions of two different responsibilities have been developed: 1) the first-arriving officer (FAO), whose primary responsibilities are to assess the situation, take immediate measures to protect life, and then to determine the source of the reported fire, and 2) a later arriving officer who assumes long-term responsibility for managing the operation. The FAO is the initial Incident Commander (IC), and his or her performance is critical to stabilizing the situation and protecting life. When a second officer arrives on scene, that individual assumes the IC function, and manages the operation, usually from a Chief's vehicle or a nearby structure.

The application described here addresses these two functions within the same simulation environment and using the same set of scenarios. As in the real world, the decision maker may not know until arriving on scene whether to act as FAO or to assume the role of Incident Commander from the first arriving officer. The IC function is very compatible with the characteristics of a high-level decision maker outlined in Section I, i.e., he or she takes in information and directs others in the resolution of the situation. The FAO functions covered by this application also focus upon the task of assessing the situation and managing resources, as opposed to technical fire fighting issues. Complete accounts of the responsibilities and tactical decision functions in high rise fire fighting have been documented by Coleman (1997), the Los Angeles Fire Department (1998), McAniff (1974), the National Fire Service (1996), Norman (1998), and O'Hagen (1977).

Use of Video Media

While computer graphics were used wherever practicable, this application relies heavily upon video media to portray visually critical elements of the environment. While more sophisticated computer graphics and virtual reality technologies could handle more of the representation, these are not currently as economical and realistic as video. The downside of the approach is that obtaining video segments that portray exactly the conditions desired is problematical. For the purpose of exercising and demonstrating the instructional system, however, it is clear that video media could be located or produced to fill in the many gaps. Interestingly, there is a wealth of fire fighting video available, but very little of it shows the critical, and less dramatic, elements that are required to represent the world to the decision maker.

Use of Text to Speech Media

Because much of the information received by the decision maker in this application is in the form of spoken messages and reports, the application also relies heavily upon text to speech capabilities. This permits verbal statements to be constructed internally via the simulation, and then played through the text to speech resource. The functions for doing this are established within D³M, thus an applicator need only pass text strings to the speech function, and they are spoken.

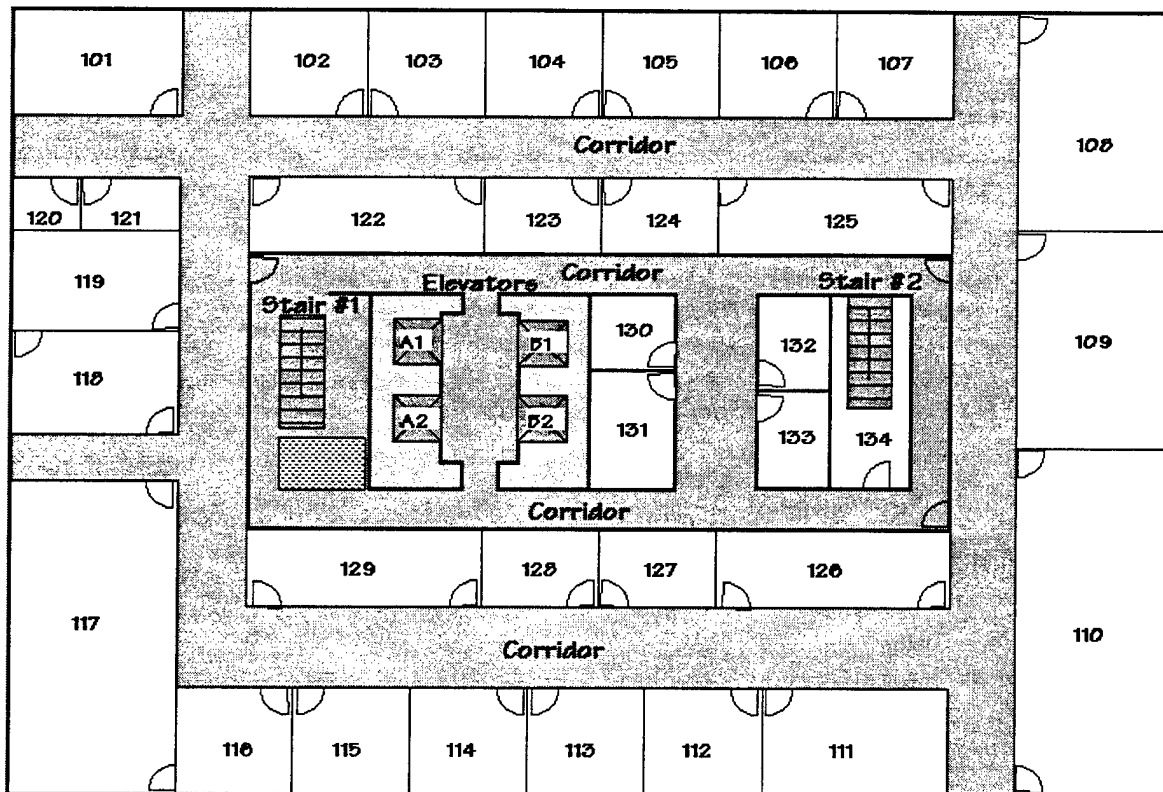
Sample Application

The particular text to speech resource currently provided in D³M is *Fonix AcuVoice Speech Synthesizer*, Version 3.4. It is installed in D³M as a standard Active X component. This system was selected after reviewing several dozen commercial and freeware alternatives (to compare the quality of different available systems, go to <http://morph.ldc.upenn.edu/lts/>). This system provides good quality of pronunciation, and it allows control of pitch and speed of the synthesized speech. Unfortunately, only a male voice is currently available, however, a female voice is under development by Fonix. The highest quality text to speech system identified was AT&T's Next-Generation, but unfortunately, this is not available for use at this time.

Applicators may employ the Fonix text to speech facility, with payment of a license fee to Fonix, they may substitute any other Active X text to speech system, or they may elect to bypass this medium altogether.

The Internal Simulation

The major element simulated in this application is a fire, initially within one compartment of a floor, but constantly on the move to adjoining areas of the floor and the building. The layout of the fire floor is represented as a set of *compartments*, each of which may be an office, apartment, storage room, or any other enclosed space with access doors, including hallways. The following figure, provided to the decision maker, is a floor plan showing compartments.



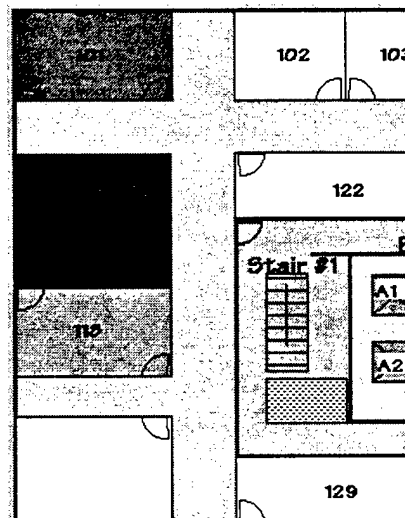
Sample Application

Each compartment carries the properties 1) *fuelLoading* that expresses how much combustible fuel is in the compartment, expressed in terms of time the compartment's contents will burn, and 2) *fireSpread*, a list of values each representing the minutes for fire to extend to each adjacent compartment. For example, the value of *fuelLoading* for compartment 108 above is 24 minutes, and the value of *fireSpread* is "109 10 107 12", signifying that 10 minutes after ignition of this compartment, compartment 109 will ignite, and 12 minutes after ignition compartment 107 will ignite from the radiant heat of this compartment. For some compartments, the fuel loading is less than the extension time, thus that compartment 'burns out' before causing adjacent units to ignite.

At the start of an exercise, the single involved compartment is put into a list of burning compartments. As the exercise progresses, the burn-time of the compartment is updated, and adjacent compartments are ignited and added to the list of burning compartment as their partitions break down. When a compartment consumes all of its fuel (its cumulative burn time equals its *fuelLoading*), it is removed from the list of burning compartments. Although the learner does not see the fire floor simulation, except in replay mode, it provides the status of the fire to all other elements.

With each updating of the fire floor, the simulation routine updates all other fire-related objects, such as the Operator Interface Panel. In addition, the status of various personnel such as assigned units and the availability of the building engineer is maintained in the simulation. Now, when the decision maker either calls for a report, checks the temperature of a door, or looks through a doorway, he or she sees a presentation that depends upon the status of the fire. Furthermore, when the decision maker acts, those actions are applied to the status of the simulated world, so that conditions can respond to the decision maker, through his or her agents.

During replays, the decision maker may view the fire simulation, via a colored view of the floor plan. Now the decision maker can see from the colors of the various compartments which are yet to ignite, which are burning, and which have consumed all their fuel. The following shows the upper left corner of the floor plan in which a fire started in compartment 120, and has spread to 121, 119, 118 and 101 across the hall.



Representing the Task Environment

Exercises begin at the fire station, with the sounding of an audio alarm and receipt of a printed message from central dispatch. The decision maker is the Battalion Chief of Engine Company 1. The report lists the time of receipt of the alarm, the address of the building involved, and the Truck crews and Engine crews that are being dispatched. While all of these factors could be varied, only the time of day is currently a scenario variable.

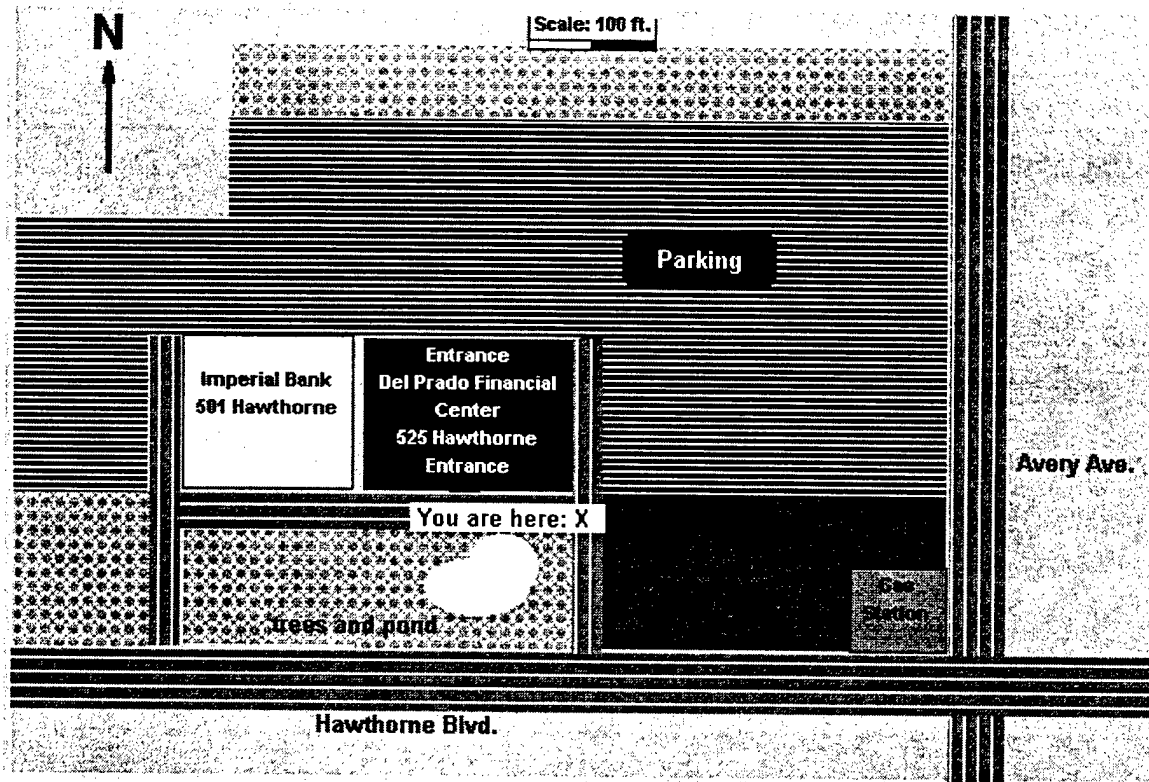
Alarm received at: 5:01 AM
Location: 525 Hawthorne Blvd., Torrance.
Dispatched: E-1 E-5 E-11 T-4 T-31

The user then sees a video of fire trucks moving to the fire scene, drawn from available fire documentaries. If the scenario occurs during the day, the video presents daytime views, otherwise travel is shown at night. Travel time to the fire scene is not varied, since one of the scenario variables is the time delay between the start of the fire and receipt of the alarm at central dispatch. Thus, the scenario can completely determine the length of time the fire burns before the department arrives on scene. The following is one frame of the video presented for this segment of the simulation.



Upon arrival at the fire scene, the user sees an overhead plot plan of the area (part of the theater of operation) and a video of the outside of the building involved. The plot plan of the area, which should affect the user's choice of command post location, is shown here.

Sample Application



If the scenario calls for fire and smoke to be apparent upon arrival – a critical issue to the arriving officer – the video shows that, otherwise it shows the same building with no fire or smoke showing. Here is a frame of the video shown when the building shows smoke.




Alternatively, the video may show people at windows or on ledges, which materially changes the situation and impacts the information that the officer should report.

Acting Upon the Simulated World

At this juncture, the user has decisions to make and actions to take. The following table lists the actions that can be made at any time, all of which produce spoken communications.

Action
Assume command
Pass command to another agency
Establish Command post and Base locations
Size up the situation
Call for additional alarm
Make assignment
Return units to station
Report conditions on current floor
Report status to dispatch
Confer with building engineer
Call in emergency medical
Call in air support
Establish staging floor

The primary mechanism for taking actions is via the following graphic, shown at all times after arrival at the fire scene.

<p>Transmit on :</p> <p><input type="radio"/> OCD channel</p> <p><input type="radio"/> fire ground</p> 	<p>Go to:</p> <p><input type="radio"/> fire control room</p> <p><input type="radio"/> stairway</p> <p><input type="radio"/> elevators</p> <p><input type="radio"/> Command post</p> <p><input type="radio"/> outside</p>	<p>See:</p> <p><input type="button" value="floor plan"/></p> <p><input type="button" value="stair and elevator plan"/></p>
---	---	---

This provides the options of:

- transmitting a radio message, on OCD (central dispatch) or fire ground channel;
- going to another location; or
- reviewing available documentation about the building.

(The *Command post* location is disabled above because it has not yet been established in the scenario from which it was taken. The *outside* location is disabled because the user is already outside the building).

Sample Application

Depending upon the scenario variables, another officer may have already arrived and begun entry to the building. If this is so, the user would hear that officer's report, and he or she *should* immediately assume the responsibility of Incident Commander (IC) from that first officer, via radio, and should establish a Command post. If another officer has not already arrived, then the user should issue a size up report and begin entry and ascent to the fire floor, to determine the situation. Of course, if there are people obviously in danger, then the operation must focus upon their safety. All of this activity, and all future action, is taken via the choices shown above. Details for each will now be given.

Transmitting Messages

Upon selecting either *OCD* or *fire ground*, under the Transmit option, the user sees the following menu of report types:

A screenshot of a 'Radio Report' menu. The title bar at the top reads 'Radio Report'. Below the title bar, the text 'Click below to broadcast.' is displayed. The menu consists of several options, each on a separate line with a dotted border: 'Report size-up', 'Establish command post or base', 'Call for another alarm', 'Report conditions on floor ...', 'Assume command', 'Report conditions', 'Confer with ...', 'Call for emergency medical', 'Call for air support', and 'Pass command to other agency'. At the bottom of the menu, there is a button labeled 'That's all for now'.

Clearly, just providing this list of options introduces a measure of artificiality, since it serves to remind the decision maker of the possible options for acting. However, until speech recognition becomes more reliable than it is now, such incidental cues must be endured.

Selecting *Report size-up* produces the following sub-menu with which the user specifies the perceived gravity of the situation as well as the general strategy that he or she is ordering.

Sample Application

Radio Report

Size up & Strategy

Engine 1 is on the scene of a 35 story office building ...

Mode of operation:

nothing showing

fast attack

command mode

Life hazard:

none apparent

exists

Strategy:

offensive fire attack

defensive fire

Engine 1 will be Hawthorne Command.

Send **Cancel**

After selecting a mode of operation, the life hazard, and the strategy to be followed, the user selects *Send*. The entire message is then assembled and spoken by the text to speech system. The message spoken in this case would be as follows.

Engine 1 is on the scene of a 35 story office building, with fire and smoke showing. We're going fast attack. We have life hazard. This is a defensive fire. Engine 1 will be Hawthorne Command.

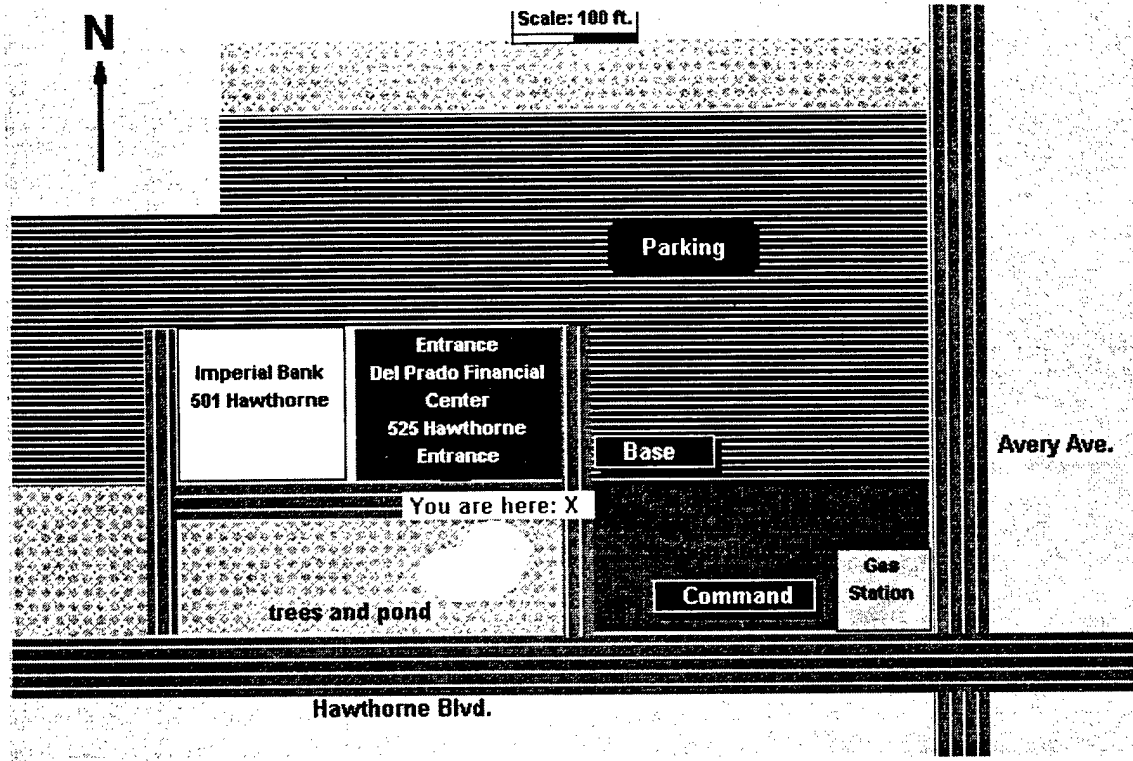
Selecting *Establish command post or base*, under the main Transmit option, returns the user to the overhead view of the fire scene shown above, and presents this instruction:

**To set the Command post or Base:
Drag the Command or Base block to the
location you want to establish. A radio
announcement will be broadcast immediately.**

Base **Command**

Now, the user drags the Base and/or Command box to any desired location on the site plan. The following shows the site plan after Base and Command have been positioned.

Sample Application



The radio report sent when the user establishes these positions is:

“Base is at parking directly east of Del Prado Center.”
 “Command is at open field at corner of Hawthorne and Avery.”

Approximately ten different locations are defined, and scoring rules exist to assess the appropriateness of each, as described later.

The following table summarizes the use of each of the remaining broadcast options.

Broadcast type	Sample broadcast, after selection of options
Call for another alarm	“Engine 1 to OCD. We’ll need another alarm here for a working fire.”
Report conditions on floor ...	“Conditions on floor 13 are clear. This floor is good for staging.”
Assume command	“Engine 1 is assuming command.”
Report conditions ...	“The situation here is under control.”
Confer with ...	Building engineer or IC provides briefing
Call for emergency medical	“Hawthorne Command to OCD. We need emergency medical here now.”
Call for air support	“Hawthorne Command to OCD. We need air support now.”
Pass command to other agency ...	“Hawthorne Command is passing command to Police Department.”

Relocating

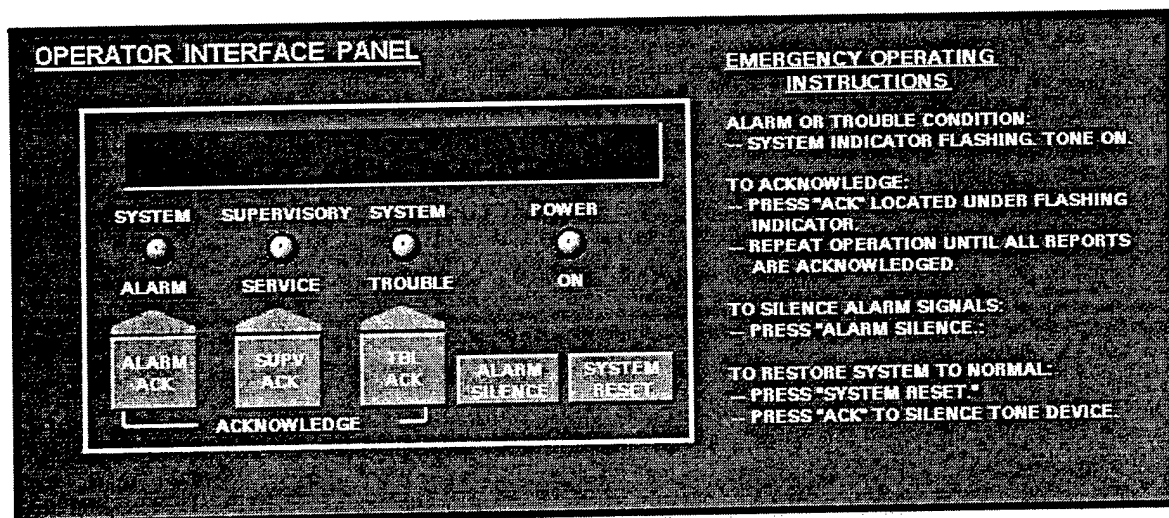
The decision maker may move to various locations, for the purpose of acquiring information. The following table lists 1) the key locations to which the decision maker may move in the simulated world to acquire information; 2) the information available at each location, and 3) the media employed to present that information.

Location	Information Source	Depiction Media
Fire Station*	Notice from central dispatch	text
Fire Station*	Audio alarm	sound (.wav)
Arrival at scene	Building external condition	Video
Arrival at scene	Aerial view of buildings and roads	Static graphics
Fire Control Room	Operator Interface Panel	Graphical simulation
Fire Control Room	Floor plan	Static graphics
Fire Control Room	Stair and elevator plan*	Static graphics
Stairway	Door condition (temperature)	text
Hallway	Condition (clear, smoky, flames)	video
Elevators	None; just means for moving vertically	Video and graphical simulation
Fire floor	Status of rooms/offices	video
Command Post	Current assignments and status	Graphical simulation
All locations	Incoming voice reports & dispatch	Text and Text to speech

* Initial location

Except for the static information (aerial view, floor plan, and stairway/elevator plan) the content of these information sources is determined dynamically from the generated scenario and the simulation. In the dangerous environment of a high-rise fire, even the process of obtaining information can be extremely hazardous, thus a learner may commit serious errors when moving from one floor to another, in order to gain more information. Note that the graphical simulation of the fire floor is only visible to the learner during replays, as it provides information the decision maker cannot see in the real world. This information source, however, is the means for maintaining the status of the fire over time, which in turn determines the information provided as the scenario unfolds.

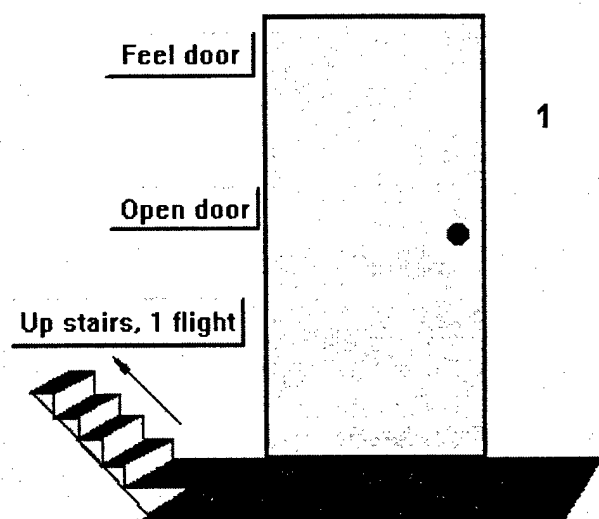
Fire Control Room. At the building's fire control room, the user sees and hears the fire alarm enunciator panel, as shown here.



Sample Application

This happens to be the Simplex model 4100 unit, a digital system used in many modern buildings. The data provided here reflect the scenario conditions. The user may silence the audio alarm and then view each of the alarms sensed by the system. With each depression of the ALARM ACK button another alarm, if any, is displayed. By analyzing the information here, the officer may be able to draw some inferences about the original location of the fire, how it has spread, the possibility that the alarm is false, how long a fire has been burning, and the degree to which smoke and toxins may be involved.

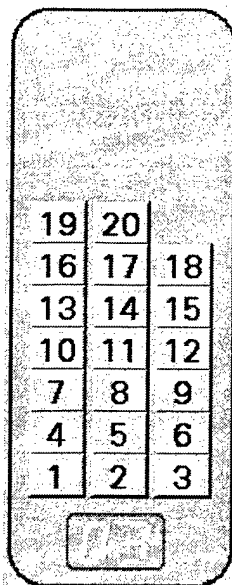
Stairway. To change floors by stair, the user selects *stairway* and sees the following graphic (all floors above the first also provide the option for descending).



Here, the user may feel the door, open the door, or ascend one floor. If *Feel door* is selected, a textual message informs the user whether the door is “cool”, “warm”, or “hot”. Upon opening the door, the user sees a video of a hallway that is either clear or smoky. Upon selecting *Up stairs, 1 flight*, the user sees a video of two firemen stepping up one flight of stairs, and then sees the figure above again, now with floor ‘2’ shown.

Elevator. Ascent and descent by elevator is similar. After selecting an elevator, possibly a critical decision, the user sees a video of fire fighters entering the elevator, then sees these controls for selecting the floor (the number of floor buttons displayed depends upon the scenario variable for total floors in the building).

Sample Application



Upon selecting a floor, the user sees a video of fire fighters in a closed elevator, and the floor number changing until the selected floor is reached. Then a video is played showing the fire fighters exiting the elevator. A sample frame is shown here.



Command Post. If the user has elected to serve as Incident Commander, and has established a Command Post, then he or she may go there to run the operation for the duration of the incident. The following interactive status/assignment board, which is similar to the Resource and Situation Status Record, form F-666 used by the Los Angeles Fire Department, is shown. As units arrive on scene, their status is automatically changed from *En route* to *Base*, which is where they would report upon arrival (in reality an assistant makes this change to the status board).

Sample Application

ASSIGNMENT/STATUS	Alarm 1						Alarm 2					
	E-1	E-5	E-11	T-4	T-23	BC-1	E-4	E-9	E-21	T-5	T-14	BC-2
En route												
Base												
Command Post												
Staging	✓											
Search		✓										
Evacuate occupants												
Determine fire floor				✓								
Attack						✓						
Backup			✓									
Lobby control					✓							
Elevator control												
Ventilation												
Return to service												

Now, the IC can assign any unit that is on scene by clicking the mouse in any box. If the IC calls for a second alarm, a second set of units appear on the *En route* row, and they change over time to *Base* as they arrive on scene. With each assignment, a corresponding verbal report is automatically spoken. The message spoken for the assignment of Truck Unit 23 (T-23) to Lobby control is:

“Truck 23, you will handle lobby control.”

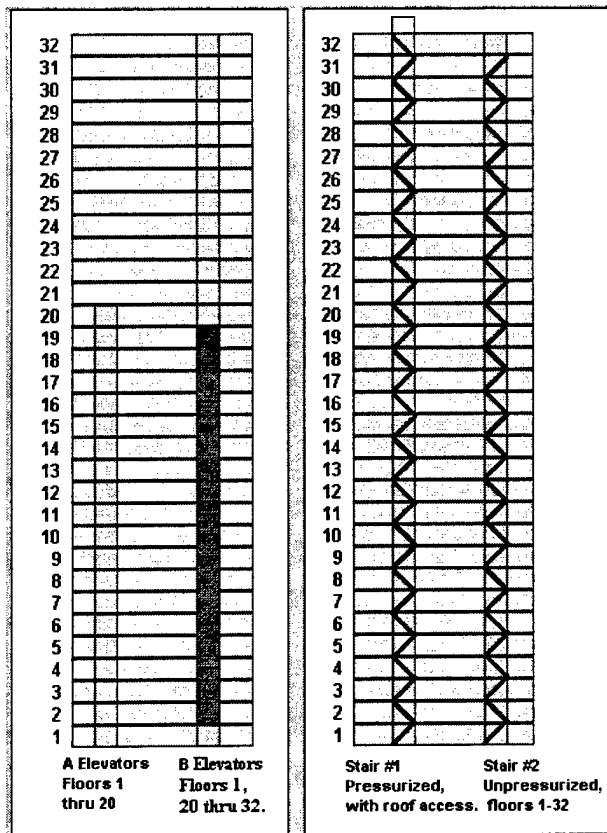
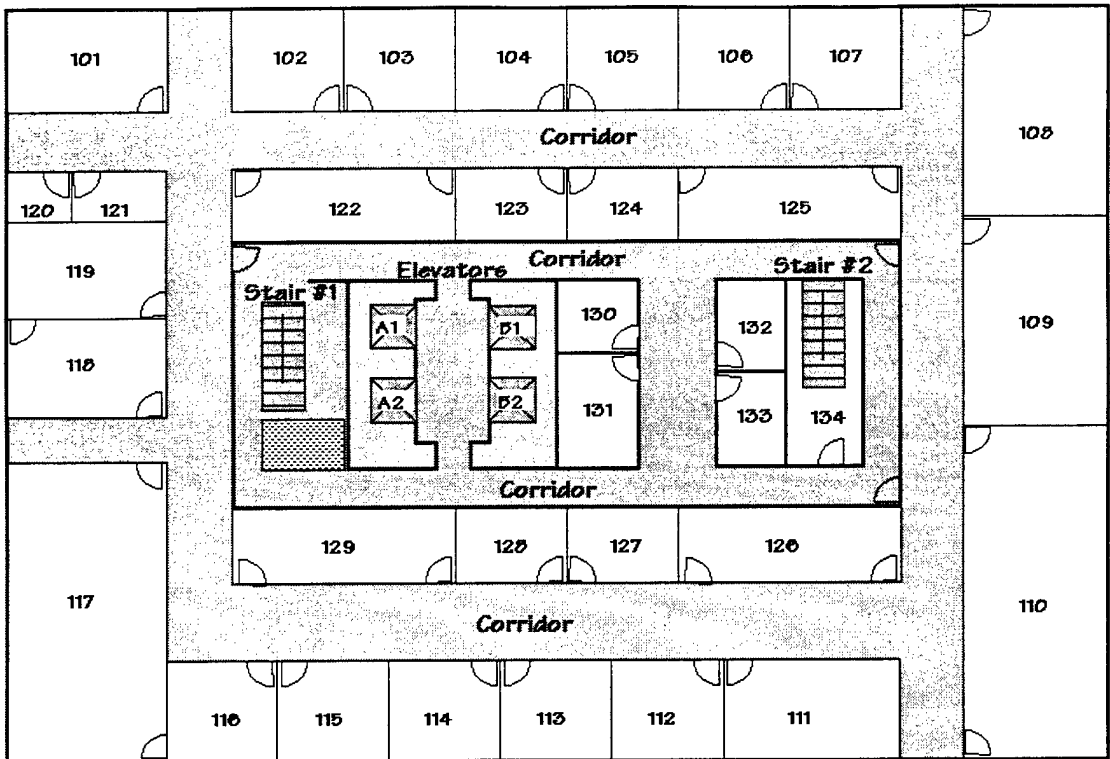
See

At the fire control room the user can obtain plans for the building’s floor layout, the stairways, and elevators. These are shown below.

The elevator plan reveals that the A elevators terminate at the 20th floor, and could thus be safely used if the fire is on floor 25 or above. Likewise, stairway #1 provides roof access, and is pressurized. Depending upon the density of civilian traffic coming down, the user might elect to use this stairway for bringing up department personnel and equipment.

Proper use of elevators and stairways is a critical aspect of this task, and scoring rules sense errors in this facet of the task. In addition, scoring rules check to see if the user reviewed this documentation at all. The next sections will elaborate upon the instructional content for this application, including the learning objectives, the scoring rules, and the scenario sets.

Sample Application



Instructional Content

This section will summarize the instructional content of the application in terms of the learning objectives, scoring rules, scenarios, and exercise conditions.

Learning Objectives

Appendix E lists the twenty learning objectives defined for the FAO, and the eleven defined for the IC. As a convenience, each objective name starts with either *FAO* or *IC*, signifying the role to which the objective applies.

Scoring Rules

The following eighteen scoring rules have been defined for particularly critical errors, and many more could be added.

Scoring Rule Name	Score Change
Failure to call additional alarm	-50
Calls an unnecessary additional alarm	-15
Failure to issue size up in a timely fashion	-20
Selects inappropriate command option during size up	-40
Failure to issue adequate periodic status reports during ascent	-20
Failure to report fire floor	-40
Failure to check door before opening	-50
Failure to close a door	-30
Failure to determine status above fire floor	-30
Failure to report staging floor during ascent	-20
Failure to observe enunciator panel before ascending	-20
Failure to study floor plan before ascending	-20
Failure to study stair and elevator plan before ascending	-20
Failure to confer with IC prior to assuming command	-40
Failure to confer with building engineer	-10
Unsafe use of elevators	-40
failure to establish and announce Command Post	-40
failure to establish and announce base	-30

The first scoring rule, failure to call additional alarm, is defined as:

`fireSmokeShowing > 1 and cumulativeBurnMinutes > 3 and faoAddAlarm = 0`

This rule fires if the FAO fails to call an additional alarm in a timely fashion. The rule detects that 1) fire and/or smoke is showing upon arrival, a variable set by the scenario, 2) the fire duration since arrival is greater than three minutes, and 3) the FAO has not called for an additional alarm, per a variable initialized to 0 and set to 1 if the user does call for an additional alarm.

The remaining parameters set for this rule are these:

Sample Application

- the value to add to the user's score: -50;
- the learning objective(s) to mark: #3 (fao call for additional alarm)
- the explanation number associated with scoring: #9 ("You have not called for an additional alarm. This is a critical action to take immediately upon observing fire or smoke upon arrival at the scene.")

Scenarios

The scenario variables that are set in the scenario dialog box are these:

Variable name	Meaning
FireStartTime	Time at which fire started, if any
FireRoom	Room number in which fire started, if any
NumberOfAlarms	Number of alarms registered at the fire control panel
AlarmType1-4	Type of alarm 1-4 on enunciator panel
AlarmDelay1-4	Minutes since fire started to alarm 1-4
AlarmFloor1-4	Floor indicated by alarm 1-4
TransitTimeE1	Time (min.) for Engine 1 to arrive at the fire scene
TransitTimeE5	Time (min.) for Engine 5 to arrive at the fire scene
FireSmokeShowing	fire/smoke is not showing (1) or it is showing (2)
EngineerArrival	Time (min.) for building engineer to arrive on scene (or 0)
BldgFloors	Number of floors in the building involved
BldgHeight	Height of involved building, in feet
BldgOccy	Number of people in building at time of reported fire
FireFloorOccy	Number of people on the fire floor

As noted in section II, the fire floor is related non-linearly and non-continuously to exercise difficulty, thus it is set within the application_startExercise routine.

Since any scenario can be combined with different sets of events, a small number of prototype scenarios support a relatively broad range of instructional conditions. The following table lists the prototype scenarios that have been defined.

Name	Situation Presented
False1	smoke alarm at control panel, no fire
False2	multiple manual alarms, no fire
False3	no alarm, no fire
Simple	all indications correct, low occupancy, smoke showing
BadID	no smoke showing, smoke alarm identifies higher floor
NoEngr	nighttime fire, building engineer unavailable
Delay	long time between start of fire and first alarm at panel
Evac	high building occupancy, much smoke, active fire

The False3 scenario, which involves neither alarms nor fire, is combined with a bomb-threat event, producing a situation that the department might well respond to, since it involves the threat of fire. The next section outlines the exercise conditions.

Sample Application

Exercises

By combining scenarios with various events, a large number of exercise conditions can be produced. The following were produced using just some of the possible combinations.

Exercise Condition
Simple false alarm
Pranksters set manual alarms; no fire
Telephone call received warning of bomb in building; none found.
Telephone call received warning of bomb in building; bomb discovered.
Simple fire with smoke showing, no problems
Simple fire with smoke showing, dangerous toxins encountered
No smoke showing, smoke alarm id's higher floor
Nighttime fire, building engineer unavailable, janitor has heart attack
High building occupancy, much smoke, active fire, poor occupant behavior
High building occupancy, much smoke, active fire, excellent occupant behavior
Fire in top floor restaurant; danger of explosion
Security encounters arson in progress; fires on 3 floors
No warnings; terrorists set fires on 3 floors
Distraught employee takes fellow employee hostage; manual alarm set off

Summary

The following table provides a summary of the application.

Simulation Element	High Rise Fire Fighting Application
elements simulated	fire progress, elevator locations, fire control panel
theater of operation	fire scene plan, floor plan, stairway design, elevator facilities
simulation variables	location of decision maker, status of resources, status of alarms, etc.
scenario variables	floors in building, time since alarm, type of alarm, time of day, etc.
exercise initialization	place decision maker at fire station, set fire condition at alarm time
special functions	setting of fire floor was done via an inserted program statement
simulation update	fire condition, location of decision maker, elevator location, etc.

Potential for Additional Task Instruction

If desired, additional instructional content could be developed for the functions of building engineer and floor fire warden, both civilian functions that are vital to effective response to fire-related emergencies. Such an extension would require development of additional learning objectives, explanations, scoring rules, and action options. The simulation, theater of operation, and instructional strategy would support these additional functions without change.

V. Conclusions

This section will reflect briefly upon the instructional process of D³M and aspects of applying it to a real world task.

Instructional Issues

Decision Making by Experts and Novices

As noted at the outset, much of expert decision making appears to be implicit, unconscious, and ongoing as opposed to explicit, discrete, and conscious. Yet, novices must learn what alternatives to consider, what factors to recognize in those considerations, and how to trade off the ubiquitous conflicts and how to assess the risks associated with uncertainties.

The training approach presented here has the capacity to present and discuss particular task performance issues, given that an expert 1) has created instructive prototype scenarios, 2) has effectively represented his or her thinking during performance of those scenarios, and 3) has composed reliable conditions that sense the quality of the learner's performance. While it appears that experts have considerable difficulty expressing what they do and why they do it *in generalized terms*, the fire fighting commanders we worked with had little difficulty composing instructive practice situations and explaining their performance *in those particular situations*. We suspect that experts would not easily author explanations except in the unique situation of doing the task, and reflecting upon the conditions they are observing and attempting to handle.

Shallow Simulation versus Deep Simulation

While a D³M task environment is simulated to a degree sufficient to enable the learner to perform the task, the level of simulation can actually be quite superficial, i.e., it is nothing akin to that required by engineering analysis or design. Such a superficial simulation, however, does not support automated production of instructional dialogs, as is done in such systems as DIAG (Towne, 1999). While the design of D³M does not preclude one from producing and exploiting such a deep model of a particular dynamic decision making task, the effort to do so for any one task would be at least equivalent to that invested in the analysis of fault diagnosis decision making, and most likely would be far greater, owing to the less well-defined problem environments and decision processes.

By relying upon the expert to construct the instructional elements, the system achieves broad applicability across the domain of dynamic decision making tasks. Further, this design facilitates ongoing expansion of instruction, possibly by multiple parties who could contribute new scenarios, additional or revised scoring rules, and enhanced instructional explanations. Such progressive development, distributed over time and space, would be essentially impossible were the instruction generated from a deep centralized model of the task environment.

Assessing Proficiency from Observable Actions

The context of the instruction, performance of a task in a simulation environment, versus interactive instruction about a task, permits the learner's ability to be assessed in terms of the observed actions performed, rather than in terms of inferred decisions made. Of course there will be those occasions when the learner performs the correct action for the wrong reason, and the instructional system will have no clue of that fact. However, this possibility also exists when a human expert is personally coaching a learner, and generally expert coaches seem willing to tolerate this possibility rather than to disrupt the exercise to inquire about the intentions and beliefs that accompany correct performance.

Development Issues

Development Alternative

As the fire fighting application was being developed, it became increasingly clear that virtually identical functionality could have been achieved by exploiting the scheduled event element within D³M, and completely bypassing the simulation development phase. A modest series of scheduled events could have produced the same progress of fire over time as does the fire simulation, all transparently to both the end user and other elements within the environment. It appears that this alternative is equally viable for many other domains, such as hostage negotiation, response to natural disasters, and forest fire fighting. Such tasks as air traffic control or CIC functions, however, probably would be better implemented via object-oriented simulation of the numerous elements in the problem environment.

Use of Video

Representing the functions of the First Arriving Officer was far more difficult than that of the Incident Commander, owing to the fact that the FAO personally experiences the incident. While static photographic images may have sufficed to represent the conditions within the building and the current movements of the learner, the motion and sound of video adds considerable realism. If nothing else, that increased realism seems to heighten the tension of the presentation, possibly better preparing learners to work in the real environment. Even with use of video, some aspects of the task, such as sensing temperatures, nature of smoke, and directions of drafts were either done via textual substitutes, or not provided at all.

Use of Text to Speech

As noted in Section IV, text to speech resources were used to convey spoken messages to the learner, and to speak the messages and orders composed by the learner. While voice quality was generally acceptable, the realism of spoken messages, in terms of voice tension, background noise, and static was quite low. Surprisingly, in the fire fighting application, realism would have been greatly enhanced had there been a way to degrade the quality of the sound, while maintaining the realism of the pronunciation. This is not

Conclusions

infeasible, since the synthesized speech is first written to file, then played through the standard Windows Media Player. There may be filtering resources that could degrade the quality of the speech file, making it resemble that which is heard over cell telephones or radio sets in an environment of high background noise and high tension.

Adaptation

The three primary adaptive processes within D³M are 1) producing scenarios of a desired level of difficulty, and 2) assessing the learner's proficiency via authored scoring rules, and 3) comparing the learner's proficiency score with expert scores. We conclude this report with a brief consideration of the validity of these processes, and alternatives for applying them.

Scenario Adaptation Issues

Section III described the alternatives for interpolating scenario variables, including linear interpolation, non-linear interpolation, and non-continuous table look-up. A broader question concerns how one assigns particular difficulty levels to the two prototype scenarios, for if these assignments are inaccurate the interpolation method becomes irrelevant.

The instructional system places no constraints on how this assignment is done. Four possible approaches are:

- 1) have a single expert make subjective judgements of scenario difficulty;
- 2) involve a panel of experts, using quantitative methods to arrive at consensus;
- 3) use the proficiency scores experts achieve in performing the scenarios to represent their difficulty; and
- 4) use the proficiency scores that novices achieve in performing the prototype scenarios.

Of these, the fourth alternative appears to be the one most justifiable. There does enter the question of whether or not relative difficulty across scenarios is consistent as learner expertise increases, and the most likely belief is that it is not. If this is a concern, then learners of varying levels of expertise could work the prototype scenarios, and measures of difficulty would result that could be used across the range of proficiency. Ultimately, a method might be devised in which the difficulty of the prototype scenarios is first established according to the individual's ability, prior to generating one to be delivered to that individual.

Secondly, there is the issue of independence of scenario variables. The current method of manipulating variables within D³M assumes independence, varying each one to produce the desired exercise difficulty. An alternative approach would have the applicator specify interactions or at least relationships among the individual variables. At the simplest, this could be a set of weights expressing the impact of each variable upon scenario difficulty. If this were done, then some of the variables could be automatically set more difficult than called for by the desired exercise difficulty, and some set easier, with the final result that the exercise difficulty is as desired.

Conclusions

This approach would yield the benefits of 1) removing the independence assumption, which is probably not true in general, and 2) producing a richer, more varied, set of practice conditions. This comes, of course, at the cost of considerably greater development effort due to likely difficulties in specifying interactions among variables.

Scoring Systems

D³M accepts virtually any scoring system one wishes to specify. The system used in the fire fighting application awards 100 points at the start of an exercise, and deducts for errors. Other scoring systems could award points for achieving various objectives, and could permit the number of points to vary according to the outcome achieved.

An ideal system might use real-world measures such as loss of property and loss of life to score the performance, however this then 1) forces the applicator to confront highly difficult questions about the value of property and life, and 2) forces the developer to carry through the simulation to accurately reflect the consequences of all actions. In a CIC setting, for example, this approach would require a much more powerful simulation of weapons effectiveness and range, so that the consequences of all decisions could be determined.

Normalizing Student Scores

D³M also imposes no constraints upon the manner of obtaining expert proficiency scores, for normalizing raw student proficiency scores. Perhaps the most legitimate way of doing this is to have a panel of experts perform the task in D³M to the level of proficiency they deem to be acceptable. Given sufficient agreement in the resulting criterion scores, those can be the basis for increasing and decreasing exercise difficulty, and therefore the basis for grading exercises as failed or mastered.

References

- Coleman, J.F. (1997). *Incident Management for the Street-Smart Fire Officer*. PennWell Publishing Company, Saddle Brook, NJ.
- Gordon, D., Schultz, A., Grefenstette, J., Ballas, J., and Perez, M. (1994). User's guide to the Navigation and Collision Avoidance Task. Technical Report AIC-94-013.
- Los Angeles Fire Department (1998). *Emergency High Rise Operations*.
- McAniff, E.P. (1974). *Strategic Concepts in Fire Fighting*. PennWell Publishing Company, Saddle Brook, NJ.
- National Fire Service (1996). *Model Procedures Guide for High-Rise Fire Fighting*. Fire Protection Publications, Oklahoma State University.
- Norman, J. (1998). *Fire Officer's Handbook of Tactics*. PennWell Publishing Company, Saddle Brook, NJ.
- O'Hagen, J.T. (1977). *High Rise/Fire and Life Safety*. PennWell Publishing Company, Saddle Brook, NJ.
- Stretton, M.L. and Lackie, J.H. (1996). *Shipboard instructor training and support: Scenario development functional architecture overview*. (SITS program technical review paper).
- Tannenbaum, S.I., Beard, R.L., and Salas, E. (1992). Team building and its influence on team effectiveness: An examination of conceptual and empirical developments. In K. Kelly (Ed.), *Issue theory, and research in industrial/organizational psychology*. Amsterdam: Elsevier.
- Towne, D.M. (1998) *Development of Scenario Tutors in a Generalized Authoring Environment: Feasibility Study* (ONR Final Report No. 119). Los Angeles: Behavioral Technology Laboratories, University of Southern California.
- Towne, D.M. (1995) *A configurable task environment for learning research* (Technical Report No. 115). Los Angeles: Behavioral Technology Laboratories, University of Southern California.
- Towne, D.M. (1999) "Automated production of instructionally appropriate scenarios" Proceedings, Eighth Conference on Computer Generated Forces and Behavioral Representation, Orlando, May 1999.
- Towne, D.M. (1999) *Diagnostic Instruction and Guidance* (Technical Report No. 120). Los Angeles: Behavioral Technology Laboratories, University of Southern California.
- Zachary, W.W., & Ryder, J.M. (1997). Decision-support systems: Integrating decision aiding and decision-training. In M.G. Helander, T.K. Landauer, & P. Prabhu (Eds.), *Handbook Of Human-Computer Interaction (2nd Edition)* (pp. 1235-1258), Amsterdam: Elsevier.

Appendix A. Detailed Student Performance Files

A detailed data file is created each time a student or the author performs an exercise. This file records each action performed, and the time at which the action was made. For students, the data file carries the name of the student, followed by the exercise number, and the suffix '.dat'. The author's latest performance is contained in the file author.dat.

The following is the content of file *fred1.dat*, reflecting Fred's work on exercise 1:

```
Fred1.dat
10/5/99 10:14 AM 1.0
FIRE1 1 none 3
0002 0 r Button doorway 1995,1305
0005 0 r Button elevator 1785,1680
0010 0 r Button fireControlRoom 1860,2070
0014 0 r Button done 9180,7995
0018 0 r Button CommandPost 1695,2385
0021 END
9999
```

The file format is as follows:

Record 1: file name
Record 2: date of performance, time of performance, and D³M version number
Record 3: scenario name (FIRE1)
 theater of operation number (1)
 event file name (none)
 current difficulty level (3)

Each of the succeeding records, except for the final two, provides data for one student action in chronological order. The data fields of each such record indicate:

- the time at which the action occurred, in seconds (2)
- '0', a placeholder required to simplify other processes
- 'r', indicating that this is a replayable action
- the object actuated by the student, ('Button doorway' for the first action)
- the x and y coordinates of the mouse action (1995,1305)

The next to last record contains 'END' in the second field, indicating that the exercise ended at the time given in field 1.

The final record is the marker '9999'

Appendix B. Student Status and Progress File

One data file is automatically maintained for each student to reflect progress and status on all exercises. The file name is the student's log-in name, followed by the suffix '.stu', e.g., *fred.stu* is Fred's student progress file. This file contains one header record followed by one record per exercise.

Here is file *fred.stu*:

```
Fred 1 940 11101000
3 6 0.85 0.75 320 2
2 4 0.60 0.73 620 1
0 0 0.00 0.00 000 0
```

The Header Record contains:

- the student's name (Fred)
- the number of the last exercise worked (1)
- cumulative time expended on all exercises (940 seconds)
- status of the learning objectives, 1 character per objective, where '0' means not yet attained and '1' means attained. Fred has attained learning objectives 1,2,3, and 5.

Each of the remaining records provide data for one exercise, as follows:

- number of trials completed (3, for exercise 1)
- last difficulty level completed (6, for exercise 1)
- average proficiency performing this exercise (0.85)
- proficiency on last performance of this exercise (0.75)
- time expended on all trials of this exercise (320 seconds)
- status of this exercise, where:
 - '0' means not yet attempted,
 - '1' means in progress,
 - '2' means failed, and
 - '3' means mastered

Appendix C. Applicator Routines

The following routines are set up in D³M for the applicator to employ as desired. These give the applicator access to the simulation update cycle as well as to exercise initialization and termination.

application_vals

This routine is called by D³M to evaluate the authored scoring conditions and functions. The author is only required to provide references to the domain-specific variables used in these conditions and functions, so that D³M can obtain current values. See the documentation within *application_vals* for more details.

application_startExercise

This routine is called at the start of each exercise. Any required domain-specific model initialization statements are placed here. See D³M's *startExercise* routine for documentation of its built in processing.

application_events

This routine is called whenever D³M determines that a ("do"-type) scheduled event is ready to be executed. Here are placed the programming statements necessary to execute the event. Note that most events, such as "change <some variable> to <some value> over <some duration>", are executed automatically by D³M and do not require any programming by the developer. Only special events that cannot be expressed with D³M's event scripting language must be placed within this routine. See D³M's *doCurrentEvents* routine for documentation of its built in processing.

application_updateSim

This routine is called each time D³M updates the simulated world, which it does as fast and as often as it can. D³M automatically keeps track of elapsed time in the simulation, as well as the time since the prior simulation update, both in decimal seconds. The statements inserted in this routine typically update time-related processes in the simulated world, such as locations of moving entities.

application_terminateExercise

This routine is called at the conclusion of each exercise. Here you may do any special scoring or data recording not performed automatically by D³M. Typically there is no need to place any programming statements here. See D³M's *terminateExercise* routine for documentation of its built in processing.

Appendix D. Help Text

The following text is displayed for the authoring functions when **Help** is selected.

Scheduled Events

Any number of scheduled events may be defined and saved per data file.
Each exercise can refer to a data file of scheduled events.

Select the CHANGE button to change some simulation attribute, either instantly or over time.
Select the DO FUNCTION button to execute a domain-specific function written by the developer.
All time values entered are in seconds.

The Property value entered may be a single word, such as "temperature" or it may be an attribute of an object, such as "floors of group building"

Any numerical value in an event can be an interval, such as 20-40.
The value will be set automatically according to exercise difficulty.

You may also specify what difficulty levels an event will occur at by selecting one of :<, >, >=, <=, <>, = from the drop-down list, and entering a difficulty level.

For example, "> 8" signifies that the event will occur when the difficulty of the associated exercise exceeds 8.

Scenario Setup

The entries shown on this screen specify one scenario prototype.
You may define as many scenario prototypes as you like.

To create a new scenario, select New Scenario from the author menu on the main page, then complete this screen and then select Save Scenario.
To modify an existing scenario, use Open scenario, make the modifications, then select Save scenario.
To delete an existing scenario, delete its file from your directory.

Exercises

The entries shown on this screen specify the makeup and presentation of one exercise.
You may define as many exercises as you like. Use the forward and back arrows to display any exercise.

To create a new exercise, select 'Create new exercise' and fill out the form. The exercise will be saved automatically.
To modify an exercise, step to it then change any of the displayed values. It will be saved automatically.
To delete an existing exercise, step to it then select 'Delete this exercise'. It will be deleted immediately.

For each exercise:

Select one of your defined scenarios to be presented, and a defined event file or "none".
Select a theater number, or enter 0 to specify that no theater of operation should be displayed.
Enter a time limit for the exercise, or enter a very large number to impose no time limit.
Maximum trials must be equal to or larger than Minimum trials. If you wish, set them both to 1 to present the exercise just one time.

The defined learning objectives are listed in the lower portion of the screen.
Click on an objective to specify that success on the exercise proves mastery of the learning objective.
The selected objective will be highlighted, and added to the list of objectives associated with this exercise.
To disassociate a highlighted objective from an exercise, click it. It will be unhighlighted.
When an exercise is mastered by the learner, all the associated learning objectives are marked as 'mastered'.

Explanations

This screen is used to create explanations that are presented when a learner plays an instructor's performance. Most explanations justify some observable action by the instructor, but they may also be used to explain why no action is being taken at some time, or to point out important aspects of an exercise when they emerge. Explanation items may be added, deleted, or modified on this screen. The changes are saved immediately and automatically.

While instructors perform exercises, they may click a special Explain Now button to mark that point in the performance. The simulation then pauses, and this screen is shown. The expert may then:

1. select any of the existing explanations for presentation at that time in the exercise or
2. create and select a new item for presentation, or
3. continue the exercise without selecting an explanation.

Course authors may also delete and add explanations to existing exercises when they play them. Alternatively, a text editor may be used to modify the explanations referred to in replay files.

Scoring

The scoring rules specify how the learner's score is computed as a result of performance. You may define as many scoring rules as you like, and they will be applied to all exercises.

For each scoring rule, enter a Rule Name, for your reference.

The Condition is an expression that evaluates to true or false, in terms of system properties that you define and maintain in your application. The condition is repeatedly evaluated during an exercise, unless it has been disabled (see below).

An example condition is: `timeSinceWarning > 120 and safetyShutOff is false`
where the variables `timeSinceWarning` and `safetyShutOff` are maintained by your application.

When a condition becomes true, the user's score is changed by an amount specified in the Score Change field. The entry may be: 1) a single value, like 8 or -10, or 2) a function defined using your properties

You may also enter a list of associated learning objective numbers, separated by commas.

When the Condition becomes true, the listed learning objectives will be set to:

- unmastered, if the score change is negative
- mastered, if the score change is positive and the learner is doing the exercise at the target level of difficulty.

There are three choices for specifying when the scoring condition will again be checked, after it has become true.

1. to never again evaluate the condition in the current exercise;
2. to have the condition checked again only after it has once gone back to false; or
3. to resume evaluation after some specified time.

Instructional Strategy

The instructional strategy controls the presentation of exercises for a course of instruction.

Select one of the four alternative exercise presentation schemes.

(A 'completed' exercise is one which has either been mastered, or has been failed due to reaching the time limit or maximum number of trials for that exercise.)

Set Recency Weight to a value between 0 and 1, to indicate how heavily to weigh the just-completed exercise in computing average proficiency. All previous exercises will be weighted with 1 minus this value.

Enter three values to indicate how exercise difficulty should be adjusted in accordance with the user's average proficiency.

The first value should be less than the second, and the second less than the third.

The Total Time limit is the time allowed for a user to complete the instruction.

If you wish to provide unlimited time, set this number to a very large value.

Appendix E. Learning Objectives

These are the learning objects defined in the high rise fire fighting application:

Objectives for First Arriving Officer (FAO)

FAO Correct choice of broadcast channel

When broadcasting reports, the officer selects the proper radio channel (OCD or fire ground).

FAO assumes command

Upon arrival, after another of lower rank has assumed command, the Officer confers with the current IC and then assumes command.

FAO Call for additional alarm

Upon arriving at the fire scene, the officer observes the building for smoke and/or fire. If either is visible, the officer calls for additional resources.

FAO Report size up upon arrival

The officer broadcasts a size up upon arriving at the scene of the reported fire.

FAO assumes command

Upon arrival, after another of lower rank has assumed command, the Officer confers with the current IC and then assumes command.

FAO Correctly report life hazard in size up

During size up, the officer correctly determines and reports the apparent life hazard.

FAO Correctly select offensive or defensive strategy

During size up, the officer correctly determines and reports whether the strategy is offensive or defensive.

FAO Broadcast periodic status reports during ascent

The officer broadcasts status reports every four to six floors, during ascent.

FAO Determine and report fire floor

Upon observing fire conditions, the officer reports the fire floor.

FAO Check door before opening

Prior to opening any door, the officer checks it by hand.

FAO Close doors when leaving floors

The officer closes any opened door prior to stepping to another floor.

FAO Determine and report conditions above fire floor

The officer ascends one floor above the fire floor to observe and report conditions.

FAO Determine staging floor during ascent

During ascent, the officer examines the floor two below the fire floor for service as the staging floor. If this floor is not appropriate for staging purposes, the officer continues downward until an appropriate floor is identified. At this time, the officer notifies dispatch of the staging floor number.

FAO Use of enunciator panel information

Prior to ascent, the officer proceeds to the fire control room and examines the alarm information at the annunciate panel.

FAO Use of floor plans

Prior to ascent, the Officer examines the floor plans to guide in assessing conditions and conducting search.

FAO Use of stair and elevator plans

Prior to the use of stairs and elevators, the Officer examines the stair and elevator plans to determine safe means of ascent.

FAO Confer with building engineer

The officer confers with the building engineer, if possible, to determine the status of the occupants

FAO Safe use of elevators, going above fire floor

To go above fire floor, only use elevator that has blind shaft to the fire floor.

FAO Safe use of elevators, in ascending to fire floor

To ascend to fire floor, only use elevator that terminates at least five floors below fire floor.

FAO assumes command

Upon arrival, after another of lower rank has assumed command, the Officer confers with the current IC and then assumes command.

Objectives for Incident Commander (IC)

IC Establish base and command post

After assuming command, the IC establishes and announces positions for Base and Command.

IC Avoid repositioning forces in action

The Officer uses uncommitted forces to take alternative positions to working crews, rather than repositioning first responding units.

IC Relieve crews as their fatigue and supplies demand

The Officer relieves crews when their fatigue or air supply conditions would hamper further safe and effective performance.

IC Determination of defensive strategy

When there is no known threat to human safety, the fire conditions are extremely difficult to combat, and the value of the property is marginal, a defensive strategy should be adopted.

IC Call for emergency medical when necessary

When there have been injuries, or there are likely to be injuries, the IC calls in emergency medical units.

IC Call for air support when necessary

When fire has reached, or is expected to reach, floors near the roof, and occupants have taken positions on the roof, call in air support.

IC Roof control

If occupants have taken position on the roof, assign fire department personnel there for control.

IC Determination of defensive strategy

When there is no known threat to human safety, the fire conditions are overwhelming the available resources, and the value of the property is marginal, a defensive strategy should be adopted.

IC Confer with building engineer

The IC confers with the building engineer, if possible, to determine the building configuration and ventilation characteristics.

IC Call for evacuation

When fire conditions overwhelm the available resources, and an offensive strategy cannot be pursued, the Incident Commander calls for a general evacuation.

IC Correct choice of broadcast channel

When broadcasting reports, the officer selects the proper radio channel (OCD or fire ground).
New Objective