AFRL-RH-WP-TR-2009-0027

Extending Interactive Electronic Technical Manuals (IETMs) with Real and Virtual Animated Content for Maintenance Task Training

Norman I. Badler
Ben Sunshine-Hill
Center for Human Modeling and Simulation
University of Pennsylvania
Philadelphia PA 19104-6389

Patrick J. Vincent
Northrop Grumman Information Technology
2555 University Boulevard
Fairborn OH 45324-6501

December 2008

Final Report for November 2007 to December 2008

Air Force Research Laboratory
711th Human Performance Wing
Human Effectiveness Directorate
Warfighter Readiness Research Division
Logistics Readiness Branch
Wright-Patterson AFB OH 45433-7604

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th Air Base Wing Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RH-WP-TR-2009-0027 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

//SIGNED//
DENNIS J. RIECHMAN
Work Unit Manager
Logistics Readiness Branch

//SIGNED//
DANIEL R. WALKER, Colonel, USAF
Chief, Warfighter Readiness Research Division
Human Effectiveness Directorate
711th Human Performance Wing
Air Force Research Laboratory

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
December 2008	Final	November 2007 – December 2008

4. TITLE AND SUBTITLE

Extending Interactive Electronic Technical Manuals (IETMs) with Real and Virtual Animated Content for Maintenance Task Training

5a. CONTRACT NUMBER
FA8650-04-D-6546 DO#11

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62202F

6. AUTHOR(S)

[1]Norman I. Badler, [1]Ben Sunshine-Hill, [2]Patrick J. Vincent

5d. PROJECT NUMBER
7184

5e. TASK NUMBER
D2

5f. WORK UNIT NUMBER
7184D211

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

[1]Center for Human Modeling and Simulation
University of Pennsylvania
Philadelphia PA 19104-6389

[2]Northrop Grumman Information Technology
2555 University Boulevard
Fairborn OH 45324-6501

8. PERFORMING ORGANIZATION REPORT

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Materiel Command
Air Force Research Laboratory
711th Human Performance Wing
Human Effectiveness Directorate
Warfighter Readiness Research Division
Logistics Readiness Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSOR/MONITOR'S ACRONYM(S)
711 HPW/RHAL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-RH-WP-TR-2009-0027

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

88th ABW/PA cleared on 18 May 2009, 88ABW-2009-2090.

14. ABSTRACT

This report documents the research associated with the design and development of a prototype virtual training system referred to as the Course Resource with Active Materials (CRAM) which is intended to help augment the training of aircraft maintenance procedures. The system is intended to help Air Force personnel receiving initial skills training in aircraft maintenance career fields gain deeper insight and knowledge of system, procedural, and safety/hazard information associated with a maintenance task prior to performing the task on the actual aircraft. The system uses a virtual coach agent to guide the user's instruction, giving information on hazards as necessary to maximize knowledge retention. The system allows multiple users to collaborate on a training procedure, and to communicate during the procedure. The system also allows instructors to augment the standard instructions for completing a maintenance procedure with their own course material. Also discussed is the implementation of such a system in a manner which maximizes reusability and the efficiency of creating new content.

15. SUBJECT TERMS Course Resource with Active Materials (CRAM), Interactive Electronic Technical Manuals (IETMs), Avatars, Maintenance Task Training, Virtual Maintainers

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
U	U	U	SAR	40	NA

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. 239.18

THIS PAGE LEFT INTENTIONALLY BLANK

Table of Contents

THIS PAGE LEFT INTENTIONALLY BLANK

1.0 INTRODUCTION

The "Extending Interactive Electronic Technical Manuals (IETMs) with Real and Virtual Animated Content for Maintenance Task Training" research task was sponsored by the Air Force Research Laboratory's Logistics Readiness Branch (AFRL/RHAL) under the Technology for Agile Combat Support (TACS) contract (FA 8650-D-6546, Delivery Order #11). The period of performance for this effort extended from 1 November 2007 to 31 December 2008. The purpose of this effort was to investigate the selective addition of multimedia content, especially segmented video and 3D animated virtual maintainers, to existing IETM presentations as a training aid. This research effort produced a prototype system referred to as the Course Resource with Active Materials (CRAM) system – an interactive virtual reality system – that could potentially be used to augment current aircraft maintenance initial skills training for Air Force personnel.

1.1 Summary

CRAM is an interactive computer application that allows aircraft maintenance trainees to become familiar with and practice maintenance procedures in a virtual reality simulated environment. The primary purpose of this research effort was to design and implement such an application in a manner which would maximize the benefit of the application (from a learning perspective) in the context of a maintenance training course. The course selected as the focus of our research was the Air Force Fighter Aircraft Maintenance Apprentice Course (J3AQR2A333A025A) conducted by the 82nd Training Wing at Sheppard AFB, TX. The key components of CRAM include a Virtual Coach agent to guide trainees' instruction, as well as a Wiki component to assist trainees helping each other, particularly in tasks that would involve multiple personnel. One of the key objectives of CRAM is to help extend student knowledge of system, procedural and safety uptake, and retention in the context of aircraft maintenance.

1.2 Scope

The CRAM system is intended to help train Air Force personnel receiving initial skills training in aircraft maintenance career fields to perform various maintenance procedures. It is intended that any procedure described in an Electronic Technical Manual (ETM) or

Interactive ETM (IETM) should be able to be converted into a virtual training procedure. For demonstration the research focused on one particular maintenance procedure: aircraft jacking procedures for the F-15 fighter aircraft.

1.3 Background

Any virtual simulation system which is intended for use in a training environment must be evaluated along two axes. First, the practicality of virtual simulation versus actual hands-on training must be considered. In the case of surgical procedure training, for instance, a virtual training system can offer trainees the opportunity to participate in a procedure as many times as desired, without having to recruit a volunteer to undergo surgery each time. Moreover, mistakes made in the simulation, as opposed to mistakes when operating on an actual patient, carry no real-world consequences. In contrast, training on a procedure such as rope tying does not significantly benefit from virtual simulation: Rope is cheaply available, and the consequences for an improperly tied rope (at least during training) are negligible. A training program for a procedure which is problematic to practice in the real world stands to gain significantly from virtual simulation. Secondly, the actual benefit to trainees from using the system must be considered. Recent research into virtual training simulation has made the point that such simulation has the potential to provide practice without providing a deeper understanding of the system. That is, a trainee who practices a procedure in a virtual simulation may improve his skill at completing the simulation without actually becoming significantly better at completing the actual procedure. This becomes a danger whenever the parameters of the simulation are not exactly those of the real-world procedure: When the graphics are not fully immersive, for example, or when the haptic (touch) interaction in the simulation is significantly different from that of the real procedure.

The use of virtual training simulation for aircraft maintenance clearly ranks well on the first axis: the cost of maintaining an aircraft is high, and the danger to trainees from unsupervised practice would be great. On the second axis, there is considerable difficulty. Because the CRAM system is intended to be used on readily available consumer-level hardware with user input through a mouse or keyboard and visual output through a 2D monitor, the gulf between simulation and reality is vast. Therefore, the CRAM system begins with a significant disadvantage: it cannot hope to exactly replicate the real world experience associated with performing aircraft maintenance tasks. Instead, our system concentrates on

enabling learning by giving trainees an opportunity to familiarize themselves with a procedure, and more importantly, by facilitating the uptake of systems knowledge. That is, the trainee should not only begin to remember the steps of a procedure, but to understand the reasoning behind the steps as it relates to the functionality of the aircraft as well. (Note: this was the basis for the Virtual Coach agent that is described in Section 2.2).

1.4 Historical Background

Early computer-based training simulations were extremely limited, requiring trainees to carry out a set of tasks in a carefully predefined fashion (Tennyson & Buttrey 1980). This was as much a function of the strict limitations of the computer hardware available at the time as it was a reflection of the prevailing attitudes towards the most effective educational mode for training simulations. Later approaches often took the opposite approach, using the enriched simulation tools offered by advancing computer technology to give trainees a more open-ended experience; the idea was to simulate, as closely as possible, the actual nature of the real-world scenarios the trainee was being trained to correctly interact with (Steinberg, 1977). Although there is no doubt that the resultant simulations were richer and more realistic, studies have suggested that the increased learner control over the simulation that naturally evolved with the enrichment of the simulation itself was counterproductive (Brown, 2001). Other studies have broken this effect down further, correlating several variables with a trainee's ability to effectively self-direct this training. In particular, younger subjects without a college education and with limited self-education experience were less likely to use self-directed training simulations to effectively absorb the material presented (Williams, 1993). Although some designers have tried to combat this by giving trainees ongoing feedback about their performance on a simulation (Kluger & DeNisi 1996), this requires that the simulation incorporate precise and detailed ongoing diagnostics, which themselves can interfere with the learning process.

1.5 Context for CRAM Research

The CRAM system is not intended to serve as a standalone instructional system. Despite the promising successes in using virtual reality technology as an instructional tool (e.g., Washburg & Gosen, 2001, show an average 10 percent improvement from simulation-based training over conventional learning methods), it is generally felt that virtual training

alone is not sufficient for providing technical training where the demonstration of skill proficiency is essential to assessing performance (Thomas & Hooper 1991). Hands-on practice is essential for both familiarizing trainees with the physical "feel" of the procedureas well as receiving the guidance of trained instructors. In fact, many failures in the field of virtual training can be traced to systems which did not augment the virtualized component with real-world instruction (Bell, Kanar, & Kozlowski 2008). Instead, CRAM is intended to be used in the following contexts:

- Before training for a procedure has begun, as a low-risk way to explore the procedure.
- After the training for a procedure has begun, during a trainee's discretionary time, if the trainee feels he needs greater familiarity with the procedure.
- After training is complete, to maintain familiarity with the procedure.

Although the creation of specific training content for the CRAM system was not a formal element of this research task, the use of some test content was important to correctly orient the design of the system as well as to test the design decisions that were made. For purposes of development, the task of jacking and lowering an F-15 aircraft was chosen.

1.6 Structure of the Report

Section 2 identifies and discusses the features of CRAM designed to meet the challenges of a virtual training simulation environment. Section 3 describes the user interface components of CRAM, and Section 4 discusses prototype demonstrations of the CRAM system and feedback. Section 5 discusses conclusions from the research and proposes recommendations for future work. The appendices discuss and analyze the technical design of the CRAM system, and give a detailed chronology of the development process.

2.0 CRAM PRIMARY RESEARCH CONSIDERATIONS

This section discusses the key research considerations that were addressed for the CRAM prototype system. These features were considered essential to the successful demonstration of CRAM as a training application and are discussed in the following sections. Additional technical considerations related to the design of other components, as well as considerations are addressed in more detail in Appendix A.

2.1 Multiuser Integration

Although using a computer is an inherently single-person activity due to the nature of the input devices, maintaining and repairing an aircraft is not. For instance, up to eight people were simultaneously involved in the F-15 aircraft jacking procedure, each with a distinct role to carry out. There is a significant difference in accomplishing a maintenance procedure alone versus performing the task as part of a team for the following reasons:

- **Separation of Responsibilities**. A team carrying out a maintenance procedure will assign each participant a role, either explicitly at the outset of the task procedure or through emergent behavior that occurs during the task. For instance, a participant may decide to work entirely with a single aircraft jack. This decision encourages the participant to think of the procedure in terms of everything happening to that specific jack, which may enable insights that would be lost if the participant were constantly moving to other sites. Of course, only working on that jack would limit the trainee's overall exposure to the procedure, meaning that trainees should rotate through roles for maximum benefit.
- **Inter-Participant Communication**. If a procedure has any required ordering on steps which are delegated to different participants, and it is not immediately and fully obvious to all participants which steps have or have not been completed, communication between participants is key. This is no different in a virtual simulation than in real life. A multiuser training simulation forces participants to communicate with each other, vocally or otherwise, regarding which steps have been completed and which participant is expected to act next. Multiuser integration, therefore, gives trainees exposure to a non-physical dimension of the procedure which would be lost without collaboration.
- **Coordinator Role**. Closely related to the preceding two points, just as a participant may not be directly involved in all physical actions taken during the procedure, so some participants may not act physically at all. This is particularly the case in the jacking procedure, wherein one participant is required to act as a spotter to ensure that jacks are raised evenly, reducing the risk of the aircraft falling off the jacks. In a single user environment, there is no need for this role; the participant must run back and forth between jacks anyway, so he can himself step back from the aircraft and assess its attitude on a regular basis. The addition of a multiuser component, therefore, gives trainees experience in a unique maintenance role which would otherwise be ignored.

Some maintenance task procedures require multiple trainees to perform the task, such as jacking an aircraft; however, CRAM does not require that such procedures be carried out with several participants. Flexibility is key: If trainees are encouraged to participate in training simulations on their own schedule, an individual trainee may at times wish to participate when a full team of participants is not available. If only a few trainees are available, they should have the ability to double up on roles in order to carry out the entire procedure, even to the point of a single participant fulfilling all roles. Additionally, just as separation of responsibilities gives participants an otherwise unavailable experience, completing a multi-participant procedure as a single trainee offers another perspective.

2.2 The Virtual Coach

In the early days of virtual reality training, it was felt that giving trainees a rich, realistic simulation environment and allowing them to direct their own learning was enough to realize the full benefits of virtual training. The research of Johnson et al., however, has suggested that entirely self-directed training is not as effective as externally directed training (Johnson et al., 1998). Self-directed students are not impartial judges of their own progress and are apt to spend too little time on portions of the material that they do not immediately see as useful or entertaining. On the other hand, an overly directed simulation would not be a simulation at all: merely a sequence of mouse clicks advancing through the steps of a procedure with little opportunity for the trainee to build an understanding of the steps he would be completing.

The most effective solution to this dilemma, as suggested by Johnson et al., is a Virtual Coach agent, whose presence and input are carefully curtailed to interfere only in situations in which it is necessary to optimize learning. Under normal circumstances, the Virtual Coach agent (henceforth referred to as the "Coach") would not have a visible presence in the virtual simulation. Like an instructor supervising trainees in a procedure in which they were already experienced, it's normal role is to stand back and observe, ensuring that the situation remains safe and that trainees are following the steps of the procedure correctly. If, however, the Coach sees that an unsafe condition has arisen (or is about to arise), or the procedure is not being correctly followed, the coach takes on actual, visual presence within the simulated environment and gives the trainees help with the procedure. This type of help may simply take the form of a message delivered to the trainees (either text, pre-recorded

audio, or text-to-speech), or may involve multimedia content. Once the Coach has successfully intervened to correct the problem, the virtual presence is removed until it is needed next.

2.2.1 Coaching Content

The presentation of the message is important. Recall that a stated purpose of CRAM is to extend system, procedural, and safety knowledge. Simply telling trainees that they have made a mistake would correct their behavior, but it would miss an opportunity to extend their knowledge. A better approach is to formulate the correctional message in the form of a targeted hypothetical: "You have done 'A'; if you do 'A'; 'B' could happen." The message should clearly connect the error and the consequence of the error, and should be placed in the proper context of the actual operation of the system. For instance, compare the message communicated to a trainee who fails to electrically ground an aircraft before beginning maintenance. A conventional computer-based training (CBT) system would simply give a message such as "You must electrically ground the aircraft," which may correct the trainee's future behavior during the procedure but fails to leverage any deeper training opportunity. A better message would be "You have failed to electrically ground the aircraft, which could lead to an explosion." This message ties the error to the consequences, with the result that the trainee will form a stronger negative association with the act of failing to ground the aircraft. Even better, however, would be "Failing to electrically ground the aircraft before beginning maintenance increases the risk of accumulated static charge causing a spark, which could ignite fuel and cause an explosion. Grounding the aircraft safely dissipates static charge." In addition to tying error and consequences, this message uses the error as an opportunity to explain the specific cause-and-effect linkages involved in the consequences. This could result in the trainee generalizing the knowledge beyond the specific step of the specific procedure; for instance, double-checking the grounding wire following a fuel spill.

As previously mentioned, the information presented to a trainee following an error need not be limited to text and speech, but can incorporate multimedia content. One key type of multimedia envisioned is a physically simulated animation of the potential consequences of the erroneous actions. For instance, in the case of an unevenly jacked aircraft, the system could display an animation of such an aircraft falling off the jacks due to the overbalance. Like the ideal correctional message formulation previously discussed, such an animation

makes a clear and educational argument about the importance of following the procedure correctly, and the inherently compelling nature of seeing catastrophes in progress. Even if only simulated, this should dramatically improve retention over simple textual messaging.

2.3 Three-Dimensional (3D) User Interaction

This section discusses key issues involved in designing a three-dimensional (3D) training simulation from the point of view of maximizing ease of use and training effectiveness. First, the importance of a three-dimensional system is discussed. Then approaches to dealing with the inherent difficulty of navigating and operating in a three-dimensional environment are explored.

2.3.1 The Importance of Dimensionality

Conventional CBT modules often consist of two-dimensional (2D) imagery, although the images may be of 3D objects. The content of these modules may appear realistic, but trainees are not free to navigate spatially in the simulated 'world' to arbitrary points. Rather, the trainee is constrained to look at what the module designers want the trainee to look at, presumably the areas that are to be manipulated during the given procedure. This is arguably suboptimal in two respects:

- The trainee does not have an opportunity to form a spatially directed mental map of the areas under maintenance. Witmer et al., demonstrated that the ability to navigate a simulated region dramatically improves users' ability to locate objects and places in the real world situations which were simulated over users who were merely allowed to study a map or schematic (Witmer et al., 1996). Even showing the subject a video with the camera moving around the region did not compare. In order to get this "spatial learning," it seems the user must be in control of a camera which can be moved around the scene at will.
- The maintenance area is often laid out in a partially schematic manner which facilitates site identification and overall mechanical comprehension. This is due to the difficulty of representing many 3D systems in a compact 2D form. This stylistic decision can give trainees a confusing or incorrect assessment of the actual layout of the corresponding area in the real world.

2.3.2 Intuitive Navigation

Navigating through a 3D world using a two-dimensional interface is an inherently difficult task. People have deep seated instincts regarding how they look around and how to get from place to place in the real world, learned since birth. Replacing these instinctual motions with mappings to a keyboard and a mouse imposes a great cognitive burden on a user; even skilled users often subconsciously lean or crane their neck when playing video games, despite these actions having no effect in the virtual world. The dual burdens presented by a three-dimensional training simulation—of fighting instinctive navigational behaviors and of absorbing and applying training information—can cause a situation known as "cognitive overload," in which a student, overwhelmed by the learning demands being simultaneously placed on him, becomes unable to usefully absorb and retain the knowledge being presented. For CRAM to be useful, cognitive overload must be minimized by making the system as intuitive to use as possible. In "The Design of Future Things," an influential work in human-computer interaction, Donald Norman explains ways an interface may be made more intuitive (Norman, 2007). One of the most important is that an interface be culturally conventional. For instance, when people approach a door with a fixed bar which is horizontal, as opposed to vertical, they naturally push it, because the modern convention is for pull-handles to be mounted vertically.

What cultural conventions exist for navigating a virtual world through a computer? Because consumer-level three-dimensional virtual interactions were unknown until recently, such conventions do not have a long pedigree. However, now that video games have become commonplace, motion control schemes have sprung up, competed with each other for primacy, and evolved into a de facto standard for navigating a virtual world. For example, video games known as "first-person shooters" where the user engages in simulated firearm-based combat, the mouse is used both to orient the view as well as to pick objects of interest; other video games known as "role-playing games" tend to use the keyboard for orienting the view and the mouse for picking objects, allowing finer-grained control of non-shooting tasks. Additionally, all control schemes tend to use the W, A, S, and D keys for navigation, because the right hand is used for the mouse and the left hand is more naturally oriented to use these keys than the arrow keys on the right hand side of the keyboard. Waller demonstrated that users familiar with computer games—to include most learners in the new generation—are

more than capable of applying the control skills learned there to virtual training simulations, and thereby reaping the benefits of avoiding cognitive overload (Waller, 2000). By carefully following the established video game control conventions, the CRAM system leverages trainees' existing knowledge to maximize usability.

2.4 Procedure Description

When reviewing lessons given by USAF maintenance instructors to their trainees, one invariant was obvious: education always went beyond the material in the Technical Orders (T.O.). An instructor who simply recited the T. O. material — even repeating pieces of it at appropriate moments in a demonstration — would not have effectively trained students to carry out the procedure in question. This is not the result of a deficiency in the T.O.s themselves. Indeed, if a T.O. did contain all the information conveyed by an instructor in the course of a lesson, it would be hopelessly pedantic and verbose for use in actual maintenance. The extra information not contained in the T.O. does not take the form of corrections or extra steps, but rather peripheral information, timely warnings, and insights engendering systems knowledge. For instance, the T.O. for aircraft jacking specifies, at one point, that the mechanic should check that all three jack feet are firmly seated on the ground. During instruction, the instructor demonstrated how to use the cover of the maintenance manual itself as a feeler gauge to check seating. This detail is anything but crucial: a mechanic would have access to more conventional feeler gauges during maintenance and, in any case, could figure out a way to check the seating on his own, given some time. Rather, the inclusion of this detail streamlines the practice of the procedure, maximizing efficiency in a clever and generally useful way.

An ideal virtual training system must make allowances for this sort of "oral tradition." So, although the textual content for the procedure is taken directly from the T.O., the system allows instructors to augment that information with their own content. This can simply take the form of text, placed below the T.O. text on the same panel, or it can consist of multimedia content such as video of the instructor explaining or demonstrating the point. The instructor-added information which is displayed to a given trainee can be dependent on the instructor to whom that trainee is assigned.

2.5 Wiki

A second "oral tradition" for information additional to that contained in the T.O. is relayed from one student to another. During real-world training, students collaborating on a procedure may explain confusing steps, give advice on correcting problems, etc., based on their own experience or on information that has been relayed to them. This oral tradition is less reliable than the first for two reasons. First, a trainee with the necessary knowledge for a particular situation is not necessarily around when that situation arises. Secondly, there is a chance that the information relayed could be erroneous, and be either unhelpful or counterproductive (or even dangerous). Because of this, allowing trainees to add their own notes to a T.O. to be viewed by other trainees is not necessarily a good idea.

An ideal tool for coping with the first shortcoming of the trainee-trainee oral tradition would be a Wiki. A Wiki is a web-based system for collaborative knowledge sharing. Any person with access to a Wiki can add, change, or delete information at will. The anarchy inherent to this unrestricted approach to knowledge sharing is tempered by making the reversion of unhelpful or misguided changes as simple as possible, making the act of making mistakes much more difficult than the act of fixing those mistakes.

The Wiki approach may or may not correct for the second shortcoming of the trainee-trainee oral tradition, that of misleading information being added. Trainees may be hesitant to correct mistakes made by others in Wiki content if they are not absolutely certain that the information is wrong. Giving instructors the ability to review all changes to the Wiki before they become visible to other trainees may fix this issue, but at the cost of making the Wiki less immediately responsive and increasing the instructor's workload. Further evaluation is necessary to determine how crucial an issue is this factor.

3.0 CRAM PRIMARY USER INTERFACE COMPONENTS

In this section, the main elements of CRAM's trainee user interface are discussed. The software details associated with the process of trainee authentication, plug-in installation, etc., which are only meaningful in the context of integrating the CRAM system into a larger on-line instructional system, are not included in this discussion. The interface elements described in this section are visible to the trainee once the trainee has authenticated himself and chosen to begin virtual training of a particular procedure.

3.1 Trainee User Interface Elements

3.1.1 Task List Panel

This panel, located on the left side of the interface depicted in Figure 1, displays the top-level sequence of steps in the T.O. The next step which must be completed to advance through the procedure is clearly indicated. Additionally, all steps are clickable; selecting a step causes the details of the step to be displayed in the step detail panel.

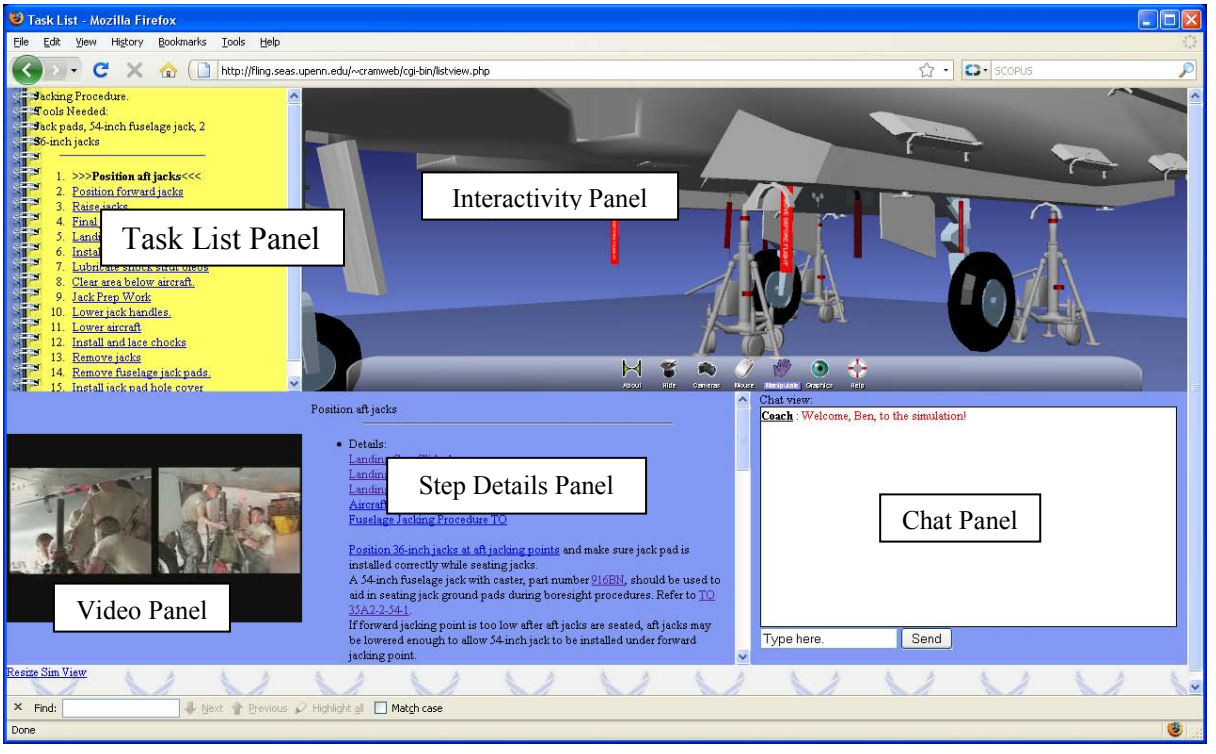

Figure 1: The CRAM User Interface

3.1.2 Step Detail Panel

This panel, located in the bottom center of the interface in Figure 1, displays details of the currently selected step. These details are taken verbatim from the T.O., but may be supplanted with reminders and advice from the instructor, as well as whatever multimedia content is desired. This multimedia, when selected, plays in the video panel. This panel also contains a text box to allow trainees to ask questions and make content suggestions, to be reviewed later by the instructor.

3.1.3 Video Panel

This panel, located to the left of the step detail panel in Figure 1, is responsible for playback of video and other multimedia content included by the instructor.

3.1.4 Interactivity Panel

This panel, located directly above the step detail panel in Figure 1, displays the interactive 3D world in which the procedure takes place. Trainees can move in the 3D world and manipulate objects using a control scheme familiar to them from 3D video games. During multi-trainee collaboration, other trainees are visible as avatars in this world. Additionally, the Virtual Coach agent appears in the virtual environment when necessary to give advice to the trainee as described in section 2.2.

3.1.5 Chat Panel

This panel, located in the lower right of Figure 1, allows trainees to communicate during multi-trainee collaboration, and its design reflects standard online "instant messaging" behavior. It consists of a text entry field to allow a trainee to type a message, which is then sent to all other trainees participating in the simulation. Past messages sent by all trainees are visible above this field, in chronological order. Additionally, the Virtual Coach uses this space to send its own messages to the trainees whenever necessary.

3.2 Using the Interface

The design of the interface is oriented around the concept of giving a trainee various tools to complete a given procedure in the interactivity panel. At any time, the user can use the task list panel to monitor progress through the procedure. The user can click a step to view its details in the step detail panel but, by default, that panel displays the details for the next step to be executed. The step detail panel is the user's main conduit for procedure completion information: it contains both the material in the T.O. as well as additional instructional multimedia content. Finally, the chat panel is used to communicate with other trainees and the virtual coach. For example, a trainee may begin the procedure by clicking through the steps in order to see what needs to be done. The trainee then returns to the details of the first step to view any associated video content before beginning, and after doing so, performs the step in the interactive pane. The system, recognizing that the step has been

completed properly, advances the simulation and displays the details for the next step. The trainee can go on to complete that step, but may first review the details of the first step if desired.

The instructor's interface when using the CRAM system is very similar to the trainee's interface. The primary differences are twofold. First, questions and comments made by trainees are visible in this interface, to be responded to either on an individual basis or by adding content to the training unit. Secondly, the instructor has the ability to add content to each step's details, which will appear when trainees are viewing or performing the step. This content may consist of text, images, or links to video content or other external multimedia content.

4.0 CRAM PROTOTYPE DEMONSTRATION AND TESTING

This section describes the demonstration and in-house user testing for the CRAM software that occurred during the research effort. The key features associated with each of these versions are discussed in more detail in Appendix B.

CRAM Version 1.2 was demonstrated for the Northrop Grumman TACS program manager and AFRL/RHAL project managers as an initial proof-of-concept of all major features. The primary feedback received regarding the Virtual Coach functionality was that the Coach's presence was too confusingly fragmented. Because the Coach's avatar did not visibly move at the same time the Virtual Coach messages were given to the trainee, the two aspects were not associated with each other. The fact that clicking the Coach's avatar toggled the visibility of the Wiki pane deepened the confusion. Additionally, the lack of an aircraft for an aircraft jacking procedure caused confusion, and the limited amount of screen space dedicated to the jacking procedure meant that the interface for navigating and completing the procedure was unwieldy. Finally, it was suggested that the demonstration procedure be expanded to cover both jacking and lowering of the aircraft.

CRAM Version 1.3 (described in Appendix B) was demonstrated at Sheppard AFB to several USAF maintenance instructors with the goal of checking assumptions and receiving suggestions regarding the relative utility of different features and the proper positioning of the tool within the larger context of aircraft maintenance training. Reaction from USAF instructors was generally positive, although some instructors expressed concern that the CRAM user interface could be unfamiliar to trainees and difficult for them to understand.

One instructor also pointed out that trainees should not necessarily be allowed to give advice to other trainees through the Wiki system, as that could perpetuate misconceptions. Finally, the instructors unanimously and stridently asked about the possibility of using a system like CRAM to augment training of the "aircraft safe for maintenance" procedures, as a significant amount of time is spent training that procedure and the wash-back rate is high. This was taken under consideration but is beyond the scope of this effort.

Informal user testing was performed using CRAM Version 1.4 on three computer science students who had not previously had experience with the CRAM system. Subjects were given a short textual introduction to the CRAM system and were then directed to complete the aircraft jacking and lowering procedure. Although all subjects completed the procedure correctly in well under the allotted time, they experienced difficulty comprehending various aspects of the interface. In particular, most were confused with the semantic difference between the "next step to complete" and the "step currently being viewed." These results led us to improve the user interface by clearly indicating whether the step being viewed was the next step to perform, and allowing users to jump directly to the next step to perform if it was not.

5.0 CONCLUSIONS AND RECOMMENDATIONS

The CRAM system is an efficient, effective, and even enjoyable tool for training certain aircraft maintenance task procedures. As a software prototype, it is robust and implements all required functionality to be maximally effective as a training tool for practical, everyday use. Although the system has yet to be formally evaluated (this will be accomplished as part of the TACS DO-15 research effort) in a controlled study, maintenance instructors at Sheppard AFB appeared to react positively to the CRAM demonstration, and felt that the system had great potential in the context of a course of instruction. The facility with which the F-15 jacking task could be represented within CRAM suggests that the architectural underpinnings of the system, specifically the layered approach to the procedure content and their training, represent a useful and novel paradigm for modeling mechanical procedures, their relationship to deeper systems understanding issues, and their support of multi-person coordinated tasks. Finally, the existing body of research into guided training simulations suggests that the Virtual Coach agent has the potential to greatly improve the effectiveness of the virtual training process. The wiki component, while potentially useful for

improving the educational experience, has not been thoroughly vetted; more research into the dynamic between trainee and instructor is necessary to discover the optimal modality for the wiki system.

As mentioned previously, a controlled study is necessary to assess how useful an application such as CRAM is relative to other instructional mediums such as traditional computer based instructional systems. Instructors at Sheppard AFB recommended that the training of maintenance procedures involving making the aircraft "safe for maintenance" might be an even more useful context for a system like CRAM for the purpose of: a) helping extend the general knowledge of system, procedural, and safety information associated with this task; and b) allowing trainee review and self-test outside of the classroom with the goal of helping reduce student washback rates associated with this training. Future research might focus on implementing and evaluating user interactivity in CRAM to support training for this type of maintenance task. Additionally, the possibility of implementing the CRAM system on a mobile or handheld device could be explored to provide broader access to trainees.

BIBLIOGRAPHY

1. Badler, N.I., Allbeck, J., Megahed, A., & Whitmore, M. (2006). RIVET: Rapid interactive visualization for extensible training. *Proceedings of the 2006 Conference on Habitation Research and Technology Development*, February 5-8, Orlando, FL.

2. Bell, B.S., Kanar, A.M., & Kozlowski, S.W.J. (2008). Current issues and future directions in simulation-based training in North America. *International Journal of Human Resource Management*, 19(8), 1416-1434.

3. Brooks, F.P. (1995). Sharp tools. In F.P. Brooks, *The mythical man-month: Essays on software engineering* (2nd Ed., pp. 127-140). Reading, MA: Addison-Wesley.

4. Brown, K.G. (2001). Using computers to deliver training: Which employees learn and why? *Personnel Psychology*, 54(2), 271-296.

5. Johnson, W.L., Rickel, J., Stiles, R., & Munro, A. (1998). Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environments*, 7(6), 523-546.

6. Kluger, A.N. & DeNisi, A. (1996). The effects of feedback interventions on performance: A historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psychological Bulletin*, 119(2), 254-84.

7. Norman, D.A. (2007). *The design of future things*. New York, NY: Basic Books .

8. Stalhane, T., Dingsøyr, T., Hanssen, G.K., & Moe, N.B. (2003). Post mortem - An assessment of two approaches. *Lecture Notes in Computer Science*, 2765, 129-141.

9. Steinberg, E.R. (1977). Review of student control in computer-assisted instruction. *Journal of Computer-Based Instruction*, 3(3), 84-90.

10. Tennyson, R.D. & Buttrey, T. (1980). Advisement and management strategies as design variables in computer-assisted instruction. *Educational Communication and Technology Journal*, 28(3), 169-176.

11. Thomas, R. & Hooper, E. (1991). Simulations: An opportunity we are missing. *Journal of Research on Computing in Education*, 23(4), 497-513.

12. Waller, D. (2000). Individual differences in spatial learning from computer-simulated environments. *Journal of Experimental Psychology: Applied*, 6(4), 307-321.

13. Washbush, J. & Gosen, J. (2001). An exploration of game-derived learning in total enterprise simulations. *Simulation & Gaming: An Interdisciplinary Journal*, 32(3), 281-296.

14. Williams, M.D. (1993). A comprehensive review of learner-control. *Proceedings of Selected Research and Development Presentations of the Annual Conference of the Association for Educational Communications and Technology* (pp. 1083-1114). January 13-17, New Orleans, LA.
15. Witmer, B.G., Bailey, J.H., Knerr, B.W., & Parsons, K.C. (1996). Virtual spaces and real world places: Transfer of route knowledge. *International Journal of Human-Computer Studies*, 45(4), 413-428.

APPENDIX A

CRAM Design Discussion and Analysis

The material in this Appendix focuses on discussing approaches and considerations (with supporting analysis) related to the design and implementation of CRAM features dealing with task procedures, Wiki content, Virtual Coach messages, etc. that are part of the CRAM prototype system.

A.1.1 Content Representation Architecture

Three high-level paradigms (or models) were considered for CRAM content representation. These models include: a) integrated code and content; b) simple scripted content; and c) layered content. Each of these approaches is described below from a technical perspective, including the advantages and disadvantages of each, and concluding with a summary of our analysis.

a. Integrated Code and Content

Description

In this approach, there is no formal separation between the underlying behavior code driving the simulation as a whole, and the actual training content which is presented to users: the two intermix in a monolithic representation. This organization (or lack thereof) is often referred to as "hard-coded," reflecting the inflexibility of the resultant software. This approach most closely resembles most common standalone computer-based training (CBT) modules. All content representation is in the form of program code for the language in which the application is written; there is no domain-specific language. Simple abstractions may be used to reduce code repetition, but nevertheless all content (other than multimedia content such as images and sounds) is a part of the same code as the underlying facilities of the application (e.g., tracking mouse motion).

Advantages

The primary advantage of such a simple approach is that work can proceed on content

creation almost immediately. Hard coding monolithic content requires very little up-front design work: because no framework needs to be designed or implemented to support standalone content, up-front design is limited to deciding on various development parameters. Additionally, the lack of a domain-specific language means that, as long as the application language is one with which the programmers are familiar, there is no lead time necessary for the programmers to familiarize themselves with the technology in use.

A secondary advantage is that since no decisions go into the design of an underlying framework (because there is no such framework), it is not possible to make those decisions incorrectly. The most common reason for a codebase to become inflexible is that it is invested in abstractions which prove to be unwieldy in implementing design decisions made late in the development process. With a monolithic approach such as this, only minor, low-risk abstractions are used, and this issue does not arise. Therefore, feature changes can be made in response to feedback even if they involve changes to the underlying design principles involved in earlier iterations. This can be ideal for an experimental project such as this one.

Disadvantages

A monolithic approach to content, however, misses out on many of the benefits of modern software development. Of primary concern is the lack of reusability. The code written for a given procedure, under a given set of feature assumptions, will likely be nearly useless in developing code for a new procedure or with a new set of feature assumptions. Because the content and the framework are interleaved throughout the application, changing one entails rewriting both. Additionally, this development approach tends to entail a lot of code duplication, leading to a longer development cycle particularly for complicated procedures.

Debugging is also problematic. Without reusable abstractions, bugs that occur are likely to be localized to a particular step, and will not manifest themselves except on that step. This makes detection of bugs more time consuming: A short “test” procedure is not sufficient to find bugs in the code except those arising directly from the test procedure itself. Worse, where systemic bugs do exist, propagating the fix to all steps is likely to be time-consuming and prone to introducing additional bugs.

Finally, and most compellingly, the design of such a monolithic application does not lend itself to insightful post-analysis. Because any needed feature can be coded in on an ad-hoc basis, no high-level patterns emerge as clearly as they would in the context of a framework.

Simply stated, a monolithic approach to content leads to a system which is larger, less maintainable, and more inscrutable than it needs to be.

Analysis

The quick turnaround time of the monolithic approach is appealing, given limited development time and several development cycles. However, we felt that we had a clear enough idea of the requirements of the system that we could build a framework in a manner that limited the risk of it turning out to be poorly conceived. Furthermore, we felt that the lessons learned from implementing this framework would be useful in future iterations of the application.

b. Simple Scripted Content

Description

As with the fully integrated approach described in the previous section, this content representation describes a procedure in terms of programming code. Unlike the monolithic approach, however, this approach clearly separates the code relating directly to procedures from the code relating to underlying application services. To accomplish this, a Domain Specific Language (DSL) is created, specifically designed to be ideal for the representation of virtual training procedures. The application then consists of two parts: the program code (written in the DSL) directly encoding the procedure, and the program code (written in the underlying platform programming language) which can interpret and execute the DSL program code. (In keeping with standard practice, in the remainder of this section, we refer to code written in the DSL as a "script," and the code written in the underlying platform programming language simply as "code" or "native code.")

Advantages

The primary advantage of separating out procedure code and giving it a dedicated language is that developing new procedures can be dramatically more efficient. The efficiency gain comes from the use of a language which is designed to be ideal for the task, allowing the procedure to be expressed in natural terms rather than having to be translated into the underlying native code representation. Such an approach also reduces debugging

time as well as the chance that bugs will go undetected. Bugs in the implementation of the DSL are generally easy to localize and correct when tested with short procedures intended for that task, and once the DSL implementation is debugged, as much scripting code as desired may be written with little need for further testing on the implementation. The scripting code may itself have bugs, but bugs in such languages tend to manifest themselves obviously with a minimum of testing (though this is by no means always the case). In an ideal case, such an approach can also "future-proof" content created for the application: If the application is re-implemented later on a different platform, only the DSL interpreter needs to be re-implemented; the content can be reused.

Finally, the design of the DSL itself becomes a useful exercise in requirements elicitation, as it forces the designer to consider all capabilities that those implementing individual procedures are likely to ever want. The degree to which procedures may be easily implemented in the resultant DSL becomes a test of the success of that elicitation. If several procedures, spanning different areas of maintenance, are successfully implemented without any workarounds for deficiencies in the DSL, that success makes a compelling case that the design of the DSL reflects a fundamentally correct apprehension of the needs of a virtual training application, no matter what its implementation.

Disadvantages

Although in the long run the use of a DSL can improve development efficiency, its initial development cost can be dramatically higher than the monolithic approach. The design of a DSL is no easy task; as mentioned above, it is impossible to do well without extensive requirements elicitation. Additionally, if the DSL turns out to be incapable of supporting a feature that is recognized late in the development process, modifying it to support the feature may require extensive recoding of content already written.

Analysis

Stalhane et al., have thoroughly demonstrated the enormous potential value to be gained from the development of a carefully-structured domain-specific content representation (Stalhane et al., 2003). This benefit far outweighs the potential development costs. Moreover, such development costs may in fact be more favorable than the integrated approach: software

engineering professionals have long felt that such domain-specific tools more than pay for their initial development cost in the long run (Brooks, 1995).

c. Layered Content

Description

The layered content approach, eventually settled on for CRAM content representation, is very much like the previous approach, in that procedure code and the underlying application code are separated. However, this approach stratifies the content itself as well. Rather than the content for a procedure consisting of a single file, it consists of multiple interacting code entities, each one a different "layer." There are many potential layers to choose from. In our implementation, the following layers are used:

Physical Layer. This layer consists of a description of the physical properties of the system, such as determining what happens when a given valve is turned. It provides limited qualitative description but is primarily quantitative in nature.

Practical Layer. This layer provides a method of tracking, and qualitatively describing, the "state" of the objects in the training simulation at any given time. For instance, in the case of a training simulation involving aircraft jacking, the practical layer would include whether the release valve on each jack was open or closed. The practical layer takes information directly from the physical layer for updates, but the qualitative information it contains can be expressed independently of the information from the physical layer.

Procedural Layer. This layer provides a top-level description of what step of a given technical order the trainees are currently completing. The content for this layer is drawn directly from the technical order. Unlike the other two layers, this layer stores no information except that which can be directly derived from the practical layer: merely by examining the state of the simulation it is possible to determine what the "next step" should be, or to determine that the trainee has failed to complete a particular step successfully. Each step within a procedure has a set of success conditions, qualitative items which the practical layer must satisfy before the step is complete. It also contains a set of failure conditions which

indicate that the step has been completed incorrectly. This includes virtual coach messages, textual and otherwise, which can be used in response to the failure conditions.

Advantages

The primary advantage of this approach is that content in each layer can be built up and used independently. Given a sufficiently expressive physical and practical content description for the landing gear system, for instance, it should be possible to write a procedural layer for any procedure involving the landing gear, without any need to rewrite or change the physical or practical content. In addition, this approach subsumes most of the advantages of the simple scripting approach. For instance, we believe that the future-proofing advantage of the simple scripting approach is even more pronounced with this approach: Although devices with differing graphical and interactive capabilities (e.g., cell phones) will most likely require a re-write of the physical layer, the practical and procedural layers need not be changed because the data they receive is still the same. Additionally, the efficiency gain of the scripting approach is even greater with this approach, because different programmers can be tasked to different layers simultaneously.

Disadvantages

The potential disadvantages of this approach are the same as those of the simple scripting approach: The development of a DSL takes time and development resources, and an incorrect design can lead to problems later in the development process.

Analysis

As with the simple scripting approach, we feel that the potential benefits more than make up for the developmental risk. Additionally, we feel that the layering approach will simplify the development of several novel features of the application, such as the virtual coach, by localizing their functionality to a single layer of the content model.

A.1.2 Application Platform

The "platform" for an application consists of the computer hardware required to run the application, coupled with operating system software and presentation software such as a web browser or a Java virtual machine. The choice of an application platform dictates both the

hardware/software configuration required to successfully execute the application, and the capabilities of the application. The following application platforms were considered:

Microsoft Windows, running on an x86-based architecture. This would involve designing the application to interact natively with the driver software, as well as the creation of a user interface using standard Windows tools. This is by far the most common hardware/software configuration in personal use today, meaning the majority of existing computers could be used for the simulation. Such a "close to the metal" approach would also maximize the potential performance of the application by minimizing overhead. However, desktop applications of this type are inherently single-user, having no standardized inter-computer communications framework. Additionally, deployment of native applications is inherently more difficult than that of other platforms, because of the installation process involved.

Apple OSX, running on Apple hardware. This is similar to the previous option, but market penetration is considerably lower.

Pure web-based client-server. With this option, the application on the client would consist of a web page, programmed with Javascript, interacting with a web server. The primary advantage of this platform is hardware independence: All modern operating systems have standards-compliant web browsers installed, and would thus be able to run the application. Such a platform would also be ideal for structured text with multimedia inclusions, a format that has been the web's bread and butter for a decade. However, simulation on such a platform would be problematic: There is no standard method to display 3D content without installing additional software. Simulation would therefore be limited to simple 2D representations.

Web-based client-server with plug-in based interactivity. Like the previous option, this platform houses the application on a web page. However, it uses a separately installed software program known as a "plug-in" to offer interactive 3D content within the web browser window. This plug-in will generally communicate with the surrounding web page using Javascript method calls, allowing integration between the plug-in and the rest of the

web page. The interactive capabilities of such plug-in content can far surpass those of plain web pages, since the plug-ins are generally written specifically for interactive content. These plug-ins are generally not as adept at web pages at rendering structured textual data, but due to the integration with the web page, that sort of content can still be displayed in the rest of the browser window. The main disadvantage of this blended approach is a more complicated development environment: the interactive content must be developed using a different language and content pipeline, increasing the amount of domain knowledge necessary to comprehend all the code involved in the application.

Despite the added complexity, the web page/plug-in platform was determined to be the optimal platform for this task. The differing natures of the interactive 3D area and the surrounding, primarily textual content meant that it was prudent to assign them to different programmers anyway, so there was no need for one programmer to familiarize himself with both subplatforms. The web technologies current in use consist of the following:

Hypertext Markup Language, or HTML. This is the language used by the World Wide Web to format the content of web pages.

PHP (not an acronym). This is a programming language run on a Web server to generate customized web pages to serve to browsers, as well as to perform server-side tasks such as database queries.

Javascript. This language is used to customize the behavior of web pages by executing directly on the client computer.

Asynchronous Javascript And XML (AJAX). This technology allows interactive updating of web page content at any time, with no need for users to manually reload the web page.

A.1.3 Interactive Plug-In

The decision to use Web technology for most of the display, but not the interactive simulation component raised an additional question: which interactive plug-in to use? The following options were considered:

Adobe Flash. This is the de facto industry standard for multimedia-rich interactive content. Most web-based multimedia games, for instance, are developed using Flash. Additionally, although Flash is not installed by default with any existing web browser, most users choose to install Flash due to the prevalence of Flash content as well as the high percentage of modern web sites which require the plug-in to be installed in order to be viewed correctly. Flash has support for full-motion video using a proprietary file format, as well as bitmap and vector graphics. However, its support for 3D content is limited at best, requiring all mathematics involved in the 3D rendering process to be performed manually. This sharply limits the level of display quality which is feasible while still maintaining a reasonable frame rate. Because we felt that a freely navigable and realistic training environment was essential for the system, Flash was thereby an untenable option.

Virtual Reality Modeling Language, or VRML. This format was created in 1994 as a standard to support the first generation of interactive 3D content on the World Wide Web. While sufficient for displaying 3D content, interactivity is limited to mouse clicks. Additionally, while a primary advantage of using an industry standard format would normally be its widespread adoption, in practice VRML is rarely used, and the average user does not have a VRML plug-in installed.

X3D. This standard is the official successor to VRML. Capabilities and adoption are similar to that of VRML.

Hypercosm. First released in 1999, Hypercosm is a proprietary technology developed specifically for web-based 3D interactivity. Hypercosm supports real-time physical simulation, and can use 3D objects developed in 3ds max, an industry-standard modeling applications. Training is one of the main applications for which Hypercosm was intended, making it ideal for our purposes. The primary disadvantages of Hypercosm are its high cost ($1000 for the content authoring environment, although the player plug-in is free) and the sharp learning curve for OMAR, its proprietary scripting language.

Analysis

As a proprietary solution, choosing to use Hypercosm subjects the project to greater vendor lock-in than choosing to use the other, more widely implemented technologies would. However, the fact that Hypercosm was developed specifically to enable solutions in this specific area (i.e., 3D simulations of manual procedures for training purposes), it was decided that the benefits far outweighed the risks. Other than its lack of support for 3D graphics, Flash is an ideal technology as well, and we would probably favor it for applications that did not stand to gain from 3D interactivity over 2D interactivity.

A.1.4 Multi-user Strategy

The choice of technology involved in connecting trainees together for multi-user simulation has important consequences for the usability of the system. The two main approaches for networking of this sort are known as client-server and peer-to-peer. We discuss each one of these approaches below:

Client-Server. The client-server networking model consists of an unequal partnership between client computers, which are directly used by trainees, and a single server computer, which is responsible for coordinating the client computers. In a theoretically pure client-server model, the server computer is not directly used by a trainee, but in practice it is possible, and at times expedient, to have one of the client computers additionally acting as a server. A web site is a classic example of a client-server model; one web server provides web site content to many client web browsers.

Peer-to-Peer. The peer-to-peer networking model consists of several "peer" computers, none of which serves as an unequal coordinator. Any computer can communicate with any other and all share the responsibility of maintaining a coherent application state across the different peers. Few, if any purely peer-to-peer applications are in use today due to the difficulty of peers finding each other over a wide-area network such as the Internet, but some applications, most notably file sharing applications, operate on a primarily peer-to-peer model, with a server providing only matchup services.

Analysis

The main advantage of the client server model, beyond ease of development, is the existence of a canonical shared state. In a peer-to-peer application, for instance, a single peer that changes the state of the simulated world must communicate that change to all peers. This can lead to a situation where different peers have a different conception of the state of the world at a given time, and coordinating between the peers can become quite tricky. In computer science this is known as the "Dining Philosophers" problem, and solutions tend to be complex and fragile. Therefore, we use the client-server model to coordinate the large-scale state of the simulation. Specifically, the server coordinates changes to the practical layer. Additionally, the client-server model was used to deploy all procedure scripts to clients; thus removing the need to manually install the procedures on each computer.

For continuous changes to fine-scale positioning of objects, however, we use peer-to-peer communication directly through Hypercosm. This approach, which leverages off-the-shelf functionality already available in Hypercosm, minimizes the latency time between one user moving an object and the other users seeing the results of that motion.

A.1.5 Other Design Parameters

In this section we describe other, less important development parameters of the CRAM system, including the handling of video content, graphics, and documents, as well authoring of 3D content.

Video Content. We encode all video content using the MPEG-4 audio/video compression standard, and industry standard format. We use Apple Quicktime, embedded in the web browser, to display the video content (although any embeddable MPEG-4-compatible player can be used alternatively).

Graphics. We encode still images, such as diagrams and photographs, in the PNG format, which provides both "lossless" compression (useful for diagrams) and "lossy" compression (preferred for photographs).

Documents. We use the Adobe Acrobat format to store long, pre-formatted documents such as technical orders.

3D Modeling Pipeline. Autodesk 3ds Max was selected for creating 3D content. Hypercosm tools are available that convert exported 3ds Max models to Hypercosm-playable modules.

APPENDIX B

CRAM Development Chronology

This section gives a summary history of the software development activities coinciding with each version of the CRAM prototype in chronological order.

B.1.1 Version 0.1

This initial version represented an attempt to produce a simple proof-of-concept virtual simulation engine based on the RIVET interactive training tool (Badler et al., 2006). It was felt that by leveraging the RIVET system's existing support for task-based semantic description of a procedure, development risks and time could be minimized. In attempting to reuse the RIVET codebase, the following shortcomings manifested themselves:

- The RIVET codebase was written entirely in Java, and had no support for connecting to other platform technologies. While this was suitable for the RIVET project, which used a dedicated handheld device, we felt that our solution should use technologies more useful for widely available consumer hardware.
- The RIVET codebase did not include support for the virtual coach functionality, nor did it directly support the hazard modeling necessary to implement such functionality.
- The RIVET codebase was exclusively single-user. Retrofitting multiuser capabilities may have required extensive revision.
- The presentation layer in the RIVET codebase did not support the rich multimedia content envisioned.

Although the RIVET codebase was not directly leveraged in building the CRAM system, several lessons were taken from its design. Most importantly, we felt that the RIVET system's step-based view of a procedure, even in a system designed to be partly exploratory, was an ideal way to orient the user towards the successful completion of the procedure.

B.1.2 Version 0.2

This version included an initial mockup of the user interface to be used for the CRAM system. This nonfunctional design test was meant to help elucidate the relative importance of

various user interface elements, as well as to provide a jumping-off point for discussion. In particular, this was the time at which the precise role of the Virtual Coach was first being nailed down.

The fundamental layout embodied in this mockup would be used in the final implementation of CRAM. Various other graphical elements, however, were discarded as taking up too much screen real estate, and the design as a whole was simplified for future versions.

B.1.3 Version 1.0

This version saw the first code which would be used in the final implementation of CRAM. While having very little graphical presence (the interactivity consisted of a set of check boxes) this version had most of the computational functionality needed for the final version: full implementations of the procedural and practical layer reasoning engines. For this version, a very simple procedure for testing was used: the task of making tea. Although this is obviously not a difficult procedure, it allowed us to test our hazard modeling ideas, finding that transient hazards (which require no corrective action) and persistent hazards (which remain until corrected) required subtly different content representation.

B.1.4 Version 1.1

This version was the first version incorporating Hypercosm. This added the physical layer, the final layer in the semantic model of the procedure. As before, we used making tea as our test procedure. This version also saw the first revision of the Virtual Coach system, with messages from the Virtual Coach appearing in a red text box.

B.1.5 Version 1.2

This version incorporated an initial demonstration of the aircraft jacking procedure, with graphical models of the jacks but no graphical model of the aircraft itself. (Rectangles were used as stand-ins for the jack pads.) Additionally, chat and wiki functionality was added here, the latter as simply a longer-persisting chat system. Finally, a 2D model representing the Virtual Coach was added, and some video content was integrated into the procedure description.

B.1.6 Version 1.3

This version incorporated several substantive changes. The Virtual Coach was given physical presence in the world, appearing and pointing to the subjects of messages. An aircraft model replaced the stand-in jack pads. The ability to resize the 3D interactivity window was added. Finally, the demonstration procedure was expanded to cover lowering the aircraft.

B.1.7 Version 1.4

At this point, in response to instructor feedback the wiki component was split into student-to-instructor and instructor-to-student portions, removing persistent student-to-student content entirely. Instructor-to-student content was then incorporated visually into the step details pane, removing the need for a separate wiki pane (which dramatically increased the screen space available to the interactive simulation).

It is anticipated that minor additional changes will be made prior to final delivery with content as part of DO-15.