REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15 Oct 2001	Final 10 Dec 1998 – 30 Dec 1999

4. TITLE AND SUBTITLE

Javelin EPBST Software Development

5. FUNDING NUMBERS

DAAH01-97-D-R005 D.O. 013

6. AUTHORS

Stephen J. Dow

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

The University of Alabama in Huntsville
Huntsville, AL 35899

8. PERFORMING ORGANIZATION REPORT NUMBER

5-20417

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

U.S. Army Aviation & Missile Command
Redstone Arsenal, AL 35898

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This task continued work on a new trainer for the Javelin weapon system, called the Enhanced Producibility Basic Skills Trainer (EPBST). The author's work was part of a team effort which transformed early prototype software into a full implementation of the EPBST requirements. This report provides an overview of the EPBST software and detailed discussion of the methods used to display scenes containing terrain and moving targets.

14. SUBJECT TERMS
Javelin, trainer

15. NUMBER OF PAGES
8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15 Oct 2001	Final 10 Dec 1998 – 30 Dec 1999

4. TITLE AND SUBTITLE

Javelin EPBST Software Development

5. FUNDING NUMBERS

DAAH01-97-D-R005 D.O. 013

6. AUTHORS

Stephen J. Dow

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

The University of Alabama in Huntsville
Huntsville, AL 35899

8. PERFORMING ORGANIZATION REPORT NUMBER

5-20417

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

U.S. Army Aviation & Missile Command
Redstone Arsenal, AL 35898

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This task continued work on a new trainer for the Javelin weapon system, called the Enhanced Producibility Basic Skills Trainer (EPBST). The author's work was part of a team effort which transformed early prototype software into a full implementation of the EPBST requirements. This report provides an overview of the EPBST software and detailed discussion of the methods used to display scenes containing terrain and moving targets.

14. SUBJECT TERMS
Javelin, trainer

15. NUMBER OF PAGES
8

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

PLEASE CHECK THE APPROPRIATE BLOCK BELOW

DAO# _____

☐ _____ copies are being forwarded. Indicate whether Statement A, B, C, D, E, F, or X applies.

☑ DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

☐ DISTRIBUTION STATEMENT B:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES ONLY; (indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT C:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND THEIR CONTRACTS (Indicate Reason and Date). OTHER REQUESTS FOR THIS DOCUMENT SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT D:
DISTRIBUTION AUTHORIZED TO DoD AND U.S. DoD CONTRACTORS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT E:
DISTRIBUTION AUTHORIZED TO DoD COMPONENTS ONLY; (Indicate Reason and Date). OTHER REQUESTS SHALL BE REFERRED TO (Indicate Controlling DoD Office).

☐ DISTRIBUTION STATEMENT F:
FUTHER DISSEMINATION ONLY AS DIRECTED BY (Indicate Controlling DoD Office and Date) or HIGHER DoD AUTHORITY.

☐ DISTRIBUTION STATEMENT X:
DISTRIBUTION AUTHORIZED TO U.S. GOVERNMENT AGENCIES AND PRIVATE INDIVIDUALS OR ENTERPRISES ELIGIBLE TO OBTAIN EXPORT-CONTROLLED TECHNICAL DATA IN ACCORDANCE WITH DoD DIRECTIVE 5230.25. WITHHOLDING OF UNCLASSIFIED TECHNICAL DATA FROM PUBLIC DISCLOSURE, 6 Nov 1984 (indicate date of determination). CONTROLLING DoD OFFICE IS (Indicate Controlling DoD Office).

☐ This document was previously forwarded to DTIC on _____ (date) and the AD number is _____.

☐ In accordance with provisions of DoD instructions. The document requested is not supplied because:

☐ It will be published at a later date. (Enter approximate date, if known).

☐ Other. (Give Reason)

DoD Directive 5230.24, "Distribution Statements on Technical Documents," 18 Mar 87, contains seven distribution statements, as described briefly above. Technical Documents must be assigned distribution statements.

Stephen J. Dow
Print or Type Name

Stephen J Dow 15 Oct 2001
Authorized Signature/Date

256-824-6406
Telephone Number

Javelin EPBST Software Development

Stephen J. Dow
Department of Mathematical Sciences
The University of Alabama in Huntsville

October 15, 2001

Final Report

F/DOD/ARMY/AMCOM/Javelin EPBST Software Development
DAAH01-97-D-R005 D.O. 13

Period of Performance: 12/10/98 to 12/30/99

1. Introduction

This task continues work on the development of a new trainer for the Javelin weapon system, called the Javelin Enhanced Producibility Basic Skills Trainer (EPBST), which superceeds a prior generation trainer simply called the Basic Skills Trainer (BST). The period of performance of this task encompasses the core development period during which a team effort transformed early prototype code into a full implementation of the EPBST requirements.

Javelin is a shoulder-launched antitank missile. Both the BST and EPBST include (1) an instructor station, which performs the processing to generate and display a terrain scene with simulated targets, and (2) a simulated command launch unit (SCLU), which the student gunner manipulates as he or she would the actual Javelin weapon system to engage targets. Whereas the BST instructor station incorporated custom proprietary graphics hardware to display rather artificial-looking terrain scenes, the EPBST uses standard Personal Computer (PC) technology to display photographic terrain scenes with targets rendered from 3D models.

The project to develop the EPBST was split into hardware development to be performed by ECC International Corporation and software development to be performed by the Software Engineering Directorate (SED) of the US Army Aviation and Missile Command. The author's work under this task was part of the software development project, a team effort performed by SED personnel, the author, and other contractor personnel. Thus this document summarizes work performed by various contributors. It should also be noted that various other documentation for this project exists within the SED organization.

2. System Architecture

The major components of the EPBST software are illustrated in the System Architecture Diagram below, which has been reproduced from the EPBST Software Requirements Specification.

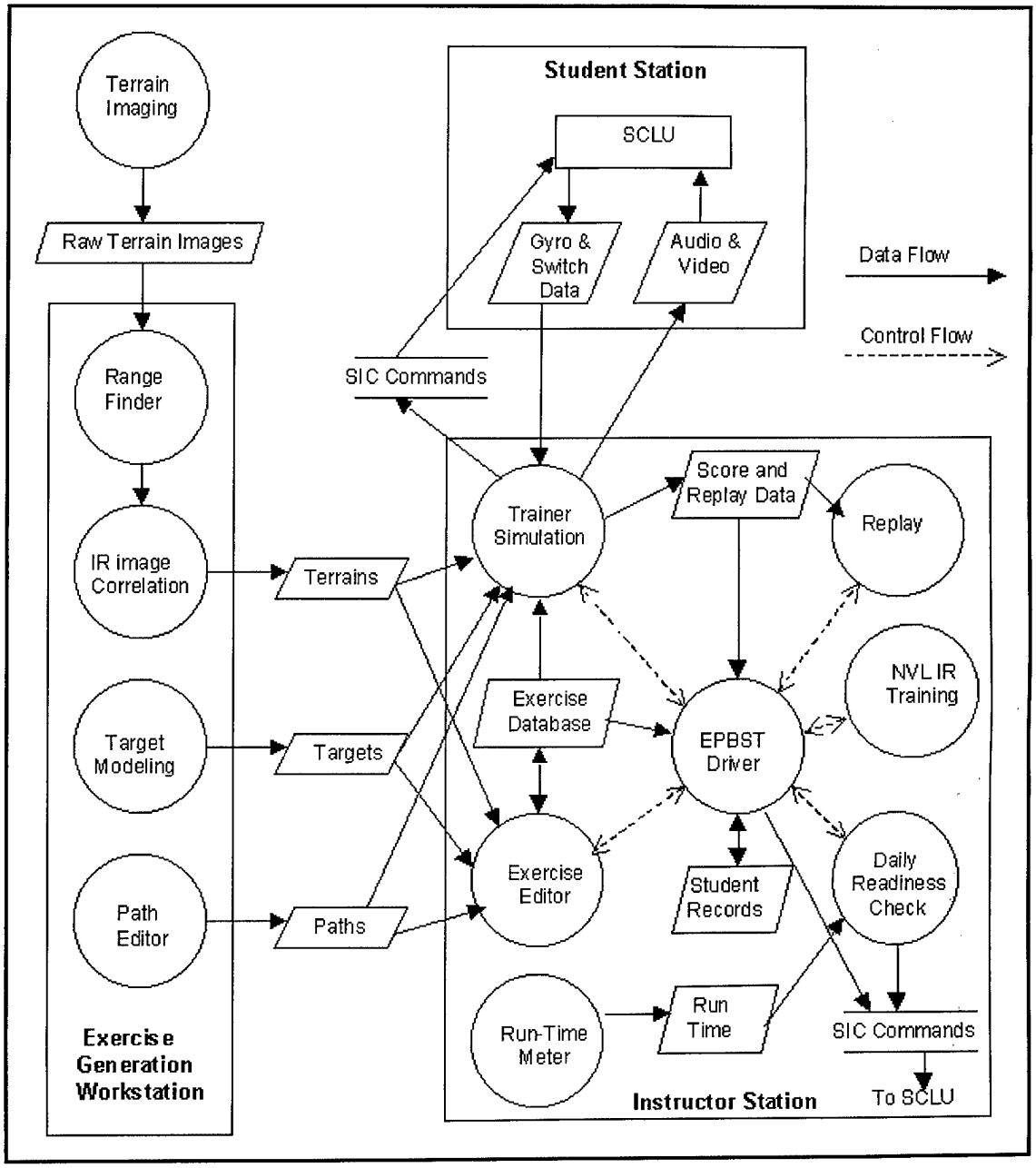

Figure 1: EPBST System Architecture

As shown in the diagram, the software mainly falls into two large groupings, Workstation and Instructor Station. The Workstation software is used offline, prior to training sessions, to create the terrain,

target, and path data needed during those sessions. The Instructor Station software is what runs during training sessions, to set up the training exercises, run the simulation (communicating with the SCLU), record scoring data, manage student records, etc. The Workstation software is used only by personnel at SED facilities, whereas the Instructor Station software is fielded to various training locations.

Versions of two of the main Workstation components, Range Finder and Path Editor, have been documented in the final reports of previous tasks by this author. The present report will focus on components of the Instructor Station software.

3. EPBST Driver

The Driver is the EPBST software module through which the user controls the execution of all the other instructor station software modules. The Driver executable file is named menu.exe. In addition to handling the startup of the other modules (diagnostics.exe, exedit.exe, tsim.exe, replay.exe), the driver program displays the menus for student sessions and incorporates the logic for scoring exercises and handling student records. The four main buttons on the EPBST main menu are:

Button	Function
DAILY READINESS CHECK	Start diagnostics.exe
JAVELIN TRAINING	Display student handler dialogs
EXERCISE EDITOR	Start exedit.exe
IR TRAINING	Start IR Training executable

Three of the four buttons, all except the JAVELIN TRAINING button, serve simply to start a separate executable program. The Driver (menu.exe) enters a wait mode until that other program exits, at which point execution returns to the Driver and the main menu appears again.

The JAVELIN TRAINING button serves to enter the student handler portion of the EPBST, which includes the dialog boxes for creating or loading a student record, for displaying a student session, and for displaying the score and critiques for a particular exercise the student has trained on. Those dialogs have buttons used to start tsim.exe ("START SELECTED EXERCISE") or replay.exe ("VIEW REPLAY"). The Driver also contains the code which displays the briefing screen prior to the start of an exercise. Interprocess communication between menu.exe and tsim.exe allows the briefing screen to appear while the simulation data is being loaded.

4. Exercise Editor

An EPBST Exercise consists of a terrain, a set of target paths across that terrain each with a specified target model to travel along that path, and additional parameters such as weather condition, number of available rounds the gunner will have, etc. A data file, named `est.dat`, stores the exercise information and is used by several EPBST modules, including the Driver, Trainer Simulation, and Exercise Editor. Only the Exercise Editor modifies the file however. The Exercise Editor allows the user to create new exercises, modify existing exercises, and copy exercises between EPBST workstations. The exercises are organized into directories; new directories may be created within the Exercise Editor. Some of the delivered directories are designated as read-only; exercises in those directories cannot be modified or deleted using the Exercise Editor. However these exercises may still be loaded into the Exercise Editor, and modified versions of them may be stored in other directories.

5. Trainer Simulation

The Trainer Simulation (TSIM) is the software module which actually runs the simulation, sending video and audio to the SCLU for the gunner to see and hear, receiving data from the SCLU corresponding to gunner aimpoint motion and switch input, and responding appropriately to simulate the behavior of the Javelin. This module corresponds to the program executable tsim.exe. It handles all aspects of generating the video with real time panning and target movement and audio sound effects needed to produce the simulation. Other software modules are responsible for setting up the data needed to start up the simulation; thus on startup tsim.exe enters directly into the simulation mode, all interaction with the instructor to set up exercise parameters, etc having already occurred. When the simulation is complete or is aborted, tsim.exe exits and software control returns to the Driver module.

Except for the message areas at the bottom of the screen, which are masked out so that the gunner does not see them, the display presented to the gunner is the same as that shown on the the instructor station screen. The display consists of the terrain scene with targets and overlaid symbology, surrounded by icons which indicate the system state.

TSIM performs some of its display logic in our own C code and some using DirectX hardware support. An overview of the logic for rendering 3D target models and inserting them into the terrain scene has been given in the previous report entitled "Trainer Software Development." Here we will discuss the subsequent display logic for composing all elements that appear on the screen.

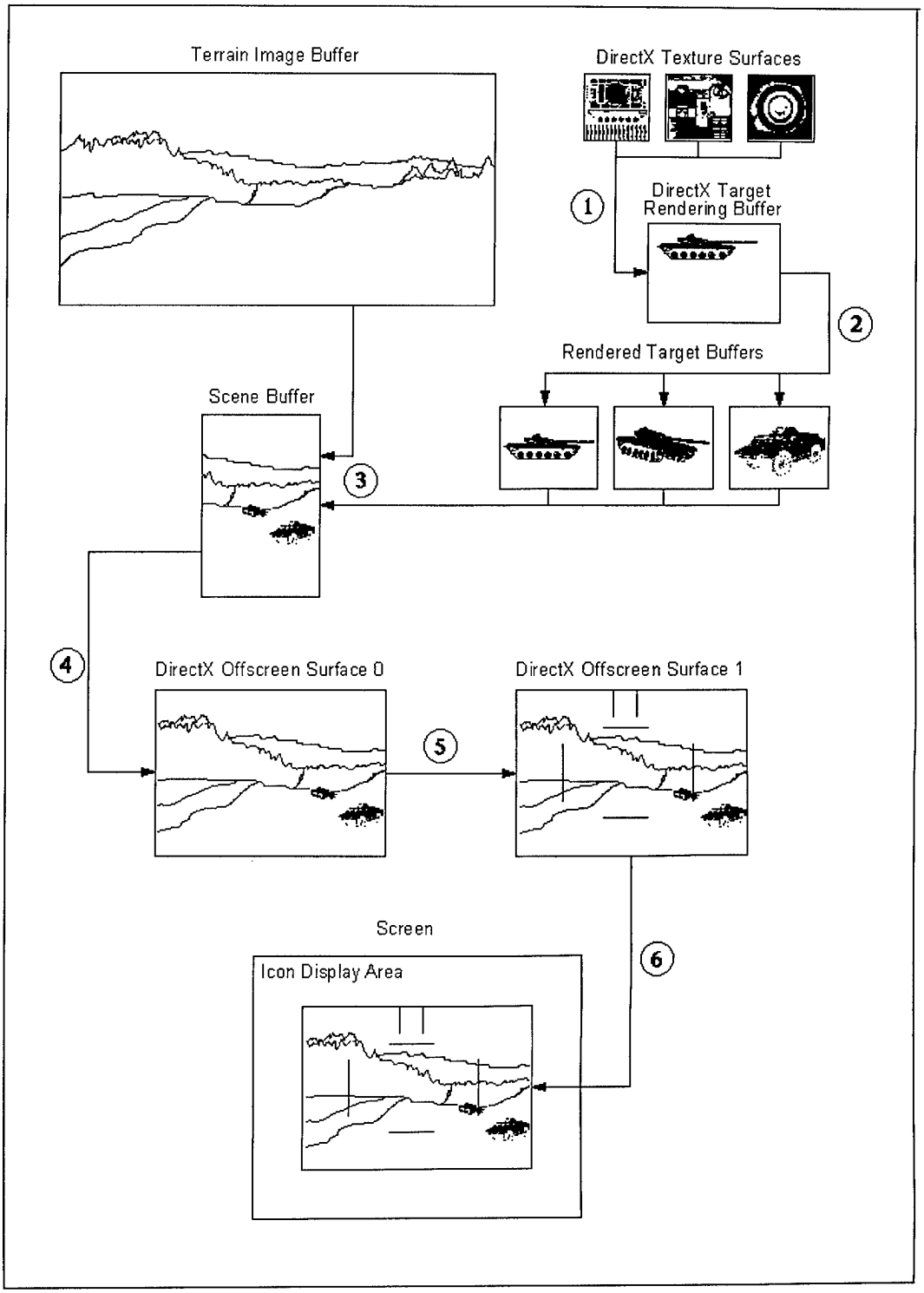

Figure 2: Display Path Diagram

At initialization several offscreen DirectX surfaces (chunks of video memory) are allocated:

(1) one texture surface for each target to hold its texture images;

(2) a rendering surface where the target (without terrain) is rendered;

(3) two offscreen surfaces ("Surface 0" and "Surface 1") for construction of the final display.

Since these surfaces are in video memory, their data is in the pixel format of the screen, which is 16-bit "high color" for TSIM. In addition to the DirectX surfaces there are image buffers allocated in system memory to hold

(4) the original terrain images;

(5) texture images;

(6) the rendered target images;

(7) the portion of the scene currently being updated;

(8) seeker image data.

The Display Path Diagram below has boxes indicating the various DirectX surfaces and image buffers, except for items 5 (texture images) and 8 (seeker image data) above. The Javelin system has several viewing modes: a visible spectrum mode (DAY), wide field-of-view infrared (WFOV), and narrow field-of-view infrared (NFOV). Three separate terrain images are held in memory to be used for these modes, the terrain image buffer in the diagram represents whichever of those three is currently in use. During target engagement the gunner enters a fourth viewing mode in which the image presented is from the missile seeker; the software uses the NFOV image buffer to construct the displayed image in seeker mode.The WFOV image has the same resolution as the visible image; NFOV has twice the resolution in each of X and Y. Also during initialization, all the texture images used by all the targets in the exercise are read from disk into system memory buffers. Each texture image is copied into a portion of the DirectX texture surface assigned to a given target. When switching between DAY and one of the night viewing modes, the texture images are swapped out, so that the proper images for the current viewing mode are in the DirectX surface.

Let's call the rectangular portion of the screen where terrain and targets appear the "display rectangle." In the code, this is either `display_rect` or `seeker_rect`, depending on the viewing mode. The sequence of steps used to fill this rectangle with the proper imagery are numbered 1-6 in the Display Path Diagram.

Step 1 represents the rendering of a target from its VRML vertex data and texture data into the DirectX rendering surface. That surface then contains a single target rendered on top of a background color, which is specified as a color (currently a fixed shade of purple) we do not expect to occur within the target. Immediately after rendering, the result is copied from the rendering surface into the system memory buffer for that target (Step 2 in the diagram). In the code, steps 1 and 2 occur within a call to EstRenderTarget (which in turn calls Dx functions to perform the rendering and transfer the data).

The function EstUpdateTargets loops through all the targets in the exercise, calling EstRenderTarget for each one that it deems to be in need of an updated rendering. EstUpdateTargets also maintains each target's

current rectangle in terrain pixel coordinates and its current range. It also computes the current frame and update rectangle for any active video clips (used for example to display the explosion when a target is hit).

Step 3 represents the construction of a portion of the scene containing terrain and possibly one or more targets. In the code this occurs in the function EstDrawSceneRect. It first copies the specified rectangle of terrain into the scene buffer; then it overwrites pixels where targets or animation frames appear using data from the corresponding target or animation buffers. This latter step is handled by a z-buffer algorithm; i.e. a range buffer for the update rectangle is initialized with terrain ranges; then it loops through the targets, copying non-background target pixels to the scene buffer when the target range is smaller than the corresponding value in the range buffer. Then active animations are drawn; alpha values are used to merge the animation pixel value with the underlying terrain/target pixel value. When applicable (i.e. WFOV or NFOV viewing modes), the contrast/brightness mapping and focus processing are applied to the resulting scene buffer during this step.

Step 4 represents transfer of the scene buffer just constructed to the first DirectX offscreen surface, Surface 0, which contains what's currently on the screen within the display rectangle, except for the overlay symbology (day stadia, crosshairs, trackgates, etc). This occurs within TsUpdateTerrainRect, when it calls DxCopyBmpToSurface, except in seeker mode, which is discussed separately below.

Step 5 represents copying the image from Surface 0 to Surface 1 and adding the overlay symbology.

Step 6 is just a direct copy of the data in Surface 1 to the center portion of the screen. Steps 5 and 6 occur in TsUpdateScene.

Additional processing occurs at Step 4 of the display logic when in seeker mode, to simulate the 64 x 64 resolution of the seeker. In seeker mode the dimensions of the display rectangle are `xview` = 230 and `yview` = 227. Three additional system memory buffers are allocated at initialization for use in seeker mode only. One, named `seeker_pix_lo`, is a 64 x 64 pixel buffer. The other two, named `seeker_pix_hi` and `seeker_pix_d`, have the dimensions `xview x yview` of the display rectangle. At Step 4 of the display path, instead of directly transferring the scene buffer constructed by EstDrawSceneRect into DirectX Surface 0, the data is first copied into `seeker_pix_hi`. That data is resolution-reduced by averaging small blocks of pixels to form each pixel of `seeker_pix_lo` and then stretched by pixel replication to create `seeker_pix_d`. The resulting data is then copied into DirectX Surface 0 for use in steps 5 and 6 as in before.

The code is structured so that a typical pass through the TSIM main loop (TsProcessFrame) requires only small portions of the display rectangle to be constructed by EstDrawSceneRect. The main causes for updates are target movement, active animations, and panning (changes to the line-of-sight). Within each main loop iteration, the rectangles which need to be updated due to target movement or animation changes are recorded, as well as those rectangles along the edge of the scene which come into view due to panning. If there are no changes such as a change in viewing mode which would require an update of the entire display, then the individual rectangles are passed to EstDrawSceneRect separately. The data already in DirectX Surface 0 or `seeker_pix_hi` from the previous iteration is shifted according to the new line-of-sight, and the

individually scene patch rectangles are constructed and copied into DirectX Surface 0 or `seeker_pix_hi` as described earlier. A single call to TsUpdateScene at the end of the loop iteration then handles the remaining seeker buffer processing and steps 5 and 6.

6. Replay

When the trainer simulation program (tsim.exe) exits, it writes a file containing a log of events that occurred during the running of that exercise, including all the line-of-sight changes, gunner switch inputs such seeker and fire trigger pulls, missile launches and target kills, etc. The file is stored in the `epbst\bin` directory with filename `ai.log`. This file is then used both by the driver program for scoring and by the replay program to replay the exercise.

When control returns to the driver program, it reads the file, computes a score, and generates critique messages. The score and critiques are displayed on the score dialog, which has "Okay" and "Cancel" buttons at the bottom. If the instructor hits the "Okay" button, the file ai.log is moved to the `epbst\replays` directory and renamed `student_name_XX_YY.log`, as explained below. This file is henceforward used only for replays; the scoring information which was computed is stored in the student record on floppy disk.

The replay filename `student_name_XX_YY.log` is constructed as follows: student_name is the student name as given in the student record, except with each space replaced by an underscore. YY is the exercise number (as it exists at the time the exercise is run), numbers bigger than 99 are coded as a two letter combination. XX are two additional characters chosen so that (1) the full filename above does not match an existing file in the replays directory, and (2) the four characters XXYY together do not match an existing replay id in this student's record.

The four characters XXYY are together called the replay id, and are stored in the student record along with the other information for the given exercise such as the score. If the student has previously attempted the current exercise, so that an existing replay id is found in the student record, then that same id is maintained, causing the previous replay file to be overwritten (if the attempt was on the same instructor station and the file still exists). If two students have the same student_name and are alternating use of the same instructor station, the XX part of the file names will keep them separate.

Whenever a replay file is about to be stored (i.e. moved and renamed from where tsim stored it), the disk usage of all *.log files in the replays directories is checked, and if it is greater than a preset limit (50 mb), replay files are deleted (oldest first), to bring the total disk usage back below the limit. This occurs after the above name determination.

The VIEW REPLAY button on the student session dialog is enabled or disabled based on whether there is a corresponding replay file for that student and exercise in the replays directory.