

TECHNICAL REPORT
NATICK/TR-10/008



AD _____

INDIVIDUAL COMBATANT'S WEAPONS FIRING ALGORITHM PHASE I OPTION

by
Amy E. Henninger

**Soar Technology, Inc.
Orlando, FL 32817**

April 2010

Final Report
October 2003 – November 2003

Approved for public release; distribution is unlimited.

**Prepared for
U.S. Army Natick Soldier Research, Development and Engineering Center
Natick, Massachusetts 01760-5020**

DISCLAIMERS

The findings contained in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of trade names in this report does not constitute an official endorsement or approval of the use of such items.

DESTRUCTION NOTICE

For Classified Documents:

Follow the procedures in DoD 5200.22-M, Industrial Security Manual, Section II-19 or DoD 5200.1-R, Information Security Program Regulation, Chapter IX.

For Unclassified/Limited Distribution Documents:

Destroy by any method that prevents disclosure of contents or reconstruction of the document.

REPORT DOCUMENTATION PAGE					Form Approved OMB No. 0704-0188																	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.																						
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.																						
1. REPORT DATE (DD-MM-YYYY) 21-04-2010		2. REPORT TYPE Final		3. DATES COVERED (From - To) October 2003-November 2003																		
4. TITLE AND SUBTITLE INDIVIDUAL COMBATANT'S WEAPONS FIRING ALGORITHM – PHASE I OPTION				5a. CONTRACT NUMBER DAAD16-02-C-0034																		
				5b. GRANT NUMBER																		
				5c. PROGRAM ELEMENT NUMBER																		
6. AUTHOR(S) Amy E. Henninger				5d. PROJECT NUMBER																		
				5e. TASK NUMBER																		
				5f. WORK UNIT NUMBER																		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Soar Technology, Inc. 3361 Rouse Road, Suite 175 Orlando, FL 32817				8. PERFORMING ORGANIZATION REPORT NUMBER																		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Natick Soldier Research, Development and Engineering Center ATTN: RDNS-TSM (T. Gilroy) Kansas Street, Natick, MA 01760-5020				10. SPONSOR/MONITOR'S ACRONYM(S) NSRDEC																		
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) NATICK/TR-10/008																		
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.																						
13. SUPPLEMENTARY NOTES																						
14. ABSTRACT <i>Report developed under Small Business Innovation Research contract.</i> Based on findings in Phase I, this effort developed a prototype data acquisition tool that can be used to collect large amounts of data over the WWW. In testing this tool, we acquired more data for the ICWFA that includes fuzzy estimates of factors influencing the target selection prioritization scheme, aim point, mode of fire, and estimates on Phit/Pmiss for a single SME. Also undertaken in this phase of the study was the development of a movie/concept demonstration to be distributed to other government agencies so that we can begin efforts in enhancing commercialization of product. Target markets will include both the analysis and training communities.																						
15. SUBJECT TERMS <table style="width: 100%; border: none;"> <tr> <td style="width: 25%;">SKILLS</td> <td style="width: 25%;">SIMULATORS</td> <td style="width: 25%;">DECISION MAKING</td> <td style="width: 25%;">TARGET SELECTION</td> </tr> <tr> <td>INTERNET</td> <td>PROTOTYPES</td> <td>VIRTUAL REALITY</td> <td>INTELLIGENT AGENTS</td> </tr> <tr> <td>SIMULATION</td> <td>SBIR REPORTS</td> <td>DATA COLLECTION</td> <td>VIRTUAL ENVIRONMENT</td> </tr> <tr> <td>ALGORITHMS</td> <td>WEAPONS FIRING</td> <td>MILITARY TRAINING</td> <td>INDIVIDUAL COMBATANTS</td> </tr> </table>							SKILLS	SIMULATORS	DECISION MAKING	TARGET SELECTION	INTERNET	PROTOTYPES	VIRTUAL REALITY	INTELLIGENT AGENTS	SIMULATION	SBIR REPORTS	DATA COLLECTION	VIRTUAL ENVIRONMENT	ALGORITHMS	WEAPONS FIRING	MILITARY TRAINING	INDIVIDUAL COMBATANTS
SKILLS	SIMULATORS	DECISION MAKING	TARGET SELECTION																			
INTERNET	PROTOTYPES	VIRTUAL REALITY	INTELLIGENT AGENTS																			
SIMULATION	SBIR REPORTS	DATA COLLECTION	VIRTUAL ENVIRONMENT																			
ALGORITHMS	WEAPONS FIRING	MILITARY TRAINING	INDIVIDUAL COMBATANTS																			
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT SAR	18. NUMBER OF PAGES 80	19a. NAME OF RESPONSIBLE PERSON Tom Gilroy																	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (include area code) (508) 233-5855																	

This page intentionally left blank

TABLE OF CONTENTS

LIST OF FIGURES.....	iv
LIST OF ACRONYMS AND ABBREVIATIONS.....	v
1 INTRODUCTION AND OVERVIEW	1
2 ENHANCEMENT OF PAST WORK - ALGORITHM EXTENSION.....	1
2.1 MOUT TARGET SELECTION ALGORITHM	1
2.2 CONSIDERATION OF HUMAN SCIENCE/MODELING EFFORTS.....	2
2.2.1 Review of Report: Squad Synthetic Environment Study (May, 2002).....	2
2.2.2 Review of Report: McKenna MOUT Study (September, 2002).....	2
3 PREPARATION FOR FUTURE WORK.....	2
3.1 KNOWLEDGE ACQUISITION TOOL.....	2
3.2 SME ARRANGEMENTS AND FUTURE DATA COLLECTION EFFORTS.....	2
3.3 DEMONSTRATION MOVIE TO DEVELOP PARTNERSHIPS IN EFFORT	4
4 CONCLUSIONS AND SUMMARY.....	4
REFERENCES	5
APPENDIX A: SME FUZZY ESTIMATES ON FACTORS AND ESTIMATES ON PHIT/PMISS	6
APPENDIX B: FITKAT USER MANUAL	34
APPENDIX C: FITKAT SOURCE CODE.....	44

LIST OF FIGURES

Figure 1. Relationships of WFA Factors.....1

Figure 2. Engagement Skills Trainer 20003

LIST OF ACRONYMS and ABBREVIATIONS

AAR	After Action Review
CGF	Computer Generated Force
HBR	Human Behavior Representation
IC	Individual Combatant
ICT	Institute for Creative Technologies
ICWFA	Individual Combatant Weapon Firing Algorithm
IOBC	Infantry Officer Basic Course
KA	Knowledge Acquisition
MOUT	Military Operations in Urban Terrain
PEO	Program Executive Office
RDECOM	Research, Development, and Engineering Command
RDT	Rapid Decision Trainer
RHS	Right Hand Side
ROE	Rules of Engagement
SAF	Semi-Automated Forces
SME	Subject Matter Expert
SOF	Special Operations Forces
SSCOM	Soldiers System Command
STRI	Simulation Training and Instrumentation
STTC	Simulation and Training Technology Center
WFA	Weapon Firing Algorithm

EXECUTIVE SUMMARY

Research Objective:

Report developed under a Small Business Innovation Research Program 2000.2 contract for topic A01.2-203. Based on findings in Phase I, this effort developed a prototype data acquisition tool that can be used to collect large amounts of data over the WWW. In testing this tool, we acquired more data for the ICWFA that includes fuzzy estimates of factors influencing the target selection prioritization scheme, aim point, mode of fire, and estimates on Phit/Pmiss for a single SME. Also undertaken in this phase of the study was the development of a movie/concept demonstration to be distributed to other government agencies so that we can begin efforts in enhancing commercialization of product. Target markets will include both the analysis and training communities.

Work Accomplished:

- Collected data on fuzzy estimates of factors identified in Phase I, aim points, mode of fire, and collected estimates on Pmiss/Phit based on those factors.
- Developed prototype data acquisition tool that can be used for web-based data collection effort.
- Developed concept demonstration to be used in commercialization efforts.
- Developed contacts for equipment and SMEs needed in future on-sight data collection efforts at USMA, Ft. Bragg, and RDECOM-STTC.

Conclusions:

The majority of work in this phase of project was allocated to the development of data acquisition system. While the immediate benefits of this tool are somewhat unrealized, we believe the investment will poise us to both collect and analyze data more efficiently in the future. Initial efforts to test the system have proven that it's easier for SMEs to work with and provides us data in format that facilitates analysis. Further, as this tool is targeted for use over the WWW, we believe it will provide us with a wealth of data that can be used to support ICWFA.

1 Introduction and Overview

Essentially, our work performed in Option Phase of project focused on extending algorithm developed previously and on preparing for future data collection efforts. As such, this document is partitioned accordingly, where the next section, Section 2, presents the enhancement of the previously developed algorithm and then the following section, Section 3, presents efforts undertaken to prepare for future data collection efforts.

2 Enhancement of Past Work - Algorithm Extension

2.1 MOUT Target Selection Algorithm

The algorithm developed in Phase I focused mainly on target selection. That is, given some combination of factors, this algorithm presented priority scheme generalized over a number of subject matter experts (SMEs). As evidenced in Figure 1, however, a number of other factors are



Figure 1. Relationships of WFA Factors

important in an individual combatant weapons firing algorithm (ICWFA). Information lacking in that initial algorithm but partially available in previously collected data include: percentages associated with dissimilar prioritization schemes, firing mode, aim points, and estimates of $P_{hit}/P_{kill}/P_{miss}$.

Data not available from initial effort was collected and may be seen in Appendix A. Appendix A is an output file from a prototype data acquisition tool we have developed named FiTKAT. This tool is detailed further in Section 3.0. These data must still be cleanly integrated into the existing algorithm. Once finished, this will provide fuzzy estimates of factors of target threat and a point estimate of SME's expectation of the probability of hitting the target given that set of factors.

2.2 Consideration of Human Science/Modeling Efforts

2.2.1 Review of Report: Squad Synthetic Environment Study (May, 2002)

We reviewed title report to see how it could be used in the ICWFA. While investigation collected a variety of data on a number of applicable scenarios involving SMEs interacting in a virtual environment, the data were presented in aggregate form and thus only minimally useful to our effort. For example, the engagement times reported include time to detect target, identify target, and shoot target. In algorithm being developed here, perceptual events (e.g., detection and identification) are beyond the scope of the effort. Of note, however, are data collected on aiming techniques and aim point, which complement the ICWFA. Also of note is the point made on future plans to statistically compare data in this report with data collected in the follow-on McKenna MOUT Study (see Section 2.2.2) to determine whether data collected in simulated environments can adequately represent data collected in live training exercises. The results of this comparison are of interest, as use of virtual environment to facilitate data collection is being considered for development of parts of the ICWFA.

2.2.2 Review of Report: McKenna MOUT Study (September, 2002)

We reviewed title report to see how it could be used in the ICWFA. While investigation collected a variety of data on a number of applicable scenarios, the data were presented in aggregate form and thus only minimally useful to our effort. For example, the engagement times reported include time to detect target, identify target, and shoot target. In algorithm being developed here, perceptual events (e.g., detection and identification) are beyond the scope of the effort. Of note, however, are data collected on aiming techniques and aim point, which complement the ICWFA. Also of note is the point made on future plans to statistically compare data in this report with data collected in the Squad Synthetic Environment Study (see Section 2.2.1) to determine whether data collected in simulated environments can adequately represent data collected in live training exercises. The results of this comparison are of interest, as a similar approach is being proposed for development of parts of the ICWFA.

3 Preparation for Future Work

3.1 Knowledge Acquisition Tool

By and large, the biggest part of our work has focused on the development of a Knowledge Acquisition Tool we can use to automate the knowledge acquisition process. Initial prototype of this tool is called First Target Knowledge Acquisition Tool (FiTKAT) and is developed around the target selection task. However, with bulk of work produced in prototype effort, the tool can be generalized to accommodate different types of data acquisition needs (e.g., fighting positions, Team Leader target assignments, use of suppressive fire, etc).

The user manual for this tool can be seen in Appendix B and the source code is available in Appendix C. These files are both provided electronically as well, and the latter is in executable form for any machine with Java interpreter installed. Because this tool is implemented in Java, our future plans include enabling access to it over the World Wide Web (WWW). Our vision is to be able to collect data over the internet from Soldiers and Marines who are currently deployed.

3.2 SME Arrangements and Future Data Collection Efforts

In preparation for Phase II, a number of contacts who could assist with data collection devices and/or acquiring SMEs were investigated. The most promising contacts include:

- COL. Charles Stanley and CAPT Carl Jacquet of Department of Military Instruction (DMI) at United States Military Academy (USMA), West Point. COL. Stanley and CAPT. Jacquet assisted in Phase I, by hosting us and providing SMEs for data collection efforts. Upon learning of our efforts to continue the project and collect additional data, COL. Stanley graciously invited us to return to DMI, USMA for further work. He added that his department has recently acquired a number of NCOs returning from Afghanistan and Iraq. We anticipate this to be a strong, reliable source of data for us.

- LTC. Rick Matthews (U.S. Army, Ret), now employed at PEO STRI and LTC. Joseph Giunta, PM for Ground Systems at PEO STRI have offered their assistance in securing equipment and SMEs at Ft. Bragg for use in our data collection efforts. Specifically, they have contacts that can help us to use the Ft. Bragg Simulation Lab's Engagement Skills Trainer 2000 (EST 2000) and get SMEs (both from Special Forces and from 82nd Airborne) to work with at Ft. Bragg. The EST 2000 (see Figure 2) is used to simulate weapons training events which leads to live-fire individual/crew weapon qualifications. It is used primarily as a unit/institutional, indoor, multipurpose, multiline, small arms, crew served and individual anti-tank training simulator. And, it provides initial and sustainment marksmanship training, static unit collective gunnery and tactical training, and Shoot/Don't Shoot training. It includes MOUT scenarios. Additionally, the



Figure 2. Engagement Skills Trainer 2000

Research Development and Engineering Command (RDECOM) Simulation and Training Technology Center (STTC) in Orlando currently has an EST 2000 for display. Thus, we can likely prepare for data collection locally in Orlando, and then only deploy for the actual data collection effort itself.

- Mr. Bill Pike of RDECOM-STTC is overseeing the development of a PC-based tactical decision trainer for newly commissioned lieutenants in the US Army Infantry School at Ft. Benning, GA. This low-cost, PC-based, rapid decision trainer, provides "virtual live-fire" training opportunity for all IOBC students to participate in a key leadership role as squad or platoon leader. Scenarios allow students to demonstrate their tactical knowledge by requiring them to make decisions in the virtual simulation that are similar to those required during a live-fire exercise. Further, the trainer allows single-player or multi-player participation. The project is creating basic scenarios for each range (Ware and Griswold) replicating the live-fire exercises currently being performed at IOBC and it tracks and evaluates the decisions made by the students as they participate in the virtual training event.

This document reports research undertaken at the U.S. Army Natick Soldier Research, Development and Engineering Center, Natick, MA, and has been assigned No. NATICK/TR-10/008 in a series of reports approved for publication.

References

1. Pike, W. and Hart, C. (2003). Infantry Officer Basic Course (IOBC) Rapid Decision Trainer (RDT). In Proceedings of 2003 Interservice/Industry Training, Simulation, and Education Conference.
2. Statkus, M., Dutoit, G., Garrett, A., Salvi, L., McMahon, R., Auer, R., Sampson, J., Short, P., and Middleton, V. (2002). Human Science/Modeling and Analysis Data: Squad Synthetic Environment Study, 23 April 2001 Through 4 May 2001. Technical Report NATICK/TR-02/015L, U.S. Army Soldier and Biological Chemical Command.
3. Statkus, M., Auer, R., Cronk, A., Garrett, A., McMahon, R., Middleton, V., Salvi, L., Sampson, J., Short, P., and Spring, R. (2002). Human Science/Modeling and Analysis Data: McKenna MOUT Study, 9 July 2001 Through 20 July 2001. Technical Report NATICK/TR-02/024L, U.S. Army Soldier and Biological Chemical Command.

Appendix A: SME Fuzzy Estimates on Factors and Estimates on Phit/Pmiss

Name: Rick Matthews
Rank: Officer (MAJ)

1: 43 or higher
2: Over 20
3: Platoon Leader
4: 2-5 years ago
5: No Response
6: 3-9
7: 100%
8: Situational Awareness
9: More than 6 years

FILENAME: group1_1.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	172		181	278	290
		152	170	284	291	0					0	0	0				
2	3			1	3	2						2	223		230	281	292
		211	227	288	311	211	227	311	343								
3	3			1	3	2						2	0	0	0	0	
		482	487	280	305	479	486	314	333								



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 0 0
AimPoint: miss

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group1_2.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	203		217	342	363
		195	231	358	422	195	223	418	502								
2	3			1	3	2						2	654		657	363	370
		652	660	374	406	0			0			0	0				



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 0 0
AimPoint: miss

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group1_3.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	1			3	2	1			3	313		325	293	308
		308	334	307	332	0		0	0	0				
2	2			3	2	1			3	408		415	304	313
		406	418	312	331	410	418	330	333					
3	3			1	3	2			2	579		603	233	262
		573	620	257	354	579	617	352	474					



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 0 0
AimPoint: miss

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group1_4.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	111	126	310	329
		107	136	329	363	0		0	0	0			
2	3			1	3	2			2	548	555	390	402
		556	583	395	407	595	618	397	405				
3	3			1	3	2			2	660	670	320	335
		654	685	332	386	655	679	385	452				



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 109 328
AimPoint: miss

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group1_5.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	277		284	323	335
		272	293	334	351	0		0	0	0				
2	3			1	3	2			5	0	0	0	0	
		534	566	371	381	574	587	374	379					
3	3			1	3	2			2	620		628	324	335
		617	638	334	370	619	634	369	415					



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 280 336
AimPoint: torso

After-Action Review

M4Mode:
Double Tap
p_hit:
80

FILENAME: group1_6.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	3			1	3	2						2	129		141	428	441
		125	151	437	452	136	158	460	478								
2	2			3	2	1						3	0	0	0	0	
		234	253	383	401	0				0		0		0			



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 137 436
AimPoint: head

After-Action Review

M4Mode:
Single Shot
p_hit:
50

FILENAME: group1_7.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	291	300	263	275
		281	306	269	285	278	303	292	314				
2	3			1	3	2			2	629	640	259	268
		624	642	267	285	627	641	290	315				



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 635 272
AimPoint: torso

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
50

FILENAME: group1_8.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	285		293	326	331
		280	331	297	345	285	298	355	377					
2	3			1	3	2			2	615		624	324	332
		608	629	331	345	616	630	349	370					



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 289 330
AimPoint: head

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
50

FILENAME: group2_1.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	343	362	194	219
		350	389	214	295	358	389	291	387				
2	3			1	3	2			2	427	435	245	254
		429	448	254	285	434	448	284	329				
3	3			1	3	2			2	600	615	215	234
		599	630	233	292	599	620	291	367				



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 433 253
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
30

FILENAME: group2_2.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	446		454	210	219
		446	461	218	245	449	462	244	281								
2	3			1	3	2						2	539		551	181	195
		537	561	197	247	540	561	247	310								



SME Target Selection Results

TargetSelected: 2

Coordinates of Shot: 543 208

AimPoint: torso

After-Action Review

M4Mode:

Double Tap

p_hit:

80

FILENAME: group2_3.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	384		388	199	206
		382	398	206	233	0				0		0	0				
2	3			1	3	2						2	467		479	170	187
		478	494	186	233	472	491	231	293								



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 384 211
AimPoint: torso

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
50

FILENAME: group3_1.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	569		582	259	272
		559	583	267	306	553	591	296	349					
2	3			1	3	2			2	0	0	0	0	0
		618	652	254	285	619	649	282	311					
3	3			1	3	2			2	0	0	0	0	0
		601	620	111	130	0			0		0			



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 576 263
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
50

FILENAME: group3_2.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	445		455	256	265
		442	462	263	285	441	458	280	312					
2	3			1	3	2			2	505		521	257	272
		499	525	272	300	492	529	296	347					
3	3			1	3	2			2	0	0	0	0	0
		538	555	116	132	0		0		0		0		



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 512 261
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
50

FILENAME: group3_3.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	0	0	0	0	
		324	343	279	307	328	355	307	341								
2	3			1	3	2						2	495		520	289	316
		493	551	308	364	495	544	352	440								
3	3			1	3	2						2	0	0	0	0	
		539	568	359	387	563	614	361	383								



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 522 331
AimPoint: torso

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group3_4.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	0	0	0	0	
		527	544	101	116	0					0		0				
2	3			1	3	2						2	521		536	285	300
		516	550	296	314	551	565	294	311								



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 531 293
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group4_1.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	462		472	419	430
		455	476	430	475	454	478	461	523								
2	3			1	3	2						2	501		513	412	428
		500	521	424	469	500	522	459	529								



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 505 422
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group4_2.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	185		194	304	313
		180	201	310	335	180	202	332	367								
2	3			1	3	2						2	286		293	306	312
		284	296	312	324	285	296	319	336								
3	3			1	3	2						2	641		652	302	314
		638	652	310	334	639	654	332	362								



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 192 320
AimPoint: torso

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
70

FILENAME: group4_3.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	0	0	0	0
		169	190	318	362	0			0		0		
2	3			1	3	2			2	0	0	0	0
		240	252	326	346	0			0		0		
3	3			1	3	2			2	0	0	0	0
		601	614	310	340	0			0		0		



SME Target Selection Results

TargetSelected: 1
Coordinates of Shot: 174 326
AimPoint: torso

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
70

FILENAME: group4_4.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	229		240	288	297
		233	263	292	310	240	268	303	329					
2	3			1	3	2			2	382		394	271	283
		381	399	279	304	384	398	296	329					
3	3			1	3	2			2	0	0	0	0	0
		523	536	268	282	0			0		0			
4	3			1	3	2			2	577		586	294	305
		572	594	303	325	576	593	322	339					



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 0 0
AimPoint: miss

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
70

FILENAME: group5_1.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	416		427	314	326
		412	437	323	354	417	436	350	368					
2	3			1	3	2			2	502		510	301	312
		498	512	310	334	504	515	331	363					



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 503 308
AimPoint: head

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
70

FILENAME: group5_2.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	245		252	278	286
		242	258	283	307	245	258	302	334								
2	3			1	3	2						2	583		592	413	422
		578	599	419	446	582	603	445	486								



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 591 424
AimPoint: torso

After-Action Review

M4Mode:
Three-Round Burst
p_hit:
80

FILENAME: group5_3.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	2			3	2	1						3	0	0	0	0	
		388	406	384	403	0					0		0				
2	3			1	3	2						2	466		487	396	417
		469	516	410	449	468	514	443	498								



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 477 415
AimPoint: head

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group5_4.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	147	173	280	309
		141	207	306	405	157	206	404	520				
2	3			1	3	2			2	567	588	299	319
		565	605	310	380	567	608	372	442				
3	3			1	3	2			2	736	746	254	265
		729	753	266	283	0			0	0	0		



SME Target Selection Results

TargetSelected: 2
Coordinates of Shot: 577 320
AimPoint: torso

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group5_5.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	247	261	459	470	
		242	270	466	501	247	270	499	537					
2	3				1	3	2			2	350	366	437	448
		0	0	0	0	0	0	0	0					
3	3				1	3	2			2	610	626	447	465
		605	640	463	519	605	636	512	571					



SME Target Selection Results

TargetSelected: 3
Coordinates of Shot: 614 470
AimPoint: torso

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group5_6.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax	l_ymin	l_ymax
--------	----------	------------	----------	---------	--------------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

1	3			1	3	2						2	48		71	323	345
		41	89	340			414			48	87	410		484			
2	3			1	3	2						2	237		253	274	287
		228	257	287	305	0					0		0				
3	3			1	3	2						2	782		811	320	350
		760	834	339	449	752	813	426	567								



SME Target Selection Results

TargetSelected: none
Coordinates of Shot: 0 0
AimPoint: miss

After-Action Review

M4Mode:
Double Tap
p_hit:
100

FILENAME: group5_7.jpg

Target	Exposure	Visibility	Distance	Posture	FiringStatus	h_xmin		
h_xmax	h_ymin	h_ymax	t_xmin	t_xmax	t_ymin	t_ymax	l_xmin	l_xmax
l_ymin	l_ymax							

1	2			3	2	1			3	208	224	301	312
		0	0	0	0	0	0	0					
2	3			1	3	2			2	267	278	288	300
		265	286	298	334	268	286	328	365				



SME Target Selection Results

TargetSelected: none

Coordinates of Shot: 50 342

AimPoint: miss

After-Action Review

M4Mode:

Three-Round Burst

p_hit:

80

null

Appendix B: FiTKAT User Manual

First Target Knowledge Acquisition Tool (FiTKAT) User Manual

Section 1 – Installing and Running FiTKAT

1. Ensure that you are running an OS that has a Java run-time environment installed. You can check to see if this is the case by typing “java –version” at a terminal or DOS command prompt.
2. If you wish to run this tool from your hard drive (and not from the included CD-ROM), ensure that you extract the following files to an empty directory. Each of these files are included in the self-extracting zip file fitkat.exe
 - *start.bat*
 - *MainApp.class*
 - *UserInfoWindow.class*
 - *TargetWindow.class*
 - *AfterActionReviewWindow.class*
 - *image_params.dat*
 - *all .jpg files referred to within image_params.dat (group1_1.jpg ...)*
3. Ensure that your desktop resolution is at least 1152x864 pixels to ensure FiTKAT can fully display each window
4. If you have edited the *image_params.dat* file (see Section 3), ensure that each image file listed exists in the /bin directory. Also, check the file to make sure each listing adheres to the correct format as shown in the provided *image_params.dat* file.
5. From the /bin directory of your CD-ROM, (or from the empty directory created on your hard drive) execute 'start.bat'.

Section 2 – Working with FiTKAT

FiTKAT includes three knowledge-acquisition windows that accept user data: the questionnaire, drill section, and after-action review section. The SME can navigate through these windows using a standard 2-button mouse and keyboard.

Section 2.1 - SME Questionnaire

The opening window displayed by FiTKAT is a questionnaire that asks the SME basic questions about his military background. The SME is not required to provide answers to any of these questions; however he is encouraged to submit a complete survey. To complete this questionnaire, navigate the mouse to the answer blanks for each question. For text answers, click in the white space provided and type in your response. For multiple-choice questions, simply click on the blank button indicating your answer. If a question permits multiple answers, your selections will display as check marks instead of filled-in circles. Use the scroll bar on the right-hand side of the window to navigate to the latter questions in the survey. When you are finished, click the 'Submit' button to proceed to the drill section of FiTKAT.

First Target Acquisition Tool - User Info

Name
SME

Rank
Major

1. How old are you?

☐ 19-25

☐ 26-30

☒ 31-36

☐ 37-42

☐ older than 42

2. How many years have you served in the military?

☐ 1-4

☐ 5-9

☒ 10-14

☐ 15-20

☐ more than 20

3. What infantry positions/roles in unit have you held? (check all that apply)

☐ Platoon Leader

☒ Squad Leader

☐ Fireteam Leader

☒ Platoon Sergeant

☒ Rifleman

☐ Sniper

☐ Machine Gunner

☐ SAW Gunner

☐ Grenadier

☒ Other

other positions

4. When is the last time you participated in training or evaluation of

Infantry Drill Training

Figure B1 – The Questionnaire Window

Section 2.2 - Drill Section

For the drill section, FiTKAT will display a list of images in the order that they are provided in the *image_params.dat* file (see Section 3). When an image appears, use the left-mouse button to fire on the target that you would most likely fire upon first if presented the situation in real life. Be sure to click (only once) on the target at the precise point on his body where you would most likely fire. After you have made a selection, refrain from clicking until a new image has loaded into the drill window.



Figure B2 –The Drill Window

Section 2.3 – The After-Action Review Window

After FiTKAT has displayed each image, the After-Action Review window instantiates. Here, each image will be displayed a second time with a set of crosshairs indicating where the user has fired. Below the image is a set of questions designed to probe the SME for more information regarding his selection. Use the mouse and keyboard to fill in the appropriate responses. Feel free to leave questions blank if you wish. When you have completed your responses for the image displayed, click the 'Submit' button to proceed to the next image.

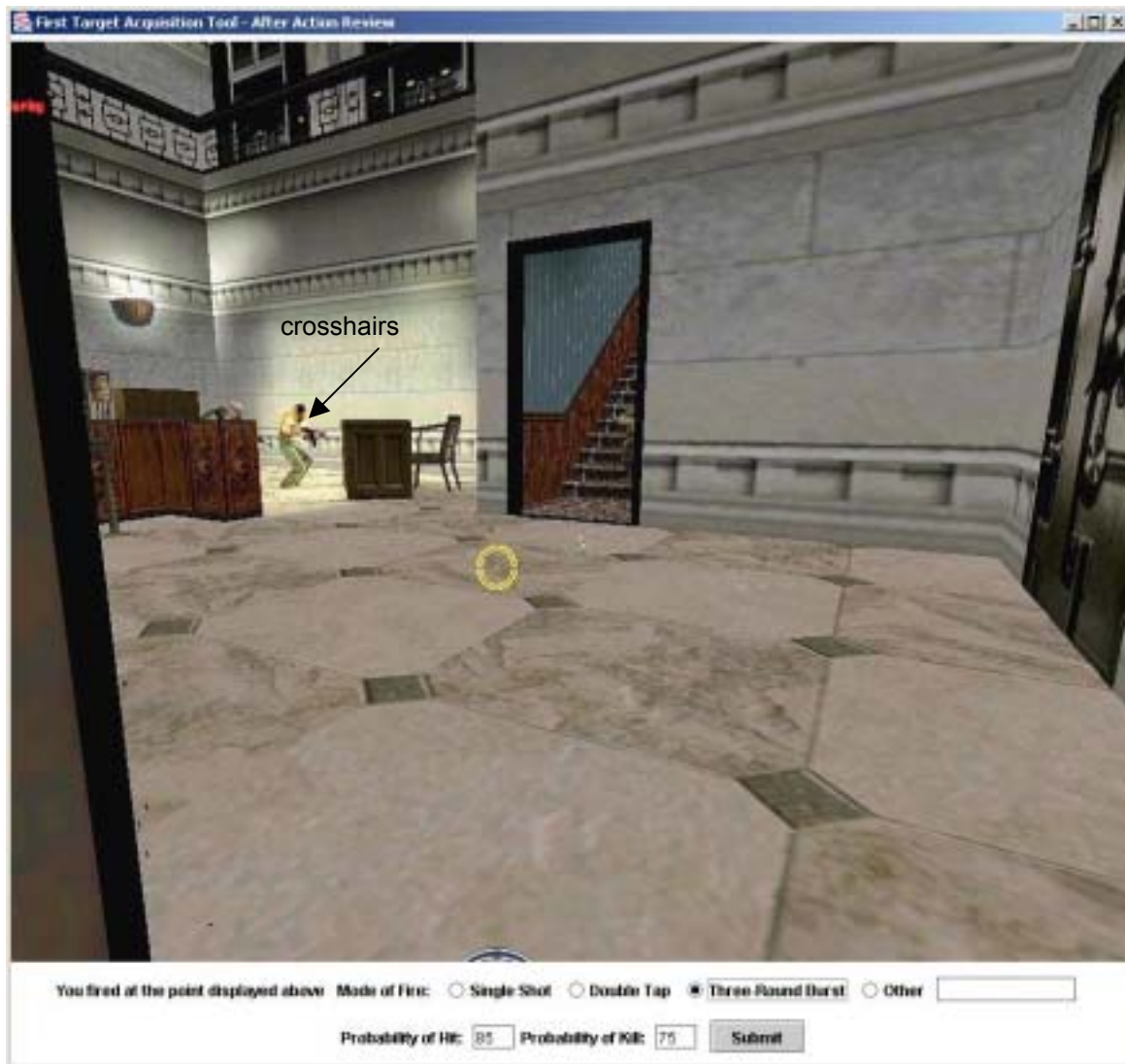
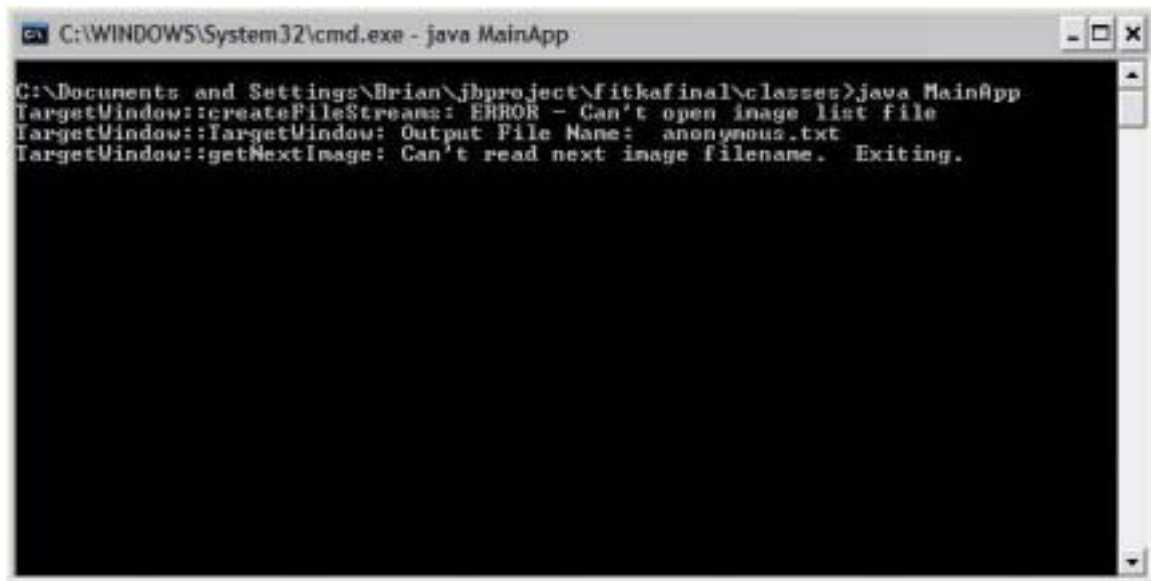


Figure B3 – The After-Action Review Window

Section 3 – The Console Window

During execution, a console window in which you executed *start.bat* will be displaying messages about the current run. If you encounter a problem while running FiTKAT, check this window to see if any error messages have printed. If an error has occurred during your run, the reason for that error will likely be displayed in this window.

In Figure B4, FiTKA has executed without the *image_params.dat* file. The program is able to execute the user survey, but when it begins loading images for the drill section it runs into the error above. The *TargetWindow::createFileStreams* text represents the class and method in which the error has occurred.



```
C:\WINDOWS\System32\cmd.exe - java MainApp
C:\Documents and Settings\Brian\jbproject\fitkafinal\classes>java MainApp
TargetWindow::createFileStreams: ERROR - Can't open image list file
TargetWindow::TargetWindow: Output File Name: anonymous.txt
TargetWindow::getNextImage: Can't read next image filename. Exiting.
```

Figure B4 – The Console Window and an Example Error Message

Section 4 – Modifying Target Selection Images with *image_params.dat*

The *image_params.dat* file contains both a listing of each image to be displayed by FiTKAT along with the coordinates of each target depicted by those images. More specifically, the locations of the head, torso, and legs of target are each identified within *image_params.dat* as a 4-tuple that defines a coordinate 'box' around that region (see figure B5 below). These box coordinates are actually pixel positions within the image file. In figure B5, for instance, the coordinate (t_xmin, t_ymin) represents the top-left vertex of the blue torso box, whereas (t_xmax, t_ymax) represents the bottom-right vertex.

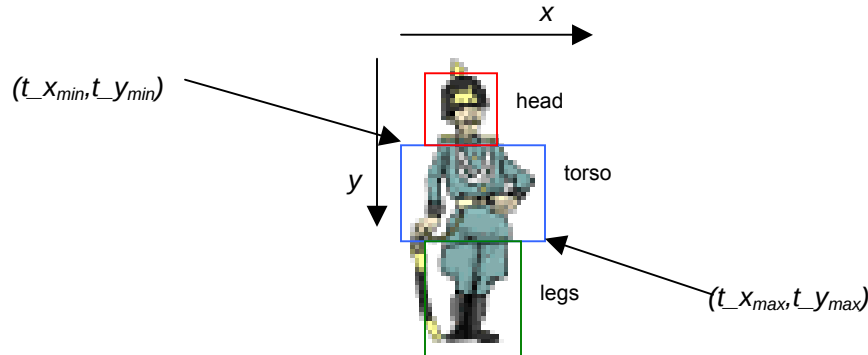


Figure B5 – Defining the Body-Part Regions of a Target with 4-Tuple 'Boxes'

In addition to these positions, *image_params.dat* also contains fields for the visibility, distance, exposure, posture, and firing status of the target in relation to the environment in which he is seen. In the provided *image_params.dat*, each field takes on a value between 1 and 3 based on the degree to which each target portrays the trait.

The fields are an attempt to portray a complete description of each target's threat within a given image. When the subject-matter expert selects a target, that selection can be mapped to the values of these fields in relation to those of the non-selected targets. As the FiTKAT program does not do any internal computations with these values, the values can be modified to span any desired numerical range or orientation.

To add an image to be displayed by FiTKAT during the drill exercise and AAR, simply add the filename and image information to *image_params.dat* using the exact format used in the provided file. Note that you can designate where in the sequence this image is displayed, as FiTKAT reads and presents each image in the exact order it is presented by *image_params.dat*.



Figure B6 – A Sample Slide from the Drill Section

image_params - Notepad

File Edit Format View Help

```

-----
1      2      3      2      1      3      203      217      342
2      3      1      3      2      2      654      657      363
-----
FILENAME: group1_3.jpg
*****
Target Exposure visibility distance Posture Firingstatus h_xmin h_xmax h_ymin
-----
1      1      2      2      2      2      313      325      293
2      3      2      2      2      2      408      415      304
3      1      2      2      3      3      579      603      233
-----

```

Figure B7 – Defining the Targets and their Parameters

When adding an image, it is necessary to include the locations of each target along with the image name. To identify these positions using the FiTKAT tool, first add the image name as the first image to be displayed. Provide values for visibility, posture, etc. (for each target) and enter 0 for all coordinate values needed. Figure B7 is an example *image_params.dat* file with the parameters of image *group1_3.jpg* shown. Figure B6 is an illustration of this image with each of its three targets identified. In Figure B7, note that a value for the exposure, visibility, etc. is assigned to each of the three targets – for example, target 2 is assigned visibility = 3 (most visible) whereas target 3 is given visibility = 2.

After these values have been assigned for each target, execute FiTKAT and proceed to the image you have added. Right-click on the upper-left and lower-right boundaries of the boxes you wish to create for each region of your targets. Note that after each click, a set of coordinates will show up in the FiTKAT console window. The coordinates of the upper-left region will correspond to x_{min} and y_{min} for the appropriate region of the target you are defining, while the lower-right coordinates translate to its x_{max} and y_{max} values. These values can be plugged into the appropriate spaces within *image_params.dat*.

4-tuple for head region of target 1

Target	Exposure	visibility	distance	Posture	Firingstatus	h_xmin	h_xmax	h_ymin	h_ymax	t_xmin	t_ymax
1	1	2	2	2	2	313	325	293	308	308	3
2	1	3	2	2	2	408	413	384	311	408	4
3	1	2	2	2	2	579	603	233	262	573	6

Figure B8 – The 4-Tuple Boxes in *image_params.dat*

Section 5 – Retrieving Knowledge Obtained from FiTKAT

Each time the FiTKAT application is run, a text file is created and saved as **username*.txt* (where **username** is the name provided by the SME in the questionnaire). At the beginning of this file are the responses that user submitted in the leading questionnaire. What follows is a copy of the *image_params.dat* file with added values containing the coordinates and body positions of the shot, along with the responses given to the after-action review for each image.

```
SME example - Notepad
File Edit Format Help
Name: SME example
Rank: none
-----
1: 19-25
2: No Response
3: Platoon Sergeant Rifleman
4: No Response
5: No Response
6: No Response
7: No Response
8: Situational Awareness
9: No Response
*****
FILENAME: group1_3.jpg
*****
target exposure visibility distance posture #firingstatus h_xmin h_xmax t
1 2 3 2 1 3 172 181
2 3 1 3 2 2 223 230
3 3 1 3 2 2 0 0
-----
SME Target Selection Results
-----
TargetsSelected: 2
Coordinates of shot: 218 295
AimPoint: torso
After-Action Review
-----
MAMode:
Single Shot
p_hits:
50
p_kill:
75
-----
FILENAME: group1_2.jpg
*****
target exposure visibility distance posture #firingstatus h_xmin h_xmax t
1 2 3 2 1 3 203 217
2 3 1 3 2 2 654 657
-----
```

user responses from questionnaire

copied from *image_params.dat*

user responses from drill window

user responses from after-action review

Figure B7 – *username*.txt

Appendix C: FiTKAT Souce Code

MainApp.java

```
/*
 * Title:      MainApp
 * Description: This is the FiTKAT main application. It
 *              calls and sizes each window to be used
 *              (UserInfoWindow, TargetWindow,
 *              AfterActionReviewWindow)
 * Author:     Soar Technology, Inc.
 * Version:    1.1
 * Date:       11.24.03
 */

import javax.swing.*.*;
import java.awt.event.*;

public class MainApp
{
    public static void main(String [] args)
    {
        // create the user info window and define size and orientation
        UserInfoWindow infoWindow = new UserInfoWindow("FiTKAT - User
Info");
        infoWindow.setSize(900, 1050);
        infoWindow.setLocation(300, 50);
        infoWindow.setVisible(true);

        // hold until info window is closed
        while (infoWindow.isVisible()) {}

        // create the first target window
        TargetWindow targetWindow = new TargetWindow("FiTKAT - Drill
Excercise", infoWindow.getIoFileName());
        targetWindow.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
        targetWindow.setVisible(true);

        // hold until window closed
        while (targetWindow.isVisible()) {}

        // exit program if file error occurred
        if (targetWindow.didRunCorrectly() == false)
            System.exit(0);

        // create the after action review window
        AfterActionReviewWindow aarWindow = new AfterActionReviewWindow
            ("FiTKAT - After Action Review", infoWindow.getIoFileName());

        aarWindow.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
    }
}
```

```
    }  
  });  
  
  // hold until window closed  
  while (aarWindow.isVisible()) {}  
  
  System.exit(0);  
}  
}
```

UserInfoWindow.java

```
/*
 * Title:      UserInfoWindow
 * Description: This is the Window JFrame that contains the
 *              user survey. This Frame is responsible for
 *              displaying the survey and recording user
 *              responses by writing them to a pre-existing
 *              file containing the list and description
 *              of each image to be displayed.
 * Copyright:  Copyright (c) 2003
 * Author:     Soar Technology, Inc.
 * Version:    1.1
 * Date:       11.24.03
 */

import java.awt.*;
import javax.swing.*;
import java.io.*;
import java.awt.event.*;

public class UserInfoWindow extends JFrame implements ActionListener,
ItemListener
{
    //JFrame stuff
    Container contentPane;
    UserInfoWindowPanel panel, column1panel, column2panel;

    // file names and streams
    FileWriter ioFileStream;
    static String imageInfoFileName = "image_params.dat";
    String ioFileName;

    // buttons, checkboxes and textfields
    JTextField userName;
    JTextField userRank;

    ButtonGroup question1answers;
    JRadioButton q1a, q1b, q1c, q1d, q1e;
    ButtonGroup question2answers;
    JRadioButton q2a, q2b, q2c, q2d, q2e;
    JCheckBox q3a, q3b, q3c, q3d, q3e, q3f, q3g, q3h, q3i, q3j;
    JTextField q3jText;
    ButtonGroup question4answers;
    JRadioButton q4a, q4b, q4c, q4d;
    ButtonGroup question5answers;
    JRadioButton q5a, q5b, q5c, q5d;
    ButtonGroup question6answers;
    JRadioButton q6a, q6b, q6c, q6d;
    ButtonGroup question8answers;
    JTextField q7Text;
    JRadioButton q8a, q8b, q8c, q8d;
    JTextField q8dText;
    ButtonGroup question9answers;
    JRadioButton q9a, q9b, q9c, q9d;

    JButton submitButton;
```

```

/* Method: UserInfoWindow
   Description: This is the Constructor for UserInfoWindow. It creates
the frame along with
               the scrolling feature and it calls the function to
display each question and
               answer field
*/
UserInfoWindow(String title)
{
    // call constructor for parent class
    super(title);

    // create frame and add content panel
    contentPane = getContentPane();
    panel = new UserInfoWindowPanel();

    contentPane.add(panel);

    // add scroll pane for frame
    JScrollPane jscrollpane = new JScrollPane(panel,
ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS,
ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
    contentPane.add(jscrollpane);

    // add question and answer fields to panel
    addFields();
}

/* Method: addFields
   Description: This method adds each question and answer field to the
panel to be displayed.
*/
private void addFields()
{
    // create a layout for the questions

    panel.setLayout(new GridLayout(1,2));
    column1panel = new UserInfoWindowPanel();
    column1panel.setLayout(new GridLayout(30,0));
    column2panel = new UserInfoWindowPanel();
    column2panel.setLayout(new GridLayout(30,0));

    // Name and Rank fields
    UserInfoWindowPanel namepanel = new UserInfoWindowPanel();
    GridLayout nameRankLayout = new GridLayout(2,2);
    namepanel.setLayout(nameRankLayout);

    JLabel userNameLabel = new JLabel("Name");
    userName = new JTextField(10);
    namepanel.add(userNameLabel);
    namepanel.add(userName);

    UserInfoWindowPanel rankpanel = new UserInfoWindowPanel();
    rankpanel.setLayout(nameRankLayout);
    JLabel userRankLabel = new JLabel("Rank");
    userRank = new JTextField(10);
    rankpanel.add(userRankLabel);
    rankpanel.add(userRank);
}

```

```

column1panel.add(namepanel);
column1panel.add(rankpanel);

// question 1
JLabel question1 = new JLabel("1.  How old are you?");
column1panel.add(question1);

question1answers = new ButtonGroup();
q1a = new JRadioButton("19-25");
q1a.setBackground(Color.white);
q1b = new JRadioButton("26-30");
q1b.setBackground(Color.white);
q1c = new JRadioButton("31-36");
q1c.setBackground(Color.white);
q1d = new JRadioButton("37-42");
q1d.setBackground(Color.white);
q1e = new JRadioButton("older than 42");
q1e.setBackground(Color.white);
// add each answer to buttongroup
question1answers.add(q1a);
question1answers.add(q1b);
question1answers.add(q1c);
question1answers.add(q1d);
question1answers.add(q1e);
// add item listener for each button in group
q1a.addItemListener(this);
q1b.addItemListener(this);
q1c.addItemListener(this);
q1d.addItemListener(this);
q1e.addItemListener(this);

column1panel.add(q1a);
column1panel.add(q1b);
column1panel.add(q1c);
column1panel.add(q1d);
column1panel.add(q1e);

// question 2
JLabel question2 = new JLabel("2.  How many years have you served in
the military?");
column1panel.add(question2);

question2answers = new ButtonGroup();
q2a = new JRadioButton("1-4");
q2a.setBackground(Color.white);
q2b = new JRadioButton("5-9");
q2b.setBackground(Color.white);
q2c = new JRadioButton("10-14");
q2c.setBackground(Color.white);
q2d = new JRadioButton("15-20");
q2d.setBackground(Color.white);
q2e = new JRadioButton("more than 20");
q2e.setBackground(Color.white);
question2answers.add(q2a);
question2answers.add(q2b);
question2answers.add(q2c);
question2answers.add(q2d);
question2answers.add(q2e);
q2a.addItemListener(this);

```

```

q2b.addItemListener(this);
q2c.addItemListener(this);
q2d.addItemListener(this);
q2e.addItemListener(this);

column1panel.add(q2a);
column1panel.add(q2b);
column1panel.add(q2c);
column1panel.add(q2d);
column1panel.add(q2e);

// question 3
JLabel question3 = new JLabel("3. What infantry positions/roles in
unit have you held? (check all that apply)");
column1panel.add(question3);

q3a = new JCheckBox("Platoon Leader");
q3a.setBackground(Color.white);
q3b = new JCheckBox("Squad Leader");
q3b.setBackground(Color.white);
q3c = new JCheckBox("Fireteam Leader");
q3c.setBackground(Color.white);
q3d = new JCheckBox("Platoon Sergeant");
q3d.setBackground(Color.white);
q3e = new JCheckBox("Rifleman");
q3e.setBackground(Color.white);
q3f = new JCheckBox("Sniper");
q3f.setBackground(Color.white);
q3g = new JCheckBox("Machine Gunner");
q3g.setBackground(Color.white);
q3h = new JCheckBox("SAW Gunner");
q3h.setBackground(Color.white);
q3i = new JCheckBox("Grenadier");
q3i.setBackground(Color.white);
q3j = new JCheckBox("Other");
q3j.setBackground(Color.white);
q3jText = new JTextField(20);

q3a.addItemListener(this);
q3b.addItemListener(this);
q3c.addItemListener(this);
q3d.addItemListener(this);
q3e.addItemListener(this);
q3f.addItemListener(this);
q3g.addItemListener(this);
q3h.addItemListener(this);
q3i.addItemListener(this);
q3j.addItemListener(this);

column1panel.add(q3a);
column1panel.add(q3b);
column1panel.add(q3c);
column1panel.add(q3d);
column1panel.add(q3e);
column1panel.add(q3f);
column1panel.add(q3g);
column1panel.add(q3h);
column1panel.add(q3i);

```

```

UserInfoWindowPanel q3jpanel = new UserInfoWindowPanel();
q3jpanel.setLayout(new FlowLayout(FlowLayout.LEADING));
q3jpanel.add(q3j);
q3jpanel.add(q3jText);
column1panel.add(q3jpanel);

// question 4
JLabel question4 = new JLabel("4. When is the last time you
participated in training or evaluation of");
column1panel.add(question4);

JLabel question4b = new JLabel("Infantry Battle Drills?");
column1panel.add(question4b);

question4answers = new ButtonGroup();
q4a = new JRadioButton("Never");
q4a.setBackground(Color.white);
q4b = new JRadioButton("Less than 2 years ago");
q4b.setBackground(Color.white);
q4c = new JRadioButton("2-5 years ago");
q4c.setBackground(Color.white);
q4d = new JRadioButton("More than 6 years ago");
q4d.setBackground(Color.white);
question4answers.add(q4a);
question4answers.add(q4b);
question4answers.add(q4c);
question4answers.add(q4d);
q4a.addItemListener(this);
q4b.addItemListener(this);
q4c.addItemListener(this);
q4d.addItemListener(this);

column1panel.add(q4a);
column1panel.add(q4b);
column1panel.add(q4c);
column1panel.add(q4d);

// question 5
JLabel question5 = new JLabel("5. When is the last time you
participated in real-world execution of");
column2panel.add(question5);

JLabel question5b = new JLabel("Infantry Battle Drills?");
column2panel.add(question5b);

question5answers = new ButtonGroup();
q5a = new JRadioButton("Never");
q5a.setBackground(Color.white);
q5b = new JRadioButton("Less than 2 years ago");
q5b.setBackground(Color.white);
q5c = new JRadioButton("2-5 years ago");
q5c.setBackground(Color.white);
q5d = new JRadioButton("More than 6 years ago");
q5d.setBackground(Color.white);
question5answers.add(q4a);
question5answers.add(q4b);
question5answers.add(q4c);
question5answers.add(q4d);
q5a.addItemListener(this);
q5b.addItemListener(this);

```

```

q5c.addItemListener(this);
q5d.addItemListener(this);

column2panel.add(q5a);
column2panel.add(q5b);
column2panel.add(q5c);
column2panel.add(q5d);

// question 6
JLabel question6 = new JLabel("6.  How many real fire fights have
you been in?");
column2panel.add(question6);

question6answers = new ButtonGroup();
q6a = new JRadioButton("None");
q6a.setBackground(Color.white);
q6b = new JRadioButton("Less than 3");
q6b.setBackground(Color.white);
q6c = new JRadioButton("3-9");
q6c.setBackground(Color.white);
q6d = new JRadioButton("10 or more");
q6d.setBackground(Color.white);
question6answers.add(q6a);
question6answers.add(q6b);
question6answers.add(q6c);
question6answers.add(q6d);
q6a.addItemListener(this);
q6b.addItemListener(this);
q6c.addItemListener(this);
q6d.addItemListener(this);

column2panel.add(q6a);
column2panel.add(q6b);
column2panel.add(q6c);
column2panel.add(q6d);

// question 7
JLabel question7 = new JLabel(
    "7.  What percentage of those fights were successful? ");
q7Text = new JTextField(5);
q7Text.setText("%");

UserInfoWindowPanel q7panel = new UserInfoWindowPanel();
q7panel.setLayout(new FlowLayout(FlowLayout.LEADING));
q7panel.add(question7);
q7panel.add(q7Text);

column2panel.add(q7panel);

// question 8
JLabel question8 = new JLabel(
    "8.  What would you consider the most important aspect of
executing");
column2panel.add(question8);

JLabel question8b = new JLabel("        Infantry Battle Drills?");
column2panel.add(question8b);

question8answers = new ButtonGroup();
q8a = new JRadioButton("Quick Response");

```

```

q8a.setBackground(Color.white);
q8b = new JRadioButton("Teamwork");
q8b.setBackground(Color.white);
q8c = new JRadioButton("Situational Awareness");
q8c.setBackground(Color.white);
q8d = new JRadioButton("Other");
q8d.setBackground(Color.white);
q8dText = new JTextField(20);
question8answers.add(q8a);
question8answers.add(q8b);
question8answers.add(q8c);
question8answers.add(q8d);
q8a.addItemListener(this);
q8b.addItemListener(this);
q8c.addItemListener(this);
q8d.addItemListener(this);

column2panel.add(q8a);
column2panel.add(q8b);
column2panel.add(q8c);

UserInfoWindowPanel q8dpanel = new UserInfoWindowPanel();
q8dpanel.setLayout(new FlowLayout(FlowLayout.LEADING));
q8dpanel.add(q8d);
q8dpanel.add(q8dText);
column2panel.add(q8dpanel);

// question 9
JLabel question9 = new JLabel(
    "9. How many years of combat training have you received?");
column2panel.add(question9);

question9answers = new ButtonGroup();
q9a = new JRadioButton("None");
q9a.setBackground(Color.white);
q9b = new JRadioButton("Less than 2 years");
q9b.setBackground(Color.white);
q9c = new JRadioButton("2-5 years");
q9c.setBackground(Color.white);
q9d = new JRadioButton("6 years or more");
q9d.setBackground(Color.white);
question9answers.add(q9a);
question9answers.add(q9b);
question9answers.add(q9c);
question9answers.add(q9d);
q9a.addItemListener(this);
q9b.addItemListener(this);
q9c.addItemListener(this);
q9d.addItemListener(this);

column2panel.add(q9a);
column2panel.add(q9b);
column2panel.add(q9c);
column2panel.add(q9d);

// submitButton
submitButton = new JButton("Submit");
submitButton.setBackground(Color.white);
submitButton.addActionListener(this);

```

```

        column2panel.add(submitButton);
        panel.add(column1panel);
        panel.add(column2panel);

    }

    /* Method: writeData
       Description:  this is a function to write a given string to the file
                    corresponding to the userDataFile buffer
    */
    private void writeData(String data)
    {
        try {
            ioFileStream.write(data);
        }
        catch (Exception E) {
            System.out.println("UserInfoWindow::writeData: ERROR - Can't write
to file");
        }
    }

    /* Method: getFileName
       Description: this retrieves the fileName
    */
    public String getIoFileName()
    {
        return ioFileName;
    }

    /* Method: actionPerformed
       Description: This function is required for a frame to implement
       ActionListener.
                    It fires when the submit button is pressed and records
all of the
                    answers indicated at the time of submittal to the i/o
file
    */
    public void actionPerformed(ActionEvent e)
    {
        /* when submit button is pressed, create a new file called and write
the user
        survey answers to that file.  Name the file *username*.txt */
        if (e.getSource() == submitButton) {
            try
            {
                if (userName.getText().length()==0) ioFileName =
"anonymous.txt";
                else ioFileName = userName.getText() + ".txt";

                ioFileStream = new FileWriter(userName.getText() + ".txt");
            }
            catch (Exception E) {
                System.out.println("UserInfoWindow::actionPerformed: ERROR -
Can't open output file");
            }

            // name and rank
            writeData("Name: ");
            writeData(userName.getText() + "\r\n");
            writeData("Rank: ");

```

```

writeData(userRank.getText() + "\r\n");
writeData("-----\r\n");

// question 1
writeData("1: ");
if (q1a.isSelected())
    writeData("19-25 \r\n");
else if (q1b.isSelected())
    writeData("26-30 \r\n");
else if (q1c.isSelected())
    writeData("31-36 \r\n");
else if (q1d.isSelected())
    writeData("37-42 \r\n");
else if (q1e.isSelected())
    writeData("43 or higher \r\n");
else writeData("No Response\r\n");

// question 2
writeData("2: ");
if (q2a.isSelected())
    writeData("1 to 4 \r\n");
else if (q2b.isSelected())
    writeData("5 to 9 \r\n");
else if (q2c.isSelected())
    writeData("10 to 14 \r\n");
else if (q2d.isSelected())
    writeData("15 - 20 \r\n");
else if (q2e.isSelected())
    writeData("Over 20 \r\n");
else writeData("No Response\r\n");

// question 3
writeData("3: ");
if (q3a.isSelected())
    writeData("Platoon Leader ");
if (q3b.isSelected())
    writeData("Squad Leader ");
if (q3c.isSelected())
    writeData("Fireteam Leader ");
if (q3d.isSelected())
    writeData("Platoon Sergeant ");
if (q3e.isSelected())
    writeData("Rifleman ");
if (q3f.isSelected())
    writeData("Sniper ");
if (q3g.isSelected())
    writeData("Machine Gunner ");
if (q3h.isSelected())
    writeData("SAW Gunner ");
if (q3i.isSelected())
    writeData("Grenadier ");
if (q3j.isSelected())
    writeData(q3jText.getText());
if (q3a.isSelected()==false && q3b.isSelected()==false &&
q3c.isSelected()==false &&
    q3d.isSelected()==false && q3e.isSelected()==false &&
q3f.isSelected()==false &&
    q3g.isSelected()==false && q3h.isSelected()==false &&
q3i.isSelected()==false &&
    q3j.isSelected()==false)

```

```

        writeData("No Response");

writeData("\r\n");

// question 4
writeData("4: ");
if (q4a.isSelected())
    writeData("Never \r\n");
else if (q4b.isSelected())
    writeData("Less than 2 years ago \r\n");
else if (q4c.isSelected())
    writeData("2-5 years ago \r\n");
else if (q5c.isSelected())
    writeData("More than 6 years ago \r\n");
else writeData("No Response\r\n");

// question 5
writeData("5: ");
if (q5a.isSelected())
    writeData("Never \r\n");
else if (q5b.isSelected())
    writeData("Less than 2 years ago \r\n");
else if (q5c.isSelected())
    writeData("2-5 years ago \r\n");
else if (q6c.isSelected())
    writeData("More than 6 years ago \r\n");
else writeData("No Response\r\n");

// question 6
writeData("6: ");
if (q4a.isSelected())
    writeData("None \r\n");
else if (q4b.isSelected())
    writeData("Less than 3 \r\n");
else if (q4c.isSelected())
    writeData("3-9 \r\n");
else if (q5c.isSelected())
    writeData("10 or more \r\n");
else writeData("No Response\r\n");

writeData("7: ");
if (q7Text.getText().charAt(0) == '%')
    writeData("No Response\r\n");
else writeData(q7Text.getText() + "\r\n");

// question 8
writeData("8: ");
if (q8a.isSelected())
    writeData("Quick Response \r\n");
else if (q8b.isSelected())
    writeData("Teamwork \r\n");
else if (q8c.isSelected())
    writeData("Situational Awareness \r\n");
else if (q8d.isSelected()) {
    writeData("Other: ");
    writeData(q8dText.getText() + "\r\n");
}
else writeData("No Response\r\n");

// question 9

```

```

        writeData("9: ");
        if (q9a.isSelected())
            writeData("None \r\n");
        else if (q9b.isSelected())
            writeData("Less than 2 years \r\n");
        else if (q9c.isSelected())
            writeData("2-5 years \r\n");
        else if (q9d.isSelected())
            writeData("More than 6 years \r\n");
        else writeData("No Response\r\n");

        writeData("*****\r\n \r\n");

        try
        {
            ioFileStream.close();
        }
        catch (Exception E)
        {
            System.out.println("UserInfoWindow::actionPerformed: ERROR -
Can't close file");
        }
        this.setVisible(false);
    }
}

    public void itemStateChanged(ItemEvent e) {}
}

class UserInfoWindowPanel extends JPanel
{
    UserInfoWindowPanel()
    {
        setBackground(Color.white);
    }
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        // write stuff in panel
    }
}

```

TargetWindow.java

```
/*
 * Title:          TargetWindow
 * Description:    This is the Window JFrame that displays
 *                each image. In addition, this Frame
 *                is responsible for recording user target
 *                selections for each image by writing
 *                them to a pre-existing file containing the
 *                results of the user survey along with the
 *                list and description of each image to be
 *                displayed.
 * Author:         Soar Technology, Inc.
 * Version:        1.1
 * Date:           11.24.03
 */

import java.awt.*;
import javax.swing.*;
import javax.sound.sampled.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.io.RandomAccessFile.*;

public class TargetWindow extends JFrame implements MouseListener
{
    // constants
    static int MAX_TARGETS = 10;
    static int CHAR_TO_INT_OFFSET = -48;
    static int NUM_BODY_AREAS = 3;
    static int head = 0;
    static int torso = 1;
    static int legs = 2;
    static int miss = 3;

    // jpanel
    FirstTargetWindowPanel panel;
    JLabel theImageLabel;
    Toolkit theToolkit;
    Container contentPane;

    // text fields
    String tempStr;
    String theImageName;
    String outputFileName;
    static String targetInfoFileName = "image_params.dat";

    // file streams
    RandomAccessFile outputFileStream;
    BufferedReader targetInfoStream;

    // internal vars
    int imageCount;
    int numTargets;
    int targetCount;
    int targetSelected;
    int aimPoint; // 0=head, 1=torso, 2=legs
}
```

```

    int aimCoordinateX;
    int aimCoordinateY;
    boolean ranCorrectly;

/* Method: TargetWindow
   Description: This is the constructor for Target Window. It
instantiates the window and
               the mouse listeners for it. In addition, it attempts
to open the file created
               by UserInfoWindow that contains the results of the user
survey, and opens the
               first image for viewing
*/
TargetWindow(String title, String theFileName)
{
    // call parent constructor
    super(title);
    ranCorrectly = true;

    outputFileName = theFileName;
    contentPane = getContentPane();

    // attempt to open file for writing
    try
    {
        createFileStreams();
    }
    catch (Exception e)
    {
        System.out.println("TargetWindow::TargetWindow: ERROR - The image
list file could not be read.");
        this.setVisible(false);
        ranCorrectly = false;
    }
    System.out.println("TargetWindow::TargetWindow: Output File Name: "
+ outputFileName);

    // initialize image count and retrieve first image
    getNextImage();
    imageCount = 1;
}

/* Method: getNextImage
   Description: This method retrieves the next image from the
input/output file to be
               displayed
*/

private int getNextImage()
{
    // clear the frame
    contentPane.removeAll();
    ImageIcon tempImage;

    // attempt to read next image string from file
    try
    {
        tempStr = " ";
        while ((tempStr.startsWith("FILENAME:") ==
false)&&(tempStr!=null))

```

```

        {
            tempStr = targetInfoStream.readLine();
            outputFileStream.writeBytes(tempStr + "\r\n");
        }

        theImageName = tempStr.substring(10,tempStr.length());
    }
    catch (Exception E)
    {
        System.out.println("TargetWindow::getNextImage: Can't read next
image filename.  Exiting.");
        this.setVisible(false);
        return 0;
    }

    if (theImageName == null) // exit if no more images present in file
    {
        System.out.println("TargetWindow::getNextImage: No more images.
Exiting.");
        this.setVisible(false);

        // indicate failure to retrieve image
        return 0;
    }
    else // otherwise, place image in frame and repaint frame
    {
        // create new image from filename
        tempImage = new ImageIcon(theImageName);

        // debug image name
        System.out.print("TargetWindow::getNextImage: Image Displayed:
");
        System.out.println(theImageName);

        // create new imageLabel from image
        theImageLabel = new JLabel(tempImage, JLabel.NORTH_EAST);
        theImageLabel.addMouseListener(this);

        // resize frame to compensate for variable image sizes
        this.setSize(tempImage.getIconWidth()+10,
tempImage.getIconHeight()+20);

        // add image to frame and repaint
        contentPane.add(theImageLabel);
        repaint();
        show();

        // indicate success
        return 1;
    }
}

/* Method: openImageListFile
   Description:  Attempts to open the input/output file that currently
stores the
               results of the user survey as well as the list and
description of
               all the images to be shown
*/
private void createFileStreams()

```

```

{
    // attempt to open the file
    try
    {
        // create new buffer for reading params file
        FileReader reader = new FileReader (targetInfoFileName);
        targetInfoStream = new BufferedReader(reader);

        // create output buffer for output file
        outputFileStream = new RandomAccessFile(outputFileName,"rw");
        outputFileStream.seek(outputFileStream.length()); // go to end of
file to write

    }
    catch (Exception e)
    {
        System.out.println("TargetWindow::createFileStreams: ERROR - Can't
open image list file");
        this.setVisible(false);
        ranCorrectly = false;
    }
}

/* MouseListener functions that need to be included but have no
functionality */
public void mousePressed(MouseEvent e){}
public void mouseReleased(MouseEvent e) {}
public void mouseEntered(MouseEvent e) {}
public void mouseExited(MouseEvent e) {}
/*          *          *          *
*/

/* Method: MouseClicked
Description: Catches the event where the user has clicked the screen
and thus
coordinates has made a target selection for the current image. The
of the mouse click are stored in local memory and also
written to the i/o file *username*.txt along with information of
where shot occurred (which enemy, where on body)
*/
public void mouseClicked(MouseEvent e)
{
    if ((e.getModifiers() & InputEvent.BUTTON1_MASK) ==
InputEvent.BUTTON1_MASK)
    {
        // record coordinates
        aimCoordinateX = e.getX();
        aimCoordinateY = e.getY();
        repaint();
        show();

        // sift through formatting
        tempStr = " ";
        while (tempStr.startsWith("-") == false)
            try
            {
                tempStr = targetInfoStream.readLine();
            }

```

```

        outputFileStream.writeBytes(tempStr + "\r\n");
    }
    catch (Exception E) {}

    boolean selectedTargetFound = false;
    boolean noMoreTargets = false;
    targetCount = 1;
    targetSelected = 0;
    aimPoint = miss;
    while (!selectedTargetFound && !noMoreTargets)
    {
        tempStr = " ";
        try
        {
            tempStr = targetInfoStream.readLine();
            outputFileStream.writeBytes(tempStr + "\r\n");
        }
        catch (Exception E) {
            noMoreTargets = true;
        }
        //System.out.println("First number read as " +
        charToInt(tempStr.charAt(0)));
        if (charToInt(tempStr.charAt(0)) != targetCount)
            noMoreTargets = true;
        else
        {
            // read in and crop all of the unused numbers (6 total)
            int numsread = 0;
            while (numsread < 6) {
                String numString = "";
                while (charIsANum(tempStr.charAt(0)) == false) // go through
blank spaces
                    tempStr = tempStr.substring(1, tempStr.length());
                while (charIsANum(tempStr.charAt(0)) == true) { // get
digits in number
                    char firstChar = tempStr.charAt(0);
                    numString = numString + firstChar;
                    tempStr = tempStr.substring(1, tempStr.length());
                }
                numsread++;
            }

            // boxes for head, torso, legs stored in a 3x4 entry array
            int[][] targetAreaBox = new int[NUM_BODY_AREAS][4];
            for (int partCnt = 0; partCnt < NUM_BODY_AREAS; partCnt++)
                for (int boxCnt = 0; boxCnt < 4; boxCnt++)
                {
                    String boxBoundString = "";
                    while (charIsANum(tempStr.charAt(0)) == false) { // go
through blank spaces
                        tempStr = tempStr.substring(1, tempStr.length());
                    }
                    while (charIsANum(tempStr.charAt(0)) == true) { // get
digits in number
                        char firstChar = tempStr.charAt(0);
                        boxBoundString += firstChar;
                        // might be at eol
                        if (tempStr.length() > 1)
                            tempStr = tempStr.substring(1, tempStr.length());
                        else

```

```

        break;
    }
    targetAreaBox[partCnt][boxCnt] =
stringToInt(boxBoundString);
    }

    //System.out.println("Target shot at (" + aimCoordinateX + ",
" + aimCoordinateY + ")");

    // did we get 'em in the head?
    if (aimCoordinateX > targetAreaBox[head][0] &&
        aimCoordinateX < targetAreaBox[head][1] &&
        aimCoordinateY > targetAreaBox[head][2] &&
        aimCoordinateY < targetAreaBox[head][3]) {

        selectedTargetFound = true;
        targetSelected = targetCount;
        aimPoint = head;
        System.out.println("TargetWindow::mouseClicked: User
Selected Target " +
                                targetCount + " at Coordinates (" +
aimCoordinateX +
                                ", " + aimCoordinateY + ") (head)");
    }
    else // in the torso?
    if (aimCoordinateX > targetAreaBox[torso][0] &&
        aimCoordinateX < targetAreaBox[torso][1] &&
        aimCoordinateY > targetAreaBox[torso][2] &&
        aimCoordinateY < targetAreaBox[torso][3]) {
        selectedTargetFound = true;
        targetSelected = targetCount;
        aimPoint = torso;
        System.out.println("TargetWindow::mouseClicked: User
Selected Target " +
                                targetCount + " at
Coordinates (" + aimCoordinateX +
                                ", " + aimCoordinateY + ")
(torso)");
    }
    else // in the legs?
    if (aimCoordinateX > targetAreaBox[legs][0] &&
        aimCoordinateX < targetAreaBox[legs][1] &&
        aimCoordinateY > targetAreaBox[legs][2] &&
        aimCoordinateY < targetAreaBox[legs][3]) {
        selectedTargetFound = true;
        targetSelected = targetCount;
        aimPoint = legs;
        System.out.println("TargetWindow::mouseClicked: User
Selected Target " +
                                targetCount + " at
Coordinates (" + aimCoordinateX +
                                ", " + aimCoordinateY + ")
(legs)");
    }
    else // either we missed or we didn't hit this target
        targetCount++;
    }
}

```

```

try {
    // more formatting
    while (tempStr.startsWith("-") == false) {
        tempStr = targetInfoStream.readLine();
        outputFileStream.writeBytes(tempStr + "\r\n");
    }

    outputFileStream.writeBytes("\r\n");
    outputFileStream.writeBytes("SME Target Selection Results
\r\n");
    outputFileStream.writeBytes("-----
\r\n");

    // write target info to file
    if (targetSelected != 0)
        outputFileStream.writeBytes("TargetSelected: " +
targetSelected +
                                "\r\n");
    else
        outputFileStream.writeBytes("TargetSelected: " + "none \r\n");

    String aimPointString;
    switch (aimPoint) {
        case 0: {
            aimPointString = "head";
            break;
        }
        case 1: {
            aimPointString = "torso";
            break;
        }
        case 2: {
            aimPointString = "legs";
            break;
        }
        default: {
            aimPointString = "miss";
            break;
        }
    }

    outputFileStream.writeBytes("Coordinates of Shot: " +
aimCoordinateX +
                                " " + aimCoordinateY + "\r\n");
    outputFileStream.writeBytes("AimPoint: " + aimPointString +
"\r\n \r\n");

    outputFileStream.writeBytes("After-Action Review \r\n");
    outputFileStream.writeBytes("----- \r\n");
    outputFileStream.writeBytes("M4Mode:\r\n");
    outputFileStream.writeBytes("
\r\n");
    outputFileStream.writeBytes("p_hit:\r\n");
    outputFileStream.writeBytes("    \r\n");
    outputFileStream.writeBytes("p_kill:\r\n");
    outputFileStream.writeBytes("    \r\n");
}
catch (Exception E) {}

```

```

        // retrieve new image and refresh frame
        if (getNextImage() == 1)
            imageCount++;
        else {
            try {
                outputFileStream.close();
                targetInfoStream.close();
                this.setVisible(false);
            }
            catch (Exception ex) {}
        }
    }
    else
    {
        //right clicked
        System.out.println("TargetWindow::mouseClicked: Target right-
clicked at (" + e.getX() + ", " + e.getY() + ")");
    }

}

/* Method: didRunCorrectly
Description: Returns the value of ranCorrectly, which becomes
false if there is an error
reading any of the required files
*/
public boolean didRunCorrectly()
{
    return ranCorrectly;
}

/* Method: charToInt
Description: This function returns the value of numerical
characters (i.e. charToInt('1') = 1)
*/
private int charToInt(char c)
{
    return (int)c + CHAR_TO_INT_OFFSET;
}

/* Method: stringToInt
Description: This function returns the value of numerical string
(i.e. stringToInt('321') = 321)
and uses the charToInt function
*/
private int stringToInt(String s)
{
    int numDecimalPlaces = s.length();
    int currentPlace = s.length();
    int intLength = s.length();
    int value = 0;
    while (currentPlace > 0)
    {
        value +=
            (int) (charToInt(s.charAt(intLength -
currentPlace)) * Math.pow((double)10, (double) (currentPlace-1)));
        currentPlace--;
    }
    return value;
}

```

```

    /* Method: charIsANum
       Description: Returns true if the char provided is a numerical
       character (i.e. charIsANum('1') == true)

       (however charIsANum('a') == false)
    */
    private boolean charIsANum(char c)
    {
        if (c >= '0' && c <= '9') return true;
        else return false;
    }
}

// JPanel class definition for image JLabel
class FirstTargetWindowPanel extends JPanel
{
    FirstTargetWindowPanel()
    {
        setBackground(Color.white);
    }
}

```

AfterActionReviewWindow.java

```
/*
 * Title:      AfterActionReviewWindow
 * Description: This is the Window JFrame that displays the
 *              after-action review for each image.
 *              In addition, this Frame is responsible for
 *              recording aar responses for each image
 *              by writing them to a pre-existing file
 *              containing the results of the user survey,
 *              the drill exercise, and the list and
 *              description of each image to be displayed.
 * Copyright:  Copyright (c) 2003
 * Author:     Soar Technology, Inc.
 * Version:    1.1
 * Date:       11.24.03
 */

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.image.*;
import java.io.*;
import java.io.RandomAccessFile.*;
import java.lang.*;

public class AfterActionReviewWindow extends JFrame implements
ActionListener, ItemListener
{

    // jframe stuff
    Container contentPane;

    // filestream for reading/writing
    RandomAccessFile outputFileReaderStream;
    String fileName;

    //tempstr for sifting through formatting
    String tempStr;

    // coordinates of shot
    int aimPointX; int aimPointY;

    // buttons and text fields
    JButton submitButton;
    JRadioButton singleShot, doubleTap, threeRoundBurst, other;
    JTextField other_text, p_hit, p_kill;

    // constants
    static int CHAR_TO_INT_OFFSET = -48;

    /* Method: AfterActionReviewWindow
       Description: This is the constructor for AfterActionReviewWindow.
       It instantiates the window and
       the and attempts to open the file created by
       UserInfoWindow that contains the results
       of the user survey and drill exercise
    */
}
```

```

public AfterActionReviewWindow(String title, String outputFileName)
{
    super(title);
    fileName = outputFileName;

    // activate contentPane for window
    contentPane = getContentPane();
    contentPane.setBackground(Color.white);

    // attempt to open outputFileName
    try {outputFileReaderStream = new RandomAccessFile(new
File(outputFileName), "rw");}
    catch (Exception E)
    {

System.out.println("AfterActionReviewWindow::AfterActionReviewWindow:
ERROR - Could not open " + outputFileName + " for filestream.");
        this.setVisible(false);
    }

    // retrieve first image
    getNextImage();
}

/* Method: getNextImage
   Description: This method clears the active window and loads the
next image referred to in *username*.txt along
               with the after action review questions. In addition,
this method is also responsible for adding
               the crosshairs of the shot taken by the user on the
current image during the drill exercise
*/
private int getNextImage()
{
    // clear the frame
    contentPane.removeAll();
    contentPane.setLayout(new FlowLayout(FlowLayout.CENTER));
    ImageIcon tempImage;

    // attempt to read next image string from file
    try
    {
        tempStr = " ";
        while ((tempStr.startsWith("FILENAME:") ==
false)&&(tempStr!=null))
            tempStr = outputFileReaderStream.readLine();
    }
    catch (Exception E) {return 0;}

    String theImageName = tempStr.substring(10,tempStr.length());

    if (theImageName == null) // exit if no more images present in file
        return 0;
    else // otherwise, place image in frame and repaint frame
    {
        // create new image from filename
        tempImage = new ImageIcon(theImageName);

        // debug image name

```

```

        System.out.print("AfterActionReviewWindow::getNextImage: Image
Displayed:  ");
        System.out.println(theImageName);

        // display shot coordinates on image
        String tempString = " ";
        while (tempString.startsWith("Coordinates of Shot") == false)
            try {tempString = outputFileReaderStream.readLine();} catch
(Exception E) {}
        tempString = tempString.substring(20).trim();
        String xcoordString = tempString.substring(0,tempString.indexOf('
')).trim();
        String ycoordString = tempString.substring(tempString.indexOf('
')).trim();

        aimPointX = stringToInt(xcoordString); aimPointY =
stringToInt(ycoordString)+5;

        // add bullet to image
        Image imageWithBullet = tempImage.getImage();
        int w = imageWithBullet.getWidth(this);
        int h = imageWithBullet.getHeight(this);
        int pixels[] = new int [w * h];
        PixelGrabber pg = new
PixelGrabber(imageWithBullet,0,0,w,h,pixels,0,w);
        try {pg.grabPixels();} catch (Exception E) {}

        // create crosshairs around target
        for (int x = (aimPointX - 10); x < (aimPointX + 11); x++)
            for (int y = (aimPointY); y < (aimPointY + 1); y++)
                {
                    pixels[w*y + x] = 0xff000000 | Color.ORANGE.getRed() << 16 |
Color.ORANGE.getGreen() << 8 | Color.ORANGE.getBlue();
                }

        for (int y = (aimPointY - 10); y < (aimPointY + 11); y++)
            for (int x = (aimPointX); x < (aimPointX + 1); x++)
                {
                    pixels[w*y + x] = 0xff000000 | Color.ORANGE.getRed() << 16 |
Color.ORANGE.getGreen() << 8 | Color.ORANGE.getBlue();
                }

        imageWithBullet = createImage(new
MemoryImageSource(w,h,pixels,0,w));
        tempImage.setImage(imageWithBullet);

        // create new imageLabel from new image
        JLabel theImageLabel = new JLabel(tempImage, JLabel.NORTH_EAST);

        contentPane.add(theImageLabel);

        // resize frame to compensate for variable image sizes
        this.setSize(tempImage.getIconWidth()+10,
tempImage.getIconHeight()+ 200);

        JLabel targetSelectedLabel = new JLabel("You fired at the point
displayed above");
        contentPane.add(targetSelectedLabel);

        // add AAR questions and answer slots

```

```

        AfterActionReviewPanel modeOfFirePanel = new
AfterActionReviewPanel();
        modeOfFirePanel.setLayout(new FlowLayout(FlowLayout.CENTER));
        JLabel M4ModeLabel = new JLabel("Mode of Fire: ");
        ButtonGroup M4ModeGroup = new ButtonGroup();
        singleShot = new JRadioButton("Single Shot");
        singleShot.setBackground(Color.white);
        singleShot.addItemListener(this);
        M4ModeGroup.add(singleShot);
        doubleTap = new JRadioButton("Double Tap");
        doubleTap.setBackground(Color.white);
        doubleTap.addItemListener(this);
        threeRoundBurst = new JRadioButton("Three-Round Burst");
        threeRoundBurst.setBackground(Color.white);
        threeRoundBurst.addItemListener(this);
        other = new JRadioButton("Other");
        other.setBackground(Color.white);
        other.addItemListener(this);
        M4ModeGroup.add(singleShot);
        M4ModeGroup.add(doubleTap);
        M4ModeGroup.add(threeRoundBurst);
        M4ModeGroup.add(other);
        other_text = new JTextField(10);

        modeOfFirePanel.add(M4ModeLabel);
        modeOfFirePanel.add(singleShot);
        modeOfFirePanel.add(doubleTap);
        modeOfFirePanel.add(threeRoundBurst);
        modeOfFirePanel.add(other);
        modeOfFirePanel.add(other_text);
        contentPane.add(modeOfFirePanel);

        AfterActionReviewPanel probPanel = new AfterActionReviewPanel();
        JLabel p_hitLabel = new JLabel("Probability of Hit: ");
        p_hit = new JTextField(3);
        JLabel p_killLabel = new JLabel("Probability of Kill: ");
        p_kill = new JTextField(3);

        probPanel.add(p_hitLabel);
        probPanel.add(p_hit);
        probPanel.add(p_killLabel);
        probPanel.add(p_kill);
        contentPane.add(probPanel);

        submitButton = new JButton("Submit");
        submitButton.addActionListener(this);
        contentPane.add(submitButton);

        // repaint and show
        repaint();
        show();

        // indicate success
        return 1;
    }
}

/* Method: charToInt
Description: This function returns the value of numerical
characters (i.e. charToInt('1') = 1)

```

```

*/
private int charToInt(char c)
{
    return (int)c + CHAR_TO_INT_OFFSET;
}

/* Method: stringToInt
   Description: This function returns the value of numerical string
(i.e. stringToInt('321') = 321)
               and uses the charToInt function
*/
private int stringToInt(String s)
{
    int numDecimalPlaces = s.length();
    int currentPlace = s.length();
    int intLength = s.length();
    int value = 0;
    while (currentPlace > 0)
    {
        value +=
            (int) (charToInt(s.charAt(intLength -
currentPlace)) * Math.pow((double)10, (double) (currentPlace-1)));
        currentPlace--;
    }
    return value;
}

public void itemStateChanged(ItemEvent e)
{}

/* Method: actionPerformed
   Description: This function is required for a frame to implement
ActionListener.
               It fires when the submit button is pressed and
records all of the answers indicated at the time of submittal to
*username.txt* and then calls getNextImage to retrieve the next image
*/
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == submitButton)
    {
        // put entries into text file

        String tempString = " ";

        // M4Mode
        try
        {
            // find M4Mode text area in *username*.txt
            while (tempString.startsWith("M4Mode") == false)
                tempString = outputFileReaderStream.readLine();

            // enter user response for M4Mode
            String outputString = " ";
            if (singleShot.isSelected())
            {
                outputString = "Single Shot";
            }
        }
    }
}

```

```

        else if (doubleTap.isSelected())
        {
            outputString = "Double Tap";
        }
        else if (threeRoundBurst.isSelected())
        {
            outputString = "Three-Round Burst";
        }
        else if (other.isSelected())
        {
            outputString = other_text.getText();
        }
        else
        {
            outputString = "No Response";
        }

        outputFileReaderStream.writeBytes(outputString);
    }
    catch (Exception E)
    {
        System.out.println("AfterActionReviewWindow::actionPerformed:
ERROR - Cannot find appropriate place to write M4Mode data.");
        this.setVisible(false);
    }

    // repeat for p_hit and p_kill

    // p_hit
    try
    {
        while (tempString.startsWith("p_hit") == false)
            tempString = outputFileReaderStream.readLine();

        String outputString = p_hit.getText();

        outputFileReaderStream.writeBytes(outputString);
    }
    catch (Exception E)
    {
        System.out.println("AfterActionReviewWindow::actionPerformed:
ERROR - Cannot find appropriate place to write p_hit data.");
        this.setVisible(false);
    }

    // p_kill
    try
    {
        while (tempString.startsWith("p_kill") == false)
            tempString = outputFileReaderStream.readLine();

        String outputString = p_kill.getText();

        //long numBytesInTempString =
        Long.getLong(tempString).longValue();

        //outputFileReaderStream.seek(outputFileReaderStream.getFilePointer() -
        numBytesInTempString);
        outputFileReaderStream.writeBytes(outputString);
    }

```

```

        catch (Exception E)
        {
            System.out.println("AfterActionReviewWindow::actionPerformed:
ERROR - Cannot find appropriate place to write p_kill data.");
            this.setVisible(false);
        }
    }
    if (getNextImage() == 0)
    {
        try
        {
            System.out.println("AfterActionReviewWindow::actionPerformed:
Attempting to close " + fileName);

outputFileReaderStream.setLength(outputFileReaderStream.getFilePointer()
);
            outputFileReaderStream.close();
        }
        catch (Exception E)
        {
            System.out.println("AfterActionReviewWindow::actionPerformed:
ERROR - Cannot close " + fileName);
        }
        this.setVisible(false);
    }
}

}

// JPanel class definition for image JLabel
class AfterActionReviewPanel extends JPanel
{
    AfterActionReviewPanel()
    {
        setBackground(Color.white);
    }
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        // write stuff in panel
    }
}

```