Behavior Research Methods
2005, 37 (4), 591-601

Revised Space Fortress: A validation study

WAYNE L. SHEBILSKE
Wright State University, Dayton, Ohio

RICHARD A. VOLZ
Texas A&M University, College Station, Texas

KEVIN M. GILDEA and JUDSON W. WORKMAN
Wright State University, Dayton, Ohio

and

MAITREYI NANJANATH, SEN CAO, and JONATHAN WHETZEL
Texas A&M University, College Station, Texas

We describe briefly the redevelopment of Space Fortress (SF), a research tool widely used to study training of complex tasks involving both cognitive and motor skills, to be executed on current-generation systems with significantly extended capabilities, and then compare the performance of human participants on an original PC version of Space Fortress (SF) with the revised Space Fortress (RSF). Participants trained on SF or RSF for 10 sets of eight 3-min practice trials and two 3-min test trials. They then took tests involving retention, resistance to secondary task interference, and transfer to a different control system. They then switched from SF to RSF or from RSF to SF for 2 sets of final tests and completed rating scales comparing RSF and SF. Slight differences were predicted on the basis of a scoring error in the original version of SF used and on slightly more precise joystick control in RSF. The predictions were supported. The SF group started better but did worse when they transferred to RSF. Despite the disadvantage of having to be cautious in generalizing from RSF to SF, we conclude that RSF has many advantages, which include accommodating new PC hardware and new training techniques. A monograph that presents the methodology used in creating RSF, details on its performance and validation, and directions on how to download free copies of the system may be downloaded from www .psychonomic.org/archive/.

Richard Volz and his colleagues[1] redeveloped Space Fortress (SF), a research tool extensively used to study training of complex tasks involving both cognitive and motor skills (Gopher, Weil, & Bareket, 1994; Mané & Donchin, 1989; Shebilske, Goettl, Corrington, & Day, 1999; Shebilske, Goettl, & Regian, 1999; Shebilske, Jordan, Goettl, & Day, 1999). The potential for experiments with SF to advance our understanding of how component skills interact within complex skills (Shebilske, Goettl, & Regian, 1999) remains. However, a redevelopment effort was necessary to continue to allow SF to be used as a research platform, because advances in hardware and software have rendered earlier versions unusable on current platforms.

This research was conducted with funding from the Air Force Office of Scientific Research, Multiple University Research Initiative Grant 664594. Correspondence should be addressed to W. L. Shebilske, Department of Psychology, Wright State University, 3640 Colonel Glenn Hwy., Dayton, OH 45435-0001 (e-mail: wayne.shebilske@wright.edu).

Note—This article was accepted by the previous editor, Jonathan Vaughan.

The advantages of the revised SF (RSF) include not only its (1) ability to be executed on current-generation platforms, but also its (2) being written in a mostly platform-independent manner,[2] (3) providing a general experiment definition mechanism, (4) supporting teamwork through being able to flexibly assign different input controls to different members of a team, (5) maintaining all data in a central database, so that data sets do not have to be merged manually after the fact, (6) being executable in a distributed environment, (7) having a capability for playback of any trial, and (8) having hooks built in for the incorporation of intelligent agents playing various roles.

In the present experiment, we compared performance of human participants on SF and RSF. We (1) trained participants on either SF or RSF for 10 lessons consisting of eight 3-min practice trials and two 3-min test trials; (2) tested for retention, resistance to secondary task interference, and transfer to a keyboard control; (3) switched participants from SF to RSF or from RSF to SF for two sets of final tests; and (4) administered after each set a questionnaire comparing RSF and SF.

We had two rationales for making predictions. The first concerned predicting the consequences of correcting an error in scoring control (see note 1). The participants who

trained on SF and RSF were told that the control score would be lowered when they flew outside the hexagon. However, a scoring error in SF made it possible for the participants using SF to fly almost anywhere on the screen without lowering their control score. In contrast, the correct scoring of control in RSF made it impossible for the participants using RSF to fly outside the hexagon without lowering their control score. On the one hand, flying inside the hexagon is more difficult to learn, because it requires precise control of the ship's velocity and position in frictionless space. On the other hand, Frederiksen and White (1989) found that circumnavigating the fortress while flying inside the hexagon is the optimal control strategy, resulting in the best performance. The participants who fly inside the hexagon not only get higher control scores (with a correct scoring system), but also do better on other subscores. Especially, they achieve higher velocity scores, because they must fly slower to maintain their position within the hexagon. Learning better control of their ship also affords an advantage for the speed score. They achieve higher point scores because they are in a better position to hit and destroy the fortress. Research on the developing of strategies for performing complex tasks (Yechiam, Erev, & Gopher, 2000) suggests that the scoring error in SF may reduce the likelihood of developing this optimal strategy. The scoring error eliminates immediate feedback that flying in the hexagon can afford an advantage. Without this immediate feedback, participants will be less likely to invest in learning a strategy that will not pay off until later. Accordingly, we predicted that, relative to the participants training with SF, the participants training with RSF would have more of a tendency to fly within the hexagon after training. As a result, they would score worse in the beginning of training, especially on the control score, and better in most or all of the scores by the end of training, although there was no reason to believe that they would perform better on the control score. On crossover tests, the participants who trained with RSF would maintain their advantage when they switched to SF, because they would have the habit of flying within the hexagon and would not experience the consequences of the scoring error in SF. In contrast, the participants who trained with SF would fall even lower when they switched to RSF, because they would have the habit of flying outside the hexagon, which would reduce their control score in the correct RSF scoring system.

A second rationale yielded similar predictions. Although SF and RSF process joystick information identically, the joystick reading routine used in DOS for SF and the one used in Windows for RSF are different[3] and may introduce slight differences beyond the control of the SF or RSF code. Experts comparing SF and RSF before the present experiment agreed that they were very similar, but they also believed that the joystick control was slightly more sensitive for RSF. Higher sensitivity would afford an advantage in the long run, but it would be less forgiving and more difficult in the beginning. Therefore, relative to the participants training with SF, the participants training with RSF would score worse in the beginning of training and better by the end of training. On crossover tests, the participants who trained with RSF would be able to adapt quickly to a more forgiving joystick. They would therefore maintain most or all of their advantage when they switched to SF, depending on how well they could compensate for the less precise control. In contrast, the participants who trained with SF would fall even lower when they switched to RSF, because they would have difficulty controlling the less forgiving joystick. Consider an extreme example of crossing between controls with different precision. Imagine a pilot being asked to do a barrel roll in a Piper Cub or a jet, which has more sensitive controls. A pilot crossing from a jet to a Piper Cub would be able to adapt more quickly than a pilot crossing over from a Piper Cub to a jet. The difference in joystick sensitivity between SF and RSF was much less extreme. It was almost imperceptible, but if it had an effect on performance, it would result in a pattern of results similar to the extreme example and similar to the pattern predicted as a consequence of the scoring error in SF.

METHOD

Revised SF Development

SF was originally developed at the University of Illinois (Mané & Donchin, 1989) under the auspices of the Department of Defense to facilitate research on training participants to perform complex sensory/motor/cognitive tasks. It was subsequently ported to a PC by Gopher (Gopher et al., 1994). Progress in both computer hardware and software have rendered earlier versions unusable, and current training research requires additional capabilities, such as support for intelligent agents. Equally important is a need for techniques that allow real-time simulation systems to execute on standard platforms that are, fundamentally, nonreal time—that is, that do not provide response time guarantees.

Objectives. The objectives of our development, then, were threefold.

1. Create techniques whereby many real-time training simulations can be built on standard hardware platforms with standard operating systems.

2. Redevelop SF so that research utilizing this tool can continue.

3. Extend SF to include capabilities that permit the study of advanced training protocols using intelligent agents.

Design requirements and considerations. Three kinds of requirements ensue from the above objectives: (1) those concerned with the timing performance and platform independence, (2) those concerned with matching, at a program level, the simulation integrity between RSF and SF, and (3) those specifying additional capabilities.

In terms of the platform-level requirements, we used the Java programming language, because of its platform independence, and various versions of the Windows operating system, because Windows is the most widely used system on which researchers would like to execute RSF.

The key real-time issue is achieving the simulation cycle times required without significant interference. Both Java and Windows have limiting characteristics, however. Java's garbage collector runs from time to time, perhaps taking hundreds of milliseconds. Windows runs a number of system tasks, interrupting application programs while doing so. Uncontrolled, both of these interfere with achieving the desired simulation cycle times. It is possible, however, to achieve some control over the Java garbage collections, in order to limit external interference to the RSF application so that the residual is small enough to allow RSF and many other applications to execute successfully. The details are described in the supplementary material in the archive.

In principle, one would expect that matching the behavior of the RSF and SF simulation systems should be as simple as reading the original code, determining all critical parameters, and then including these in the new simulation. However, the process is complicated by several factors. The original graphics support system is outdated and will not execute on current systems. The replacement drawing systems have different timing characteristics and behave in a rather different manner. Moreover, the original PC implementation makes a number of unessential approximations (e.g., approximating the hexagons by circles, for purposes of determining whether the ship is inside or outside a hexagon) and, in one case, has a significant error (scoring when a ship is outside the hexagon). Furthermore, although the original specification for a game cycle was 40 msec, the implementation actually achieved only a somewhat variable period; for one of the common widely used sets of hardware, the cycle time was measured to be approximately 46 msec. The design process to address these issues, therefore, was actually a spiral process of implementation and testing of the resulting system by expert-level players.

The operational enhancements desired for RSF enable both the study of advanced training protocols and a more organized conduct of experiments. More specifically, the additional features of RSF are the following.

1. The ability to utilize a variety of intelligent agents. We have already used an intelligent agent as a partner in our revised version of the original PC SF, in which case the agent performs either the joystick or the mouse functions and a human partner performs the others. RSF allows any input to be provided by either a human or an agent (He, Nanjanath, & Volz, 2004). Intelligent agents also can be used to adapt the training protocol to individual needs. We have already implemented a simple form of this, in which the transition between components in a Multiple Emphasis on Components protocol is based on an individual performance, rather than on a schedule (Bodine, 2003).

2. The ability to include multiple players. Our goal in this area was to have each individual input assignable to any participant in any combination. Furthermore, an observation-only mode (on a separate processor) is supported.

3. The ability to easily configure an experiment to include a variety of experimental conditions. We divide the specification of an experiment into several levels. A Space Fortress experiment (SFE) is composed of a number of Space Fortress experiment conditions (SFECs). Multiple SFEs and SFECs are allowed to execute concurrently. After passing a qualification test, trainees are randomly assigned to an experimental condition within their SFE. The conduct of an experiment may require several groups to participate at different times. Each SFEC is composed of a selectable number of sessions. Each session, consisting of a selectable number of trials, is to be composable differently. Each trial is configurable, with any input assigned to any player in the simulation.

4. The ability to conduct training scenarios in a distributed environment. Individuals (human or software agents) in a multiple player simulation are able to participate in the simulation from different workstations. The experiment proctor can control experiment sessions from remote locations.

5. The ability to manage experimental data in an organized manner. RSF has a centralized data collection capability. The original SF simply wrote scores to the local disk after each trial. Subsequently, the files had to be moved to a central computer and postprocessed to generate a single data file that could be analyzed. In RSF, the data collected is automatically sent over the network to a central data collection computer and is stored in a database for easy retrieval.

6. The ability to replay a game (after action review). At a minimum, all inputs must be captured and stored, including not only explicit user inputs, such as joystick movement or mouse clicks, but game-generated inputs, such as the appearance of mines and bonus opportunities as well.

System-level validation results. Several levels of validation were applied to RSF. The focus of the lowest level of unit testing was on confirming that the basic game parameters did indeed match those of SF. These included such things as object dimensions (ship, fortress, hexagons, mines, shells, and missiles), maximum object speeds, frequencies at which entities such as mines and bonuses appear, and so forth. Although the initial values were determined by reading the SF code, measurements were taken on the screen to identify any incorrect values. Next, we had experts execute both simulations and report any discrepancies that they noted. The purpose in these tests were to discover any methodological differences that might have crept into the coding. Then we performed extensive timing tests, to determine whether, ultimately, we were able to obtain adequate real-time performance.

The lowest level of SF feature testing did not uncover any intellectually interesting differences; all differences were subsequently brought into concurrence. The testing of the two versions by experts revealed more interesting results. In the PC version of SF, the determination of whether a ship was outside the hexagon was approximated by determining whether it was outside an approximating circle. Due to a scaling error, the actual radius of the circle used encompassed nearly the entire screen. Thus, the con-

trol scores acquired while flying outside the large hexagon were higher in SF than in RSF. After considering the alternatives carefully, it was decided to implement the hexagon check and to control scoring correctly in RSF and study the effect of the resulting difference. Second, expert users stated that the joystick felt slightly more sensitive with RSF than with SF. This occurred despite the joystick calibration. No test participant reported noticing any difference, however; only those expert in the game could notice a difference. The lowest level system routines used to read the joystick are not part of the SF or RSF software per se and are not identical. Since we do not have the source code for the joystick routines used in RSF, we do not know exactly what measurement technique was used. Further calibration would thus be very difficult to accomplish. As a result, it was decided to use the existing joystick reading and calibration routines and conduct tests to determine whether there was a measurable difference.

In terms of the timing performance of RSF, although we have made many fine-grained measurements of specific function times, we will show typical data only for the most important real-time parameter, the stability of the simulation cycle. Figure 1 shows the actual cycle time versus the cycle number for 733-MHz Dell Precision PCs running Windows 2000, since that was the environment in which RSF was used in our experiments. Although it is easy to set the cycle time as desired, we chose a nominal 46 msec in order to match common system performance in the original PC version of SF. Except for the larger cycle times that correspond to either a ship or a fortress destruction, it can be seen that the cycle times are quite stable and around the nominal value. In other tests, we found that we were able to control the Java garbage collection so that we experienced only 1.7 msec of delay once every 800 msec; this is negligible within our 46-msec period.

Summary of RSF development. The underlying design and implementation has been found, through extensive testing, to be sound, with the exception of the error in SF that was corrected in RSF and the slight joystick sensitivity difference felt by expert players. These two differences formed the motivation for the experiments reported below.

Participants

The participants were 125 right-handed males between the ages of 18 and 61 years. Some of the participants were students, whereas others were paid employees of a temporary employment agency. The students received research credits for the 1st hour of the study and were compensated at $6.75/h for the remainder of their time. The temporary employees received $6.75/h for all hours worked.

The selection of participants was based on screening of performance on an aiming task (Mané & Donchin, 1989). The participants were not permitted to take part in the experiment if they failed to obtain a minimum aiming score of 780 points on at least one of their aiming games.

Apparatus

Two adjacent laboratory rooms, each equipped with eight computer workstations, were used for data collection. Each workstation was equipped with an IBM PC-compatible computer, a joystick for the right hand, and a three-button mouse for the left hand. Two chair-desks sat in front of each station, with the mouse placed on the left desk and the joystick attached with Velcro straps to the right desk. All the participants sat in the right chair-desk. Sound effects were delivered on headphones.

Tasks

The aiming task (Mané & Donchin, 1989) consisted of three 3-min trials. The aiming task display contained the participant's ship in the center of the screen. The ship could be rotated left and right by corresponding movements of the joystick. Mines appeared on the screen every few seconds, and the participants attempted to quickly and accurately destroy the mines. The mines disappeared im-

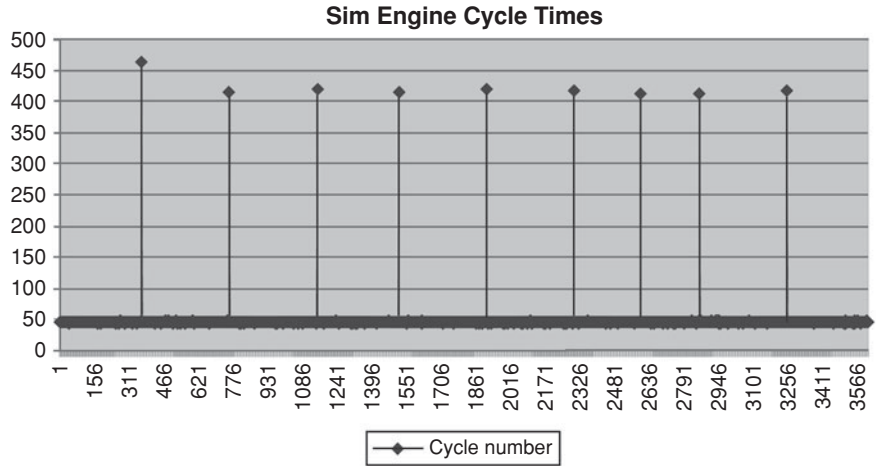

Figure 1. Stability of timing achieved in the Revised Space Fortress.

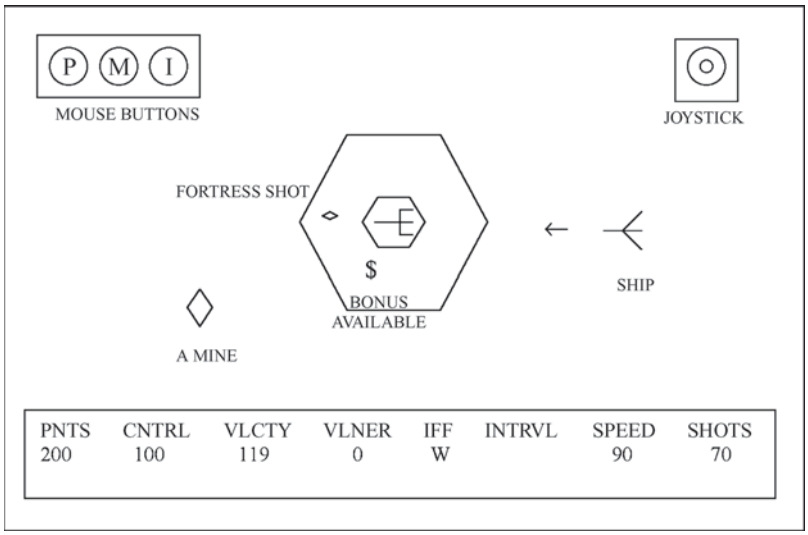

Figure 2. Space Fortress game screen.

mediately if hit or after a few seconds if missed. The total aiming score was a function of the number of mines destroyed and the speed with which they were destroyed.

The task screen was the same for SF and RSF. Figure 2 shows the starting positions of a computer-controlled fortress, which was centered within two concentric hexagons, and of a ship that was controlled by a joystick. The fortress rotated to track the ship and to fire shots at it. The small diamond between the fortress and the ship is a shot from the fortress. The arrow is a missile from the ship. The larger diamond is a mine, which appeared every few seconds. The dollar sign signals an opportunity for a bonus.

A control panel was shown below the area in which the ship and mines flew. It displayed five subscores: points (PNTS), control (CNTRL), velocity (VLCTY), speed (SPEED), and the vulnerability (VLNER) of the fortress. It also displayed an indicator used to identify friend or foe (IFF), an interval (INTRVL), which indicated the time between IFF responses, and shots (SHOTS), which indicated the number of missiles remaining on the ship.

The velocity score was designed to increase at the rate of 7/sec when the ship was flown below a slow criterion velocity and to decrease at the rate of 7/sec when the ship was flown above the criterion velocity.

The control score was designed to increase at the rate of 7/sec when the ship was within the hexagonal boundaries and to increase by 3.5/sec when the ship was on the screen but outside the outer hexagon. It was designed to decrease by 35 when the ship flew off the screen and by 5 when the ship flew into the inner hexagon.

The speed score was determined by how quickly the trainee properly identified and destroyed the mines that appeared on the screen. The interval between a mine's disappearing from the screen and appearing again was approximately 4 sec, and mines remained for 10 sec or until they hit the trainee's ship or were destroyed by the trainee. Before each trial, the trainees memorized three IFF letters that indicated foe mines. Other letters indicated friends, which the trainees were to energize with a shot. Foe mines had to be identified with a double click of the right mouse button within a 250- to 400-msec interval, after which they could be destroyed with a single shot.

The points score increased 4 for hitting the fortress, 100 for destroying the fortress, 20 for shooting a friendly mine, 30 for destroying a foe mine, and 100 for selecting the points bonus when it appeared. The points score decreased 3 for firing a missile after the supply was depleted, 50 for being shot by the fortress, 50 for being damaged by the mines, and 100 for being destroyed by the fortress.

Opportunities for bonuses were given periodically throughout the game. When those arose, the trainee could select either 100 bonus points by clicking on the left mouse button or 50 additional missiles by clicking on the center mouse button.

Experimental Design

Two between-participants independent variables and one within-participants independent variable were combined in a split-plot factorial design. The between-participants independent variables were version of task (SF vs. RSF) and order of crossover tests (trained version first vs. alternate version first). The within-participants independent variable was lesson number. All the participants completed 19 lessons. The dependent variables were the total score (sum of the subscores) and subscores averaged over the tests within each lesson, as well as the ratings on the SF/RSF rating scale.

Experimental Procedure

Table 1 lists the activities that were performed by the participants during the practice and test trials in all the experimental conditions. Acquisition lessons consisted of

Table 1
Experiment Schedule

		Activity				Space Fortress Trials	
Day	Lesson	SF-A	SF-B	RSF-A	RSF-B	Practice	Test
1			Informed consent				
			Questionnaires				
			Aiming task			0	3
2			Instruction video				
			Assigned to SF or RSF				
	0		Baseline trials			0	4
			Summary video				
	1–2		Training lessons			8	2
3	3–6		Training lessons			8	2
4	7–10		Training lessons			8	2
5	11		Retention test			0	2
	12		Secondary interference task			0	2
	13		Retention test			0	2
	14		Transfer task			0	2
	15		Retention test			0	2
	16	SF	RSF	RSF	SF	0	2
	17	RSF	SF	SF	RSF	0	2
			SF/RSF rating scale				
	18	SF	RSF	RSF	SF	0	2
	19	RSF	SF	SF	RSF	0	2
			SF/RSF rating scale				

eight 3-min training trials and two 3-min test trials. The use of two test trials following each acquisition lesson replicated the procedures used by Donchin (1989; Donchin, Fabiani, & Sanders, 1989) in his original experiments, in which SF was used, and was intended to measure the improvement in task performance that occurred during each lesson. The participants began by completing an informed consent form and a research participant payment form, as well as questionnaires on demographics, video game experience, and attitudes toward video games.

Upon completion of the questionnaires, the participants performed the aiming task. Participant scores were then examined, and those individuals scoring above 780 total points on one of their aiming games were invited to complete the remainder of the study. Unqualified candidates were debriefed and were given compensation for their participation. Qualified participants were randomly assigned to a training protocol and to computer workstations (this was to control for any differences in location and any other minor differences that may have been present from station to station).

On Day 2, the participants were shown a 17-min video containing complete instructions for the SF task prior to the start of their training. The participants were also given paper instructions and a picture of the game screen, to use in conjunction with the video. The written summary of the instructions was available throughout the experiment. When the training video ended, the participants completed four trials (baseline trials) of the complete SF task (or the RSF task, depending on their experimental condition), to establish a baseline of their performance. These four trials were run under the total score emphasis. Following the baseline trials, a 7-min summary video of the instructions was shown (and a paper summary of the instructions was given) in order to refresh the trainees' memory. After the summary video, the participants were divided into their training protocols and were given specific instructions for the training condition that they were assigned to for the remainder of the experiment. The participants then performed two training lessons.

On Days 3 and 4 of the experiment, the participants performed four lessons per day. Day 3 consisted of Lessons 3, 4, 5 and 6; Day 4 consisted of Lessons 7, 8, 9, and 10. On Day 5, the participants completed nine 2-trial lessons. Lessons 11–15 followed an ABACA design, in which A was a two-test game lesson, B was a two-test game secondary tapping task lesson, and C was a two-test game transfer test using the keyboard. Lessons 16–19 investigated participant perceptions of the task when they alternated back and forth between the two versions of the SF game. The participants in SF-A and RSF-A completed two test games in the version on which they had trained on Lessons 16 and 18. They completed two test games on the other version during Lessons 17 and 19. The participants in SF-B and RSF-B completed two test games on the version on which they had trained during Lessons 17 and 19 and completed two test games on the other version during Lessons 16 and 18.

The SF/RSF rating scale measured participant perceptions of SF and RSF on specific aspects of the task. For example, the participants rated how similar the two programs were in terms of the "ability to thrust the ship forward using the joystick." A Likert-type scale was used, where a rating of 5 indicated *very similar*, a rating of 3 indicated *similar*, and a rating of 1 indicated *very different*. This questionnaire was given after Lesson 17 without advance notice and again after Lesson 19 with notice before the lesson.

Table 2
Means and Standard Deviations for SF/RSF Rating Scale

	Rating Scale	Time One			Time Two		
	1 - - - - 2 - - - - 3 - - - - 4 - - - - 5						
Item	*Very Different* *Similar* *Very Similar*	*N*	*M*	*SD*	*N*	*M*	*SD*
1	Ability to thrust the ship forward using the joystick	124	3.19	1.19	119	3.22	1.26
2	Ability to rotate the ship clockwise	124	3.68	1.12	119	3.55	1.19
3	Ability to rotate the ship counterclockwise	123	3.63	1.16	119	3.53	1.18
4	Ability to accurately aim your ship at desired target	123	3.39	1.21	119	3.30	1.20
5	Ability to fire missiles	124	3.73	1.18	119	3.66	1.20
6	The responsiveness of the joystick to your hand movements	124	3.05	1.30	118	3.12	1.28
7	Rate at which the mines travel on the screen	122	2.94	1.17	119	2.89	1.25
8	Ability of the mines to track the movement of your ship	124	3.15	1.14	119	3.18	1.27
9	Rate at which the fortress rotates in the center of the screen	124	3.32	1.16	119	3.33	1.23
10	Rate at which the fortress fires shells at your ship	124	3.03	1.24	119	3.12	1.27
11	Ability of the fortress to track the movement of your ship	124	3.30	1.23	119	3.15	1.31
12	Ability of the fortress to accurately lock onto your ship's position	123	3.16	1.22	119	3.08	1.25
13	The responsiveness of the mouse to your finger presses	123	3.50	1.24	119	3.66	1.28
14	Ability to fly the ship at a slow rate	123	2.90	1.31	118	2.90	1.25
15	Ability to keep the ship within the two hexagons	123	3.29	1.25	116	3.14	1.19
16	Ability to fly the ship in a clockwise direction while within the two hexagons	123	3.46	1.18	115	3.43	1.19
17	Ability to perform the identify friend or foe function (IFF)	124	3.36	1.21	118	3.49	1.19
18	Ability to energize a friendly mine	122	3.88	1.02	118	3.83	1.10
19	Ability to destroy a foe mine	123	3.56	1.22	118	3.53	1.27
20	Frequency of bonus opportunities	123	3.39	1.21	117	3.60	1.21
21	Ability to select missile bonuses	123	3.84	1.09	118	3.79	1.23
22	Ability to select points bonuses	123	3.83	1.10	118	3.92	1.17
23	Tracking of fortress hits via the vulnerability counter	124	3.73	1.12	119	3.66	1.21
24	Ability to perform rapid double shot to destroy fortress (when vulnerability count is 10 or above)	123	3.73	1.13	118	3.47	1.30
25	Distance between your ship and a shell/mine that is considered a "hit" by the computer	124	3.19	1.29	119	2.99	1.21
26	The sensitivity of the vulnerability reset behavior when rapid double shots are made prior to a 10 count	123	3.32	1.22	118	3.37	1.23
27	Behavior of your ship when hit by a fortress shell	124	3.06	1.37	119	3.03	1.30
28	Behavior of your ship when hit by a mine	124	3.15	1.31	119	3.07	1.29
29	The simplicity of the foe mine identifiers presented prior to game (ABC vs. GNQ)	123	3.40	1.13	118	3.49	1.09
30	The scoring procedures used in the game	124	3.75	1.17	119	3.73	1.21

RESULTS

The SF/RSF rating scale means and standard deviations for each question at each administration were analyzed (Table 2). None of the mean ratings was significantly different from the *similar* category on two-tailed *t* tests for each question.

Figure 3 shows the results for the total score. Relative to the participants who had trained with SF, the participants who had trained with RSF performed lower before training, as was predicted [$F(1,107) = 5.98$, $p < .05$]. However, there were no statistically significant differences on Lesson 1 [$F(1,107) = 1.28$, $p > .05$] or Lesson 10 [$F(1,107) = 1.17$, $p > .05$]. A repeated measures analysis over the baseline and 10 lessons revealed no effect for training condition ($F < 1$). The same analysis revealed an effect for training lessons [$F(10,1070) = 289.26$, $p < .0001$] and an interaction between training condition and training lessons [$F(10,1070) = 4.03$, $p < .0001$], with RSF trainees starting lower and ending higher, as was predicted. The trainees who had trained on the SF and RSF versions did not have statistically significant differences on retention tests, secondary task tests, and keyboard task tests ($F < 1$ for all). The trainees who had trained with RSF did have the predicted advantage over the SF group on the two crossover trials [$F(1,107) = 18.43$, $p < .0001$, and $F(1,107) = 21.05$, $p < .0001$].

Figure 4 shows the results for the points score. Relative to the participants who had trained with SF, the participants who had trained with RSF performed lower before training [$F(1,107) = 12.96$, $p < .001$] and continued to perform lower on Lessons 1 [$F(1,107) = 11.51$, $p < .01$] through 6 [$F(1,107) = 7.21$, $p < .01$]. The RSF trainees did not have a statistically significant difference from the SF trainees on Lesson 7 [$F(1,107) = 2.50$, $p > .05$] and were not significantly different through Lesson 10 [$F(1,107) = 2.47$, $p > .05$]. A repeated measures analysis over the baseline and 10 lessons revealed an effect for training condition [$F(1,107) = 9.39$, $p < .01$] and an effect for training lessons [$F(10,1070) = 153.17$, $p < .0001$], but no interaction between training condition and training lessons [$F(10,1070) = 1.00$, $p > .05$]. Although the mean score of the SF trainees was higher than that of the RSF trainees on all retention tests, the differences were not statistically different, except for Retention Lesson 17 [$F(1,107) = 5.05$, $p < .05$]. The SF trainees also outper-

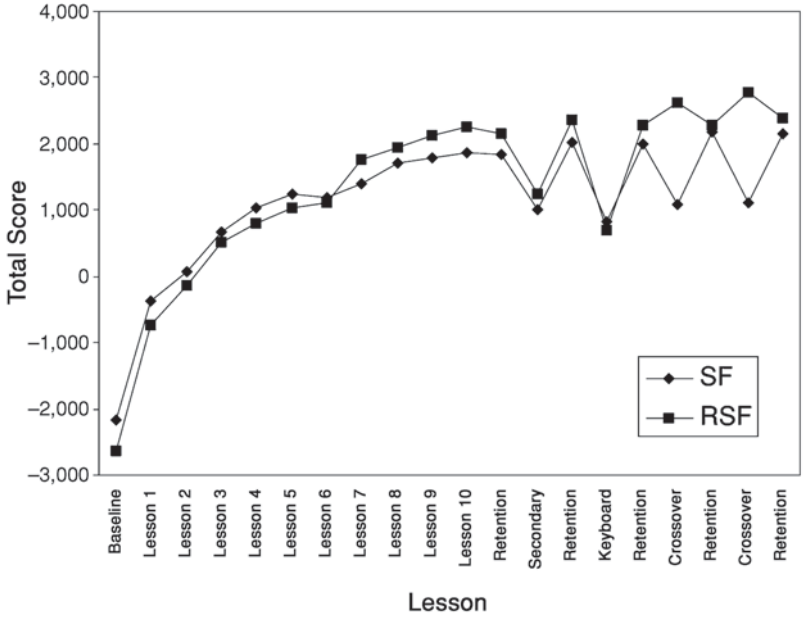

Figure 3. Total scores on the Space Fortress task between software versions.

formed the RSF trainees on the secondary task [$F(1,107) = 4.63$, $p < .05$] and the keyboard task [$F(1,107) = 6.39$, $p < .05$]. However, as was predicted, the trainees who had trained with RSF had an advantage over the SF group on the two crossover trials [$F(1,107) = 8.13$, $p < .01$, and $F(1,107) = 13.74$, $p < .0005$, respectively].

Figure 5 shows the results for the velocity score. Relative to the participants who had trained with SF, the participants who had trained with RSF performed better before training [$F(1,107) = 3.98$, $p < .05$]. However, the RSF trainees did not perform significantly better on Lessons 1 and 2 [$F(1,107) = 3.17$, $p > .05$, and $F(1,107) = 3.61$, $p > .05$, respectively]. On Lessons 3 [$F(1,107) = 6.68$, $p < .05$] through 10 [$F(1,107) = 14.19$, $p < .05$], the RSF trainees significantly outperformed the SF trainees. A repeated measures analysis over the baseline and 10 lessons revealed an effect for training condition [$F(1,107) = 10.52$, $p < .005$], an effect for training lessons [$F(10,1070) = 67.17$, $p < .0001$], and an interaction between training condition and training lessons [$F(10,1070) = 3.28$, $p < .0005$]. The trainees who had trained on the RSF version performed better on all the retention tests than did the SF

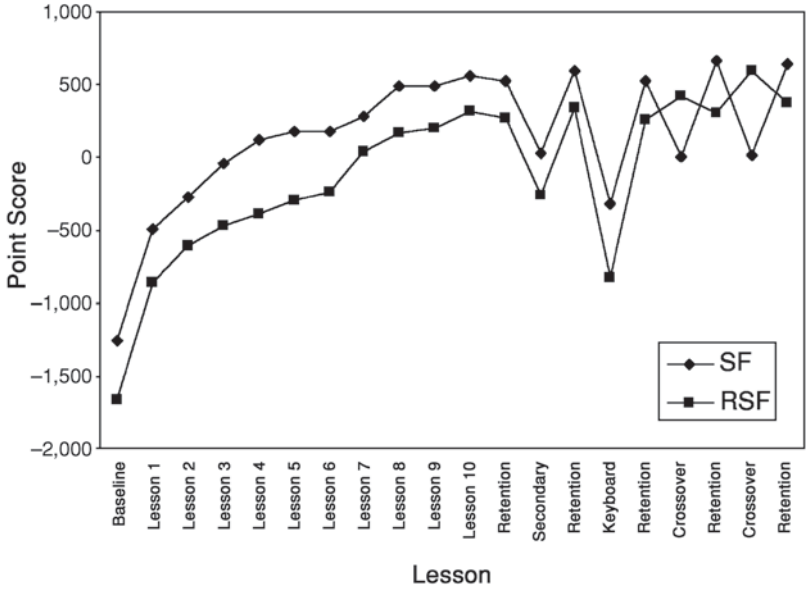

Figure 4. Point scores on the Space Fortress task between software versions.

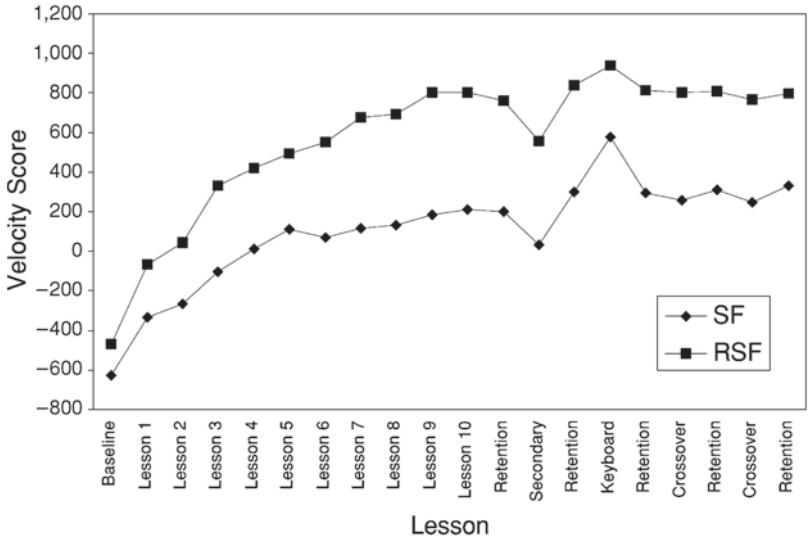

Figure 5. Velocity scores on the Space Fortress task between software versions.

trainees. The RSF trainees also outperformed the SF trainees on the secondary task [$F(1,107) = 13.22$, $p < .0005$] and the keyboard task [$F(1,107) = 10.08$, $p < .005$]. The trainees who had trained with RSF maintained their advantage over the SF group on the two crossover trials [$F(1,107) = 12.64$, $p < .001$, and $F(1,107) = 11.94$, $p < .001$, respectively].

Figure 6 shows the results for the control score. Relative to the participants who had trained with SF, the participants who had trained with RSF performed lower before training [$F(1,107) = 6.56$, $p < .05$] and on Lesson 1 [$F(1,107) = 5.88$, $p < .05$], as was predicted. Although RSF trainee mean scores were lower during training lessons, the differences were not significant between Lesson 2 [$F(1,107) = 3.79$, $p > .05$] and Lesson 10 [$F(1,107) = 0.76$, $p > .05$]. A repeated measures analysis over the baseline and 10 lessons revealed no effect for training condition [$F(1,107) = 3.49$, $p < .07$], an effect for training lessons [$F(10,1070) = 128.52$, $p < .0001$], and an interaction between training condition and training lessons [$F(10,1070) = 1.95$, $p < .05$]. The trainees who had trained on the SF and RSF versions did not have statistically significant differences on any retention tests ($F < 1$), the secondary task [$F(1,107) = 1.82$, $p < .19$], or the keyboard task ($F < 1$). As was predicted, the trainees who had trained with RSF had a significant advantage

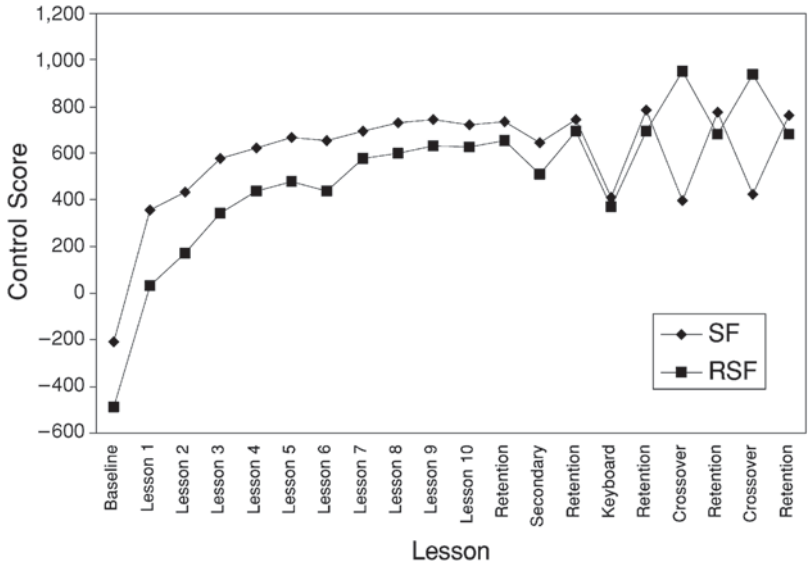

Figure 6. Control scores on the Space Fortress task between software versions.

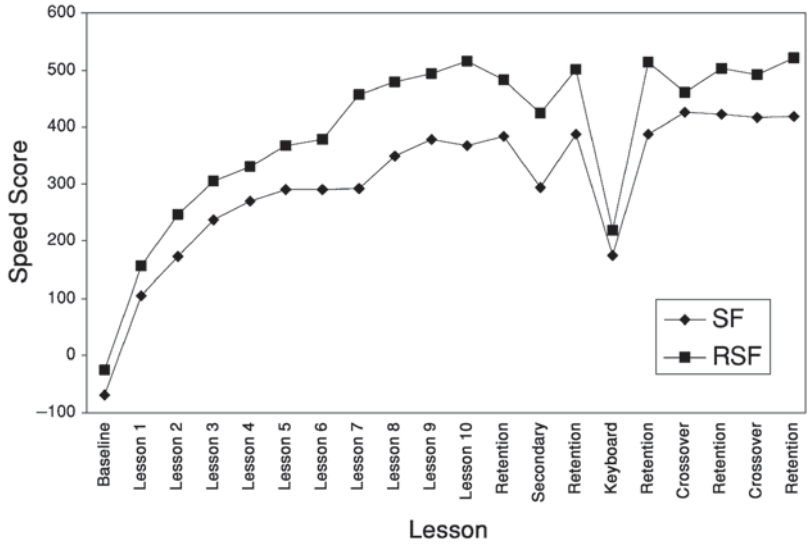

Figure 7. Speed scores on the Space Fortress task between software versions.

over the SF group on the two crossover trials [$F(1,107) = 26.94$, $p < .0001$, and $F(1,107) = 25.22$, $p < .0001$, respectively].

Figure 7 shows the results for the speed score. Relative to the participants who had trained with RSF, the participants who had trained with SF performed lower before training [$F(1,107) = 6.60$, $p < .05$]. Although their mean performance remained lower throughout training, the differences were not significant on Lesson 1 [$F(1,107) = 1.78$, $p > .05$] through Lesson 6 [$F(1,107) = 3.25$, $p > .05$] but were significant on Lesson 7 [$F(1,107) = 11.57$, $p < .001$] through Lesson 10 [$F(1,107) = 9.37$, $p < .01$]. A repeated measures analysis over the baseline and 10 lessons revealed an effect for training condition [$F(1,107) = 6.01$, $p < .05$], an effect for training lessons [$F(10,1070) = 123.69$, $p < .0001$], and an interaction between training condition and training lessons [$F(10,1070) = 2.40$, $p < .01$]. Performance of the SF and the RSF trainees varied on the retention tests. The RSF trainees performed better on Retention Lesson 13 [$F(1,107) = 4.98$, $p < .05$], Lesson 15 [$F(1,107) = 7.31$, $p < .05$], and Lesson 19 [$F(1,107) = 4.27$, $p < .05$]. There was no statistically significant difference in performance on Retention Lesson 11 [$F(1,107) = 3.79$, $p > .05$] and Lesson 17 [$F(1,107) = 2.22$, $p > .05$] between the two groups. The RSF trainees outperformed the SF trainees on the secondary task [$F(1,107) = 10.16$, $p < .005$] but performed as well as the SF trainees on the keyboard task [$F(1,107) = 1.51$, $p > .05$]. The participants who had trained with RSF did not have a significant advantage over the SF group on the two crossover trials [$F < 1$ and $F(1,107) = 2.40$, $p > .05$, respectively].

DISCUSSION

Despite the similarity of SF and RSF, as reflected in objective measurements of physical characteristics and in subjective ratings, RSF corrected SF's error in computing the control score, and RSF may have slightly more precise joystick control. This almost imperceptible difference was noted by the experts before the experiment but was not reflected in the participants' ratings on the questionnaire. On the basis of these two physical differences, performance differences were predicted, and many of the predicted differences were obtained. The total score and all the subscores, except velocity and speed, supported the prediction that participants training on RSF would do worse in the beginning. The total score and all the subscores, except speed, supported the prediction that participants training on RSF would do better on crossover trials. Only the velocity and speed scores were consistent with the prediction that the RSF group would do better than the SF group at the end of training. The failure to support the predicted advantage of RSF over SF in total score and points at the end of training suggests that we do not understand all of the consequences of the small physical differences between RSF and SF. Both the support for predicted differences before training and on crossover trials and the failure to support predicted differences at the end of training indicate that caution must be exercised in any comparison of results with SF and RSF. The observed differences in performance flag the need to test, rather than assume, the comparability of theoretically important variables across SF and RSF.

The disadvantage of having to be cautious when comparing results between SF and RSF is offset by many advantages of RSF. The advantages include reducing errors in SF, running on current-generation platforms, support for team training, support for incorporation of intelligent agents, flexible experiment definition, global management of experiment data, and storing enough data to play back each trial for detailed analysis. The accommodation of intelligent agents enables dynamic adaptive interactions, which allow, for example, simulating environmental dy-

namics, coaches, and teammates and/or adapting training protocols to individual or team performance. Although the playback feature is currently available, it was added after this experiment, so we could not use it to supplement our analysis of predicted differences. It would have been interesting to play back trials to directly investigate predicted differences in the development of the strategy of circumnavigating the Fortress. Such investigations will be possible in future experiments.

REFERENCES

Bodine, T. (2003). *Effects of goals and emphasis change on complex skills*. Unpublished master's thesis, Wright State University.

Donchin, E. (1989). The Learning Strategies Project: Introductory remarks. *Acta Psychologica*, **71**, 1-16.

Donchin, E., Fabiani, M., & Sanders, A. (Eds.) (1989). The learning strategies program: An examination of the strategies in skill acquisition [Special issue]. *Acta Psychologica*, **71**, 23-51.

Frederiksen, J. R., & White, B. Y. (1989). An approach to training based on principled task decomposition. *Acta Psychologica*, **71**, 89-146.

Gopher, D., Weil, M., & Bareket, T. (1994). Transfer of skill from a computer game trainer to flight. *Human Factors*, **36**, 387-405.

He, L., Nanjanath, M., & Volz, R. A. (2004). *RSF agent development guide* (Tech. Rep. TSSTI-TR-7-04). College Station: Texas A&M University.

Mané, A. M., & Donchin, E. (1989). The Space Fortress game. *Acta Psychologica*, **71**, 17-22.

Shebilske, W. L., Goettl, B. P., Corrington, K., & Day, E. A. (1999). Interlesson spacing and task-related processing during complex skill acquisition. *Journal of Experimental Psychology: Applied*, **5**, 413-437.

Shebilske, W. [L.], Goettl, B. [P.], & Regian, J. W. (1999). Executive control of automatic processes as complex skills develop in laboratory and applied settings. In D. Gopher & A. Koriat (Eds.), *Attention and performance XVII: Cognitive regulation of performance. Interaction of theory and application* (pp. 401-432). Cambridge, MA: MIT Press, Bradford Books.

Shebilske, W. L., Jordan, J. A., Goettl, B. P., & Day, E. A. (1999). Cognitive and social influences in training teams for complex skills. *Journal of Experimental Psychology: Applied*, **5**, 227-249.

Yechiam, E., Erev, I., & Gopher, D. (2000). On the potential value and limitations of emphasis change and other exploration-enhancing training methods. *Journal of Experimental Psychology: Applied*, **7**, 277-285.

NOTES

1. See the supporting document in the BRM Archive.
2. Except for the low-level joystick reading routines.
3. A common low-level routine was not found, and hence, they were necessarily different.

ARCHIVED MATERIALS

The following materials and links may be accessed through the Psychonomic Society's Norms, Stimuli, and Data archive, http://www.psychonomic.org/archive/.

To access these files or links, search the archive for this article using the journal (*Behavior Research Methods*), the first author's name (Shebilske), and the publication year (2005).

File: Shebilske-BRM-2005.zip

Description: The compressed archive file contains two files:

comp-meth-paper.pdf, a monograph entitled "Fine-Grained Data Acquisition and Agent Oriented Tools for Distributed Training Protocol Research: Revised Space Fortress" by Richard A. Volz, Jamison C. Johnson, Sen Cao, Maitreyi Nanjanath, Jonathan Whetzel, Thomas R. Ioerger, Barani Raman, Wayne L. Shebilske, and Dianxiang Xu. The monograph describes general methods that can be used to create real-time simulations for human interaction using standard non–real-time operating systems, details on the development of RSF, RSF performance, and directions on how to acquire program code and documentation.

comp-meth-paper.rtf, the same monograph in RTF format.

Authors' e-mail addresses: wayne.shebilske@wright.edu, volz@cs.tamu.edu.

(Manuscript received June 9, 2003;
revision accepted for publication September 11, 2004.)