

Applying Computer Game Tutorial Design Techniques To Simulation-Based Training

Steven C. Rowe

Charles J. Cohen, Ph. D.

Kevin K. Tang

Cybernet Systems Corporation

727 Airport Blvd.

Ann Arbor, MI 48108

734-668-2567

srowe@cybernet.com, ccohen@cybernet.com, ktang@cybernet.com

Keywords:

Computer game, Tutorial, Simulation, Training.

ABSTRACT: *In recent years, computer games have become remarkably complex simulators. Aside from the obvious flight and driving simulations, there are combat simulations that can be used to teach tactics, resource management, and strategic thinking, and even world simulations that can teach social interaction and city building. These complex games require equally complex skills to play them well.*

This complexity has forced game manufacturers to include tutorial levels within their games. Recognizing that simply reading the manual is not enough to fully understand how to operate the game, the manufacturers have written what amount to scenario-based training modules for the game itself. These modules familiarize the user with the user interface, the key components of the game (i.e. what's most important to pay attention to), and provide step-by-step instruction for some of the fundamental tasks required to succeed. Among the techniques used are: text instructional displays, voice-over instruction, computer-controlled "how-to" examples, and after-action review. Often, goals are set that require the student to repeat the scenario until they have achieved a certain level of proficiency.

In this paper, we explore these and other techniques used to make computer game tutorial levels successful, compare them to existing techniques used in contemporary simulation-based training, and propose how to use these lessons to improve future simulation-based training environments.

1. Introduction

It is widely believed that computer games push the technology envelope for software and hardware in the areas of graphics and applied artificial intelligence [1]. However, it may be overlooked that games push forward other less obvious technology. One such area is that of automated training.

Modern computer games are often very sophisticated simulations, or contain sophisticated simulations. These simulations need equally sophisticated user interfaces. Sophisticated user interfaces require that the user be taught how to operate them.

Game designers have risen to the challenge of providing that training, often in an innovative and entertaining way that engages the student and improves their retention of the material. The author believes that

these techniques can and should be applied to "serious" training applications.

In this paper we highlight a number of techniques used by computer games in their tutorials that might be applied to training systems for other simulations. Each item lists some strengths and weaknesses. The games presented as examples are:

- *Neverwinter Nights (NWN)* (see Figure 1.1), a single-player immersive role-playing game (RPG) [2],
- *Ghost Recon (GR)* (see Figure 1.2), a multi-player immersive tactical military simulation game [3],
- *City of Heroes (CoH)* (see Figure 1.3), a massively multi-player immersive RPG [4], and
- *Myth: The Fallen Lords (Myth)* (see Figure 1.4), a single-player real-time strategy (RTS) game [5].

We conclude with recommendations for authors of simulation-based training systems.



Figure 1.1 *Neverwinter Nights* tutorial is driven by textual conversations with characters in the game.

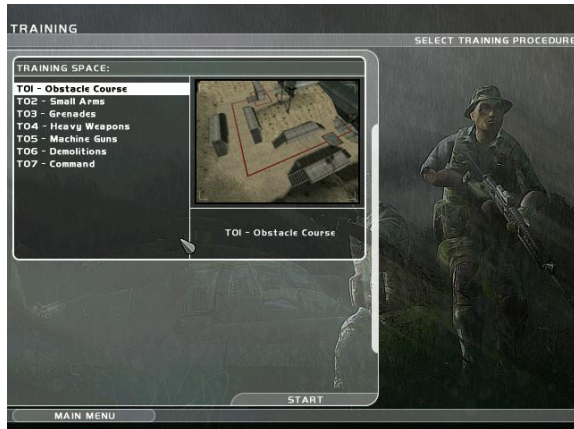


Figure 1.2 *Ghost Recon* allows the user to choose a tutorial topic from a list.



Figure 1.3 *City of Heroes* focuses on interaction with other characters.



Figure 1.4 In *Myth: The Fallen Lords*, unit and camera control are the principal focus of the tutorial because of the number of units that must be controlled in real-time.

2. Techniques

This section presents a number of techniques that are employed for training by computer games.

2.1 Written manuals

The traditional approach for training is to write a detailed technical manual. A good technical manual should be exhaustive of its coverage of the system. If the user has to “go to the manual” to learn something, the chances are excellent that he will find his answer. This completeness requires that the manual contain a useful index and topical table of contents.

Another benefit is that the manual can be used to learn even when the system is unavailable. This assumes that the student has motivation to read the manual and can retain the information without having practiced what has been read. However, this assumption does not generally hold for games, since game players have purchased the game for entertainment, and few people find reading detailed manuals entertaining. Even in classroom settings where external motivation is present, the student is not as engaged as he might be.

Another drawback to relying on written manuals for training is that the manual must be kept in sync with the system. For systems that are constantly being subtly modified for usability, this can be a problem. A manual with incorrect information is worse than no manual at all.

Most modern computer games ship with a minimal manual, more like a pamphlet to get the user started. This pamphlet contains enough information to troubleshoot start-up problems and get the program

running. From there, the user learns from the game's built-in tutorials. More complete written documentation is often sold separately as a "hint book," "guide book," or "strategy guide." These books generally assume that the reader has mastered the interface and discuss more strategic topics.

2.2 Use the simulation engine to code the tutorial

All of the games presented here are scenario-based. Generally, this is accomplished by provided the game developer with scenario creation tools including a scripting language interface to the game engine [6]. Similarly, most scenario-based training systems are scriptable. If the scenario creation tools are sufficiently flexible, the training scenarios can be written with the same tools that are used to provide simulation content.

There are a number of benefits to taking this approach. One is that the user is immersed in the real system. The user practices using the real system at the same time they are receiving instruction. Another is that there is no artificial separation of the training system from the "real" system. Because of this, the same developers that create scenario content for the simulation can also create instructional content with no additional tools to learn. *NWN*'s tutorial (see Figure 2.1) was developed entirely using the scenario conversation editor tool that is included with the game.

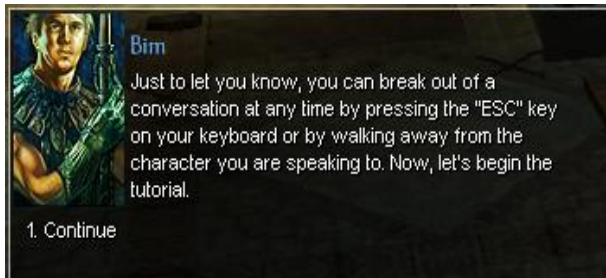


Figure 2.1 *NWN* provides all tutorial instruction through its normal conversation interface.

One drawback of this approach is that it may require extensions to the scripting language to support necessary tutorial features like pop-up text boxes or playing a voice-over.

2.3 Present the material in chunks of gradually increasing complexity

In order for learning to be effective, the lessons must move from the student's short-term memory to long-term memory. The keys to this process are repetition, and practice that takes place at least 30 seconds after a technique has been demonstrated. The reason for the

latter is that if the technique is shown immediately before the student uses it, he can operate completely from short-term memory and there is no need to internalize the information. This is *imitation*, as opposed to *learning*.

All of the example games have divided their tutorials into topical sections. *Myth* starts by teaching you how to select a single unit, then how to select multiple units. After (and only after) the user has mastered these, the tutorial progresses to how to change the camera angle. After this, combat is introduced.

The *CoH* tutorial focuses on the tasks most often performed in the game: how to interact with patrons to get missions, and how to use the character's super powers to defeat villains. It begins by presenting the student with a trivial task: walk down the street and converse with another character. That character in turn presents the student with a slightly more difficult task: pick up an object and then go talk to another character. These interactions are repetitive, as required for learning, but are different enough that the student does not lose interest. The complexity of the assigned tasks continues to increase until eventually they require the character to win a very easy combat scenario. The tutorial culminates in a small but complete "door mission" typical of the actual game play. Completion of this mission is rewarded by the character's promotion and entrance into the actual game world.

2.4 Multi-media instruction

Instruction within the simulation can be provided in three ways: text, audio, and automation. For games that allow conversational interaction with other characters, text is always used. *NWN* augments the text with a voice-over of the text being read. *GR* provides instruction in pop-up text boxes (Figure 2.2) that do not generally appear in the game after the tutorial. These pop-ups are also accompanied by a voice-over of the text being read.

Myth, which has no textual display capability, relies exclusively on verbal instruction. Audio presentation can serve to maintain the student's attention, but one must beware of the implicit assumptions that the student can hear, is fluent in the language, and understands all the words. Thus, it should be viewed as something to augment written instruction, not replace it.

Automation is the practice of actually moving the mouse cursor for the user and simulating button clicks and key presses. None of the games presented here use

automation as a tutorial technique, but the reader may be familiar with it from certain topics of Microsoft Windows® help (see Figure 2.3).

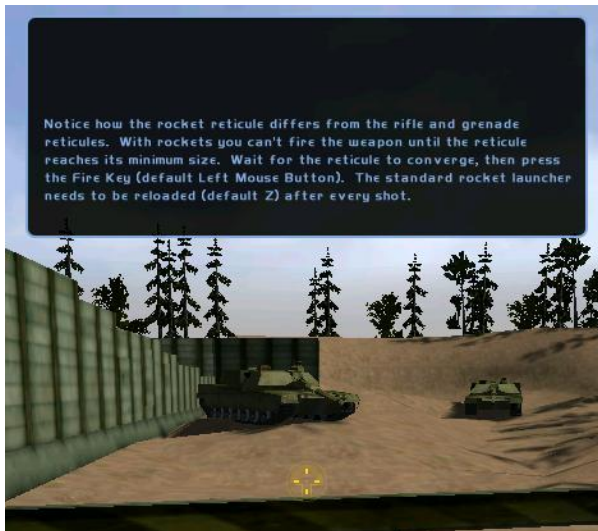


Figure 2.2 GR Uses pop-up text boxes to provide instructional content.

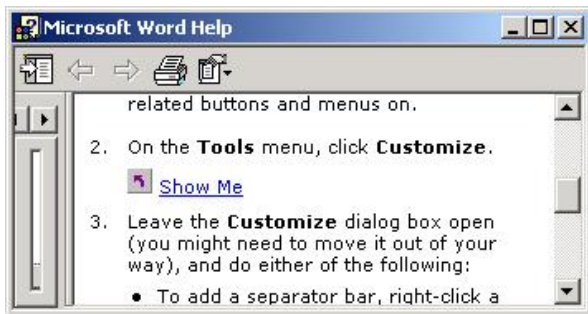


Figure 2.3 Some topics in Microsoft Help include a "Show Me" button that runs an automation sequence.

This technique is useful for very complex user interfaces where the explanation of a multi-step process might become so long as to be confusing. When designing an instruction with automation, care should be taken to leave it under the control of the user. It is generally bad form in a user interface for the system to commandeer the mouse and keyboard without the user's express permission.

2.5 Spatially or contextually separate topics

One innovation of game tutorials is to have designated areas within the simulation where training occurs. In *GR*, for example, there is a heavy weapons range, a

demolitions area, and a rifle range. Each area elicits different training material from the system. In *NWN*, different characters take on the role of instructors for various specializations. For example, there is a character that talks about how to cast magic spells, another that teaches sword combat, another that teaches archery, etc. These instructors are placed in spatially separate areas that are logical within the context of the game world. For example both the sword and archery instructors are in the gymnasium, while the magic instructor is in an arcane library.

For students unfamiliar with this type of environment, it is important that the tutorial guide the student to each lesson area in the proper order the first time through. When the lessons are separated spatially as in *NWN* and *GR*, simply placing the character in a room with one door forces the student to learn how to move and open the door. In *NWN*, early rooms have an exit door that is locked until the student has either taken the lesson or explicitly declined it.

The point of separating the lessons in this way is that by moving to a new area, or speaking to a different character, the student acknowledges a change in broad topic, similar to moving on to a new chapter or section in a textbook. This cognitive separation allows the student to mentally close the current learning experience and open a new one.

2.6 Allow the student to choose topics

If spatially separated areas are used to separate topics, care must be taken not to serialize the lessons. The student should be free to choose among the learning topics. If he must, for example, go through the entire rifle tutorial before being allowed to repeat the explosives tutorial, he will become frustrated and may skip remedial lessons that he would otherwise have taken.

Although *GR* lays its training areas out serially along a spatial path, it is not required that the student stop at each (or indeed, any) of the areas. This allows the student to skip to the advanced lessons with the only penalty being about 15 seconds of extra simulated walking.

NWN gives the student the chance to opt out of any tutorial section by simply saying "No, thanks. I'm not interested," and moving to a different area (see Figure 2.4). Equally important is the option to repeat material that is poorly understood. *NWN* offers this at the end of each sub-lesson.

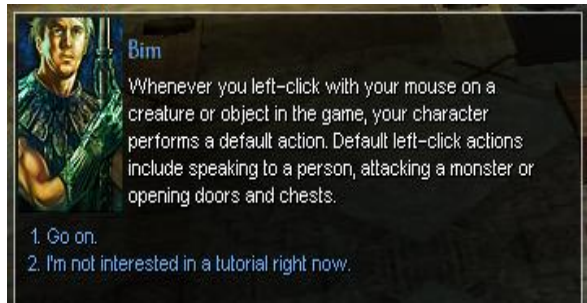


Figure 2.4 All of *NWN*'s tutorial conversations include a choice to opt out of the current lesson.

CoH failed to learn this lesson, and forces the player to go through the entire tutorial, step-by-step, every time they create a new character. Veteran players turn this into a game in itself by trying to set new speed records in completing the tutorial mission, but the fact remains that this sort of disincentive may prevent some players from creating new characters.

2.7 Include testing in the tutorial material

In order to ensure that the student has learned the material, some form of testing is required. Since the material centers on operation of the user interface, it is usually sufficient to ask the user to do something within the simulation. When they have done it, the scripting system is notified, some positive reinforcement is given, and the tutorial proceeds. If some time elapses and the student has not completed the goal behavior, hints can be given. This feedback is especially important in unsupervised training, where the student (who is by definition not a subject matter expert) must make the decision about whether he has received adequate training.

For example, in the *Myth* tutorial you are told how to move a unit. You are then asked to move a unit into an area marked on the ground. After about ten seconds, if the unit is not within the area, the voice-over says "To move a unit, select it with the mouse and then right-click on the spot you want to move to." After another ten seconds it says, "To select a unit, click the left mouse button while pointing at that unit." If at this point you press the right mouse button, the voice-over says "No, the *other* button." The tutorial does not proceed until the unit makes its way into the designated area.

In *NWN*, some lessons don't have testing events, but rely instead on the student's confidence that they know the material. Answering, "Do you understand what I just said?" in the affirmative is sufficient to move on to the next lesson. Answering "no" will provide remedial instruction.

2.8 Provide a safe learning environment

Stress affects a student's ability to learn. Generally, people learn better in a stress-free environment, but this principle must be balanced with the incentive that stress can provide as reinforcement.

NWN and *CoH* conduct their tutorials in an area that is spatially separated from the hazardous "real world" of the main game. It is very difficult to have one's character die in these circumstances, but if it does happen, the character is returned to the beginning of the lesson with a chiding message.

Myth and *GR*, being grittier combat simulations, allow the student's persona to be hurt or even killed in the tutorial. Whereas they don't have hostile characters actively trying to kill the character, being too close to a satchel charge or anti-tank rocket explosion will kill the character. Again, the character is returned to the lesson with a chiding message, providing gentle negative reinforcement for the suboptimal behavior.

The key benefit of providing these relatively safe environments during the tutorial is that the student will be free of distraction during the instructional phase. A practical consideration is that in a simplified environment there are relatively few decision paths, which make tutorial lessons easier to script.

To apply this lesson to non-game simulations, the instructor should provide an environment where mistakes are permitted, perhaps with some gentle negative reinforcement. Formal testing, i.e. lists of questions that demand to be answered, causes anxiety in most students. To avoid this anxiety, testing should be separate from material presentation and be disguised as something other than a test (as in *Myth*).

That being said, remember that stress is an integral part of some situations, and handling that stress should be part of the lesson plan. For example, physical stress on a pilot in a high-G maneuver, stress from cognitive workload in an air traffic control simulation, and anxiety in a combat simulation are to be expected and must be taken into account. The key is to provide the appropriate stresses, and not create artificial ones like test anxiety that do not reflect the environment for which the student is training.

2.9 Reinforce learning with rewards and/or punishments

Reinforcement is an important motivator in learning. Generally, positive reinforcement is better for student morale than negative reinforcement, so the tendency is

to see more of the former than the latter. This is especially true in the context of computer games, since they are supposed to be fun, and negative reinforcement techniques tend to be the opposite.

NWN rewards you for demonstrating your ability during the tutorial. For example, in the rogue (read “thief”) training room if you succeed in picking the lock to a chest, you get to keep what you find inside. *CoH* rewards the character with experience points for successfully completed tutorial sections. Rewards in games for younger players often take the form of encouraging words or a fun sound effect. Other games make new features available as a reward for progress through the tutorial.

The reward concept can be applied even to non-game simulations by awarding points to the student for the mere completion (as opposed to good/bad performance) of training sections. For example, one could arbitrarily assign the value of 5 points to tutorial section. When a student completes that section, whether on the first go or after numerous attempts, if the system responds “Points awarded: 5/5,” then the student has been positively reinforced. A perfect score encourages everyone.

As noted in section 2.8, some negative reinforcement is included in the form of chiding by the game’s voice-over in the event of catastrophic failure to learn the lesson. This should be kept to a minimum, as even the small amount present in these games is a drain on morale. A lesson learned very early in the computer game industry is that positive phraseology and encouragement is best; say “Sorry that you died. I’m sure you’ll do better next time!” instead of “You idiot! Try not to die again.”

3. Conclusion

We have presented a number of techniques employed by computer game tutorials that can be applied in non-game simulation environments.

In summary:

- Provide, but don’t rely upon, thorough written documentation. Provide a written quick-start guide to get the student into the simulation, and then let the tutorial take over.
- When designing a simulation-based training system, include the ability for the instructor to create training scenarios. This may require a few extra features that are not strictly part of the simulation.

- When designing a training scenario, divide the material into small lessons. Present lessons of gradually increasing difficulty. Verify that the student has learned each lesson before proceeding to the next.
- When laying out automated tutorial material, be sure to allow the student to bypass lessons he has already mastered. If applicable, try to use spatial separation of topics to enhance learning.
- When designing course material, minimize test-related anxiety by presenting test materials in non-traditional ways.
- Training material should contain plenty of positive reinforcement and perhaps a little negative reinforcement. Rewards as simple as a pleasant sound or points awarded are effective.

These techniques are backed by principles of cognitive science, and have moreover been demonstrated to be effective. Because computer-based simulations share a great deal in common with computer games, employing these training techniques can improve performance of students in non-game simulations.

4. References

- [1] Wen, Howard. “Hi-Tech Games: Pushing the Envelope in 2001 and Beyond”, <http://www.gamespot.com/gamespot/features/pc/hitech/>
- [2] *Neverwinter Nights* was developed by Bioware and is published by Atari. <http://NWN.bioware.com/>
- [3] *Ghost Recon* was developed by UbiSoft and published by Red Storm Entertainment. <http://www.ghostrecon.com/>
- [4] *City of Heroes* was developed by Cryptic Studios and is published by NCSOFT. <http://www.coh.com/>
- [5] *Myth: The Fallen Lords* was developed and published by Bungie. <http://www.bungie.net/>
- [6] *AI Game Programming Wisdom*, section 5, ISBN 1-58450-077.

Author Biographies

STEVE ROWE is a senior software engineer at Cybernet Systems Corporation. Besides being an avid game-player, he has taught in the classroom and also worked in the development of automated teaching materials at KnowledgeNet Corporation.

CHARLES COHEN is the Vice President of Research and Development at Cybernet Systems Corporation. He specializes in Human cognitive modeling and non-traditional human-computer interfaces including

gestures. His Ph. D. from the University of Michigan was awarded for his work on computer interpretation of human gestures and facial expressions.

KEVIN TANG is a research engineer at Cybernet Systems Corporation. His early fascination with computer games fostered his interest in human-computer interaction. His expertise in Human-centered

design is currently applied towards developing cogent techniques for tutorial-based training and simulation. Mr. Tang holds a B.S.E of Computer Science & Engineering from the University of Michigan.