



© John S. Dykes Collection c/o theisspot.com

Out of the Video Arcade, into the Office: Where Computer Games Can Lead Productivity Software

Although computer games were originally confined to the displays of large and expensive university computers, they are now ubiquitous, found in living rooms, on mobile phones, and even 35,000 feet in the air on airline entertainment systems. With this ubiquity has come inquiry, and computer game research is now a bona fide scholarly endeavor as evidenced by journals covering the field (e.g., *Game Studies*), doctoral programs centered around the discipline (e.g., Center for Computer Game Research, IT University of Copenhagen), and, of course, this column.

Of course HCI researchers have been conducting computer game research for decades, despite what

seems like recent excitement around this “new” discipline. Carroll, a pioneer in this approach, advocated using the motivational techniques found in computer games to improve the productivity-software user experience [3].

Another pioneer, Malone, studied what makes computer games enjoyable and suggested ways for making productivity software similarly enjoyable [12]. This pioneering work

has led to further research that suggests new, exciting directions for productivity software. For example, Houser and DeLoach described some of the instructional aspects of computer games and discussed how these features might be used to better productivity software

Jerrod Larson
University of Washington
jerrodl@u.washington.edu

Editor: Dennis Wixon ♦ Microsoft Game Studios ♦ denniswi@microsoft.com

[10]. Dyck et al's analysis of computer games documented the way games support effective user interaction and recommended that similar techniques be used in productivity software [9]. Chao has even presented a modification of the popular shooting game Doom in order to manage UNIX processes [8]. All of this work shares a common theme: Productivity-software designers can learn much from computer games. But while this effort has yielded important findings, and the field appears to be gaining attention, cross-pollination between computer game design and productivity-software design is still rather limited: As Dyck et al point out, "...games have evolved enormously, but many of their contributions have not been applied to general [productivity software] UIs."

There are a number of persistent usability issues in productivity software, and it is prudent to investigate whether solutions can be found in computer games. One of these issues is how a single piece of software can effectively meet the needs of both novice and expert users. Decades of usability studies have demonstrated that expert and novice software users have vastly different needs with respect to software, and theories and research on cognition help explain why [11, 13, 16].

Admittedly oversimplifying things a bit, expert software users tend to be able to handle (and appreciate) many options in their software and many tools readily available in their software interfaces. Expert users can interact with rich UIs without becoming overwhelmed, in part because they know where to focus their attention, and they know which elements in the UI are most likely to facilitate whatever task they would like to undertake. Novice users, on the other hand, tend to be overwhelmed by options, and they have limited ability to discern those important UI elements from the unimportant. Novice users, therefore, typically benefit from simple interfaces with few choices, while expert users typically desire more-complex interfaces with many choices. (There are, of course, exceptions to this rule: Both novice and expert users seem fond of Google for the simplicity of its interface). Given the apparent tension between the needs of novices and experts with respect to productivity-software UI, it seems relevant to investigate whether computer game designers have found a resolution to this tension—and if they have, what does it look like?

This article presents some strategies employed by

computer games to provide players with appropriate levels of game play regardless of player skill. Before doing so, however, it is important to acknowledge the fact that this discussion will at times highlight concepts that are, on the surface, similar to ideas suggested by HCI researchers in the past. By describing the ways in which these concepts achieve reality in computer games, we can further support previous work on computer games (some already mentioned) and add substance to their claims. Renewing interest in the groundbreaking work that has not yet made its way into standard productivity-software design is ultimately one of the goals of this article.

DIFFICULTY-REGULATION STRATEGIES

Computer games provide players with an appropriate level of game play by way of a number of distinct strategies. These strategies can be collectively called *difficulty-regulation strategies*. At times, the strategies share characteristics with the concepts of "balance" and "challenge" and the practice of "level design" described in game literature, though, as we shall see, they are much broader [14, 15, 2].

When computer game designers effectively use difficulty-regulation strategies, a novice can enjoy a game as much as an expert because implicit in the game's design are mechanisms to ensure players are: 1) provided an appropriate level of game play difficulty to roughly match their individual expertise; or 2) provided motivation for advancing through progressively more difficult game play; or 3) provided a clear progression of difficulty as skill is demonstrated; or 4) provided additional instruction or tools as needed.

These strategies are not particular to a type of computer game, either: Difficulty-regulation strategies are common across all computer game platforms (e.g., PC-based, console-based), ages (e.g., contemporary to 25 years old), and genres (e.g., driving games, first-person shooters). That said, these strategies are relatively uncommon in productivity software.

Difficulty-regulation strategies are found in several permutations. For the sake of brevity, only five will be described here: user difficulty selection, implicit and explicit stage progression, tool unlocking, hinting, and tutorial. Although most computer games primarily use one of these forms of difficulty regulation, many use a combination of some or all of them. In fact, the differ-

© ACM 1072-5220/07/0100 \$5.00



ABOUT THE AUTHOR Jerrod Larson works as a human factors designer at a U.S. aerospace company, where he conducts user research and design for various kinds of Web services and software applications. He earned an MS in technical communication from the University of Washington, where he is currently pursuing a PhD in education.

ences between some of the strategies are quite subtle, and the boundaries around each are debatable; the objective here is to draw only rough characterizations of each such that they might prove useful to HCI researchers and productivity-software designers.

This classification is not exhaustive; it does not address, for example, artificial-intelligence techniques championed by computer game researchers like Charles et al, techniques that are already informing game design and are, by definition, difficulty-regulation strategies [7]. Indeed, the strategies described herein are decidedly low-tech and have seen decades of use in computer game design.

User Difficulty Selection. One common difficulty-regulation strategy can be called user difficulty selection. When a game employs user difficulty selection, the user must choose her skill level (or how difficult she would like the game to be) at the game's outset, selecting between categories akin to beginner, intermediate, or advanced. This selection determines the difficulty of game play. In driving games, the driving courses get tighter and the opponents drive faster on more difficult levels; in shooting games the villains shoot more quickly and accurately on more difficult levels; in logic games the puzzles are more complicated, and the user has less time to complete them on more difficult levels.

Conversely, "easy" stages feature easier, more forgiving game play. On easy stages, players are presented with slower game play, less challenging adversaries and easier puzzles. Easier levels may also feature other difficulty-regulation strategies like hinting and tutorials in order to provide additional support to novice players.

In user difficulty selection, players have the power to choose what level of game play they want. That an expert player might choose the easy setting is not of chief import; what matters most is that players have the ability to choose the experience they would most enjoy.

Implicit and Explicit Stage Progression. *Stage progression* is another common difficulty-regulation strategy in computer games. In stage progression, the game begins with the easiest game play regardless of player skill. The longer the player can play without error, the more difficult the game becomes.

There are two ways in which this advancement occurs in computer games: implicitly or explicitly,

Sometimes stage progression is *implicit* and relatively unobtrusive: Screen action just gets faster or game play becomes otherwise more difficult the longer the player plays. But computer games may instead *explicitly* announce stage increases. In explicit staging, when a player completes a stage the computer game may stop to congratulate him, it may declare a stage complete, or it may simply announce that the next stage is commencing. Moreover, any combination of these options is possible.

In addition to regulating difficulty, explicit stage progression may also help to motivate players by providing them feedback for successfully completing a stage (see [3, 1, 17] for discussions about motivation in computer games).

Tool Unlocking. A less-common difficulty-regulation strategy is *tool* unlocking. With tool unlocking, games allow players to perform special functions or gain access to more-effective tools after they reach certain milestones. These functions allow users to play more effectively or to reach higher scores or levels in the game. Oftentimes, an audio-visual cue announces this unlocking, at which time the player can try out the new functionality. In a karate game, for example, tool unlocking might mean a player can "unlock" the ability to do a special kick after achieving a certain score. In a driving game, players may be able to "purchase" faster and more agile cars as they advance through the game. Tool unlocking is similar to Carroll's productivity-software "training wheels," which advocates for computer interfaces staged in terms of complexity to coincide

with user expertise [4, 6, 5].

Hinting. Similar to Dyck et al's concept of "calm messaging" and Houser and DeLoach's "performance coaching" and "brief instruction," *hinting* provides a brief audio or visual message of instruction aimed at improving a player's performance or helping her when she seems to be having trouble [10]. These hints happen during game play while the player is being scored, but are quite brief—appearing just long enough for players to read or hear them.

Tutorial. The last difficulty-regulation strategy described here is the *tutorial*. Many computer games feature sophisticated tutorials to familiarize the player with the basics of game play and control. Generally, tutorials provide the player with simple instructions on

*The five
difficulty-regulation
strategies are meant
only to inspire
creativity in
productivity-
software diagnosis.*

how to navigate and play the game. But computer game tutorials frequently go a step further: They often monitor and evaluate a player's performance as she performs tasks, providing her with new instruction only after she has demonstrated competence with a previous task. For example, in a shooting game the tutorial might teach the player how to shoot her weapon—telling her what keys to press and then asking her to try on her own. Only after she successfully completes the task does the tutorial provide her instruction on a more complex task, like reloading her weapon. The tutorial continues instructing the player until she has learned—and has demonstrated—all the skills necessary for playing the game successfully.

Lastly, unlike the strategy of hinting mentioned earlier, tutorials happen outside of game play and when the player is not being scored.

IMPLICATIONS FOR PRODUCTIVITY SOFTWARE

The aforementioned difficulty-regulation strategies are quite common in computer games, but I would suggest they are relatively uncommon in productivity software. Computer games in general and difficulty-regulation strategies in particular might influence productivity software. In fact, when productivity software has employed a particular strategy (the tutorial, for example), it is oftentimes qualitatively different than the computer game equivalent described here. Current productivity-software tutorials, for example, are fantastically low-tech compared with their computer game brethren; most are little more than a series of animations the user must passively watch. One might imagine future productivity-software tutorials providing task-based instruction to users, like in computer games, advancing users to new instructions only when they have demonstrated proficiency.

User skill selection, explicit and implicit staging, tool unlocking, hinting, and tutorial are five strategies used in computer games that allow games to meet the needs of both novice and expert players. These strategies may provide productivity-software designers with new approaches to dealing with the conflicting goals of expanding software functionality for expert users while simultaneously keeping software easy to use for novice users.

The difficulty-regulation strategies are admittedly

incomplete: They are meant only to spawn creativity and new ways of thinking in productivity-software designers. They do, however, provide concrete examples of what difficulty regulation looks like in computer games, and they hopefully begin to suggest how such strategies could work in productivity software.

Computer games do have much to teach productivity-software design. Not only does the work of game-research pioneers from the HCI community remain relevant, but computer games doubtless offer a relatively rich source for further HCI research. ♦

REFERENCES 1. Bowman, R.F. (1982). "A 'Pac-Man' theory of motivation: tactical implications for classroom instruction," *Educational Technology* 14-16 Sept. 2. Byrne, Ed. *Game Level Design*. Charles River Media, 2005. 3. Carroll, J. M. (1982). The adventure of getting to know a computer. *IEEE Computer*, 15 (11), 49-58. 4. Carroll, John M., Carrithers, Caroline. Training Wheels in a User Interface. Communications of the ACM. Volume 27, Issue 8 (August 1984) 5. Carroll, J.M. The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill. MIT Press: Cambridge, 1990. 6. Catrambone, Richard, Carroll, John M. Learning a word processing system with training wheels and guided exploration. Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface (CHI '87) April 1987. 7. Charles D, Kerr A, McNeill M, McAlister M, Black M, Kücklich J, Moore A, & Stringer K (2005). Player-Centered Game Design: Player Modelling and Adaptive Digital Games 8. D. L. Chao. Doom as an interface for process management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI Letters* 3, 1), pages 152-7. ACM Press, New York, 2001. 9. Dyck, Jeff, Pinelle, David, Brown, Barry, Gutwin, Carl. "Learning from Games: HCI Design Innovations in Entertainment Software" 2003 10. Houser, Rob, DeLoach, Scott. Instructional Design Lessons Technical Communicators Can Learn From Games. IEEE Professional Communication Conf. 1996 11. Mack, Robert L., Lewis, Clayton H., Carroll, John M. Learning to Use Word Processors: Problems and Prospects ACM Transactions on Office Information Systems, Vol. 1, No. 3, July 1983, Pages 254-271. 12. Malone, Thomas W. 1982. "Heuristics for Designing Enjoyable User Interfaces: Lessons from Computer Games." Conference on Human Factors in Computing Systems. Proceedings of the 1982 conference on Human factors in computing systems. Pages: 63-68 13. Norman, Donald A, Draper, Stephen W. User Centered System Design: New Perspectives on Human-Computer Interaction. Hillsdale: Lawrence Erlbaum Associates, Inc. 1986. 14. Rollings, Andrew, Adams, Ernest. On Game Design. Indianapolis: New Riders. 2003 15. Rollings, Andrew, Morris, Dave. Game Architecture and Design. Indianapolis: New Riders. 2004. 16. Sweller, J. Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, 12 257-285. 1988. 17. Wong, K.K. (1996) "Video game effect on computer-based learning design. *British Journal of Educational Technology*, Blackwell Publishers, Vol. 27, No. 3, pp. 230-232.