Doom as an Interface for Process Management

Dennis Chao
Computer Science Department
University of New Mexico
Albuquerque, NM 87131 USA
+1 505 277 5957
dlchao@cs.unm.edu

ABSTRACT

This paper explores a novel interface to a system administration task. Instead of creating an interface *de novo* for the task, the author modified a popular computer game, Doom, to perform useful work. The game was chosen for its appeal to the target audience of system administrators. The implementation described is not a mature application, but it illustrates important points about user interfaces and our relationship with computers. The application relies on a computer game vernacular rather than the simulations of physical reality found in typical navigable virtual environments. Using a computer game vocabulary may broaden an application's audience by providing an intuitive environment for children and non-technical users. In addition, the application highlights the adversarial relationships that exist in a computer and suggests a new resource allocation scheme.

Keywords

Cyberspace, Doom, first-person shooter, games, metaphors, operating systems, Post-Modernism, 3D user interfaces, vernacular, video games, visualization

INTRODUCTION

Those who use computers inevitably encounter some of the metaphors that allow for easier assimilation of abstract concepts. The desktop metaphor is so pervasive that most users hardly notice it [9], but the richness of the science-fiction version of cyberspace is largely confined to research laboratories and Hollywood. The application described in this paper is an initial step towards bringing a richer environment to personal computers.

There is a large gap between how we think about performing actions on our computers and how we actually perform them. For example, people who need to manage processes on a UNIX system think about the "daemons" spawning children that may need to be "killed" or "blown away." This violent language suggests a metaphor for process management:

SIGCHI'01, March 31-April 4, 2001, Seattle, WA, USA.

Figure 1: The PSDoom interface.

a first-person shooter game. Each process can be represented as a monster, and interacting with the monsters would affect the corresponding processes. The implementation of this metaphor and the great interest it generated reveal interesting insights about our computers and our society.

IMPLEMENTATION

The Doom process manager (PSDoom) is a modification of the game Doom [8] that displays representations of the processes running on a machine. Rather than using standard text-mode UNIX tools to view and manipulate processes, one surveys and shoots at a room full of bloodthirsty mutants, as shown in Figure 1. When a user starts PSDoom, currently running processes are instantiated as "process monsters" in a single room in a "dungeon." These monsters have their associated process' name and id printed on them. The program periodically polls the operating system to add newly-created processes to the game. The user may choose to view the processes from a balcony above the room, as shown in Figure 2, or to enter the room to interact with them. If the user inflicts a wound upon a process monster, the corresponding process' priority is lowered to give it fewer CPU cycles. When the monster accumulates enough damage and is killed, the associated process is also killed.

PSDoom inherits the rest of its behavior from the original Doom, and play is not noticeably affected. Monsters attempt to attack the player and each other. The hostility of the mon-

Figure 2: The view from the balcony in PSDoom.

Figure 3: Killing a process in PSDoom.

sters and the user's limited ammunition are disincentives to attack them. Conflict among process monsters could help regulate heavily-utilized systems by making crowded rooms have higher mortality rates. Killing random processes on an extremely loaded system is not an uncommon operating system strategy. When the user is "killed," he or she will be healed and placed at the entrance of the dungeon with a pistol and a modest amount of ammunition.

Doom was chosen for this project for two reasons. The first is that it is a classic game, familiar to most system administrators. The second is that its source code was recently released under the GNU General Public License (GPL) [13]. This license not only allows the author to modify the source code, but it guarantees that future derivatives of the author's work will be available to the public.

RESULTS

PSDoom received a surprisingly large reaction even though it was not publicized [24, 26, 4, 17]. Less than a week after the initial version of the code was written, the project's website was attracting tens of thousands of visitors per day. Approximately 800 responses were e-mailed to the author or www.slashdot.org within the first two months. Of these responses 27% praised the project, 23% offered suggestions for improving PSDoom, 10% found the project funny, 10% reported technical problems, 8% related PSDoom to science fiction or to the future of interfaces, 1% disliked the project, and 0.6% were frightened by its implications.

Users found the interface intuitive. One can quickly assess machine load by seeing how crowded a room is. The command line methods to slow down and kill processes are different, while PSDoom unifies them – shooting a monster with a small weapon slows down or "wounds" the corresponding process, and repeated firings or the use of a large weapon kills the process, as shown in Figure 3. The violence inflicted upon the monsters reflects the violent terminology of UNIX commands.

A significant problem with the current implementation of PSDoom is that monsters are much more likely to attack each other than expected. This causes many windows to mysteriously disappear as the program runs. For the same reason, the computer is prone to crashing because certain processes are vital to the computer's operation and should not be killed.

Many users want a larger variety of monsters in PSDoom. If larger or more important processes were represented as larger monsters, it would be easier to assess the machine load at a glance. If these monsters were also more powerful, they would be less likely to be killed by accident and be more able to defend themselves against the player. Several users made similar suggestions for altering the appearance of monsters based on certain attributes. Processes that take more memory could appear wider, while those that take more CPU time can appear taller. Sleeping processes could be represented by napping monsters.

To address some of these requests, I added code to make some of the more important processes "Barons of Hell," the largest monsters in the game. Unfortunately, they had a tendency to quickly kill all of the other processes, and the user could not interact with processes for more than a few seconds before his or her avatar is killed. Making the monsters less aggressive would allow the user to navigate among processes more easily as well as make the computer more stable.

DISCUSSION

The enormous interest that PSDoom generated naturally raises the question of why people find it so compelling. Perhaps even more interesting than the application itself is the set of issues that it raises.

Cyberspace Environments

In 1981, Vernor Vinge introduced the concept of cyberspace to the reading public in his novella *True Names* [30], in which characters could plug into a virtual universe where their actions in a fantasy world mapped to performing sophisticated actions on the network. For example, navigating a treacherous path through a swamp in cyberspace could gain the user entry into a high-security network in the real world.

The use of a cyberspace environment can make performing complex tasks easy if the mapping between the actions in the virtual world and their effects in the real world is intuitive. Communication among users may be facilitated because interacting with a group of avatars can be made as natural as interacting with a group of people. Doom places the user into a 3D first-person perspective environment that rapidly engages the senses. Its interface is particularly suitable for the task of process management because many system administrators are familiar with the game and it realizes the common metaphor of killing processes.

In PSDoom, actions that affect the processes take time and effort. This philosophy runs counter to intuition, which tells us to make everything as easy to do as possible. In a command line interface, all actions take approximately the same amount of work, making it just as easy to erase all of one's files as it is to remove a single one. In this cyberspace environment, the players are not omnipotent, so actions take an amount of effort proportional to their effects. People with different levels of authority can be given weapons of different strengths. An experienced system administrator can be given a large gun, while the beginner may be forced to deal with monsters with his or her bare hands. It would take a foolhardy player to attack a room full of monsters, just as a newbie should not kill a bunch of important processes. A more experienced sysadmin would have time to stop a novice who is trying to kill the wrong process.

The Software Vernacular

Current applications often do not leverage the rich vocabulary of contemporary mass media and video games. Their user interfaces are usually austere, reflecting the limitations of the machines of two decades ago. Our rectilinear desktops feature a Machine Aesthetic, not a human one. There have been attempts to create friendlier interfaces based on familiar physical settings (e.g. General Magic's MagicCap [21] (Figure 4) and Microsoft Bob [2] (Figure 5)), but they have often been derided as condescending or confining [12]. The failure of these more humane interfaces may be due to the denial of the computer world's own vocabulary that is distinct from that of everyday life's. In many ways, computer games have become the vernacular of the computer world because they are what children and many non-technical adults use.

Interfaces that adhere strictly to a metaphor based on the physical world are in danger of becoming simulations of reality instead of useful analogies [18]. One problem with simulations is that one must closely adhere to all the restrictions of the real system. For example, in Microsoft Bob it is disturbing to see pieces of furniture hovering over the floor in apparent violation of gravity. In contrast, it is more or less natural to have an arbitrarily large number of windows open on a computer desktop, which is a less-literal, and thus less-confining, metaphor. Cartoon-like animations have been suggested as a compromise between straightforward representation and artistic expression [6]. The idioms of cartoons form an effective vocabulary for concise visual communication with the user, but cartoons lack a crucial element of user interfaces: interactivity. Computer games have successfully integrated concise, stylized visual representations and feedback from the user.

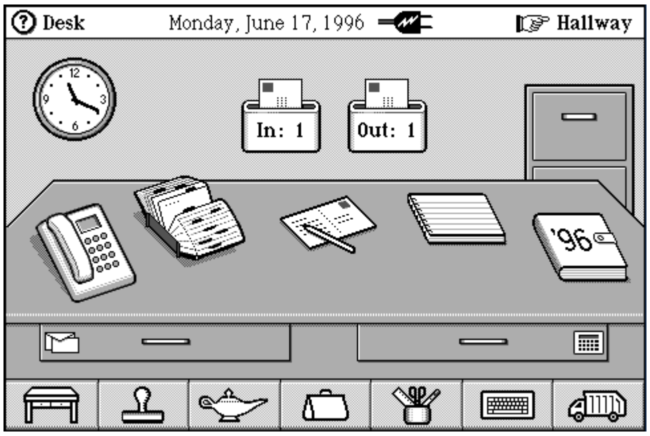

Figure 4: General Magic's MagicCap.

Figure 5: Microsoft Bob.

The desktop metaphor derives much of its power from the fact that users were traditionally office workers who spent much of their days at actual desks [12, 20]. However, in recent years computers have entered the home, and the current generation of users is gaining increased exposure to computers that are not merely office appliances. Children are now growing up on MTV and Nintendo and are therefore more literate in the languages of video games and mass media than in that of the traditional office milieu. Designers should consider taking advantage of the vocabulary of popular culture instead of trapping the next generation in dioramas of suburban homes and offices.

Like the Post-Modern architects, I would like to blur the distinction between "high" and "low" culture [29, 3] and bring popular culture into "serious" computer applications. By adding familiar cultural elements to our interfaces, we can make them appealing to the common user and children, the next generation of users. These elements would bring a sense of playfulness and ease that are usually absent from our button- and toolbar- laden windows.

Several attempts have been made to introduce game-like elements to the desktop, including the recent UBUBU Universe [27]. UBUBU replaces the hyperlinks and folders on the desktop with navigable planets in a solar system. While the concept is superficially similar to Bob, the choice of flying through space rather than walking around a house may engage the general public instead of being perceived as patronizing. Companies like UBUBU that design products for web-surfers are using playful game-like designs to tap into the lucrative demographic of computer game players.

An Adversarial Operating System

PSDoom presents an unusual perspective on our interactions with computer programs. It is unique in that it allows the processes to fight each other and the user. Thinking of our computing environments as being adversarial can be enlightening. Processes are competing for machine resources, such as memory and CPU time, and PSDoom makes this competition explicit. The user may want to kill processes to free needed resources, so from the process's perspective, the user may be its greatest threat. The processes are given the ability to shoot back and defend themselves.

Operating systems attempt to satisfy users and processes through fairness policies for resource allocation. An adversarial operating system would invert the responsibility for sharing resources by forcing the users and processes to compete directly with each other. It would simply maintain a stable arena in which processes could compete. An offensive and defensive arms race could ensue, but the system would not necessarily degrade to an hostile, anarchic environment. Under the right conditions, processes will cooperate rather than compete [1]. They would find it in their best interests to only attack when necessary because unduly belligerent processes would become injured in frequent conflicts and die.

Violence in Computer Games

It is unfortunate, especially in the light of recent schoolyard tragedies, that first-person shooters are so popular. Even though studies on the effects of violent media on youth are not conclusive [11], interface designers must proceed with caution when adding potentially aggressive aspects to interfaces. One must take into consideration the age of the expected user base and the possible use of non-aggressive alternatives.

Lucasfilm's Habitat [14], an early multi-user virtual environment, faced these ethical dilemmas with the introduction of "weapons." These weapons could "kill" avatars, though death in Habitat is no more serious than death in Doom – the avatar loses possession of the items he or she was carrying and is sent home. About half the players enjoyed this aspect of the environment while the rest were opposed to it [23]. To stem anarchy, the players established their own churches and laws. In Ultima Online [28], an online fantasy environment, player-killing is an integral part of gameplay for some, but many value the complex social interactions with their fellow players even more.

```
 12:12pm  up  2:29,  3 users,  load average: 0.08, 0.04, 0.00
51 processes: 48 sleeping, 3 running, 0 zombie, 0 stopped
CPU states:  1.5% user,  0.5% system,  0.0% nice, 97.8% idle
Mem:   63356K av,  59392K used,   3964K free,  14928K shrd,   4096K buff
Swap: 128516K av,   5460K used, 123056K free                  38904K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  273 root      10   0 14988 6036  1396 R       0  0.9  9.5   2:41 XF86_Mach64
  277 dlchao     2   0  1940 1824  1108 S       0  0.5  2.8   0:09 WindowMaker
  619 dlchao     2   0  1172 1172   700 R       0  0.3  1.8   0:00 top
  310 dlchao     0   0  1136  968   772 R       0  0.1  1.5   0:02 xterm
    1 root       0   0   112   68    52 S       0  0.0  0.1   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kupdate
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kpiod
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
  119 root       0   0   264  212   152 S       0  0.0  0.3   0:00 syslogd
  121 root       0   0   340    0     0 SW      0  0.0  0.0   0:00 klogd
  126 daemon     0   0    80    0     0 SW      0  0.0  0.0   0:00 portmap
  129 root       0   0  1264  768   564 S       0  0.0  1.2   0:00 named
  133 root       0   0    76    0     0 SW      0  0.0  0.0   0:00 rpc.statd
  136 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 lockd
  137 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 rpciod
  144 root       0   0    76    0     0 SW      0  0.0  0.0   0:00 inetd
```

Figure 6: The UNIX top application.

We have seen violent multi-user games evolve into peaceful virtual communities. Production values play a role in the suspension of disbelief essential for this to happen [19], but even text-based MUDs, which originated as hack-and-slash games, have become chatrooms [5]. The key elements seem to be the rich interactions with other players and the persistence of identity over multiple sessions. These seminal experiments in multi-user virtual worlds are showing us how to moderate violence with a sense of community.

RELATED WORK

Visualizing Processes

The standard UNIX application to view processes is top (Figure 6). It is a dynamic version of ps, a command that simply lists the running processes. As implied by its name, top displays the "top" CPU-users, refreshing at regular intervals. Its interface is primitive. The user may not interact directly with the process list – an instruction and process number must be entered, much like on the command line. Another limitation is that the number of processes displayed is limited by the number of lines on the display (typically around twenty to thirty). There is now a plethora of alternatives to top that allow more interaction with the list, such as clicking on the entries. These contribute obvious graphical additions to a process display and add little conceptually.

LavaPS [16] is a more abstract way to view processes. It displays processes as colored blobs in a window, using the size, color, and location of shapes in the window to represent certain metrics of the processes, as shown in Figure 7. More explicit information can be shown on demand. Like

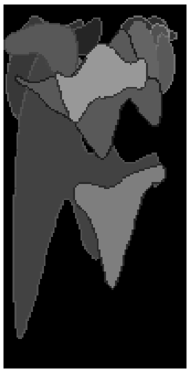

Figure 7: Heidemann's LavaPS process visualization application.

PSDoom, LavaPS represents processes as physical entities with no quantitative information. However, the application is inspired by "calm technology" [31] and is therefore intentionally passive. Its main purpose is to display the state of processes, and manipulating the actual processes can be done through a menu.

Game-like Interfaces

The *CHI 97 Workshop on Game Design and HCI* suggests that computer game design has much to offer to the HCI community, but little has been done so far [7]. A popular area for game-like interfaces is software for children. Such products range from educational games to tools that encourage creativity, such as authoring tools. It is difficult to make direct comparisons between software for children and software for adults because the expectations we have for the intended audiences are so different. Designers readily use entertaining interfaces to keep children engaged [15]. Even in educational software, in which the ultimate goal is to teach the user, the proximate goal is to keep the child engaged, or else the ultimate goal will never be reached. In application software for adults, the burden of staying focused is on the user. The measurement for success is not how enjoyable the software is but rather how effectively it accomplishes a task.

The contrast between software for these two audiences highlights the difference between enjoyment and ergonomics. A piece of children's software must look good and be fun, while software for adults must work. These are not necessarily conflicting goals, as an interface that is difficult to use will also adversely affect the user's enjoyment. However, for "serious" software, concessions to make a program more enjoyable might not be implemented because they can not be justified on ergonomic grounds.

For an application that helps the user monitor the state of a system, like PSDoom, the case for making user interfaces more engaging is compelling. Keeping the user engaged is as important as making the tasks easy to perform. In such cases, the use of randomness in a program, which adversely affects ease-of-use, may be desirable because it can make the task more enjoyable [22].

The use of game-like interfaces need not harm the performance of applications. In fact, it can reduce the learning curve. We would not need to sacrifice power – games have amazingly complex maneuvers that skilled operators can use in addition to their simple repertoires for novices.

FUTURE WORK

It would not be difficult to remedy the major source of instability of PSDoom, the belligerence of the monsters. The monsters currently have no sense of self-preservation. They will continue to attack until they or their targets are eliminated. One could simply make the monsters only attack when provoked and more likely to run away rather than fight to the end.

It would be natural to extend the program to networked environments. In fact, many internet search engines have already classified PSDoom as a network tool in anticipation of this feature. A networked version would allow multiple users to communicate verbally as well as work cooperatively [10]. Turning this single-player game environment into an online world could reduce the level of violence by increasing the sense of community.

By selecting only certain types of processes to appear in the dungeon, one can change the nature of PSDoom. For instance a filter that only allows suspicious or possibly corrupted processes [25] to appear would turn PSDoom into a security program. Alternatively, one can choose to display only large users of memory or bandwidth in order to alleviate pressure on these resources.

CONCLUSION

PSDoom presents a novel approach to process visualization and manipulation that illuminates many aspects of our relationship with computers. The popularity of this early prototype demonstrates that interfaces inspired by video games and science fiction are eagerly awaited by a large audience. Computer games are a rich and often overlooked source of technology and metaphors for user interfaces that will only increase in importance as children, the next generation of users, grow up with computers. Our computer environments are rapidly becoming an anachronism as people who have never handled actual file folders are forced to use their virtual counterparts. The process of changing interface metaphors will expose previously-obscured facets of the underlying applications, such as the genuinely adversarial interactions in our operating systems. Interested programmers can explore and modify the code for PSDoom, which is available at http://www.cs.unm.edu/∼dlchao/flake/doom.

ACKNOWLEDGMENTS

I would like to thank the Adaptive Computation Group at UNM for providing a supportive environment in which one can claim one is doing research while playing Doom for two days. The offhand remarks of Anil Somayaji provided the initial inspiration for the project. David Ackley, Ben Bederson, Patrik D'haeseleer, Stephanie Forrest, and Jason Stewart provided helpful comments that improved this paper. This work was partially supported by the National Science Foundation (grants NSF-9553623, CDA-9503064, and IRI-9711199) and the Office of Naval Research (grant N00014-99-1-0417).

REFERENCES

1. Axelrod, R. *The Evolution of Cooperation*. Basic Books, New York, 1984.

2. Bob. Microsoft Corporation, Seattle WA, 1995.

3. Brand, S. *How Buildings Learn: What Happens After They're Built*. Viking Penguin, New York, 1994.

4. Brockmeier, J. A SysAdmin's Dream. *Linux Magazine 2*, 1 (January 2000), 76.

5. Bruckman, A. MOOSE Crossing: Construction, Community, and Learning in a Networked Virtual World for Kids. MIT Media Lab, Cambridge MA, 1997.

6. Chang, B.W., and Ungar, D. Animation: From Cartoons to the User Interface. *UIST '93 Conference Proceedings* (Atlanta GA, November 1993), ACM Press, 45–55.

7. Cherny, L., Clanton, C., and Ostrom, E. Entertainment is a Human Factor: A CHI 97 Workshop on Game Design and HCI. *SIGCHI Bulletin 29*, 4 (October 1997), 50–54.

8. Doom. Id Software, Mesquite TX, 1993.

9. Erickson, T. Working With Interface Metaphors. *The Art of Human-Computer Interface Design*, edited by Brenda Laurel. Addison-Wesley Publishing Company, Inc., New York, 1990.

10. Evard, R. Collaborative Networked Communication: MUDs as Systems Tools. *Proceedings of the Seventh Systems Administration Conference (LISA VII)* (Monterey CA, November 1993), USENIX, 1–8.

11. The Federal Trade Commission. *Marketing Violent Entertainment To Children: A Review Of Self-Regulation And Industry Practices In The Motion Picture, Music Recording & Electronic Game Industries*. Washington: Government Printing Office, 2000.

12. Gentner, D., and Nielson, J. The Anti-Mac Interface. *Communications of the ACM 39*, 8 (August 1996), 70–82.

13. GNU General Public License. Free Software Foundation, Inc., Boston MA, 1991.

14. Habitat. Lucasfilm Ltd, San Rafael CA, 1986.

15. Hakansson, J. Lessons Learned From Kids: One Developer's Point of View. *The Art of Human-Computer Interface Design*, edited by Brenda Laurel. Addison-Wesley Publishing Company, Inc., New York, 1990.

16. Heidemann, J. LavaPS. 1998. Available at http://www.isi.edu/∼johnh/SOFTWARE/LAVAPS/.

17. Hwang, F. Commando Line Interface. *Wired 8*, 2 (February 2000), 52.

18. Johnson, S. Interface Culture. Harper, San Francisco CA, 1997.

19. Kim, A.J. Ultima Online: An Interactive Virtual World with Multiple Personalities. *SIGGRAPH Bulletin 32*, 2 (May 1998), 15–19.

20. Langford, D., and Jones, C. The Kitchen Interface - A Lateral Approach to GUI. *SIGCHI Bulletin 26*, 2 (April 1994), 41–45.

21. MagicCap. General Magic, Sunnyvale CA, 1994.

22. Malone, T.W. Toward a theory of intrinsically motivating instruction. *Cognitive Science 4*, 333–370.

23. Morningstar, C., and Farmer, F.R. The Lessons of Lucasfilm's Habitat. *Cyberspace: First Steps*, edited by Michael Benedikt. MIT Press, Cambridge MA, 1990, 273-301.

24. Schmidt, J. Systemverwaltung nach Hackerart. *c't: Magazin für Computer Technik*, 23 (1999), 68.

25. Somayaji, A. Automated Response Using System-Call Delays. *9th USENIX Security Symposium* (Denver CO, August 2000), ACM Press, 185-197.

26. Systemadministration mit dem Colt. *IX: Magazin für professionelle Informationstechnik*, 12 (December 1999), 44.

27. UBUBU Universe. UBUBU, Inc., San Francisco CA, 2000. Available at http://www.ububu.com/.

28. Ultima Online. Electronic Arts, Inc., Redwood City CA, 1997.

29. Venturi, R., Scott Brown, D., and Izenour, S. *Learning From Las Vegas*. The MIT Press, Cambridge MA, 1972.

30. Vinge, V. True Names, in *Binary Star #5*, edited by James R. Frenkel. Dell, New York, 1981.

31. Weiser, M., and Brown, J.S. Designing Calm Technology. *PowerGrid Journal 1.01* (July 1996).