Managing Montezuma

Handling All the Usual Challenges of Software Development, and Making It Fun: An Interview with Ed Beach

Forrest Shull

**Many (Maybe All!)** of our readers are working today on the challenge of creating quality software. Many of us are familiar with the need to make decisions about how to incorporate the right set of features while balancing other quality characteristics like reliability, maintainability, robustness, and performance.

As I reflected on our “engineering fun” issue, I thought about the challenges developers in the computer gaming industry must face, creating products that have a need for all those usual qualities plus having to be fun as well. Software engineering research and practice have been studying the usual “-ilities” for years now, but I was a bit mystified as to how a team would measure and achieve “fun” during development.

**Civilization**

To help address those questions, I turned to one of the world leaders in computer gaming, Firaxis Games. Firaxis’ flagship strategy game series, Civilization (affectionately known as Civ), is regarded by many to be the computer strategy game. There have been five games in the series so far, all helmed by Firaxis’ Director of Creative Development, Sid Meier. Having sold over 9 million units to date ([http://firaxis.com/games/game_detail.php?gameid=41](http://firaxis.com/games/game_detail.php?gameid=41)), the franchise has received an array of professional awards and fan acclaim, and possibly resulted in more person-hours lost to addictive gameplay than any other game series. If anybody knows how to engineer fun gameplay, it must be these folks.

The level of complexity in a Civilization game can sometimes make it sound more like a job than a hobby: your task as a Civ player is to manage a growing and increasingly complicated nation, starting with a single band of warriors in prehistory and progressing through a world-girdling society with spaceflight, civil unrest, and other complexities of modern life. Players need to develop a productive population to create knowledge and commerce for their nation, create defensible borders to protect their growing populations, and secure the resources needed to underpin increasingly sophisticated technologies. And of course, all this is done in competition with other nations who are all trying simultaneously to do the same, each run by artificial intelligence (AI) in the persona of a famous world leader. Much of the fun of the game comes from dealing with competitive AI leaders (including Montezuma, George Washington, Augustus Caesar, and others; see Figure 1), who can collaborate with or be in conflict with each other at different points in the game, depending on their own (simulated) interests and strategies. (The human plays one of the world leaders, and the AI runs the rest.) And since there are multiple paths to victory based on
excelling either militarily, diplomatically, culturally, or technologically, the human player and the AI leaders have a number of different strategies that need to be monitored simultaneously.

Ed Beach led the AI programming for the latest Civ game, and hence was responsible in large part for managing these complicated interactions and giving the human players a fun challenge. I was able to get some time with him to explore how the engineering under the hood was managed to result in a sophisticated, successful product in this domain.

**Interview**

**Forrest:** Ed, could you talk about your experiences on Civ V as a way to help our readers understand how you tackle all the usual challenges of software development and incorporate fun gameplay to boot?

**Ed:** I’d be happy to. Let me start by noting that I spent the first 15 years of my career writing engineering software with pretty severe reliability and precision requirements (for NASA and the wireless industry). So I’m quite familiar with those challenges of creating quality software that you mention.

In fact, that’s a great place to start this discussion. Most of the engineering principles that help you develop quality software in a more traditional setting still apply here in the game industry. For instance, it was absolutely critical that we think about issues such as modularity, encapsulating behaviors that were going to change, and extensibility when developing the Civ V AI leaders. We wanted all four programmers in our gameplay/AI development team to be able to work in parallel on different AI subsystems. If we hadn’t set these subsystems up to be properly isolated, our work would have ground to a halt. In fact, I’d say that software development for the game industry is fundamentally the same as a traditional project—with the major caveat that your requirements are significantly more fluid than normal.

Why are the requirements so fluid? Well, that’s how the “fun gameplay” gets introduced. We stand up an early prototype of the game and quickly get people playing it. With a prototype in place, we can take an evolutionary approach to development. We find out what’s fun and build around those elements. Those parts of the game that users find tedious are quickly refactored or eliminated. It’s a proven method for developing great games pioneered by Firaxis’ creative director, Sid Meier. But it means that you have to design the software to accommodate nearly constant change.

**Forrest:** While you’re working on the subsystems for the AI players, how much do you interact with other development teams working on different parts of the game? Would you ever have the case that the rules of the game change while you’re working on the AI subsystems that must reason about those rules? How do all these aspects get put together to see if they’re adding up to a good game or not?

**Ed:** The development of the AI subsystems is most closely linked to the current game rules. When a game designer adjusts a rule, there are usually direct ramifications within the AI logic. That’s one reason why the model at Firaxis is for the game design effort to be led by designer/programmers who have skills in both of those disciplines. In many cases, the designer changing the rule makes the AI adjustments himself. However, there are certainly ties to other members of the development team as well. For instance, we needed to carefully plan out the possible diplomatic responses of AI leaders so the correct set of animations and voiceovers could be created by our art and sound teams. Those are expensive

![FIGURE 1. Montezuma of the Aztecs, a proud and awesome leader, who can be represented in-game by the human or by the game’s AI logic.](image)
assets to create (especially since we had voiceovers from 18 different leaders all in the proper language ... including several ancient ones!), so any miscommunications here are costly.

You also need to listen to the feedback coming in from your testers. For instance, we found that early in many games, the first dramatic moment that elicited an emotional response from our players was when the barbarians captured one of their workers or settlers. Originally that unit was just instantly killed by the barbarians. But from our testers’ passionate reaction, we knew we’d create something even more engaging if the player could chase down the barbarians and recapture their lost unit. So we put aside other AI work to quickly teach the barbarians to escort these prisoners back to their camps. The hunt to recapture lost civilians and make those wicked barbarians pay is now a tried-and-true part of Civ V gameplay.

**Forrest: What’s a typical day look like when you’re doing this kind of work?**

**Ed:** Much of the day is spent like any of the other programmers: writing up designs, coding, testing various paths through the code. However, besides just verifying that the AI responds to individual decision points properly, we also needed a way to monitor its progress over the course of a full game of Civ. That’s 400 or more turns: something that would take six to eight hours to play through. Rather than force someone to play that long, we developed the ability to have the AI leaders play against themselves. These “autoplay” runs can be completed in under an hour since we turn off all graphic updates. We then added the ability for each AI subsystem to log all the moves it made (and often all the alternatives it discarded). By the end of a game, we typically accumulate hundreds of megabytes of log file data, most of it in comma-separated files (that can be easily imported into tools such as Excel). Through the course of the project, I spent a lot of time using spreadsheets to sort and search through these logs to make sure that our AI-controlled civilizations were advancing at the expected rate.

**Forrest:** How do you know when you’ve “got it”—when the AI leaders are doing what you wanted them to? Do you have a good sense from the beginning of the types of behavior you want to see in the AI leaders? How do you know how good the AI is in any given version of the game?

**Ed:** Once again, this comes from talking to other members of the team, testers especially. Everyone knew who was working on the AI logic: when it performed well (such as taking out a player’s city with a surprise attack), I heard about it quickly. But it was equally important to get feedback from the testers when it wasn’t working as effectively as it should. With each version of the game, I asked our external testers to collect a series of AI blunders: moments when an AI leader had an obvious good move and failed to make it. By examining and correcting each of these cases, we were able to improve the AI performance dramatically.

**Forrest:** Are smarter AI players always better, or do you need to keep a balance with other game characteristics?

**Ed:** Civilization has a community of very hard-core players. For this audience, a smarter AI opponent is always better. However, we made a real attempt with Civ V to reach out to a new group of strategy gamers and make this version the most accessible Civ ever. For this audience, a fearsome AI leader is not what you want. However, we were able to take the AI code we had in place and turn things around to help these newcomers. How? Well, we had...
an extra copy of all the AI subsystems run as a “shadow” AI player that continually monitored the human player’s own position but didn’t actually trigger any moves. We would consult that shadow AI player and present information to the user as advisor recommendations. So, in this case, making the AI smarter was still beneficial because we could turn that into quality advice for a new player.

Forrest: Once you know what you want to see in the behavior of the AI players, is it fairly straightforward to do the programming? Or would you ever have to go back and modulate your expectations for the players based on what seemed feasible to actually program?

Ed: Some of the programming is quite complex, especially when you need to plan out an optimal set of tactical combat moves for dozens of pieces on a board with thousands of spaces. Programming such an AI player to compute optimal tactical moves is several orders of magnitude more difficult than what chess programs currently have to compute. We also had to always be mindful of how much processing time our AI algorithms would need. An ideal user response time between turns in a Civ game is probably 15 seconds or less. In that brief time, we need to make all the moves for all the opposing civilizations (up to 11 opponents, not including the additional minor powers we call city states). So we do have places in the Civ V tactical AI code where we exhaustively look through a game tree and score moves to find the optimal one. However, we have to be careful to limit our use of even these relatively basic AI algorithms to just the cases where we can gain the maximum benefit for the increasing processing time.

Forrest: Is there something you wish more people knew about software development in the field of computer gaming?

Ed: Well, how about I share what I wish I knew three years ago when I started working on the Civ V AI? At that time, the one thing I underestimated was how difficult it is to validate that the AI code is all working properly. Unless specially trained in the planned capabilities of the AI players, a tester often can’t tell the difference between the AI players “not being smart enough” and one of our AI routines (that should be able to make the correct decision) failing. Often these problems would show up in our log files, but identifying errors in that fashion could be time-consuming. It’s a situation where we need a system to formally validate the AI behavior within a series of automated tests. That’s something we had to put in place for Civ V, but it’s definitely an area we’ll want to explore in the future.

Forrest: Thanks so much, Ed, for spending time with us.

S o, I hope you enjoy this issue as much as I have. The guest editors have done an impressive job of compiling articles that cover a gamut of issues related to the engineering underpinnings of an incredibly vibrant and economically successful domain. If you’re interested in learning more about the computing gaming profession, you will also find useful the webpages of career advice that Firaxis created at www.firaxis.com/jobs/career.php. 

STAFF

Lead Editor
Dale C. Strok
 dstrok@computer.org

Content Editor
Brian Bramon

Manager, Editorial Services
Jenny Stout

Staff Editors
Camber Agelius, Dennis Taylor, and Linda World

Publications Coordinator
software@computer.org

Production Editor/Webmaster
Jennie Zhu

Contributors
Thomas Centrella, Molly Gamborg, Shani Murray, Joan Taylor, and Alex Torres

Cover Artist
Fred Chuang

Director, Products & Services
Evan Butterfield

Senior Manager, Editorial Services
Lars Jentsch

Senior Business Development Manager
Sandra Brown

Membership Development Manager
Cecelia Huffman

Senior Advertising Coordinator
Marian Anderson

manderson@computer.org

CS PUBLICATIONS BOARD

David A. Grier (chair), Alain April, David Bader, Angela R. Burgess, Jim Cortada, Hakam Ergodmus, Frank E. Ferrante, Jean-Luc Gaudiot, Paolo Montuschi, Dorée Duncan Seligmann, Linda I. Shafer, Steve Tanimoto, George Thrivathukkal

MAGAZINE OPERATIONS COMMITTEE

Dorée Duncan Seligmann (chair), Erik Altman, Isabel Beichl, Krishnendu Chakrabarty, Nigel Davies, Simon Liu, Dejan Milojčić, Michael Rabinovich, Forrest Shull, John R. Smith, Gabriel Taubin, Ron Vetter, John Viega, Fei-Yue Wang, Jeffrey R. Yost


Editorial: All submissions are subject to editing for clarity, style, and space. Unless otherwise stated, bylined articles and departments, as well as product and service descriptions, reflect the author’s or firm’s opinion. Inclusion in IEEE Software does not necessarily constitute endorsement by IEEE or the IEEE Computer Society.

To Submit: Access the IEEE Computer Society’s web-based system, ScholarOne, at http://mc.manuscriptcentral.com/sw-cs. Be sure to select the right manuscript type when submitting. Articles must be original and not exceed 4,700 words including figures and tables, which count for 200 words each.


Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.