



GAME DEVELOPER MAGAZINE

DECEMBER 2003





GAME PLAN

LETTER FROM THE EDITOR

Celebrating 100

If I had a penny for every issue of *Game Developer* that's hit the streets, this month I'd finally have a dollar. (And if I had a dollar for each clump of hair I've torn out writing this column every month, I might have enough to buy a cheap wig.)

One hundred issues is a big milestone in the magazine business. Even though we serve an industry that might take an average of 18 to 24 months to put out a product, we have to put out a whole new product every single month. Deadlines are unforgiving. Slippage is not an option. Sometimes bugs slip through. But we, like you, persist in the face of these challenges, partly out of a belief that somehow our output makes the world just a tiny bit better, and perhaps partly out of some foolish optimism that things will get easier for us someday. Sound familiar?

While preparing "*Game Developer's* 100th Issue Retrospectacular" (page 28), I had a blast reminiscing with many contributors, correspondents, readers, and colleagues from yesteryear and today, many of whom I now consider friends. On several occasions I was genuinely touched by some of the *Game Developer* memories raised by taking people back five or 10 years to seminal points in their careers, and I was disappointed I couldn't fit more of those memories into the article. Many found it sobering to think quantifiably about how much the industry has matured in the past 10 years, and how yesterday's bedroom hackers are today's industry elite.

On a personal level, it's been humbling to look back at my own time with the magazine: humbling not only because it gave me pause to consider how much knowledge I have gained from the hundreds of extraordinarily talented and intellectually generous people *Game Developer* has connected me with over the years, but also because I'm 60 issues older than the naif who washed up on *Game Developer's* shores five years ago. To what degree I am wiser I can only credit to the many

readers who have offered their thoughtful, honest, and unselfish feedback over the years. My year has certainly been smoothed by the many people who are genuinely dedicated to improving games and game development by the free exchange of information and ideas.

For *Game Developer*, being an aging figure in a young industry necessitates a certain reckoning. As editor, I'm determined not to let the magazine's middle-age slow it down but rather let its experience buoy it through the development challenges yet to be faced as the sometimes inspiring, sometimes terrifying pace of game technology plods on. We're cooking up some exciting new things for you to see and read in 2004, in an effort to help you better understand the bigger picture of the ever-changing game development industry. Our goal is to continue to help game developers understand and define their craft, their business, and their creative will, lest others seize the opportunity to impose their definitions upon you. We look forward to serving you better than ever.

Game Developer's next big milestone is right around the corner; our 10th anniversary arrives next spring. I had so much fun talking with readers for our 100th issue, I'd love to hear from more of you to help us look back at our first 10 years. E-mail your *Game Developer* thoughts to editors@gdmag.com.

From the cockpit. This month we bid farewell to our managing editor of two years, Everard Strong, as trusty a first officer as any captain ever had. Everard's off to start up his own independent publication, and we wish him great luck in his new endeavor. Departments editor Jamil Moledina will step up as managing editor next month, and Kenneth Wong will join as departments editor from our sister publication *CADENCE*.

Jennifer Olsen
Editor-In-Chief

Game Developer

www.gdmag.com
600 Harrison Street, San Francisco, CA 94107 t: 415.947.6000 f: 415.947.6090

EDITORIAL

Editor-In-Chief
Jennifer Olsen jolsen@cmp.com

Managing Editor
Everard Strong estrong@cmp.com

Departments Editor
Jamil Moledina jmoledina@cmp.com

Editorial Assistance
Kenneth Wong kwong@cmp.com

Product Review Editor
Peter Sheerin psheerin@cmp.com

Art Director
Audrey Welch awelch@cmp.com

Editor-At-Large
Chris Hecker checker@d6.com

Contributing Editors
Jonathan Blow jon@number-none.com
Noah Falstein noah@theinspiracy.com
Steve Theodore steve@theodox.com

Advisory Board
Hal Barwood Independent
Ellen Guon Beeman Monolith
Andy Gavin Naughty Dog
Joby Otero Luxoflux
Dave Pottinger Ensemble Studios
George Sanger Big Fat Inc.
Harvey Smith Ion Storm
Paul Steed Microsoft

ADVERTISING SALES

Director of Sales/Associate Publisher
Michele Sweeney e: msweeney@cmp.com t: 415.947.6217

Senior Account Manager, Eastern Region & Europe
Afton Thatcher e: atthatcher@cmp.com t: 828.350.9392

Account Manager, Northern California & Midwest
Susan Kirby e: skirby@cmp.com t: 415.947.6226

Account Manager, Western Region & Asia
Craig Perreault e: cperreault@cmp.com t: 415.947.6223

Account Manager, Target Pavilion, Education, & Recruitment
Aaron Murawski e: amurawski@cmp.com t: 415.947.6227

ADVERTISING PRODUCTION

Advertising Production Coordinator Kevin Chancel
Reprints Terry Wilmot t: 516.562.7081

GAMA NETWORK MARKETING

Director of Marketing Michele Maguire
Senior Marcom Manager Jennifer McLean
Marketing Coordinator Scott Lyon

CIRCULATION



Game Developer is BPA approved

Circulation Director Kevin Regan
Circulation Manager Peter Birmingham
Asst. Circulation Manager Lisa Oddo
Circulation Coordinator Jessica Ward

SUBSCRIPTION SERVICES

For information, order questions, and address changes
t: 800.250.2429 or 847.763.5958 f: 847.763.9606
e: gamedeveloper@halldata.com

INTERNATIONAL LICENSING INFORMATION

Mario Salinas
e: msalinas@cmp.com t: 650.513.4234 f: 650.513.4482

EDITORIAL FEEDBACK

editors@gdmag.com

CMP MEDIA MANAGEMENT

President & CEO Gary Marshall
Executive Vice President & CFO John Day
Executive Vice President & COO Steve Weitzner
Executive Vice President, Corporate Sales & Marketing Jeff Patterson
Chief Information Officer Mike Mikos
President, Technology Solutions Robert Faletta
President, Healthcare Media Vicki Masseria
Senior Vice President, Operations Bill Amstutz
Senior Vice President, Human Resources Leah Landro
Vice President & General Counsel Sandra Grayson
Vice President, Group Publisher Applied Technologies Philip Chapnick
Vice President, InformationWeek Media Network Michael Friedenberg
Vice President, Group Publisher Electronics Paul Miller
Vice President, Group Publisher Software Development Peter Westerman
Corporate Director, Audience Development Shannon Aronson
Corporate Director, Audience Development Michael Zane
Corporate Director, Publishing Services Marie Myers



GamaNetwork

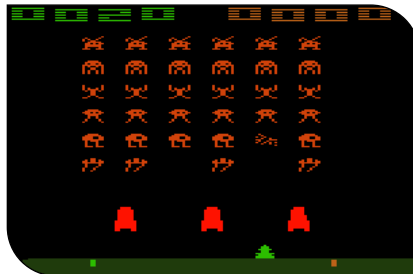


INDUSTRY WATCH

KEEPING AN EYE ON THE GAME BIZ | *jamil moledina*

Courts of L.A. Activision's TRUE CRIME: STREETS OF L.A. shipped worldwide in November as originally planned, despite a court order filed in U.S. District Court in Los Angeles by Robert Crais, the author of *L.A. Requiem* (Ballantine Books, February 2000). Crais alleged that the game's Nick Kang character was based on his hero Elvis Cole, and he sought an undisclosed sum and the destruction of all infringing works.

Former Blizzard executives spawn new studio. David Brevik, Max Schaefer, Erich Schaefer, Bill Roper, and Kenneth Williams — all former executives at Blizzard North credited with games such as DIABLO, STARCRAFT, and WARCRAFT — have come together to form a new game company called Flagship Studios. Joining them as additional cofounders are David Glenn, Peter Hu, Philip Shenk, and Tyler Thompson, the artists and programmers behind DIABLO and DIABLO II. The new company has yet to announce any games.




The Atari 2600's port of Taito's SPACE INVADERS can now safely land at your local archive.

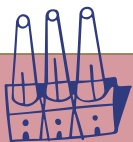
Preserving the real Atari. In response to a filing by Brewster Kahle of The Internet Archive, Lawrence Lessig of Creative Commons, and others, the Librarian of Congress granted access exemptions from copyright protection measures in the Digital Millennium Copyright Act to obsolete videogames. The exemption applies to games that require the original media or hardware as a condition of access, and it determines a format obsolete "if the machine or system necessary to render perceptible a work stored in

that format is no longer manufactured or is no longer reasonably available in the commercial marketplace." According to the original filing, the exemption was proposed in order to migrate degraded and obsolete works to modern storage systems and enable "archiving, future scholarship, and commentary."

Convergence, once more with feeling. Sony demonstrated its new PSX multifunction device at the fall CEATEC show in Tokyo. The PSX includes a PS2, a hard drive, a DVD burner, and digital television recording functionality. Sony plans to sell it in Japan by the end of the year in two capacities, a 160GB version for approximately \$720 and a 250GB version for approximately \$900. Sony expects versions for the U.S. and Europe to follow in 2004.

SYPHON FILTER(-ed). Facing mounting pressure from The Toronto Transit Authority and Quebecois politicians, Sony agreed to remove all references to levels in a Toronto subway and terrorists named the "Quebec Liberation Front" in its upcoming title SYPHON FILTER: THE OMEGA STRAIN, in all territories. Canadian newspapers drew references to the 1970 killing of government official Pierre Laporte by the Front de Liberation du Quebec, prompting Sony Canada's John Challinor to say that they "deeply regret any misunderstanding this may have caused." 

Send all industry and product release news to news@gdmag.com.



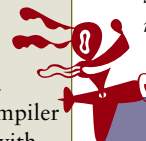
THE TOOLBOX DEVELOPMENT SOFTWARE, HARDWARE, AND OTHER STUFF

Ghost 3D unveils 3DS Max plug-in. Ghost 3D announced that its 3D modeling, conversion, digitizing, and reverse-engineering products will now be available as plug-ins for Discreet's recently released 3DS Max 6. The newest versions of Ghost products include Power Modeler Pack, Power Digitizer Pack, Resurrect, Scribe-it, and Surf-it. They offer tools for spline surface modeling, spline rebuilding, parametric object creation, surfacing, digitizing, and reverse engineering. www.ghost3d.com

Binary releases new game engine. Binary Worlds recently released a demo version of Descensor, a new game engine for producing large 3D worlds in real time. Descensor automates the process of computing 3D objects during game-

play. The landscapes can be as large as needed, since only the visible parts are computed. According to Binary, the engine is well-suited for online games, because it can re-create the same world on each player's machine with minimal bandwidth usage. www.binaryworlds.com

OpenGL compiler front-end for Linux. 3Dlabs recently announced a compiler front-end for Linux, integrated with the previously announced OpenGL Shading Language compiler for Windows. It provides developers with a single front-end for consistent cross-platform portability of the OpenGL Shading Language standard. The compiler is a free download. www.3dlabs.com/opengl2



UPCOMING EVENTS CALENDAR

DV EXPO WEST

LOS ANGELES CONVENTION CENTER
Los Angeles, Calif.
December 9-12, 2003
Cost: \$75-\$599
www.dvexpo.com/west



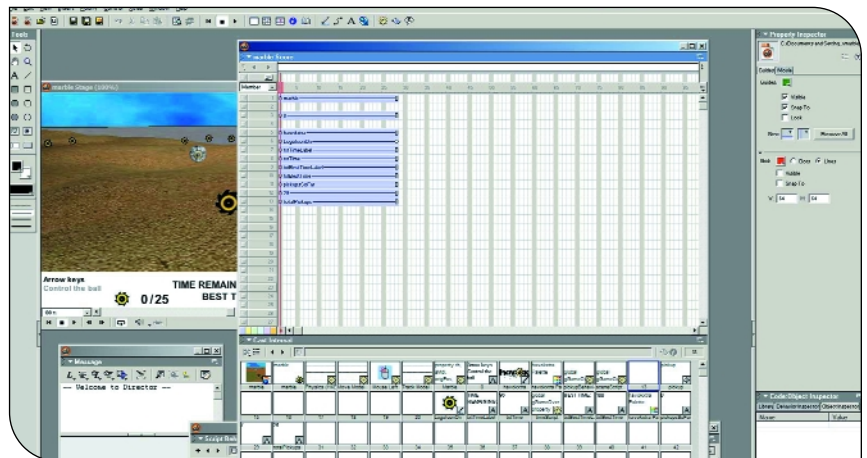
Macromedia Director MX

by Justin Lloyd

Macromedia Director MX is aimed at developing rich Internet content and web or stand-alone games, similar to the casual games found on RealArcade, Yahoo!, and MSN's sites. It's also used by some companies as a prototyping tool to quickly test out ideas and gameplay mechanics.

Director does a good job delivering on some of its promises, but unfortunately it fails on too many other ones. Director MX is the upgrade to Director 8.5 and is already beginning to show its age, both in its feature set and in its user interface paradigm.

Working with Director MX means becoming very familiar with Lingo, Director's built-in scripting language. Lingo supports a primitive object-oriented model that has no obvious way to protect members from being called by other scripts or child classes. The language is extensive, with literally thousands of commands, objects, and properties that can be manipulated, and you have to learn them all. The program attempts to sell the user on ease of use and drag-and-drop functionality, but these attempts actually get in the way when creating a large project. Director MX is nowhere near as "click here, do this, click there, do that" as it's touted to be, and due to the poor documentation and the small number of examples included, I found learning Lingo a lot harder than memorizing the DirectX SDK or the Havok Physics SDK.



Director MX's Cast and Score interfaces are useful in creating quick game prototypes.

The examples that are included are simple enough, showing two or three buttons that navigate a simple user interface. However, try to create a game interface with 20 or 30 buttons, and the resulting spaghetti code arrives in one-gallon catering packs. The interface requires constant shuffling back and forth with the slider, attempting to find the current edit point. This is in direct contrast to VisualBasic's ease in letting you navigate code.

Unlike a lot of scripting systems, Director has the ability to single-step through scripts with the integrated debugger, setting breakpoints and inspecting variables and properties with the Object Inspector panel.

Director MX adds 3D capability that automatically utilizes OpenGL, DirectX 5.2 or 7.0, or its software rendering

engine if hardware is not available. DirectX 7 appeals to the broad, lowest common denominator market but leaves out niceties such as pixel and vertex shaders. 3D content isn't really an integral part of the package; usually it is rendered in a separate window, preventing compositing with 2D elements. And 2D elements can't interact with 3D objects, confusingly also known as Sprites.

Xtra power. The real power of Director comes in the shape of Xtras (third-party plug-ins), some of which are free, many of which you have to buy (with some costing more than Director itself). Havok ships its Havok Physics Xtra for free, and it includes many great examples. I was excited to get it and play around with what Havok touts as a drag-and-drop physics solution. However, I was disappointed to find it wasn't at all intuitive, requiring a lot of scripting and several nights' work to add interactivity to the basic physics operations.

JUSTIN LLOYD | Justin has over 18 years of commercial game programming experience on almost every released platform.



Director MX's final output is a Projector file, played back with Macromedia's Shockwave player, or bundled into a stand-alone executable. Doing the latter instantly creates a 2.5MB Windows executable file. Authoring a stand-alone executable on more than one platform requires the purchase of a license for each platform.

Director MX, as its name suggests, uses a Hollywood movie metaphor to refer to many of the actions, processes, objects, and interface elements within the package. The metaphor itself isn't flawed, just the application of it at times. Director MX project files are referred to as

Movies: the Stage is the playfield where game objects (cast members) are placed, which then follow scripts. The metaphor begins to break down when placing cast members on the Stage, which are then known as Sprites. To confuse matters, everything on the Stage is a Sprite — QuickTime movies, vector shapes, and buttons all become Sprites.

The cast panel lists scripts and cast members, which can include audio clips, movies, buttons, bitmaps, scripts, and so on. For a large project, the cast panel quickly becomes cluttered; external libraries of cast members help, but there's no way to place cast members into logical groups.

The Score provides a non-linear edit suite interface to control in what Frame (an instant in time) Sprites appear and how they move between frames. Think of creating a movie in Adobe Premiere: A Sprite is placed into the next available of 1,000 possible edit channels in the Score, reusing channels for other cast members as the movie progresses.

With so many hard-coded constants floating around, there should be a way to define global and local symbolic constants, but the capability to do this in Director MX simply does not exist. After nine versions of Director, Macromedia still doesn't provide even the basics of a real programming language.

With a package of this maturity and complexity, I was expecting some refined documentation, examples, and tutorials. But alas, I was disappointed. The documentation briefly covers each aspect of the package but then leaves the rest of the explaining to third-party books and web sites, including the extensive Director Users Group and Director Mailing List.

Wrap up. A rapid prototyping tool is expected to shoulder the burden of handling mundane housekeeping tasks along with the necessary video and audio requirements — sprites, 3D, streaming movies, and physics, for example. Director MX does this in spades, but where it falls down is the archaic scripting system and non-intuitive interface. Improvements in interaction between 3D and 2D, more and varied documented examples showing off

the power of the Xtras, and a more proficient drag-and-drop interface (with different interfaces for different types of content) would go a long way toward making Director MX an ideal RAD tool.

Kaidan's 360 One VR

by sean wagstaff

Kaidan's 360 One VR is one of the best solutions I've seen for grabbing full 360-degree panoramas from the world around us. In a single shot, it captures a panoramic image that can be mapped to cycloramas for in-engine background scenery, or used as an accurate reference for painting your own background environment art.

The 360 One VR attaches to a wide range of digital cameras, either via an adapter ring or with the help of an adjustable, calibrated mounting bracket. I used the lens with a Nikon D100 SLR and a Nikkor 60mm Macro lens, which requires the optional machined-aluminum SLR Bracket (\$299) that separates the camera from the lens by a measured distance. Since you're focusing on the mirrored surface of the VR lens rather than on your actual subject, your camera needs to be able to focus at very close distances while maintaining a decent depth of field — so a macro lens or macro-focusing capability is mandatory.

Once you've mounted the lens and adjusted the focus properly, you set your camera's exposure normally (considering the lighting from every direction) and shoot away. Composing the shot is another matter. The view through the lens is extremely distorted, since it captures a donut-like image covering 50 degrees above and below the horizon. This makes composing a picture awkward at best, since it's often difficult to tell exactly what you're looking at.

You also have to get yourself out of the shot; you can choose to cower under the camera's tripod, or you can fire the camera with the self-timer (giving you enough time to run and hide behind a large object), or you can trip the shutter with a remote control.

DIRECTOR MX ★★

STATS

Macromedia
San Francisco, Calif.
415.252.2000
www.macromedia.com

PRICE

\$1,199 MSRP

SYSTEM REQUIREMENTS

For Windows: Intel Pentium II Processor or higher, Windows 98 SE/2000/XP, 128MB RAM, 1024 × 768, 16-bit color display or better, Microsoft DirectX 5.2 or OpenGL, 3D accelerator, 100MB disk space, CD-ROM drive.
For Mac: G3 Processor, Mac OS X 10.1.2 and up (10.2 recommended), 128MB RAM, 1024 × 768, 16-bit color display or better, OpenGL 1.1.2, 3D accelerator, 100MB disk space, CD-ROM drive.

PROS

1. Good, high-quality, third-party plug-ins.
2. Very low system requirements for playback.
3. Large community support.

CONS

1. Exceptionally steep learning curve.
2. Expensive for multi-platform stand-alone delivery.
3. A lot of information needs to be "hard-coded."

The PhotoWarp software (Mac and Windows) that ships with the 360 One VR instantly unwraps the image with a single click into a very normal-looking cylindrical panorama. New in version 2.0 is the capability to unwarp batches of images, which is particularly useful when you bracket exposures to make sure you've got one that works, or when you're ready to convert a day's shooting back at the studio.

The software can also automatically convert the resulting rectangular images into a QuickTime VR panorama or one of several other panoramic formats, but for game artists the process will usually end at saving the rectangular file as an image that can be texture-mapped onto a cylinder as a background. You can also combine mul-



Kaidan's 360 One VR with optional SLK mounting brace.

tiple exposures to create a high-dynamic-range image, for use as radiance maps and environmental reflection maps, but Kaidan's software doesn't offer any help in this regard — and the circular holes in the sky and ground mean the images really aren't well-suited to tasks that call for a spherical map. (Kaidan's panoramic tripod heads for shooting with fisheye lenses have more promise for creating HDR images, but this is still an awkward process awaiting an elegant solution.)

The 360 One VR does a great job of quickly and painlessly capturing panoramic scenes, and the included software effortlessly handles the conversion into usable images. However, at \$750, it is somewhat expensive for game artists. A rotating tripod head and capable stitch-

ing software can give equivalent results if you're willing to spend a little more time on the image. But if you're grabbing lots of urban or natural panoramas for use as background scenery or textures, then this strange-looking gadget is definitely worth considering. 📸

★★★ | 360 One VR
Kaidan
www.kaidan.com

Sean Wagstaff is a freelance 3D artist. You can reach him at www.wagstaffs.org.

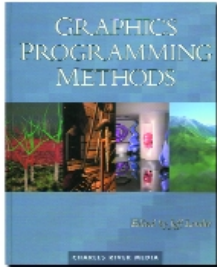
Graphics Programming Methods
Edited by Jeff Lander

reviewed by daniel sanchez-crespo

Graphics Programming Methods is a collection of papers about graphics

programming edited by Jeff Lander. The 400-page book contains over 35 papers, arranged under three areas: animation, geometry, and rendering. The authors are a mix of academic and industry people, with the former outweighing the other in terms of topics covered.

Topics covered in the different papers, though diverse, are all geared toward real-time applications; expect little content on offline rendering techniques. Though this focus makes the book a good companion for game technology programmers, this is a book about graphics programming, not game programming specifically. On the other hand, this book is not another iteration of the *Graphics Gems* or *Game Programming Gems* series: articles here are longer and more involved, and readers



are treated to more background and mathematical information. Given the professional tone of the book, this approach is welcome.

The first section of the book covers subjects such as tree animation, new methods for IK, facial animation, and soft-body animation. Though this is the shortest section, the selected papers are top-quality. John Van den Burg's paper on facial animation and Jason and Andrew Weber's text on multi-resolution dynamics for deeply hierarchical bodies are great reads.

The second section covers geometry, the most heterogeneous of them all. The papers (11 total) cover subjects as diverse as portal rendering, landscape engines, surface smoothing, and texture-mapping techniques. Terrain rendering is one of the subjects more widely covered by this book, as there are three papers in this section, along

with four in the rendering section, that deal with this subject in detail.

The book's section on rendering is its strongest section, with lots of useful material on subjects including Metropolis sampling, outdoor lighting, and volume shadows. Techniques described here are on the cutting edge, so the section is really a joy to read.

Overall, *Game Programming Methods* is a great addition to any 3D programmer's battle chest. It provides many cutting-edge algorithms along with a companion CD-ROM with ready-to-use implementations.

★★★★ | Game Programming
Methods
www.charlesriver.com

Daniel is a programmer at Novarama, a Barcelona-based development studio. He also teaches at Universitat Pompeu Fabra.

Keeping the Faith

Obsidian's Feargus Urquhart on RPGs and PCs

Feargus Urquhart likes RPG titles a lot, as he should, given his past involvement in their creation. For the last six years Feargus has been involved in Black Isle Studios' most successful titles, including *BALDUR'S GATE*, *PLANESCAPE: TORMENT*, *ICEWIND DALE*, and *FALLOUT*. Before that he was involved with other RPG titles at Interplay, Black Isle Studios' parent company.

However, he and Interplay were looking toward different futures. "Interplay changed directions a number of years ago and put most of their efforts behind next-gen consoles," Feargus told us. "However I still think PCs have a place as a gaming platform."

Following his instincts, he looked into developing his own company, and with steering help from BioWare, launched Obsidian Entertainment earlier this year. We took some time to talk to Feargus about the challenges in starting up Obsidian, why the RPG format works for him, and the future of the PC as a gaming platform.

Game Developer: How does Obsidian treat the issue surrounding non-compete agreements with its employees?

Feargus Urquhart: We don't have non-compete agreements with employees other than they can't compete with us while they are working at Obsidian. We chose not to have employees sign one because they are questionably enforceable; if someone wants to leave then we either need to let them go or do something about it. Having someone feel trapped with a company because of a non-compete doesn't help their morale on a day-to-day basis, which doesn't really help the company.

GD: What about Obsidian sets it apart from other studios?

FU: Obsidian's strengths are a wealth of experience in the business and process of making games and the ability to make those two diverging aspects work together. When we approach a new project we are able to tell a publisher what it will take to make the game, how long, and that we will budget for it effectively. Coupled with that, we have a very strong group of guys making games, with the core group having over 50 years of combined industry experience.

GD: Has your life become more chaotic or less since launching Obsidian?

FU: Weirdly enough it's become less chaotic. With Obsidian there seems to be both fewer and more working parts at the same time. If we need a new computer, we just go online and



Behind Obsidian's stone facade, Feargus is all smiles.

order it, while at Interplay it might have taken three months, six e-mails, and three meetings. That's not to say it will stay that easy. If we end up going to more than two or three teams in the future, we will have to get more structured, which will mean that we will need to have more approval processes. However, things can get a little chaotic when it takes six weeks to get medical insurance going for the company because of all these hoops to jump through and forms and more forms and more forms and more forms to fill out. Did I mention the forms? All in all, though, it's been a great experience.

GD: What kind of structures will you put in place to prevent decisions from being mired down in committees and meetings?

FU: It just has to do with focus. My opinion is that Interplay lost this focus as it grew and became a publisher. For us to stay on track, we are going to make sure that the company's largest priorities always relate to the development of our products.

GD: Why is it that so many developers eschew PCs in favor of console development?

FU: When talking to publishers in the industry, most are much more interested in hearing about ideas for your \$4 million console game rather than your great \$2 million PC game. I don't mean to fault publishers by saying this, because in many ways sales data supports why this is the case. However, there are still a bunch of games that sell very well on the PC, and they don't require console licensing fees.

GD: What surprising turns have you seen the game development industry take?

FU: I am somewhat surprised at the lack of support for the PC as a gaming platform right now. It's not that I just want to make PC games instead of console games, but there are certain kinds of games that I would like to make that just work better on the PC. I would hope those ideas would actually be for games that people would buy, but it's hard to get other people in the industry excited about a game if it is PC only. There are companies that are still very successful on the PC, like Blizzard, but many see that success as an anomaly for some reason.

GD: What five key components make up a successful RPG game?

FU: In no particular order, they are a player-driven story, a robust character development system, believable non-player characters, heroic quests, and a living world. 🎮

Predicate Logic

We are in the midst of a software engineering crisis. While games are getting larger and more complicated, our programming tools — languages, compiler systems, debuggers — are remaining stagnant. We've been saved, so far, by the fact that computers get faster every year, so our code gets worse, and we still get by. But big, sprawling, complex code will eat up as many person-hours as you can dish out, and this year more than ever we've been seeing the results.

Delays and feature cuts are nothing new in our industry, but now it feels as though we are hitting a wall: there seems to be no cutting-edge game that isn't horribly delayed, its features slashed, released as a mere shadow of its creators' intentions. Consequently, each year's cutting-edge is landing ever closer to the previous year's cutting-edge. We are traveling along a convergent series, and we will soon be stuck at the furthest point of complexity to which our development methods can reach.

So we try to improve our development methods. Toward that end, the programmers of our industry have dived headfirst into a pool of object-oriented programming (OOP) books; they bury their programs' data and flow control deep within tangled nests of C++ features. I believe that these recently adopted ideas are wrong, that these techniques are failing us. After all, the proof is in the pudding — if the techniques were working, then games using them would become shining development successes. But to the contrary, the games making heaviest use of C++ features are among the tardiest, most spectacular messes. The programmers of our industry are buying heavily into a dogma that has not been sufficiently field-tested.

But if you denounce this dogma in public, you'll be scoffed at. How could that huge stack of OOP books be wrong? What kind of joker are you to deride these things that have become standard practice? Well they can scoff at me, because I'm just not buying any of that crap. I spend a lot of time doing contract work with various developers, and I see them shooting themselves in the foot with this stuff, over and over.

Programmers are buying heavily into a dogma of object-oriented formalism that has not been sufficiently field-tested — leading to tardy, spectacular messes.

I want to explore alternative programming techniques, in areas far away from the current OOP morass. We're now in the habit of needlessly complicating our programs; we need techniques that simplify programs, enabling us to accomplish more with less exertion. This month I begin a series of articles investigating one trail off the beaten path. I don't promise that it will

revolutionize programming, but it may help some. If every game studio were doing a little bit of open-minded investigation like this, we'd all be much better off.

Data Structures and Redundancies

As experienced game programmers, there are many small problems we solve by rote — we set up some data structures and go, same as we've done for years. I'll illustrate this with an example of such a situation and then question it.

Suppose you're making a game, and you've got objects in the game world — Entities — and some of these Entities are players who can carry other Entities. We need some way to represent who is carrying what. Typically we would implement this as follows: each Entity is represented by an instance of a C++ struct or class, and each of those instances contains a list (or an array) of all items carried by that Entity. When a player gives a command to drop an item, we step through his "carrying" list to find that item, and we remove it.

Now suppose we have a magical disappearing Stone that needs to remove itself from our inventory of its own volition. Starting with only a pointer to the Stone, we need to remove that Stone from the appropriate carrying list. We could find that list by iterating over all Entities in the world, searching for the Stone in every carrying list. But that's a big pain and it's slow. So we tend to put a pointer on the Stone (actually, on every Entity) that points to the entity that carries it (or NULL if the Stone is not



JONATHAN BLOW | *Jonathan is not normally this ranty, but recently he has become a student of rant-jitsu. If you know any good instructors, send recommendations to jon@number-none.com.*

being carried). Now we face an odd issue. A single fact about the world, “The Dude is carrying the Stone,” is represented by two different pieces of state: the `carrying_list` on the Dude, and the `carried_by` pointer on the Stone. Our code must be careful to keep these two pieces of data in sync, or else we’re in trouble.

Quantifying this situation, suppose that the concept of “carrying” represents one unit of game functionality (because it’s difficult to see how you might subdivide “carrying” into sub-concepts that make sense on their own). How many units of work does it cost us to implement this one unit of functionality, “carrying”? As described earlier, we need to first (1) implement the `carrying_list`, then (2) implement the `carried_by` pointer, and then (3) maintain the constraint between them. By a naively simple method of accounting, this is three units of work, yielding one unit of functionality. Probably item (3) is more expensive than the other two, as it’s more subtle and is a long-term concern.

Ideally, to implement one unit of functionality, we want to do one unit of work (or less). Now, taken in just the isolated case of “carrying,” this three-unit situation is not a huge problem — we do the work and then move on. But as our program becomes larger, filled with analogous cases, and becomes host to a tangle of interdependent concepts, suddenly we find that we’re spending great piles of work units for mere handfuls of functionality units. We can use all the simplicity we can get.

High-Level Languages?

The stated goal of a high-level language is to reduce the amount of work required to create software. However, I think most such languages take insufficient approaches toward this goal (this is especially true of the scripting languages we have been developing for games). Often they provide features that make programming a little easier, but they fall short of the dramatic changes we need. Using a language like LISP or Objective CAML to implement our “carrying” functionality, we may end up typing a little bit less, so our three units of work will become slimmer, but they are still three pieces of work, and that’s bad.

The situation can be compared to optimizing a program’s CPU usage. Suppose you profile your program, and there are some frequently called functions that are taking up a lot of time. The straightforward approach — streamlining these functions to make them faster — will give you a speed benefit. But experienced optimizers know that it’s more effective to redesign the code to eliminate those functions (if possible). A function that you think is fast is infinitely slower than a function that doesn’t exist.

We face the same situation with another optimization problem: reducing the time we spend creating software. Many languages have set out to reduce the amount of typing you have to perform in order to make things happen, but if data manipulation and flow control still occur at roughly the level of C++, then there’s a limit to how fast programming can go, and it’s not all that much faster than what we already have. Rather than trying to make our small-grained manipulations faster, we should find ways to eliminate them altogether.

LISTING 1. DATABASE FACTS.

```
(female ann)
(male mark)
(male don)
(parent mark don)
(parent ann don)

// Inference rule:
// Is y the sister of x?
(sister ?x ?y) <- (female ?y), (parent ?x ?p), (parent ?y ?p),
(notequal ?x ?y)
```

The Predicate Logic Experiment

Which brings us to the title of this article. To eliminate redundancies like `carries/carried_by`, I want to use an unstructured database that just holds facts about who carries what. There are several ways to build such a database, but I chose the style of first-order predicate logic. Predicate logic has been deeply studied throughout the history of computer science so a simple web search provides lots of reference material.

To use predicate logic for our “carrying” example, we want to insert a fact into the database that says “Dude is carrying Stone.” Only that single fact represents the carrying relation, so there is no redundancy. We can perform queries like “What is Dude carrying?” and “Who is carrying the Stone?” with equal ease — the database engine uses a matching system to generate the answers.

To represent facts, my syntax is a set of arguments enclosed in parentheses, with the predicate listed first. So the fact “Dude is carrying Stone” is written like this: `(carrying dude stone)`. The “carrying” is just an arbitrary label that we are free to choose. In a real game, you would want “Dude” and “Stone” to be identifiers of instantiated Entities. For this simple example, though, we will just use labels for them.

To ask what Dude is carrying, we perform the following query: `(carrying dude ?x)`. The question mark indicates that `x` is a variable. The database engine looks for any facts that are three arguments long and have `carrying` and `Dude` as the first two arguments. It returns a list of all possible values for `x`; in other words, everything that Dude is carrying. If we want to know who is carrying the Stone, we can make this query: `(carrying ?x stone)`; now we will get back a list containing everyone who is carrying the Stone (which should be only one or zero items long).

As programmers who care about the speed of things, we might worry: when there are a lot of facts in the database, how do we organize them so that these queries can be answered quickly? My answer for now is, we’re simply not going to optimize. All database items will be stored in one big linked list and the search will proceed through them all. I will be changing this system rapidly over the next few articles, so it needs to be very simple and flexible.

LISTING 2. QUERIES AND RESULTS.

```
(sister mark ?x) -> { x = ann }           // "Who is mark's sister?"
(sister ?x ann)  -> { x = mark }         // "Who is ann the sister of?"
(sister don ann) -> false                // "Is ann don's sister?"
(sister ?x ?y)   -> { x = mark, y = ann} // "Who are all the sisters we know about, and who are they sisters of?"
(sister ? ?y)    -> { y = ann }         // "Who are all the sisters we know about?" (not caring who they are sister of)
(?r mark ?x)    -> { r = parent, x = don ; r = sister, x = ann } // "What relationships do we know of with mark in the second slot?"
```

In the long term, when we care about speed, we can let the script programmer set policies about which labels or argument slots get indexed by primary hashes, secondary hashes, and so on — decisions that are hopefully informed by a good profile report. These policies would be set late in development, allowing for a rapid development model where the programmer can create initial functionality without worrying about speed.

Interestingly, we now have a full implementation of the “carrying” functionality without performing any of the three work units listed earlier. We have no interdata constraints to maintain, and we don’t have to declare anything either, assuming we can just dump all the “carrying” facts into a global namespace. I would say that there’s about one work unit here, which involves remembering that the label “carrying” has been used, and that it’s a predicate with two arguments, first the guy who carries, second the guy being carried. Though this isn’t directly comparable to our earlier implementation method, it seems to be much less effort.

With this predicate logic approach applied across an entire code base, the resulting simplification would be significant. Speaking of lists of Entities, in my current game, implemented the old-school C++ way, I have many views of the same set of data: one hash table of all Entities that exist, mapped by an integer ID; one list for each type of Entity (containing all of that type); lists for Entities that have been created but not fully initialized by network traffic; lists of Entities that have been fully initialized but not yet added to the world, or that have been removed from the world but not yet destroyed; and lists for Entities that are “awake” or “asleep” according to the physics system. All of these are just the poor C++ programmer’s method of performing simple hard-coded database operations. (There are further redundant views that are beyond our scope today, such as spatial partitionings of Entities for visibility culling or collision management.) Properly juggling Entities between all these lists, in response to game events, can be a challenge.

Within a fact-database framework, we can perform database queries directly, and most of the lists above simply disappear — they go away and are replaced by nothing. It is infinitely easier to program nothing than to program a lot of small things.

The Logic Part

Besides adding bare facts to the database, we can also add rules of inference: if such and such things are true, that

implies that this other thing is true. The implied fact can be queried, just like facts that are asserted directly into the database.

Let’s look at a time-honored example, the “sister scenario,” which seems to have been expounded in every tutorial ever written about the programming language Prolog. We have some people, Ann, Mark, and Don, and some facts about them: (**female ann**) — “Ann is female,” (**parent ann don**) — “A parent of Ann is Don,” and so on. See Listing 1 for the full set of asserted facts. In addition to these, we can assert an inference rule, “Is y the sister of x?” as follows:

```
(sister ?x ?y) <- (female ?y), (parent ?x ?p), (parent ?y ?p),
(notEqual ?x ?y)
```

The <- means “is implied by,” and the comma is a logical AND. So y is the sister of x if y is female, the parent of x is some p, the parent of y is that same p, and x and y are not equal (assume **notEqual** is a built-in primitive). Now suppose we want to perform a query like “Who is Mark’s sister?” — in other words, (**sister mark ?s**). The database engine will match this query against the rule for sister, substituting the arguments **mark** for ?x and ?s for ?y, giving us the following temporary rule:

```
(sister mark ?s) <- (female ?s), (parent mark ?p),
(parent ?s ?p), (notEqual mark ?s)
```

The engine will then attempt to handle all the queries on the right-hand side. If each query can be resolved (whether by a direct assertion or another inference rule), then a result will be returned, a binding for the ?s in (**sister mark ?s**). In this case, the query will return Ann as the result. See Listing 2 for more examples of queries we can perform. More complex inference rules can be asserted, including recursive ones (as we will see in a future article). The general framework of predicate logic can include other operations besides what we have described here. However, just with assertion and implication, we can do some very interesting things.

Sample Code and Next Month

This month’s sample code (available at www.gdmag.com) provides a simple predicate logic assertion and querying system. Next month, we’ll expand the power of this system, building it into the larger framework of a full programming language. 🐉

And a Partridge in a Poly Tree

Dear Santa, it's getting around to the holidays, and I'd like to remind you that I've been a very good animator this year. I made all my milestones, more or less. I was very disciplined about keeping my source control up to date — and you know what a pain that can be. Heck, at a party, I even said some nice things about my producer — when he wasn't anywhere within earshot! So as you can imagine, I'm feeling pretty virtuous, and I've got a rather long list this year.

First off, let me say that I'll make it easy for you this time around: no feature requests! So I won't say anything about being able to play back my animations without prerendering. I won't even mention IK/FK switching, or real-time muscle systems. I've learned my lesson and I'm going to stick to practical requests.

1: A MiniDV Camcorder

I might as well start with the big one: I'd really like a miniDV camcorder for capturing reference. Trying to catch my reflection in the conference-room window while I'm crawling around on the floor pretending to be wounded has given me a serious crick in the neck. I promise this is only for study — I swear I'd never stoop to rotoscoping! I just want to be able to scrub through things, look at them from different angles, and see how the actions unfold. I especially find it useful to look at the motions playing backwards because it helps to highlight the anticipations and follow-throughs.

Please, Santa, before you skip to the next item, hear me out! Nowadays capturing video is simple and cheap. All you need is a \$20 Firewire card and a camera with a Firewire port. I won't need any fancy editing software — I can just grab clips with Windows Movie Maker or iMovie on a Mac. Since I'm only concerned with reference, I'm not asking for one of those 3-CCD monster cams. All I want is a camera that takes decent footage in natural light so I can catch myself working out moves, or go to the zoo to capture some animal reference. The camera doesn't even have to be the latest model — almost any miniDV camera built in the last three years is probably adequate. I'd be very happy with one of the early models of the Canon ZR series, for example — the latest entry level model (the ZR60) is around \$500, but the earlier models like the ZR45 can be found online for as little as \$350. I like the Canons because they have good image stabilization, so footage shot by hand doesn't have that distracting shaky-cam effect.



Any miniDV format camcorder, such as this Canon ZR60, is adequate for capturing reference footage in natural light.

Now, since I've been so good this year, would it be too much to ask the elves to include a progressive scan mode too? I know that progressive scan generally kicks the price up a bit, but it's vastly better for reference work. Progressive scan captures full-frame images, instead of half-resolution interlaced pictures, so the stills are sharper and don't suffer from that nasty interlace blur. Pro-scan video compresses better too, so it's easier to keep around on disk. I've seen new progressive scan cameras under \$650, but I'd be satisfied if you can have the elves scrounge up an older JVC GR-DVM80, which now only costs around \$500.

2: An MPEG Video Camera

I know that \$500 — or even \$350 — is a big present, so maybe you want a cheaper option. I could try one of the little SD-card-based cameras that are coming out now for around \$100, like those from Mustek and SiPix. They always squeeze a "DV" into the name but they aren't related to miniDV cameras.



STEVE THEODORE | Steve started animating on a text-only mainframe renderer and then moved on to work on games such as *HALF-LIFE* and *COUNTER-STRIKE*. He can be reached at steve@theodox.com.

Instead they shoot MPEG-4 video. Unfortunately it's usually pretty low resolution — 320×240 at about 15fps seems to be the going standard. 15fps is pretty crude for subtle timings, but at least it should be good for establishing key poses and tempos. Plus, most MPEG cameras are tiny so they're easy to carry around, and some of them (for instance, the Panasonic SVAV30) double as MP3 players, so maybe it wouldn't be too bad. For around \$200, Gateway's DV-S250 MPEG camera promises around 24fps, which should be enough for some serious work. None of these little cameras includes auto-focus or auto-exposure, and they won't be as easy to work with as standard camcorders but they are a lot cheaper.

Actually, the best cheap option is probably a Firewire- or USB 2.0-based webcam. Now, the image quality isn't as good as a camcorder, and obviously a webcam is going to be physically tied to the computer, but the throughput is high enough for 640×480×30 capturing. The frame rate and size are a lot more important for motion study than color fidelity, and I don't absolutely need auto-focus when I'm just capturing around the office. I have heard that webcams need more light than camcorders, since they use smaller sensors, so maybe you should throw in a little halogen mini-spot. It shouldn't be too much of a stretch, since a Firewire webcam such as the Orange Micro I-Bot can be had for as little as \$100. If you are thinking about the I-Bot, though, please throw in a tripod — that little wire foot thingie is rather prone to tipping over.

3: Books on Animation

Now, I know a camcorder, even a little tiny one that shoots grainy MPEGs, is a pretty big present, so I'll try to understand if you don't get it for me. But surely you can't object if I ask for books — we're all supposed to pursue self-improvement, right? So here's a list of books I'd love to fill my bookshelf with.

The very best thing you could get me from the bookstore is the complete three-volume set of Eadweard Muybridge's *Animal and Human Locomotion*. The short versions from Dover, *Human Figure in Motion* and *Animals in Motion* are okay, but the big hardcover edition that covers his whole career is much better printed and clearer. Moreover, a lot of interesting material isn't in the smaller editions — particularly unusual subjects like people walking with crutches or disabilities. Unfortunately, Volume 3 of the complete set, which covers animals, is very hard to come by, even on the web. But Volumes 1 and 2 are priceless, and the short edition of *Animals in Motion* is better than nothing. Even after 115 years, Muybridge is essential for anybody who wants to study motion over time. If only somebody would do equally good editions of other early motion photographers, such as Etienne-Jules Marey or Otto Anschütz. Maybe for next Christmas?

Another huge hardcover book I'd really love to have is Richard Williams's *The Animator's Survival Kit*. Books by traditional animators tend to get bogged down in the mechanics of timing for film or economizing drawings, but Williams soars



Cutting-edge animation tools: *The Animator's Survival Kit* and *Posefile Reference*.

above the minutiae. Williams's style translates particularly well to computer animation, because he's always argued for a very light touch on the most graphic squash-and-stretch techniques, in contrast to other old masters like Preston Blair. His staple technique focuses on the use of progressive breaks — whip action — to emphasize movement rather than wholesale deformations, and thus works much more naturally in a skeletal animation system. But *The Animator's Survival Kit* is not just a handbook on fundamentals; it's also a treasury of several decades' worth of practical solutions to common animation problems. If you think I've been really good this year, maybe you could just get me a spot in one of Williams's Animation Master Class seminars.

Another great reference that I'd really like to have in the bookshelf is *The Artist's Complete Guide to Facial Expression* by Gary Faigin. Perhaps because this book is intended for traditional artists, it's actually much easier to work with than photo reference — the drawings emphasize the main components of displaying facial expression more dramatically than the photos in many competing books. The accompanying text combines detailed knowledge of anatomy with a good theatrical sense of mood and nuance. Even though there are a number of decent books that cover some of the same ground and also discuss practical problems such as morph targets and mesh construction (notably *Animating Facial Features and Expressions* by Bill Fleming and Darris Dobbs and *Stop Staring* by Jason Osipa), Faigin's book is still the most comprehensive treatment of facial expressions and the one I'd pick first.

I'm a little embarrassed about this next one, but I swear my interest in the *Pose* series (*Basic Pose*, *Daily Pose*, and *Moving Pose*) of reference photo books from the Japanese publisher Bijutsu Shuppan-Sha (Books Nippan in the U.S.) is purely professional. The gimmick of the series is photos capturing actions from multiple angles, which is great for keeping track of things that often get obscured when the reference contains only one view. The fact that they tend to have a nude and lingerie section is just accidental. Really! They're very hard to get outside of

Japan these days, but you can usually find them in bookshops that cater to animé and manga fans. I hear that Antarctic Press, a manga-style comics publisher in Texas, is starting a similar series under the name *Posefile Reference*. The first volume includes 360-degree views, low, medium, and high shots, and a pretty impressive arsenal of prop weapons from swords to grenade launchers. There doesn't seem to be a lingerie section, although all of my Japanese-schoolgirl-in-sailor-suit-with-big-gun reference needs will be nicely covered.

The last item in my book list is a small volume that seems like a stocking-stuffer after all these enormous reference books. *Acting for Animators* by Ed Hooks is a condensed introduction to acting theory aimed at animators rather than stage actors (for a taste, turn to page 30 of this magazine for his feature "Chasing Gollum"). The book's main idea is that our job starts, rather than ends, with the mechanics of movement and timing. While it doesn't have much to offer on mechanics, *Acting for Animators* is a great reminder of what we should be striving to do in our work. If you're feeling especially generous, Santa, you could even get Ed Hooks to come out and give one of his workshops for the whole department! But seriously, Santa, performance is a terribly overlooked topic in our business. If we want to create characters that move instead of just characters that



Stickfas action figures blend the artist's mannequin with iconic action sculpting, resulting in a great tool for posing practice.

move, we're all going to have to accept the fact that animation is as much a branch of theater as it is of computer graphics. Come to think of it, how about giving a copy of this book to everybody in the department?

MUST-HAVE GAME BOOKS

1-58450-288-4 \$69.95

1-58450-277-0 \$49.95

30% Pre-Pub Discount on "Coming Soon" titles at charlesriver.com!

1-58450-258-4 \$49.95

1-58450-214-2 \$39.95

JOGD (Journal of Game Development)
Subscribe & Submit Papers at jogd.com

(800) 382-8505 www.charlesriver.com
Also available at Amazon.com, Barnes & Noble, Borders, and other fine retailers.

Slow C/C++ Builds?

Download
FREE, Fully Functional
30-Day Trial!

IncrediBuild accelerates C/C++ builds by distributing compilation tasks across the network, cutting down build time by 90% and more!

- Simple setup, requires no changes in code or settings
- Compatible with any Visual C++ Win32 project
- Fully integrated with MSVC's IDE

XOREAX

www.xoreax.com

4: Gifts for Coworkers

For Yu Tang, who has trouble with his timings: a sports stopwatch. There are two requirements. First, the watch has to be accurate to 1/100th of a second (the 1/10th of a second watches that you see for \$10 or so are usually pretty inaccurate, as well as a little crude for key-timing). Second, the watch needs enough memory for at least a few dozen laps and splits. Delta timing, which displays the offset between “laps,” which for us means timing keyframes, is a useful extra. Probably the most important feature of all is large, easy-to-use buttons that can be operated while acting something out. It’s a shame that nobody seems to make a stopwatch that works in SMPTE time-code or 30Hz frames, since it’s a pain to do the math to turn 1/100th of a second measurements into frame numbers. The Frame Master II time-code calculator from Calculated Industries doubles as a stopwatch, but at more than \$100, I’m not sure it’s worth it just to avoid some math.

For Sharon, who spends way too much time staring at her hotkey cheat-sheet: an X-Keys Desktop programmable keypad from P.I. Engineering. The X-Keys Desktop, which has 20 buttons (capable of holding up to 38 functions), has three main advantages as a hot-key replacement. First, it is very easy to program — it takes any arbitrary sequences of characters, so wacky combinations such as Ctrl-Alt-Shift-P are easy to set up. In fact, you could even program a macro string or script into a single button. Second, it stores all of its programming internally in an EEPROM, so you can unplug it from one machine and take it to another with no hassle. Finally, it comes with removable key caps, so you can print out and attach labels. The only drawback to the X-Keys box is that the device doesn’t know what application you’re running, so there’s no way to have different hotkey mappings for different programs. P.I. makes a variety of programmable input devices, including jog-shuttle dials and footpedals. The basic X-Keys box is about \$80, while specialty devices such as the jog-shuttle can run as high as \$250.

For Alexander, who needs to work on

the rhythms in his gestures: a Microsoft Sidewinder joystick. It’s not for games — it’s a simple way to let him do gestural input of timings. He can read the motion and buttons on the joystick right into Max via the Motion Capture utility, or into Maya with the freeware program JoystickServer.exe. It’s a great way for people who have good posing skills but trouble with sequencing to get some visceral feeling for the timing of their extremes. Even if he never tries to capture an animation directly off the joystick, just recording a bouncing-ball movement on a dummy object is a great way to control the tempo of a scene. If you’re not willing to shell out the \$25 for a joystick, at least give the poor guy a simple noise-canceling PC microphone. Almost any mic will do — all he needs is a way to hum, mumble, or say, “Da-dum-di-dum-di-dum!” into the Windows Sound Recorder. And once he gets used to having that nice waveform display in his timeline to help him hit his

beats, he’ll thank you for it.

For Dean, who is a great concept artist but whose characters always look like they’ve got lumbago: a set of Stickfas action figures. These little guys look like the result of crossbreeding a traditional artist’s mannequin with one of those little 1980s GI Joes. They’re very iconic in design, and since they’re not burdened with typical action-figure sculpting they’re eminently poseable. They usually come with some mix-and-match equipment, and they can even be put together in non-standard ways to create unusual characters. At less than \$10 for the basic model (ranging up to about \$25 for a dragon), Stickfas make a great stocking stuffer for anybody who needs some posing practice.

Finally, for my producer: no coal — I’m not vindictive. How about a large bottle of Tums and an audio book (does he know how to read?) of Wess Roberts’s *The Leadership Secrets of Attila the Hun*? Happy holidays! 🎁



3D PIPELINE
entertainment

PRE-VISUALIZATION. MOTION CAPTURE.

THE MOST PROVEN MILITARY 3D SIMULATION ENGINE IS ABOUT TO PRODUCE ENORMOUS COST SAVINGS FOR ENTERTAINMENT PRODUCTION.

IMAGINE THAT YOU'RE AT THE HELM, ONLY NOW IT STARTS BEHIND A PC. WITH A FULLY ROBUST MOCAP SET-UP, YOU GET FULL COVERAGE PLUS YOUR OWN PRE-VIZ POV AND TWO ADDITIONAL CAMERA ANGLES. WITH THAT, THERE'S SIMULATION OF THE FOCAL LENGTH, SO DISTANCE AND DEPTH POINTS ARE TRACKED. IMPORT LIVE OR PRE-SEQUENCED FOOTAGE, AND MODEL OR IBR-BASED ELEMENTS. USING A HIGHLY ADVANCED COMPOSITING TOOL, SET YOUR SPATIAL COORDINATES, DIFFERENTIATE BONES, DETERMINE THE CAMERA PATH. CELL SHADE TO SMOOTH OUT THE EDGES FOR THAT EXTRA POLISH. EVEN EDIT, JUST TO SEE HOW IT WILL LOOK FOR THE FINISH. ALL WITH REAL-TIME RENDERING.

YOUR IDEAS, OUR TECHNOLOGY, UNLIMITED POSSIBILITY. THIS IS THE POWER OF HYPERVIS. FROM 3D PIPELINE ENTERTAINMENT.

CONTACT: JIM SLAZAR/GUNTHER SONNENFELD 310.566.6230
INFO@3DPENTERTAINMENT.COM

Don't Pull Your Pito To Get off the Bus

There's more to dialogue localization than being multilingual. For example, in Colombia, the word *pito* refers to the cord you pull to signal the bus driver to stop. However, in Mexico, where they speak Spanish with their own regional and cultural dialects, the same word is slang for "penis."

If your plans include localizing your game for different markets, the time to prepare is at the beginning of the development process. There are several things you can do to reduce the cost of translation, making your move to other markets much more profitable.

Size does matter. When translating English into almost any other language, more syllables are required to say the same thing. For example, an English sentence translated into Japanese or German can be up to 20 percent longer.

So what happens in a cutscene with dialogue that is tightly synchronized with the graphics? If there is no space for the dialogue to spread out, either the actor has to read the lines more quickly (which will sound rushed to a native ear) or the dialogue has to be crafted ahead of time by the translator so that it will fit the allotted time. The latter scenario is better, so long as you realize the problem before the first round of recordings is made — which can otherwise become a costly mistake.

Alternatively there might be a creative way to put together the cutscene so that there is some room to maneuver the audio — again, something that won't cross anyone's mind if only an English version is being considered.

By the way, length is also a consideration for anything written in the game — can your maps, inventory lists, and menu items adapt to phrases of different lengths? Is it easy to update every single piece of text without having to recompile or re-render anything?

Don't try this at home. In the same way that an intern is probably not the right



casting decision for the voice of a Crypt Warlock, a level designer is probably not ideal for translating dialogue or voice acting just because she speaks Spanish.

You may need a translator who can create dialogue that keeps within broad regions that have localized differences. For example, Spanish pronunciation in Spain is radically different from Mexican Spanish, so a Mexican actor delivering the lines of a conquistador would be as out of place as an American actor portraying Winston Churchill. However, there are a few exceptions in which certain languages can be written and delivered in a manner that sounds generic across a fairly wide region. Consider the way all newscasters in the U.S. sound like they come from Iowa.

Cultural considerations. Localization professionals must not only get the words right but the cultural nuances as well. Consider that "football" means "soccer" to most of the world, and many non-U.S. fans prefer English terms when talking about American football. When International Contact was localizing an American football game recently, they consulted with translators and sportscast-

ers in Montreal to get French versions of idiomatic American terms, translating "go-to guy" as "*receveur désigné*" (designated receiver), for example.

Expert knowledge can also be useful when choosing voice characters — keeping the same voice types might not be the best choice for a localized version. A voice type that works well for the American market may seem weak or just plain strange in another culture, so consider allowing for some creative leeway in the casting and translation for non-English versions of your game. This approach is more likely to produce a version that seems as if it were created in the target country rather than originating elsewhere.

Speaking of being in the country, you may also want to cast and record actors in the countries you are exporting to. If your game needs a large number of different character voices, you need a large pool of talent from which to draw. It may be cheaper and easier to get a large group of good Japanese actors in Japan than in the U.S. However, casting and recording in another country requires experience and good contacts.

Local loyalty. Often, companies that take a country's language and culture into consideration are rewarded with dedicated fans in that market. Since brand loyalty is extremely significant in many countries, spending more to localize by region is worthwhile.

Yet the greatest benefit in understanding localization is simply becoming conscious of the potential hazards. With a little preparation and advanced planning, you can save time, money, and a lot of embarrassment. 🍷



JON GOLDING | Jon is co-author of *The Apple Guide to Localizing Multi-Media* and VP of International Contact, a multi-language communications agency. Contact him at jon@intlcontact.com.



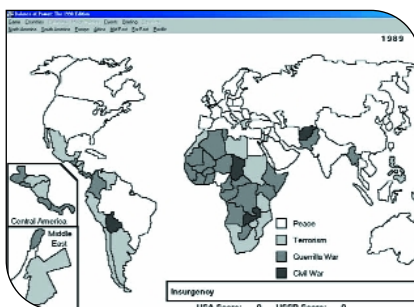
SIMON AMARASINGHAM | Simon is a composer, sound designer, and co-founder of dSonic, an audio production company dedicated to the game industry. Contact him at simon@dsonicaudio.com.

Eat Your Vegetables

This month I'll consider rules gleaned from the recently released book *Chris Crawford on Game Design* (New Riders). I should begin by stating that I have no financial ties to Chris's book, and although I've known him for many years we're not close friends. Furthermore, I've learned to be cautious in recommending his lectures or writings to others. In fact, to continue a food-related theme from my last column ("Tapas-Down Design," November 2003), I've found that my appreciation of Chris's point of view is much like my love of brussels sprouts. Sure, they can be bitter and occasionally unpleasant, but they can also be very good for you and sometimes surprisingly pleasant. Perhaps it's an acquired taste — or perhaps it's deeper than that. I participated in a study in college that suggested that there is a genetic variation among people's sense of smell related to their appreciation of cabbage-related vegetables, so I may also have the Chris Crawford appreciation gene too.

Most of the examples in the book focus on Chris's own games that go back 25 years, but I found much of his advice to be valid and topical. What really caught my eye were the "lessons" included as small sidebars. These are brief, imperative sentences intended to guide the behavior of developers — in short, much like the 400 Project rules. He has 96 of them in the book. As with much of Chris's work, I think some are right on target, others interesting, and some just plain wrong, but they're all thought-provoking.

I've tried to solicit rules from other established game developers with limited success, so I was excited to see this potential treasure trove. But alas, we're not 96 rules closer to the goal of 400 game design rules, as many of his rules are not strictly design oriented. They include producer-oriented rules like number 72: "No matter what the schedule says, give the game enough time to



Chris Crawford's *BALANCE OF POWER*: lessons from gaming history.

get it right," or 83: "Decide whether to hire an artist or an illustrator." There are quite a few programmer-oriented ones, like 80: "Use only one mathematical operation per line of code." And many range from the philosophical (94: "Integrity is an unexpected virtue") through the cynical (86: "In the games biz, trust no one") to the cranky (36: "Young programmers can be stubborn asses" — a bit of Chris's infamous curmudgeonly streak there). Lest you jump to conclusions from my excerpts, he does take the time in the main text to justify and explain each of these points.

Study your lessons. So what are some of the design-oriented rules that I found particularly useful? Here are a few:

11: "Interactivity is the essence of what you are selling." Chris has always been a proponent of this point of view, since all too often designers start to think like frustrated filmmakers and include long, boring movies.

31: "Begin each project with a one-page specification of the gameplay." Not a universal truth, but a handy, practical place to start.

43: "Always ask, 'What does the user do?'" A great precept, many amateur designers forget this repeatedly.

Chris also has a lot of solid advice about becoming a designer and thriving:

13: "Read more."

23: "Don't get a job in the games industry unless you really, really love games." Great advice — otherwise you'll be at a competitive disadvantage to the many who do.

24: "Learn everything you can."

53: "Read! Read! Read!" See a pattern here? All the great designers I know read a lot.

56: "Polish, polish, polish! Take a minimum of 6 months after alpha for polishing." A good rule — but all too often real-world contingencies get in the way.

Bitter sprouts. Some of the lessons are, in my opinion, more indicative of Chris's own prejudices than of underlying truths. For instance:

39: "Other people can't see your vision; you have to make it happen yourself." Some people do work best as individuals, but there are numerous counterexamples of game designers who have successfully shared their vision with a large team.

47: "Sequels are for entertainment, they have no artistic content." I think the folks at Blizzard as well as Shigeru Miyamoto might take exception to that one. For that matter, the implication that "entertainment" and "art" are separate categories doesn't sit well with me. As I warned at the start of this column, this book is not for everyone. But to sum up, Chris says it better than I do:

40: "Always be on guard against the tendency to think in the old ways." 🐉



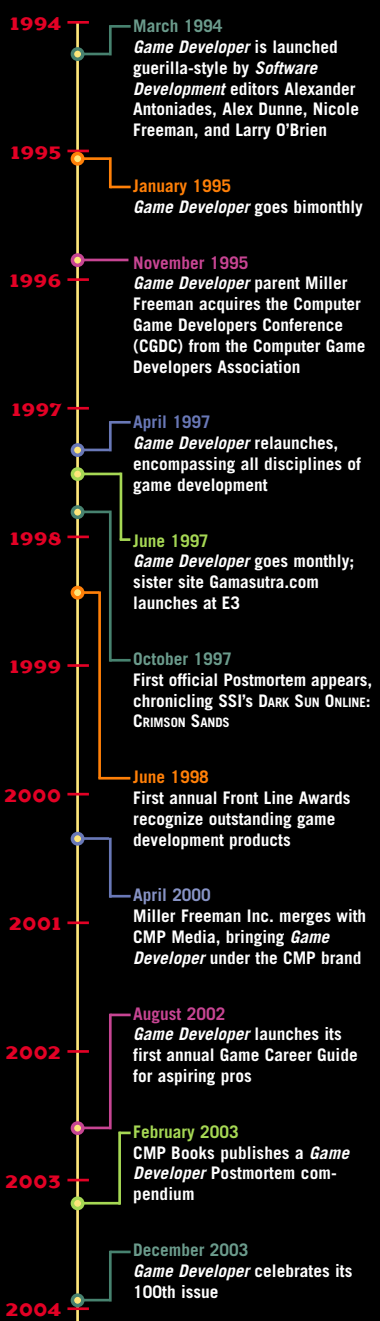
NOAH FALSTEIN | Noah is a 23-year veteran of the game industry. His web site, www.theinspiracy.com, has a description of *The 400 Project*, the basis for these columns. Also at that site is a list of the game design rules collected so far, and tips on how to use them. You can e-mail Noah at noah@theinspiracy.com.



GAME DEVELOPER'S 100TH ISSUE

Game Developer's

100th Issue Retrospectacular



Staving off the unrelenting advance of deadlines doesn't offer much opportunity for reflection for those of us who put *Game Developer* together each month. But recently, as I was unceremoniously shoving another journalistic triumph into my sacrosanct canon of complete back issues, I stumbled upon the realization that my shelf was straining under the weight of nearly 100 issues of *Game Developer*. Looking back at the long history of a magazine in a very young industry, the number seemed at the same time so many and yet so few.

The truth is, everyone and everything in the game industry is getting older. Or perhaps more mature. Yes, let's say more mature.

In the Beginning

Issue number one of *Game Developer* magazine was born into somewhat deliberate obscurity in the spring of 1994, when 20,000 copies went out to newsstands nationwide. Once the magazine was out on the street, its obscurity was limited only to its parent company, Miller Freeman, and its executives: *Game Developer* was a guerilla launch. Only after garnering widespread favorable response from the first issue did an insurgent cabal from the editorial team of *Software Development* magazine tell the purse-string holders where their money was really going.

After almost a year of volunteering their time (and some of *Software Development*'s budget) to the magazine's first issues, founding editors Larry O'Brien, Alex Dunne, Alexander Antoniadis, and Nicole Freeman got the corporate investment they needed to keep *Game Developer* going officially, though at first on a bimonthly basis. "The sell-through on that first issue was something like 60 percent, which is astronomical in newsstand terms," recalls co-conspirator Dunne, who served as editor-in-chief from 1996 to 2000 and now

produces Gamasutra.com. "It was extremely encouraging."

Fortunately for *Game Developer*, many of the folks who picked up that first issue supported it by subscribing, contributing, and spreading the word. "I received a copy of the premiere issue," remembers longtime "Graphic Content" columnist Jeff Lander. "I was amazed that there was enough of a community to support a magazine devoted to the industry. It legitimized in my mind that this could be a real, grown-up job."

Others, sensing the vacuum of available information at the time, couldn't wait to contribute. "I bought a copy of issue #1 and called the editor that same week asking for author's guidelines," recalls frequent contributor Matt Pritchard of Ensemble Studios. "He blew me off, leading me to get mad and yell at him online, which led to my first article in issue #4."

Building a Name

Gradually, the magazine picked up steam, and it would evolve substantially over the years as the industry and the market developed. Back in the earliest years, *Game Developer*'s focus was strictly on programming and almost exclusively programming for PCs. Its first tag line, "Programming for fun," reflected the idea that the industry at that time was characterized largely by hobbyists. Later, the focus grew stronger on the rising prominence of professionals, and the "for fun" factor was downplayed. In April 1997, the magazine relaunched with a new tag line, "On the front line of game innovation," to include all disciplines of game development. The magazine also began to gradually add more console-oriented content.

The first true Postmortem appeared later that year, and the feature's overwhelming popularity hasn't waned since. "One of my favorite parts of making a game is writing the Postmortem," says longtime reader and con-



OUR FAVORITE ARTICLES

tributor Jamie Frisstrom of Treyarch, who has penned four Postmortems for *Game Developer* and Gamasutra. “The games themselves appear on the shelves for a few months and disappear, but the Postmortem is forever.” Within a few years there were enough Postmortems to fill a book, and *Postmortems from Game Developer* was released earlier this year.

Into the Future

While respect for the game development profession has grown during the magazine’s lifespan, the industry and the roles developers play in it continue to evolve as gaming plays a bigger part in the entertainment and business worlds. Some of today’s trends were readily predictable back in 1994: rising production values, increasing IP crossover with other media, and consolidation with market growth. Other trends, however, were not so easily predicted: the proliferation of mobile phones and their near-immediate ubiquity as a new game platform, the spike in consumer savvy and expectations, and the rise of fledgling game studies programs at schools and universities around the world, which is shifting the knowledge-transfer base in game development from the bedroom to the classroom.

“I certainly envy the kids learning this stuff today,” confesses former columnist Lander. “I received an e-mail from a fifth grader a month or so ago who not only understood my particle system article from 1998 [“The Ocean Spray in Your Face,” July 1998], he had improved upon it in a lot of very interesting ways.” Those who’ve been in the business for a few years and are still looking at the studio down the street as their competition might want to check out the schoolyard too. *JD*

by Jennifer Olsen

- ▶ **“Game Development Myth vs. Method”** by Mark Cerny and Michael John (June 2002)
Cerny’s many collaborators over the years may not appreciate that this production manifesto has come to be known widely as “The Cerny Method,” but this article exposed the common foibles that have long besieged game production and deftly explained how to allocate resources to circumvent them. It’s the preproduction, stupid!
- ▶ **“How to Hurt the Hackers: The Inside Scoop on Internet Cheating and How You Can Combat It”** by Matt Pritchard (June 2000)
When online gaming exploded in popularity, developers quickly found themselves besieged by cheaters exploiting game systems to their advantage. Fed up, Ensemble’s Pritchard got into their diabolical minds in this widely circulated investigation.
- ▶ **“Physics, The Next Frontier”** by Chris Hecker (October/November 1996)
Hecker kicked off his seminal four-part series on how (and why) to build a physics engine seven years ago; today the tree of physical simulation is finally bearing the fruits of emergent gameplay.
- ▶ **“Postmortem: Ion Storm’s Deus Ex”** by Warren Spector (November 2000)
I have since forgiven Spector for the sleepless, deadline-added night I spent pruning his magnum opus down to the requisite 4,000 words from the staggering 9,000 he submitted. That Spector himself used to be a magazine editor only made me question his sadistic prolificacy more. Thank goodness he spins a great yarn.
- ▶ **“Postmortem: DreamWorks Interactive’s TRESPASSER”** by Richard Wyckoff (June 1999)
Nearly three years after Hecker’s physics articles, we learned that dynamically stacking crates was

- still pretty difficult. TRESPASSER was the original “and now . . . the rest of the story” Postmortem. I still remember the calls from DreamWorks’ lawyers-claiming-not-to-be-lawyers. TRESPASSER developer Austin Grossman later edited CMP Books’ collected *Postmortems from Game Developer*.
- ▶ **“Game Developer Reports: The Top 20 Publishers”** by Tristan Donovan (September 2003)
Nibbling the hand that feeds us is an irrepressible tradition for game developers, but with the trend toward consolidation in publishing, it was time to reevaluate the publishing landscape and learn how developers were feeling about their place in it. Confidential surveys showed which publishers produced happy developers and which did not. That got their attention.
- ▶ **“Postmortem: Presto Studios”** by Michael Saladino (December 2002)
We broke from tradition and did a Postmortem of a recently defunct studio, examining how circumstances at a company that did so many things right could suddenly turn so very wrong. Life is getting harder for small developers, and sadly I’ve gotten quite a few “it happened to us too” e-mails since Saladino’s article appeared.
- ▶ **“Formal Abstract Design Tools”** by Doug Church (August 1999)
Playing games is easy; talking about game design is hard. Church realized that while hardware was advancing, analytical tools for game design were not. Step one: Try to create a common vocabulary.

Many of these articles are available at Gamasutra.com. Hecker’s articles are available at www.d6.com/users/checker/index.htm. Can’t find an issue with one of our favorites? Pick up a back-issues CD-ROM at the Gamasutra.com Store.



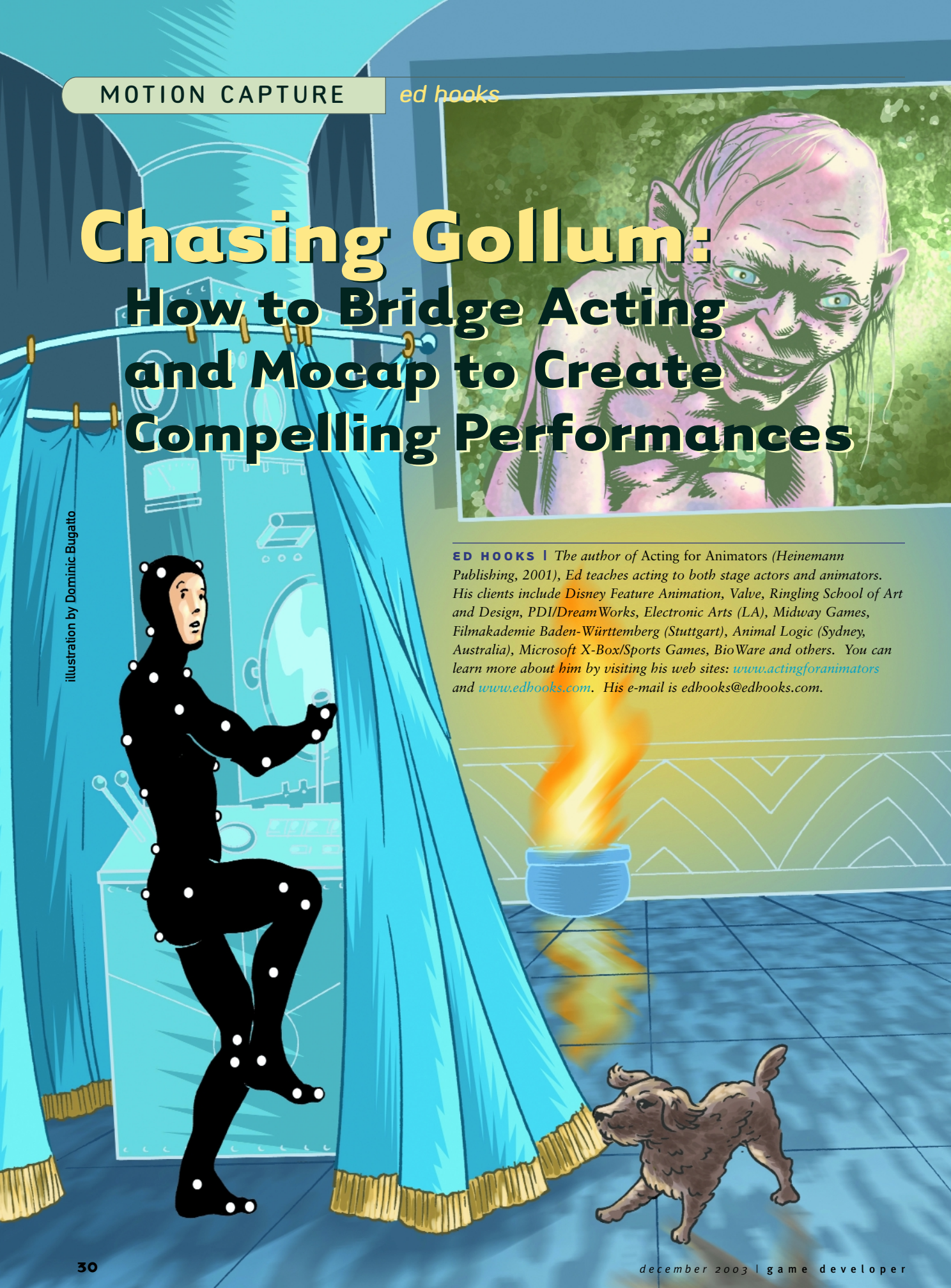
The development team behind *Game Developer*. Front row, left to right: Jennifer Olsen, Kenneth Wong, Audrey Welch. Back row, left to right: Everard Strong, Jamil Moledina, Peter Sheerin

Chasing Gollum: How to Bridge Acting and Mocap to Create Compelling Performances



illustration by Dominic Bugatto

ED HOOKS | *The author of Acting for Animators (Heinemann Publishing, 2001), Ed teaches acting to both stage actors and animators. His clients include Disney Feature Animation, Valve, Ringling School of Art and Design, PDI/DreamWorks, Electronic Arts (LA), Midway Games, Filmakademie Baden-Württemberg (Stuttgart), Animal Logic (Sydney, Australia), Microsoft X-Box/Sports Games, BioWare and others. You can learn more about him by visiting his web sites: www.actingforanimators.com and www.edhooks.com. His e-mail is edhooks@edhooks.com.*



Mocap, the strange sister in the attic, is finally being brought downstairs. Animators may not like their sibling very much, but they're learning to live with her. After years of awkward silence interrupted by angry outbursts, everybody can at least sit in the same room and watch *The Simpsons* together. And by the way, that's Gollum you see crouched in the corner over there. We can probably thank him and the team at Weta Digital for this new living arrangement.

The relationship between animators and motion capture specialists is strained for several reasons. First of all, the respective job functions are a classic left-brain versus right-brain conflict. The technical capture of motion data is arguably more science (left brain) than art (right brain), while animation, regardless of whether it is 3D or traditional, is more art than science.

Second, if peace in the family was part of the goal, mocap got off to a perfectly lousy start. It was sold as the expensive new family car that anybody at the company could drive if they had a set of keys, even if they only had a learner's license. Several years ago, someone got up at the Game Developers Conference and said that it doesn't matter if you are a producer, programmer, or animator, if you're directing mocap, you are Steven Spielberg or James Cameron. Yeah, right. And if I ride a bicycle, I'm Lance Armstrong. Given the production-quality demands of today's consumers, it is essential to recalibrate the game industry's perspective on acting. A company's animators cannot make good acting out of bad mocap.

Let's repeat that: Animators cannot make good acting out of bad mocap.

Having taught at a lot of game companies, I've perceived a systemic schism between the animators and the programmers and between animators and mocap. Given the complexity of programming that goes into the typical game nowadays, plus the astronomical costs of game development, this divide is understandable. But it's a schism that needs to be bridged going forward. More often than not, when I teach a class to game animators, someone will say to me afterward, "Gosh, if only Sam the programmer had been here," or, "I sure wish the mocap people were here. They were working today." The goal is to get the entire production team, from designers to animators, on the same page. But because animators are more right-brained than many of the other developers, it's a difficult challenge.

In an effort to bridge this creative schism, this article presents a collection of acting pointers to mocap developers, regardless of the system used or the size of the company.

Good Acting Isn't as Easy as It Looks

Several years ago, when I went to teach a class at a game company, one of the producers took me into his office to explain what he wanted the class to accomplish. He loaded up one of their current games and started playing it, telling me that he had personally acted out much of the mocap on the screen. He explained that he was a better actor than others in the company, which was why he did it himself. Then he put down the controls and demonstrated how a "good actor" takes a bullet and how a "bad actor" takes a bullet.

I stood there with a silly agreeable grin on my face as this fellow took bullets and did falls on the carpet in his office. He wanted me to teach everybody else in the company to do it like he did it. There was nothing I could do but keep grinning and nodding my head in agreement. I couldn't very well tell him that what he was doing didn't have much at all to do with acting. He may have been the Bill Gates of games and a prince among men, and I'm positive he meant well, but he was not an actor. It was a classic case of someone underestimating the art.

It is true that some people have a natural feel for acting in the same way that Eric Clapton has a natural feel for music or Seabiscuit had a natural feel for racing. However, acting is not usually something a person can do without training. Peter Jackson understood this when he hired classically trained actor Andy Serkis to help create Gollum for his *Lord of the Rings* movie trilogy. The realization of that character was a watermark in animation history, a true collaboration between an actor and the animators. Unless they were all at the top of their form, you wouldn't have had a Gollum.

When I first saw that character come slithering out of the rocks, my jaw dropped. The earmarks of a good actor are all over the place. A prime example is the way Serkis obviously understands Michael Chekhov's concept of the psychological gesture (see *Lessons for the Professional Actor* in For More Information). When Gollum is being extra crafty, note what he's doing with his arms and hands. Those are psychological gestures, and they create a complex emotional response in the viewer, capitalizing on the fact that our sense of sight is more powerful than our sense of hearing.

Casting Is Half the Battle

Good performance animation in a game requires that you begin with a skilled performer. It makes sense to consider casting a physically fit and agile, classically trained actor rather than a gymnast, dancer, stunt person, or athlete if the end



Actor Andy Serkis in full motion capture regalia, and behind the CG mask of Gollum. Photos courtesy of New Line Cinema.

character must interact with others and behave believably in complex gameplay situations. A gymnast or athlete may be a fine choice for a sports or arcade game but will probably not have a lot of knowledge about acting. A performer may be able to move beautifully but still deliver a weak performance. If you have a director who is also weak on acting theory or directing such a performer, you have a certain recipe for a wooden end result.

Shakespeare advised that actors should “Hold the mirror up to nature” (*Hamlet*, act 3, scene 2), and it was more than a superficial suggestion. Your player is hard-wired by nature to read subtle signals in other humans. That is why acting matters. If your game characters move in a stiff and unmotivated way, you will probably not receive any protesting e-mails about it because the players know it’s only a game; they cut you a lot of slack because they know you’re just spoofing. But if someone in the player’s own life moves that way, it would set off emotional alarms.

Human beings begin recognizing what kind of human movement rings true when still in the crib. By the time a person gets to be 13 years old, he or she already has a doctorate in human movement, even if he or she can’t conceptualize and explain it to you. It is primal and evolutionary; we learn from one another through a process of mimesis. We understand early on, for example, what tension in the body looks like and that it often precedes an outburst of some kind; we recognize that relaxation manifests itself as a feeling of weight, not lightness; and we recognize that there are smiles that can be trusted (the muscles around the eyes contract) and smiles that perhaps cannot be trusted (they don’t contract).

Dancers, gymnasts, and athletes do not typically move like normal people. They are trained to stand up straight and not slouch, for starters. If you pass such a person on the street, you recognize right away that he or she is different. An experienced

ballet dancer tends to walk by putting one foot directly in front of the other; an athlete or gymnast is going to be more loose-limbed than most of us. Depending on his or her sport, an athlete will likely have unusual upper-body or lower-body strength and musculature. Stunt performers move more normally but may not know much about acting. I have worked on TV shows with some stunt doubles who were actually pretty good actors, but most of them are more athlete than actor.

In a mocap session, the goal is often to capture hundreds of photo-real moves that will later fit together. From my perspective as an acting teacher, that is a technical limitation, not an end goal. I understand why you must do it and don’t envy you the task. But if you are looking for strong performance animation, movement cannot be separated from acting.

Here are some examples of how motivations and objectives affect movement in a performance, and why the director must keep them in mind to achieve optimal results. If you direct a mocap performer to “walk slowly from point A to point B and stop on the mark,” that is not actor-sensitive direction. The performer must have a reason for walking, a destination. If you don’t give him a contextual reason, he’ll do the move as you request, but the move will be hollow. A character will walk one way if she is moving across the room to take a lover into her arms and another way if she is moving to answer the phone. She will walk one way if she is moving across the room to meet her boss and another way if she is moving to greet a child. She will move one way if she is timid and another way if she is bold. She will walk across the room one way if she forgot to turn off the stove and another way if she has a rock in her shoe.

Certainly, a non-actor will understand simple contextual directions such as “Xenon is chasing you with a ray gun!” or “If you step on a land mine, you’re toast.” But a non-actor may well be weak on characterization itself. Your end charac-



Andy Serkis on the Weta Digital performance-capture stage in Wellington, New Zealand. Photo courtesy www.serkis.com.

ter may or may not be intended to walk and run like the mocap performer you have hired. An actor can likely make the proper physical adjustments; a football player probably can't.

Another example: If a guy is feeling amorous and goes out on a date, he has an objective. He may put his arm around his date's shoulder and look longer into her eyes. In acting theory, these are actions in pursuit of an objective, a concept that is workaday for a trained actor but may be Greek to a gymnast. Actor Constantin Stanislavsky, father of "the method," defined acting as "playing an action in pursuit of an objective while overcoming an obstacle." If an actor knows why he is moving or carrying out a particular action and what the obstacle is, the capture will look more believable.

Even a straightforward war-themed game can improve by understanding this principle. A soldier at war wants to survive. That is his objective. He may have signed up for patriotic reasons, but in the heat of battle he simply wants to do the job and get out of there alive. The objective of survival and the obstacle is the situation. If this scenario is enunciated in mocap session, it will make the character movement edgier and more credible. All humans act to survive. We automatically recognize and respond to the survival strategies of others we meet in the world, including the characters in a game.

Thinking Leads to Conclusions; Emotion Leads to Action

Emotion may be defined as an automatic value response. When you feel an emotion (such as fear), you tend to do something about it — run, confront the danger, scream, reach for your gun, and so on. Each of your characters has his or her own set of values and resulting emotions.

We humans relate to one another largely through our emotions; captured movement should ideally also expose emotion. Many years ago at the Disney Studios in Hollywood, Disney's drawing instructor, Don Graham, gave a famous lecture about

the importance of animating force (impulse) rather than form (movement). In general, 3D leans heavily toward the animation of form, and mocap tilts even further in that direction. One of the biggest problems in games is that the character movement often appears mental. A character decides to move from here to there rather than doing so in response to an emotion. One reason it can look like that is because mocap is capturing movement rather than acting.

Acting theorist Artonin Artaud famously pointed out that actors are "athletes of the heart." Movement that is motivated by thought instead of emotion will inform your players, but it will not move them emotionally.

Animation vs. Live Acting

That old saying about how "animators are actors" is not strictly correct. In preparation for this article, I went looking for current wisdom about mocap. I came across a man — no names, please — who evidently owns a mocap studio. He explained in his sales pitch how he often uses the animators at the client company as performers because they have a good understanding of the character and because "animators are actors, too!"

I've heard this "animators are actors" thing a thousand times, and I get the drift. I understand that an animator gets into the skin of the character and, in that sense, can be thought of as an actor. However, acting as perceived by the rest of the world is something that happens in the present moment.

Animation and live acting are two very different skills. If I kiss you on the cheek, you'll have an emotional response, pro or con. Actual actors have to deal with that reaction in the present moment. Animators don't have a present moment. They have to create the illusion of a present moment. The performers in your mocap session are acting in the present moment, but the animators that lay the last mile of the performance pipe are not. This is another reason why there is a natural tension between mocap and animators.

Scenes Begin in the Middle

Let's say a character enters from the left, crosses to the center of the room, glances this way and that, and exits on the right. Simple, right? You can direct that easily, right? Okay, let me ask you if you considered the following questions:

Where did the character come from when he entered? What happened before the sequence? Whatever happened prior to the sequence will dramatically affect the movement and dynamic of the sequence being captured.

When the character looks this way and that, will you have the actor actually see something, or will you have him pretend to be looking at something? Even if the character's face is not being captured, the body will move differently if your performer actually looks at something. Have him, for example, look at a shelf on the back wall of the studio to his left and then look at the bath-



Trained actors approximate natural movement better than other physical performers, such as athletes.

room door in the far right corner. Acting is doing. The more you can have the character actually do something rather than pretend to do it, the better it will look.

When the character exits, where is he going? You may not care because it is only necessary to get him out of the room, but in acting, the purpose of movement is destination. The character enters for a reason and exits for a reason. It is not simple movement.

You can extend this principle for sequences that involve multiple characters. Every single one of them has a “moment before” and a “moment after.” Every single one of them has a context and his or her own set of emotional responses.

Actions Act upon Actions

A character should play an action until something happens to make him play a different action. This principle is a close cousin of scenes beginning in the middle and is particularly important for your secondary characters. Remember in the early-generation games, when the player would enter a

FOR MORE INFORMATION

- Richard Boleslavsky. *Acting, the First Six Lessons*. Theatre Arts Books, 2003.
- Michael Chekhov. *Lessons for the Professional Actor*. Performing Arts Journal Publications, 1985.
- Harold Clurman. *On Directing*. Simon & Schuster, 1997.
- Uta Hagen. *Respect for Acting*. Macmillan, 1979.
- Issac Kerlow. *The Art of 3D Computer Animation and Effects* (revised third edition). John Wiley & Sons, 2003.
- Alberto Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, 2000.
- Frank Thomas and Ollie Johnston. *The Illusion of Life: Disney Animation*. Hyperion, 1995.
- Jeff Young. *Kazan, the Master Director Discusses His Films* (Interviews with Elia Kazan). Newmarket Press, 1999.

room and discover some character in there just rocking back and forth in a hold cycle? You can't get away with that kind of thing any longer. A character who is discovered in a room presumably was doing something before we arrived. Our arrival is what causes him to do something else. To capture the action of such a character, don't have him start from a static place. Come up with something he is already doing before the sequence begins.

Going back to the example of an amorous guy on a date, he may have roaming hands, but if she pushes him away, he's going to have to try another strategy if he wants to achieve his objective. He was playing an action until she did what she did. Then he played a different action.

Creating Atmosphere

A character moves differently in the cold than in heat. He may move differently at 2 a.m. than he will at 10 a.m. Each room or set has its own atmosphere that ideally will be taken into account by your director and mocap performer. A chapel, for example, has a different atmosphere from a party, and both of these have different atmospheres from a poker game or a jail cell. It is very easy to overlook this aspect of performance, but it can make a big difference to the resulting credibility.

The Point of No Return

Looking five or seven years down the road, there is no question that the most successful titles will feature characters with more human complexity and better-motivated movement. Oh sure, we'll still have games that are basically the equivalent of Mister Toad's Wild Ride. The situations in racers and football games and shoot 'em ups won't evolve all that much, and there will always be a market for them. But increasingly the players that prefer more situational games will expect more believable characters, and they will cut you less and less slack if you don't give it to them. Just as there is no going back to rubber hose animation in feature films, there is no going back to the first generation of character games.

Motion capture is here to stay, but right now the game industry is not using it to its full potential. The feature film crowd is doing better. True, feature film artists have an advantage over game artists; they can work on a single scene until utter perfection is achieved. Games need hundreds and thousands of variables depending on gameplay. The financial structure and development process between films and games is also different. But we're looking into the future, remember? Game companies are going to have to come to terms with the ever-higher expectations of players. Ten years from now we will look back on 2003 as quaint. We have already passed the point where whoever is in the office kitchen can put on the suit while somebody else in the office directs. Mocap sessions will increasingly require the skill of specialists that have a seat-of-the-pants understanding of acting principles and theory, plus the input and performance of strong actors. 🎭



Racing Toward A Vision:

Integrating Traditional Visualization Teams into Game Art and Animation

When visual style is one of the distinguishing features of a game, an ad hoc approach is not a sure way to execute a strong visual direction. In the handful of cases where such an approach has worked, usually there is a strong initial vision that helps guide the direction. The typical game development approach, however, involves floundering around in hopes that a direction will magically appear. This naiveté can be the death of polished visual style and is easily remedied. If you know where to look and know how to plan, resources are available that can help jumpstart the creative process and actually save production time.

As the scope of game projects attempt to match the entertainment value of feature film productions, game companies are specializing, adding new, more compartmentalized departments

DI DAVIES | *Di is the visual development manager at Vicarious Visions in Troy, N.Y. She has eight years of traditional animation experience in addition to 11 years of game development experience. She can be reached at di@vvisions.com.*

that mimic film production models, adopting a more disciplined approach to preproduction. This approach is not just necessary for big-budget games but also for mid-budget games. More and more, game companies will look to feature film animation professionals and CG film professionals to help answer the call.

Using *CRASH NITRO KART*, released for the Gamecube, Xbox, Playstation 2, and Game Boy Advance as an example of an approach to visual development, we'll examine the challenges faced and processes used by the visualization teams at Vicarious Visions in Troy, N.Y., and Animation Academy in Burbank, Calif., to bring a vision for a cartoon-style racer into focus.

Goals

In order to have successful art direction in games, developers must decide early on what the visual goals are and how the visual direction can complement the game design and technical goals of the project. There is an assumption that prerendered content has fewer technical constraints than real-time game content. This may be partially true, however in both cases, technology will

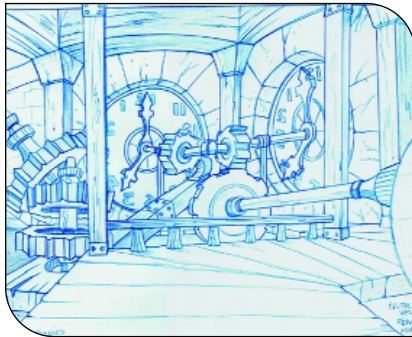


FIGURE 1. Steps showing the progress from concept to implementation for the clock tower track in CRASH NITRO KART.

often determine what can be achieved creatively. Established goals are not only critical to sustaining focus during highly creative brainstorming, they can also prevent time-consuming departures from the intended vision. Full understanding of the license and the approach ties in with the design work. For example, Disney encourages its entire team of artists to avail themselves of the script and storyboards so that the artists are continually thinking about the property and the motivation for their work.

Key questions should be asked before any visual team is sought out to begin visualizing the world, which will avert wasted effort and assist in defining goals: What is the intended effect of the visuals? (Does the overall game want to convey humor or horror?) What are the property's origins? Is it depicted in other media? (For instance, comic book heroes are born in strong graphic worlds and can serve a great starting point for visual direction.) How does the protagonist fit the world to be designed? (Color schemes can be designed to play up the main character or characters, to frame key moments in worlds, or to convey an overall mood.) What is the timeframe? (Time can certainly determine the scope of the visual development and the complexity of the world that can be created.) What are the technical constraints? (Visual direction can sometimes be ineffective if limitations are not considered before designing, taking into account the target platform and game genre.)

Preparing for the Work

First, do your homework. Learn as much as you can about your subject matter and jot down your observations. Design notes go a long way toward building a foundation for art direction. It doesn't matter if you are tackling a new license, an established license, or original IP (intellectual property). Film's visual development giants (Disney, Pixar, Warner Bros., Dreamworks) build their style guides from the notes and observations of the art director. The style guide gives production artists a direct reference for the visual direction.

The more preparation the art director can do to help the pre-production and production artists understand the intent of his or

her art direction, the quicker the implementation aligns with the vision. The pressure to be artistically specialized is greater now, and so is the need for more efficient direction. Encouraging all team artists to adopt more disciplined attitudes toward preparing for the work is critically important for efficient development. It is equally important for project managers and project planners to allow and support for this time.

Another part of preparation is asking lots of questions. Regardless of whether artists are coming from animation, comics, graphics, or illustration, there will be numerous issues that affect their approach to the work. The communication pipeline, interpretation of direction, and other work commitments can affect your schedule, your workflow, and your end result. Make sure you have the opportunity to discuss and document your processes, and be sure you learn about theirs. What is obvious to you may be completely counterintuitive to them.

In our case, three of the five professional film animation artists on our off-site team in Burbank were new to visual development for games, and we had to spend some time getting our bearings. While story and acting are key for a film animation artist, games rely on events and gameplay timing. Camera angles and game engine speed affect the artistic decisions for a game artist working on a cartoon racer, rather than character motivation or scene staging. The Burbank team played the prototype of CRASH NITRO KART not only for creative reference but also to better understand the nature of gameplay progression and timing. Gameplay events were described as opportunities for visual staging in the same way as a movie relies on cues and story progression.

Finally, don't assume too much in your preparation. Defining and proving visual direction early enough to reduce risks or challenges before too much has been invested in actual production is wise, but setting expectations based on assumptions can cause problems. Some common misconceptions can lead to big disappointment and unexpected results:

All of the work that is generated in preproduction is going to be used. Typically, 95 percent of what will be created early in the visual development phase will be scrapped, because the nature of early visual development is more like visual brainstorming; we

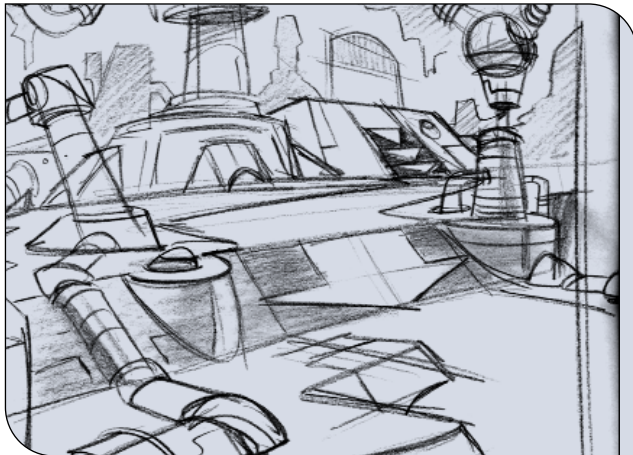


FIGURE 2. Free-form sketch — inspirational sketches from a brainstorming phase can help streamline the visual direction.

called this the “free-form process” on CNK. It was a fast and loose, highly creative process using rough sketches which helped move the visual and design direction along. Without this process, we may have committed to ideas and generated polished conceptual work that would have ultimately amounted to nothing more than a great effort wasted. The smaller percentage of actual conceptual work that is used for the game will set the tone for entire worlds or characters, and is worth that investment.

Any visualization professional can jump in and nail the style.

Without an assigned “keeper of the vision” or art director, it is difficult for artists to interpret the intent of the design and focus on a cohesive vision. If the visual development team is trying to emulate the vision of someone not available to them, then it is essential to develop a style guide before the artists start visual development, which will lend focus to the production team who must implement the visual direction. Don’t overlook the importance of finding visual artists with the skills appropriate to the style needed. If you want a cartoon-style game, hire animation professionals from the film or TV animation industry. If you want a comic book-style game, hire professional comic artists.

The development team will readily implement the visual work.

Sometimes 2D conceptual work is difficult to translate in 3D space, especially where scale and perspective issues arise. Be prepared to test the production implementation of conceptual work early on, before the visual development work is too far along, to determine whether you should modify your approach.

If we hire an off-site team, we just hand them the work and they will deliver exactly what we want. When traditional artists have to adjust their approach to account for unfamiliar technology, it is very important that at least one person be available from the game development team to support the creative process. References are very important to convey ideas and to help give context to the game design.

For CRASH NITRO KART, the non-game artists spent a lot of time looking at the design of a variety of games to help them get a feel for the work they were assigned, in addition to the visual references provided. Recognizing similarities between the design process for games and the design process for film animation was crucial for the non-game artists to frame a context for their work and made their designs stronger and more relevant to the game.

Two Industries, Separated by a Common Language

When professionals in two industries converge to work on a project, terminology can be a challenge to communication. It is important to establish a common language or choose the terminology to be used when working between industries. Our process for world visualization was formed partly from traditional animation processes and partly for custom needs. A detailed description of that process, complete with visual examples, follows. The name of each stage of the process used for game development visualization is followed by a comparative name in parentheses, which is used in animation development.

High concept (script). The high concept is a broad-strokes description of all of the game’s worlds which describes the essence of the world. The high concept tries to capture the progression and tie in elements of the world to jump-start the visual brainstorming process before the level is designed. In essence it serves as the script for visualization.

For example: *In Lava World, large pools of lava bubble by the path’s edge, steam vents intermittently blow steam across the path, and there is a large series of caverns in the distance. A giant broken statue of a Roman god lies half-sunk in the mud.*

Free-form sketch (inspirational sketches). The free-form is not tied to the map overview or to progression specifically but rather attempts to unify the high concept and thumbnail sketches to begin to shape the world and the props within it (Figure 2). The free-form is intended to encourage the creative brainstorming of visuals rather than restrict it by thinking too much about the technical constraints.

Overview map (workbook). The overview map sketch uses the track or level design generated by the design team and brings together all sections of the world in an overhead shot, directly using the shape of the track or AI path, and indicating the placement of landmarks, props, and reuse of elements. This overview helps provide contextual reference for how the player will experience the world.

Screenshot (camera layout). The screenshot can help frame a series of key moments experienced during gameplay. This image can be taken from raw playable levels, may include placeholder objects, and can cut down substantially on the time spent trying to find appropriate concept perspectives.

Moment sketch (layout). The moment sketch (Figure 3) frames key moments in the level. It is the final result of having



FIGURE 3. Early blocked out levels can help frame key moments.

taken the free-form concepts, raw screen captures of the bare track or level, if available, and developing the fully stylized versions of the key moments. The sketch should be clean and ready for color, and in the proper visual style.

Color moment (color key). The color concept (Figure 4) is the color version of the black and white, finalized moment sketch. The color version should attempt to capture color schemes accurately, as well as lighting and effects as accurately as possible. It can also convey texture map ideas. The color concept artist should be aware of the color scheme outlined either in the high concept or by the concept lead.

Gameplay sketch (layout notes). While not a sequential part of the process, the gameplay sketch (Figure 5) is a rough sketch, similar to the free-form or thumbnail, which specifically demonstrates gameplay ideas. Some gameplay sketches can be rough layouts of gameplay progression through a level, others might convey character moves or boss fight designs. Ideally, the gameplay sketch is created early in the design process to help prove design ideas.

Object detail sketch (thumbnail). The object detail sketch is a focused drawing of a particular object or path element that needs specification for the development team to implement. Often the object detail sketch is an afterthought, when a development artist or designer realizes he or she needs to see a mockup in order to fully understand the idea. Sometimes the object detail is a power-up or a hazard, again serving as a visual aid for an idea before implementation is attempted.

Model sheets (same). Character visualization has closer similarities in process between games and traditional art, since it has its own unique set of needs. Developing a character for games still requires the development of a model sheet and pose sheet to assist the animators and modelers, so in this regard it is very much like traditional methodology. One of the key differences in creating a character design for 3D games as opposed to 2D film media is that the character has to be approached more thoroughly. Planning for interaction, collision in the game environment and range of motion affects the character's girth and scale



FIGURE 4. A color version of the moment sketch conveys color schemes and lighting.

as well as the shape and design of cloth, equipment or weapons.

Production flowchart (none). The production flowchart is a simple, one-page visual diagram to help convey the overall process flow. It can be very helpful as a quick reference for the design team to understand the context of their process and to demonstrate when and how to seek out approvals. In traditional animation production, the process has been established for a long time, and most companies invest in internships to train their new artists in their processes. Since game production is still in its infancy, tools such as the flowchart help speed along the comprehension of the process structure.

Working with Off-Site Teams

In situations where there is an off-site team, it is critical for the art director to spend some time early in the process with that team. Visiting the off-site environment makes it easier to identify some of the issues the team is facing; you can learn about the team members, help with communication setup and protocol, and in general be available for the team to communicate expectations and test the process discussed.

For CRASH NITRO KART we scheduled an extended visit to Burbank right at the beginning of the collaboration. This time was hugely beneficial in establishing a smooth communication pipeline, but in retrospect, we probably could have scheduled another visit to offer the off-site team more instruction on how their work was being interpreted and to share work in progress via testing stations.

Scheduling and budgeting for on-site visits is one of the best investments a producer can make and facilitates a much smoother pipeline for your preproduction leads. If you have visual development occurring over several months or even a year, plan to schedule at least three or four visits to facilitate feedback and reviews, especially if it is not possible for your art director to be on-site full-time with the visual development team.

Part of the challenge of successful communication is taking into

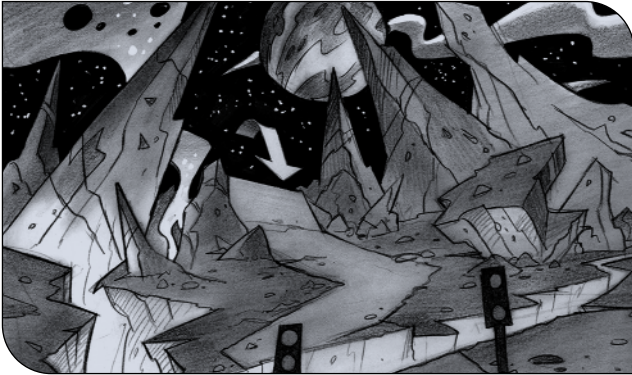


FIGURE 5. The gameplay sketch illustrates design ideas.

account personality and methods of communication. In particular, if you are coordinating a group of visualization contractors, it is critical to adapt your approach for the inevitably different needs of your visual team. For example, if a freelance artist has several commitments to juggle but you really need his or her style of work, you need a very strong communication loop with that individual to ensure the work is not slipping and that other issues are not deterring the artist from getting your work done on time.

Visualization Variables

Something to keep in mind when considering the approach described in this article is that it was appropriate for our style of the game, the time constraints, and the professionals involved. For instance, if you are looking at doing a free-roaming platform game, you do not necessarily have the same technical or design constraints present in developing visuals for a racing game. Your level of detail in off-the-path elements may be greater, you may have a greater number of concepts to develop, and you may need to introduce different steps into the process.

Visual development is usually a much greater investment for a free-roaming character platform game. For example, on a recent high-profile character license just the character development took roughly 12 months for one artist and required about 70 drawings. The time spent developing new characters for CRASH NITRO KART was closer to three months, but this was with already-established main characters. If the license required all-new characters including the main characters, and the design of the characters was completely open, it would take that much more time to develop the characters.

For the world visualization on CRASH NITRO KART, six on-site and off-site artists worked for a concentrated period of nearly five months, then a smaller core group of on-site artists worked for an additional three months. In total, 800 inspirational drawings were created, not including the final concepts which added up to 60 black-and-white moment sketches and 50 color moments (keys) for 13 worlds. Compare that output

FOR MORE INFORMATION

BOOKS

- Don Hahn. *Disney's Animation Magic*. New York: Disney Press, 1996.
- Frank Thomas and Ollie Johnson. *Disney Animation: The Illusion of Life*. New York: Abbeville Press, 1984.
- Mark Cotta Vaz. *The Art of Finding Nemo*. San Francisco: Chronicle Books, 2003.

WEB SITES

- The Animation Academy
www.theanimationacademy.com
Animation school featuring animation industry professional faculty
- Animation Industry Database
www.aiddb.com
Search for production companies, freelancers, and art schools
- Mega Animation
www.meganimation.com
Resource for layout and concept artists
- Games 411
www.games411.com
Resource for layout and concept artists
- Gamasutra.com
www.gamasutra.com/companies
Directory of game art and animation contractors

to world development for a free-roaming character platform game that required one year of preproduction to produce hundreds of inspirational drawings, and then two years of concentrated visual development with two concept artists in parallel with production.

Demands for visual excellence keep rising while budgets stay the same. Achieving solid visual development takes a strong art director, a professional concept team, and the willingness to invest proportionally more in preproduction. This investment proved that it reduces the amount of time that is wasted on art production that needs to be redone due to a lack of clear vision of what the game visually should look like. To date, the game industry has made more use of preproduction for cinematic portions of games than for the in-game visual direction. For in-game visualization, certain types of games such as character platform games or RPGs have historically made more use of preproduction. But as our efforts on CRASH NITRO KART proved, the benefits of using visual development to help focus visual content creation are clear. 🎨

ACKNOWLEDGEMENTS

Thanks to artists Rui Tong and Chongguang Zhang, who created the beautiful color work seen in this article. Thanks also to Karthik Bala for the great creative direction on CNK and for the opportunity. Kudos to the many production artists on CNK who worked hard to implement the vision. Thanks also to Tobi Saulnier and Steve Derrick for proofreading and support.

Combat System Development on BioWare's STAR WARS: KNIGHTS OF THE OLD REPUBLIC

Creating a new type of combat system inside a 60-hour RPG is a daunting task. There's a spectacular amount of complex interactions between the combat system, data structures, and in-game objects that make even visualizing such a system difficult. Yet this is exactly what BioWare did with its newest title, STAR WARS: KNIGHTS OF THE OLD REPUBLIC.

CASEY HUDSON | Casey was producer and project director on STAR WARS: KNIGHTS OF THE OLD REPUBLIC.

RAY MUZYKA | Ray is joint-CEO of BioWare Corp. and was executive producer of STAR WARS: KNIGHTS OF THE OLD REPUBLIC.

JAMES OHLEN | James was the lead designer of STAR WARS: KNIGHTS OF THE OLD REPUBLIC.

GREG ZESCHUK | Greg is joint-CEO of BioWare Corp. and was executive producer of STAR WARS: KNIGHTS OF THE OLD REPUBLIC.

M

any of the design decisions made, and project management methodologies used at BioWare during the development of *STAR WARS: KNIGHTS OF THE OLD REPUBLIC* (KOTOR) were built on the experience of our exceptional staff from our past projects such as *BALDUR'S GATE 1* and *2*, *NEVERWINTER NIGHTS*, and *MDK2*. We set our high goals for the combat system: first, we wanted our system to leverage the fun of BioWare's past RPGs and the experience we gained from them. In addition, the combat system needed to look as exciting as the battles in the *Star Wars* movies. Finally, the combat system and the game in general had to feature an interface that was very accessible; we wanted any player who likes *Star Wars*, likes playing Xbox or PC games, or likes console or PC RPGs, to have fun with the combat techniques in KOTOR.

What Went Right

1. An experienced team. Building a new and unique game system is difficult at the best of times, but the most valuable asset a team can have is people that have tried similar things in the past. Fortunately, many of the senior members of the KOTOR team cut their teeth on the original *BALDUR'S GATE*, where we first developed the processes allowing us to make enormous games with new methods of depicting RPG combat. After developing *BALDUR'S GATE 1* and *2*, *NEVERWINTER NIGHTS*, and expansions to both series, we had trained — and retained — a very strong group skilled in the development of far-reaching game systems, and we had prototyped many different combat models over the years. Add to that experience a few forays into console development (*MDK2* for the Dreamcast and PS2), and we had a team with all the experience required to navigate the pitfalls and rewards for a large, mainstream console RPG.

We leveraged the experience of our team by making use of the personal round-based system (one action per three-second round per character) used in our previous titles. In addition, we had experience in using *Dungeons & Dragons'* D20 system to create a satisfying type of party-based combat, something we wanted to replicate in KOTOR. By basing the system on the personal round-based system of the *Star Wars* D20 rules, we weren't trying to reinvent the wheel. Instead, we built upon what had succeeded in the past, adjusting it for the new goals for this particular game.

We did add highly choreographed combat actions to provide a more action-oriented experience for players. Success in this area would have been much more difficult had we not worked on a similar system for *NEVERWINTER NIGHTS*. It's also worth noting that a number of new, inexperienced people worked on *STAR WARS: KNIGHTS OF THE OLD REPUBLIC*; BioWare's matrix structure mixes new and experienced team members to build an efficient group dynamic.

GAME DATA

PUBLISHER:

LucasArts

NUMBER OF FULL-TIME

DEVELOPERS:

20 at onset, 70 at peak

NUMBER OF EXTERNAL STAFF AND CONTRACTORS:

15 QA, 5 for voice, 8 for sound effects and music

LENGTH OF DEVELOPMENT:

3 years

RELEASE DATE:

July 2003 for Xbox,

Fall 2003 for PC

PLATFORMS:

Xbox and PC

DEVELOPMENT HARDWARE:

Dual 1600+, 512-1024 MB RAM with a GeForce 3 or 4 graphics card

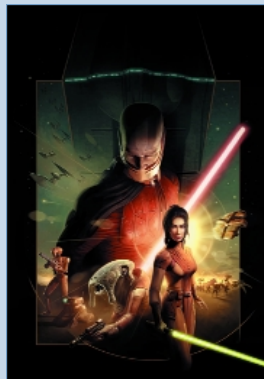
DEVELOPMENT SOFTWARE USED:

Visual Studio.NET, Borland C++

Builder, 3DS Max, Photoshop

NOTABLE TECHNOLOGIES:

BioWare Odyssey Engine



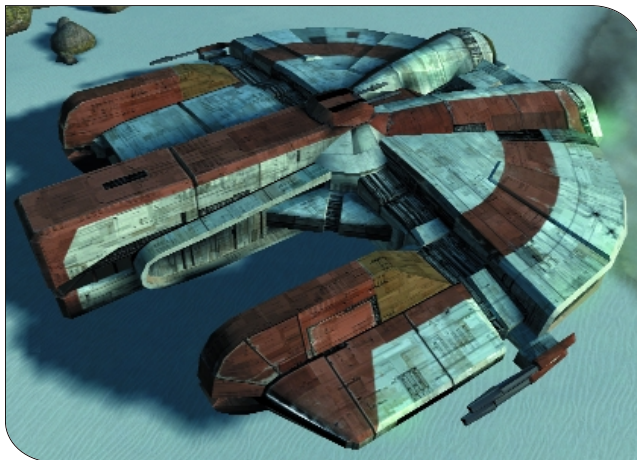
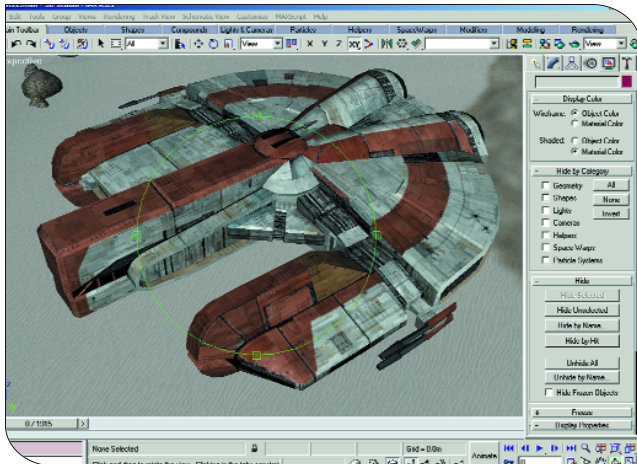
2. Focusing on mainstream players. One of our most important goals was to create a combat system that would be easy to use for a broad cross-section of players. Knowing that LucasArts' revered *Star Wars* brand would give the game widespread attention, we wanted to make sure that all types of players — not just experienced console RPG fans — could jump in and have fun.

To this end, we started with an over-the-shoulder view of the action, giving a cinematic view of the surroundings, rather than the top-down view of more traditional PC RPGs we had created in the past. We also needed the combat to look as action-packed as other mainstream games but be controllable through a simple interface. These goals prevented the system and interface from becoming overly complex and added weight to any feedback from QA or focus tests that certain parts of the combat system should be easier to use. Usability testing with fresh perspectives played a large part in the development of the combat system.

We also adjusted our implementation of the *Star Wars* D20 rules, simplifying the player's interaction with the rules in a videogame setting. Fortunately our publisher, LucasArts, supported us in this endeavor, and we were able to balance our attention to the D20 rules with playability considerations for mainstream fans.

In the end, we were surprised at how well the combat system turned out. Even though combat seemed complex at first glance, most people mastered the system before completing the game. Much of the positive feedback we received about the game was from people who have never played role-playing games before.

3. Combat choreography. While the combat system featured personal rounds for each character, we didn't want it to look like the combatants were taking turns via a strict alternating system (we at BioWare affectionately call this "cave-man combat," where each combatant politely takes turns bonking



A 3DS Max 3.1 textured version for the Ebon Hawk, and a final PC in-game version.

the other on the head with a large club). To capture the excitement of a *Star Wars* blaster or lightsaber battle, the action had to look as fluid as possible.

To achieve this kind of realism, we used a “choreographed animation” system for playing our combat animations. Each character would “lock” with one enemy, attacking for part of the round and defending for the other part of the round. Animations were therefore made in 1.5-second choreographed sequences, where one character did an attacking motion and the defender performed the appropriate countermoves. Essentially, combat actions were predetermined and synchronized between interacting combatants.

This meant that characters could do virtually any combat move that you see in the movies, and each attack would be choreographed with the defender’s animations, enabling characters to do spins, backflips, and rapid lightsaber flurries while appearing to interact physically with other characters. Making multiple animations for each combination of weapon types limited excessive repetition.

Choreographed animation was also efficient for the development process. Since both characters were animated in tandem in 3DS Max, animators were able to see exactly how the animations would look in the game, streamlining development of the system and the adjustment of animations. However, it was still essential to review all combat actions within the context of the game engine, to discover any subtle differences in playback within the engine, on both Xbox and PC.

4. Interface iteration. In the early design stages of *STAR WARS: KNIGHTS OF THE OLD REPUBLIC* we put a lot of thought into the main interface and how the game would be controlled. Since we were taking a new approach in creating a strategic, party-based console RPG, we couldn’t be absolutely confident in the interface design until we experienced it under true game conditions. As we had learned on many of our past projects, our first attempt at an interface simply initiated a process of repeated iteration (including usually two to three major revisions and multiple smaller iterations of each major revision) that lasted right up until the project was complete.

The first version of the interface was crude, lacking the elegance and simplicity that we later realized was needed. A set of combat actions was attached to each button, which would generate a menu of context-specific combat choices. The complexity was confusing to casual players.

After watching the uninitiated struggle with the first interface at E3 2002, the team leads, QA, and other senior members of the company spent time that fall to record all known concerns about the main interface, which prompted a fundamental change in how we let the player interact with the world during combat.

As a result of this feedback, we decided to create more rigid frameworks for player control. Players could interact with enemies only by entering a combat mode. In combat mode, the camera would focus on the hostile target, and non-hostiles were non-selectable. In addition, we felt it was important to depict the range of possible combat actions to players in the form of an always-visible combat action menu (we extended this to non-combat as well to keep it consistent, and called this the “horizontal action menu”). These two factors made the combat much less confusing, and with the addition of action icons instead of drop-down menus, we finally had an interface that was becoming easy and fun to use.

Between spring 2002 and summer 2003 (when the Xbox version of the game was released; the PC version followed later in 2003), we did around 10 smaller iterations to arrive at the final interface. After the most coarse, time-consuming, radical changes (such as the ones just described), we eventually tunneled down to dozens of smaller changes that took only a few hours to make. We incorporated extensive feedback from QA teams at BioWare and LucasArts, plus valuable feedback from usability and focus testers at Microsoft, internal usability tests at BioWare, and multiple rounds of comments from the press, compiled during various press tours and demos. This iterative process gave us confidence that players would be able to control the game easily, but

more importantly, we were starting to have fun playing it ourselves.

5. Using feedback to tune the game.

One of the main problems with many RPGs is game balance: it is easy to make a game that is perceived as either too difficult or too easy, but balance is elusive. Thus, we sought more effective methods of getting fun-factor feedback from our QA department during the course of development. While we'd used feedback from QA to improve balance and fun factor in previous games, we explored new ways of doing it this time.

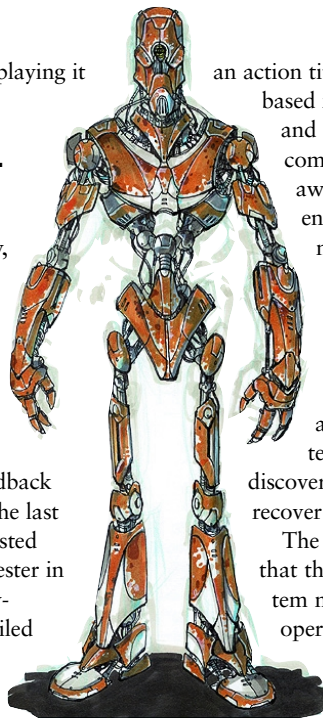
We devised a system to document combat feedback that was used extensively by our testers during the last few months of development. These documents listed every combat encounter in the game, and each tester in the QA department filled it out during their play-through. The document included fields that detailed the general difficulty of each encounter, the amount of credits and experience points they received, and the tactics and Force powers they used to defeat the encounter.

Most of the testers could finish the game in a weeklong session, so at the end of each week we would review all of the data from the QA departments at BioWare and LucasArts. We identified levels that were too easy or too difficult. We looked at the overall treasure allotment and decided whether it needed to be increased or reduced. We also reviewed level progression and then tweaked the experience point system throughout the game. This was an ongoing, repetitive process that lasted through the last few months of development; only through our quality assurance teams' painstaking effort, where they repeatedly documented combat difficulty, were we able to hone the gameplay experience for both casual and hardcore players.

Balance testing was also aided by BioWare's and LucasArts' QA teams' game-playing skills. Because QA was always trying to plow through the game as quickly as possible, they would discover which Force powers and combat tactics gave them the best playing advantage. Once we identified these overpowered culprits, we'd "nerf" them, forcing the testers to use a broader variety of tactics and powers. By the end of the testing cycle we discovered that people used a very large range of tactics, and no one method proved superior.

What Went Wrong

1. Using a round-based system. Demonstrating the first playable version of the game at E3 2002 uncovered one of our biggest hurdles in the combat system's development: the graphics and camera angle made the game look so much like



An early color concept for the HK-47 character.

an action title that people didn't intuitively play it in a turn-based manner. Novice players wanted to mash buttons and twirl the thumbsticks during battle, breaking the combat system and making the game look extremely awkward. The interface's discrete character control enabled players to disrupt their current attack by moving or accidentally selecting a container while attempting to engage the enemy.

We battled this problem to the end of the development cycle, and it required a lot of concerted planning and work to overcome. Carefully controlling the player's access to gameplay functions in and out of combat produced a more intuitive system, but consumer trade shows are not the place to discover such problems in the first place, and we had to recover from some of the poor press at the show.

The fundamental lesson learned, albeit in retrospect, is that the scale of player actions allowed by a combat system must match the scale of actions on which the system operates; if a combat system is based on doing discrete combat actions at any time (like throwing a single punch or firing a single magic spell, as occurs in our upcoming Xbox RPG JADE EMPIRE), you should allow equivalent "per-action," low-level player control. Games like STAR WARS: KNIGHTS OF THE OLD REPUBLIC with higher-level strategic systems should actually restrict players to only controlling higher-level strategic actions.

2. Tutorial handling. Though we were able to find a good balance between strategic control and ease of use, the final combat system still required considerable player adjustment, because at the time no similar system had been implemented. This uniqueness led us to attempt to train the player in all of the basic control and combat systems in the first few areas of the game. Many players found the complexity of the resulting tutorial areas detracted from their initial immersion in the story.

The tutorial condensed too much information into the first hour of the game, when it would probably have been better for us to spread the tutorial elements throughout three or four hours of gameplay. Not only would this have been better for the flow of the story, but also for the player's retention of the information. However, a benefit of condensing the tutorial to a small number of areas was that iteration of the tutorial itself (such as rewriting the tutorial text as interface changes occurred) could occur more frequently and be tested separately, with less risk to the overall project.

Another option would have been to include a separate, stand-alone tutorial. We considered this in the initial design meetings for the game but decided against it, fearing too many players would skip the tutorial. To improve the game's accessibility, we felt console players unfamiliar with PC RPG conventions needed training before getting into the bulk of the game.

3. Lack of interface prototyping. Through an iterative process of implementation, testing, feedback, and redesign, we arrived at a main interface that exceeded our original design goals. That process, however, was extremely lengthy — over one full year — and required a huge amount of manpower, which drove up our development costs. The key missing element from our early paper sketches and graphical mockups was interactivity.

After several iterations of the interface, it became clear that we were simply waiting to see how certain actions would “feel” with the Xbox controller or PC keyboard and mouse, and see how the game would respond. If we had an interactive method for quickly prototyping our ideas, we could have drastically shortened the iteration period. Though we had actually experienced similar issues in all of our past RPGs, KOTOR proved once and for all that more complex RPGs require early hands-on design to develop powerful, transparent, easy-to-use interfaces.

On future RPGs like JADE EMPIRE, we hope to test our interface ideas interactively much earlier in development, so we can work out the “feel” issues well in advance of implementing the interface in the actual game.

4. Sequencing of resources. One of the challenges in working in a multi-project company like BioWare is that sequencing of resources is a constant learning experience. Over the years we have developed a number of techniques to optimize the use of our matrix personnel system (see Ray Muzyka and Greg Zeschuk’s “Managing Multiple Projects,” March 2003), but there is always room for improvement.

Since the combat system in STAR WARS: KNIGHTS OF THE OLD REPUBLIC was different from anything we had encountered in the past, we planned to mitigate risk with lots of early preproduction and prototyping. We initially planned on putting designers on the project early to prototype gameplay into placeholder areas of the game, but things didn’t pan out as expected. Instead, we ended up doing a fair amount of retroactive design while balancing the requirements for story and event scripting.

To solve this problem on future projects, we now pay closer attention to the planning of personnel scheduling based on what we learned in our past projects’ schedules, with the aim of maintaining adequate staffing throughout each project. In some cases this involves outsourcing or hiring additional staff much earlier than we have in the past, anticipating their need later on in the project and allowing sufficient ramp-up time. Hiring the right staff early enough can actually reduce the overall development time and cost of a project.

5. Not using enough feedback to tune the game. Our use of feedback was both a strength and a weakness in the development of KOTOR. While we extensively used team and QA feedback to fine-tune the interface and balance the combat system, there were several areas where we could

have made much better use of feedback.

First, our measurement of the game balance was more of an art than a science. We didn’t have adequate metrics in place early enough (or in some cases, ever) to determine whether the game was doing what we wanted it to as players fought enemies, gained experience, and moved up in level. QA testers didn’t have a formalized way to report combat-balancing statistics until the later stages of testing. We also didn’t build enough automated testing systems into the game engine to track statistics on character experience, abilities, and combat encounters.

We also underestimated the importance of the feedback system in the game. Intended only as a means of letting the player know exactly what happened in a tough battle, the message screen that formed the heart of the feedback system wasn’t given much priority. However, during testing, bugs in the feedback system made it extremely difficult for testers to determine whether the game was performing properly (appearances are often deceiving). It required a large amount of work very close to the end of development to make the feedback system work well enough to be used as a testing tool, presenting a significant risk to the schedule. Planning for a clear and robust in-game feedback system much earlier in the project is now something we now consider essential in our RPGs.

Fruits of the Mainstream

Despite the challenges encountered during development, STAR WARS: KNIGHTS OF THE OLD REPUBLIC ended up being a big success. Microsoft touts the Xbox version of the game as the fastest-selling Xbox title ever released, and the game is also one of the highest-rated RPGs of all time according to Gamerankings. Based on this success, we have high hopes for the PC version, which expands on what we learned in the development of the Xbox version.

In addition, we are actively applying the lessons on what worked well and what didn’t work as well to the three new intellectual properties in development at BioWare, one of which, our recently announced Xbox-exclusive title JADE EMPIRE, will be published by Microsoft.

Credit must be given where it is due: STAR WARS: KNIGHTS OF THE OLD REPUBLIC’s success is entirely due to the hard work of the team at BioWare and also that of our publisher, LucasArts, who fully supported our development efforts and who shares the high quality standards toward which we were all aiming. The team who worked on the game, like the other teams at BioWare, are all hard-working, smart, creative, passionate individuals, and it is an honor for the authors to work with all of them. Our goal at BioWare is to try to exceed the quality of our past games with each new game we make; we felt we accomplished that with STAR WARS: KNIGHTS OF THE OLD REPUBLIC, and we will continue to strive toward this goal in the future. ✍️

Suppose that you're the owner of a game development studio and that the publisher with whom you are negotiating your next title insists on a nasty little clause that forbids you from making another game for anyone else, not only while the two of you are under contract, but for years afterwards, thus effectively putting you out of business the minute your relationship sours. You'd never sign such an egregiously unfair piece of tripe, right?

Then why should a no-compete clause in an employment contract be treated any differently?

Last September, Ubi Soft Montreal filed for an injunction to prevent a handful of former employees from joining the city's

No-Competes: Bad for Business

new EA studio, claiming that their contracts barred them from working for any other game company, anywhere in North America, for an entire year.

And I thought serfdom had been abolished. Silly me.

No-Competes Serve Nobody's Interests

All legal merits of the Ubi Soft case aside, the entire practice of no-compete agreements between employers and employees is bad for individual developers, not much better for companies, and terrible for the industry as a whole.

As an individual, you are signing away your right to earn a living in the future. That's a bad bargain, even in a tight job market. Some enlightened jurisdictions may refuse to enforce an overly aggressive contract (such as California, which has a statutory prohibition against enforcing no-competes except under extreme circumstances), but getting out of it may require an ugly, expensive court battle. And should the judge decide against you, you may suddenly find yourself unemployable, at least in regards to making games: not a pretty sight, especially with two kids and a mortgage on your hands.

If you're a studio manager, the no-compete is one of the least effective human resource strategies imaginable. If you're hiring, it will offend candidates, driving the best of them away and

stranding you with employees who can't attract attention from your more liberal competitors. Disgruntled staff, on the other hand, will stick around longer than they should, because they can't legally go elsewhere. Predicting the effect on morale and performance, especially at crunch time, is left as an exercise for the reader. Even good employees who would be perfectly happy otherwise may falter in an environment driven by fear — successful companies behave like teams and families, not harems.

And finally, as an industry, can we really afford to kick our best people to the curb? Let's face it: if Danny Developer, who has made games all his life and never really considered doing anything else, is forced to spend a year or two away under threat of legal action, there's a pretty good chance that he'll find a non-game job that pays better and demands fewer all-nighters. Once he's used to the money and to seeing his wife more often than if they were both on death row, he might not come back.

Meanwhile, another studio hires newbies and misses milestones.

It's Your Job to Keep Your Staff Happy

Bottom line: a no-compete is not the way to protect a company. Ditch the attorney and create an environment in

continued on page 55

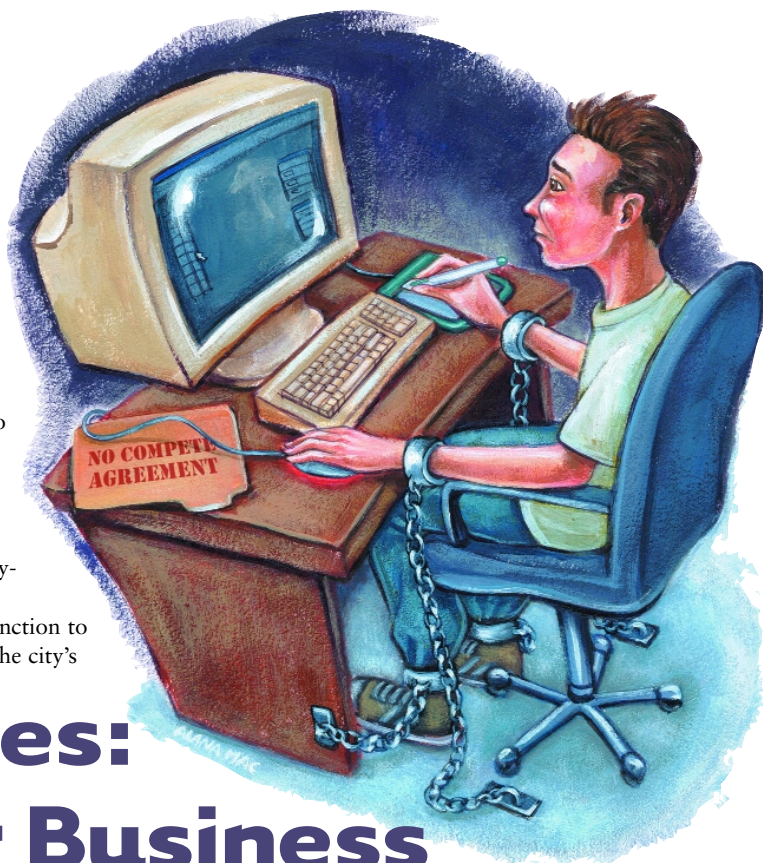


Illustration by Alana Machniki

continued from page 56

which your employees are happy. For game developers, this doesn't take more than creating interesting projects and being treated with common decency. If half of your team quits on the same day, it's your fault, not theirs, and there isn't a coercive contract in the world that could fix the problem.

But what if you treat your employees like your firstborns and they still want to leave anyway?

Let them. If they were crucial to your team's success, trying to enforce a no-compete is not going to win them back. In fact, it might trigger a larger exodus. If they weren't crucial, you are wasting your time and your money by blocking the door. Hire someone else.


Quid pro Quo

Employment is voluntary. You pay me X amount of money for Y amount of work. When I'm done, you've received what you've paid for, and probably a lot more, given this industry's propensity for unpaid overtime. Employers aren't entitled to anything more.

You don't want me to work for your evil competitors? Fine, pay me to sit at home and write a novel. This way, whatever

trade secrets I know are safe, which is the only thing you have a right to care about. And it's probably going to cost you a lot less than dragging me to court.

Do Something

If you're concerned about abusive employment practices in the game industry, the IGDA's Quality of Life Committee will be releasing a white paper at GDC 2004 covering many issues, including no-competes. They will also be hosting roundtables, which will help to gather input from the community and orient its endeavors for 2004 and 2005. You can also voice your experiences and ideas related to quality-of-life issues on our web forum, found at www.igda.org/qol. 

FRANÇOIS DOMINIC LARAMÉE | *A 12-year veteran of the industry, freelance designer and writer Laramée is the chair of the IGDA's Quality of Life Committee. He is also editor and principal author of Game Design Perspectives and Secrets of the Game Business (Charles River Media) and a regular speaker at the Game Developers Conference. You can contact him at francoislaramee@videotron.ca.*
