

# gd

GAME DEVELOPER MAGAZINE

FEBRUARY 2003





# GAME PLAN

LETTER FROM THE EDITOR

## Where's the Party?

**N**ow is not the best time in the world to be a third-party game developer. Whereas in the past the natural rhythms of game industry cycles and fortunes mirrored big-bang-big-crunch cycles of consolidation and start-ups, the game industry is now large enough that a long-range forecast of continued consolidation seems more likely.

Going first-party will provide only short-term security for some developers if the next year sees a major shakeout among publishers, as analysts are beginning to predict will happen. Second parties barely exist as such anymore, most of the prominent ones having been fully absorbed or cut loose by their publishers in this console cycle.

Is the industry headed for a shakeout in 2003? If there's one thing to learn from games, it's that very few match-ups don't end up with winners and losers. Consolidation has been a consistent feature of the maturing of most other entertainment media, and currently the odds don't seem good that we will see as many game publishers at the end of 2003 as there are today. Those that go down will take their first-party studios with them, leaving some developers in unfamiliar hands through mergers and acquisitions, and leaving others out in the cold entirely. Developers and publishers in all three major markets — Japan, North America, and Europe — continue to face unique but significant challenges in their respective economies.

The hardware makers are holding — and exerting — an increasing amount of power, making tycoons out of winners and paupers out of losers. On the developer end of the food chain, budgets are being clamped down and plum projects seem to be harder to come by, even for

those studios for whom they used to come relatively easy. If you now work for a small studio and the name printed in the corner of your paychecks hasn't changed to that of a larger company in the past couple of years, you could be in for a bumpy ride.

**New this month.** This issue features the debut of a new editorial property, *Game Developer Mobile*. So much information is swirling around the nascent industry of games for mobile devices that we've settled on a newsletter format, with reporting and analysis from Ben Calica, a veteran game industry writer and analyst. Our goal is to cut through the hype with an independent perspective, while providing the most relevant, up-to-date information and trends in a context that keeps developers' issues and concerns at the forefront. Look for this new feature bi-monthly, and be sure to let us know what kinds of news and information you'd like to see in future editions by e-mailing us at [editors@gdmag.com](mailto:editors@gdmag.com).

The next 12 months will be crucial for mobile game developers. In order to carve out a successful role in this market, game developers must provide their clients real value in the form of well-executed games, while asserting their strategic value in a market where most business deals remain ad hoc and few workable standard models have emerged. Handset manufacturers and mobile operators seem convincingly committed to supporting a market for games; let's hope they give developers enough leverage and incentive to make good ones.

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

### Publisher

Jennifer Pahlka [jpahlka@cmp.com](mailto:jpahlka@cmp.com)

### EDITORIAL

#### Editor-In-Chief

Jennifer Olsen [jolsen@cmp.com](mailto:jolsen@cmp.com)

#### Managing Editor

Everard Strong [estrong@cmp.com](mailto:estrong@cmp.com)

#### Production Editor

Olga Zundel [ozundel@cmp.com](mailto:ozundel@cmp.com)

#### Product Review Editor

Daniel Huebner [dan@gamasutra.com](mailto:dan@gamasutra.com)

#### Art Director

Audrey Welch [awelch@cmp.com](mailto:awelch@cmp.com)

#### Editor-At-Large

Chris Hecker [checker@d6.com](mailto:checker@d6.com)

#### Contributing Editors

Jonathan Blow [jon@number-none.com](mailto:jon@number-none.com)

Hayden Duvall [bayden@confounding-factor.com](mailto:bayden@confounding-factor.com)

Noah Falstein [noah@theinspiracy.com](mailto:noah@theinspiracy.com)

#### Advisory Board

Hal Barwood LucasArts  
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 Independent

#### ADVERTISING SALES

##### Director of Sales/Associate Publisher

Michele Sweeney e: [msweeney@cmp.com](mailto:msweeney@cmp.com) t: 415.947.6217

##### Senior Account Manager, Eastern Region & Europe

Afton Thatcher e: [athatcher@cmp.com](mailto:athatcher@cmp.com) t: 828.350.9392

##### Account Manager, Northern California & Southeast

Susan Kirby e: [skirby@cmp.com](mailto:skirby@cmp.com) t: 415.947.6226

##### Account Manager, Recruitment

Raelene Maiben e: [rmaiben@cmp.com](mailto:rmaiben@cmp.com) t: 415.947.6225

##### Account Manager, Western Region & Asia

Craig Perreault e: [cperreault@cmp.com](mailto:cperreault@cmp.com) t: 415.947.6223

##### Account Representative

Aaron Murawski e: [amurawski@cmp.com](mailto:amurawski@cmp.com) t: 415.947.6227

#### ADVERTISING PRODUCTION

**Vice President, Manufacturing** Bill Amstutz

**Advertising Production Coordinator** Kevin Chanel

**Reprints** Cindy Zauss t: 909.698.1780

#### GAMA NETWORK MARKETING

**Director of Marketing** Greg Kerwin

**Senior MarCom Manager** Jennifer McLean

**Marketing Coordinator** Scott Lyon

#### CIRCULATION



Game Developer is BPA approved

**Group Circulation Director** Catherine Flynn

**Circulation Manager** Ron Escobar

**Circulation Assistant** Ian Hay

**Newsstand Analyst** Pam Santoro

#### SUBSCRIPTION SERVICES

**For information, order questions, and address changes**

t: 800.250.2429 or 847.647.5928 f: 847.647.5972

e: [gamedeveloper@balldata.com](mailto:gamedeveloper@balldata.com)

#### INTERNATIONAL LICENSING INFORMATION

**Mario Salinas**

t: 650.513.4234 f: 650.513.4482 e: [msalinas@cmp.com](mailto:msalinas@cmp.com)

#### CMP MEDIA MANAGEMENT

**President & CEO** Gary Marshall

**Executive Vice President & CFO** John Day

**Chief Operating Officer** Steve Weitzner

**Chief Information Officer** Mike Mikos

**President, Technology Solutions Group** Robert Faletta

**President, Healthcare Group** Vicki Masseria

**President, Electronics Group** Jeff Patterson

**President, Specialized Technologies Group** Regina Starr Ridley

**Senior Vice President, Global Sales & Marketing** Bill Howard

**Senior Vice President, HR & Communications** Leah Landro

**Vice President & General Counsel** Sandra Grayson

**Vice President, Creative Technologies** Philip Chapnick



United Business Media

## GamaNetwork



# INDUSTRY WATCH

KEEPING AN EYE ON THE GAME BIZ | *everard strong*

**No more Mr. Nice Box.** Microsoft has no plans to ease up on its efforts to carve out a position in the \$10 billion videogame market, rather than cutting its losses and exiting from the venture. Industry analysts expect Microsoft to spend more than \$2 billion in the next five years on the Xbox. Microsoft's Xbox Live broadband service has seen early success, with a reported 150,000 kits sold in the first week of release.

**Metis brands Acclaim.** Marc Metis has joined Acclaim Entertainment as its new senior vice president of brand. In his position, he will manage Acclaim's franchises and new brands.

**No rust in Activision deal with Stainless Steel.** Known for the PC strategy title *EMPIRE EARTH*, Stainless Steel Studios has signed an exclusive multi-title, multi-year agreement with Activision. Their first title will be another historical RTS game.

**NewKidCo posts loss.** Due to the delayed release of two titles — *LITTLE LEAGUE BASEBALL 2002* (GBA) and *TOM & JERRY WAR OF THE WHISKERS* (PS2) — NewKidCo announced an operating loss of \$2.4 million for Q3 2002, compared



*EMPIRE EARTH* developers Stainless Steel Studios have signed a multi-title deal with Activision.

to a \$607,000 operating loss for the same period in 2001.

**PC game sale revenue up for 2002, but unit volume down.** NPDTechworld revealed that PC game revenues were up 1.2 percent — from \$945 million for the first 10 months in 2001 to \$956 for that same time period in 2002 — but the number of units sold fell by 6.2 percent, from 44.4 million in 2001 to 41.6 million for those same 10 months in 2002. The research also pointed out that the average selling price of PC games rose from \$32 in 2001 to \$37 in 2002, but it did not say if this price increase caused the lower volume of unit sales or if there were other,

outside factors, such as increased competition from the console industry or other broader economic trends.

**Take-Two takes its own Angel.** Take Two has acquired Angel Studios, developers of *MIDNIGHT CLUB* and *SMUGGLER'S RUN*, for an aggregate \$28 million in cash and around 235,000 shares of restricted common stock. Angel Studios will now be known as Rockstar San Diego.

*Send all industry and product news to [news@gdmag.com](mailto:news@gdmag.com).*



## UPCOMING EVENTS CALENDAR

### GAME DEVELOPERS CONFERENCE

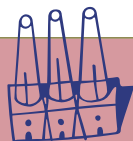
SAN JOSE CONVENTION CENTER  
San Jose, Calif.  
March 4-8, 2003  
Cost: \$150-\$1,975  
[www.gdconf.com](http://www.gdconf.com)

### GDC MOBILE

SAN JOSE CONVENTION CENTER  
San Jose, Calif.  
March 4-5, 2003  
Cost: \$895  
[www.gdcmobile.com](http://www.gdcmobile.com)

### MILIA (NEW DATES)

PALAIS DES FESTIVALS  
Cannes, France  
March 26-28, 2003  
Cost: variable  
[www.milia.com](http://www.milia.com)



## THE TOOLBOX

DEVELOPMENT SOFTWARE, HARDWARE,  
AND OTHER STUFF

**Calculate game royalties and payouts.** PLX Systems introduced a new royalty accounting and licensing system for large game publishers. The product, PLXware Right Track Suite (PLXware/RTS), can be used to process royalties for authors, artists, composers, producers, and other rights-holders. [www.plxsystems.com](http://www.plxsystems.com)

**New tree creation software.** IDV recently released SpeedTreeRT, a tree creation tool for game and visual simulation development. The program delivers low-polygon, realistic trees, with

adjustable wind effects, seamless LOD transitions, and a library of over 90 trees from 30 core species.

[www.idvinc.com](http://www.idvinc.com)

**New Intel C++ compiler.** Intel's newest C++ compiler, version 7.0, integrates with Microsoft Visual Studio, and the Linux version provides GNU compatibility to C++. The compiler also include an auto-parallelization option that automatically looks for opportunities in applications to create multiple execution threads.

[www.intel.com](http://www.intel.com)



## New Worlds, New Battlefields: Massively Multiplayer Online Game Middleware

by mitch ferguson and michael ballbach

**C**reating a massively multiplayer online game is game development's equivalent of a moon shot. It's expensive, technically difficult, and can take years to complete, and yet everyone wants to give it a try.

The technology required is not easy for newcomers to develop. Sensing the need for software that handles a variety of functions, including billing, patching, support, administration, client messaging, stable servers, data persistence, and server clusters, several companies are developing middleware libraries and products to ease these hurdles.

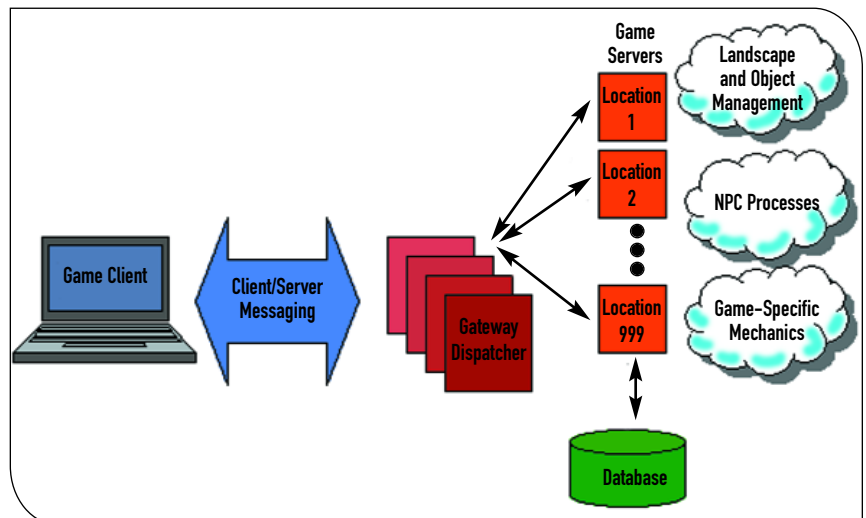
Butterfly.net and Zona's Terazona are two products that provide nearly complete MMOG solutions. Both supply the skeleton of an MMOG, leaving only the skin of a graphics engine, a billing system, and the guts of the actual game mechanics to be defined.

Taken at the highest level, Butterfly.net and Terazona use the same solution strategy. The details are different enough that the environment and personality of the developer will clearly suggest one over the other. Their highest-level strategy is not appropriate for all future MMOGs, however, and whether you would use Butterfly.net or Terazona depends on if your game fits into their mold.

### Common Parts of the High-Level Design

#### Messaging to dispatcher or gateway.

Both systems have a simple API to connect, validate an account, get available game characters, and connect with a



A simplified version of Butterfly.net and Terazona architecture.

particular character. This uses a client-server messaging system, which is easily extensible, reliable, and simple to integrate into a game client.

**Game servers and their landscape.** The developer divides up their world into fixed sections, with the size of each section dependent on how much action is expected to occur there. The actor/object positioning and landscape collision for each section is simulated on a single server. There can be multiple sections on a single server, but there cannot be multiple servers for a single section. These section

servers are the final destination of the game client messages. The boundaries between these world locations are implemented in different ways for Butterfly.net and Terazona, but both make information about things on the other sides of these boundaries available to the game client.

These servers are divided into two separate halves: the generic landscape functionality and the game-specific mechanics. The developer writes the game-specific mechanics to take care of the simple rules such as trading, grouping, and targeting. In addition, the devel-

**MITCH FERGUSON** | *Mitch was a lead engineer on Maxis's THE SIMS ONLINE and Jaleco's LOST CONTINENTS, and now works for a MMO startup in San Francisco called Perpetual Entertainment.*

**MICHAEL BALLBACH** | *Mike has been a server engineer at VR1/ Jaleco for six years, was the lead server engineer on Jaleco's LOST CONTINENTS, and now works for Perpetual Entertainment.*

oper must write a separate process that simulates monsters.

**Game-specific mechanics.** Messages from the client are forwarded to the game-specific functionality in case special physics or movement rules are in play. Butterfly.net has you write this code in Python, which is interpreted by the built-in interpreter. Terazona asks that you link in a shared library to receive these callbacks. Either way, this rule-checking code is used to validate game state and client messages.

**NPCs and monsters.** When it comes to monster AI, both systems pass the buck. Terazona has NPC servers connect as a privileged game client. Butterfly.net has a separate NPC server for every section. In both cases the developer is in charge of how these separate servers are designed and coded. There is an API for receiving callbacks and sending messages to the landscape and the game clients.

**Persistence.** The player data structure (or parts of it) is automatically persisted by both systems. Butterfly.net requires

that you define all the state variables that are needed (also in the database), and it persists the player's data in the clear when it can. Terazona saves out the player data as a block into the database. This saves you from having to define a table, but it also prevents you from performing any simple queries on the player data.

**Games that work with these assumptions.** While both systems claim to be very extensible, in fact both vendors are open to deals that include the source. It should then be possible to shoehorn something unique into either of these frameworks, though this could offset the original reason for using one of these middleware solutions in the first place.

Most of the last generation of massive multiplayer RPGs would have been able to use either of these systems. Their landscapes are stable, the population density of various areas is easily predicted, and the NPC load isn't outrageous. Both systems allow the tuning of the performance characteristics to allow for a faster, twitch style of game.

However, many of the coming generation of MMOGs are starting to break the old mold. THE SIMS ONLINE and TABULA RASA both will have a very dynamic system of landscapes and server processes coming in and out of existence. Also, if someone decides to try a game with a random landscape system or an often modified and persisted landscape, they might have difficulty.

## The Details

**Terazona.** Terazona server components are implemented almost entirely in Java. A lot of components make up the server suite, including the authentication server; these include the administration component; the NPC servers; the GSS, (or Game State Servers); and others. Intercomponent communication is realized by an implementation of the Java Message-oriented middleware standard provided by ICE Technology.

The Game State Servers require the developer to implement a set of functions (their GSAPI, Game State API) in C/C++ and compile them into a DLL or shared

## Selecting Networking Middleware

When considering networking middleware, things aren't as straightforward as you may think. Networking is not just a numbers game; you can't look at fill rate and rendering features as you might graphics engines. Networking is more closely tied with players rather than the game's features. It's what allows your players to play against each other, and unlike the game, your players aren't such a known or predictable quantity.

— Crosbie Fitch

### SELECTED PROVIDERS:

#### TERAZONA

See page 12 for contact details

#### BUTTERFLY.NET

See page 12 for contact details

#### OPEN SKIES

Cybernet Systems  
Ann Arbor, Mich.  
[www.openskies.net](http://www.openskies.net)

#### VAST

Rebel Arts  
Calabasas, Calif.  
[info@rebelarts.com](mailto:info@rebelarts.com)  
[www.rebelarts.com](http://www.rebelarts.com)

#### TERRAPLAY

Terraplay AB  
Solna, Sweden  
+46 (8) 764 91 00  
[www.terraplay.com](http://www.terraplay.com)

#### DEX

Horizon, a Glimpse of Tomorrow  
Monrovia, Calif.  
(626) 446-8925  
[www.horizongot.com](http://www.horizongot.com)

#### NET Z AND ETERNA

Quazal  
Montreal, Quebec, Canada  
(514) 395-4646  
[www.quazal.com](http://www.quazal.com)

#### SIMINTERNET

MÄK Technologies  
Cambridge, Mass.  
(617) 876-8085  
[www.mak.com](http://www.mak.com)

#### LITHTECH DISCOVERY SYSTEM

Lithtech  
Kirkland, Wash.  
(425) 739-1659  
[www.lithtech.com](http://www.lithtech.com)

#### TURBINE ENGINE

Turbine Entertainment Software  
Westwood, Mass.  
(781) 407-4000  
[www.turbinegames.com](http://www.turbinegames.com)

#### NEL

Nevrax  
London, U.K.  
[business@nevrax.com](mailto:business@nevrax.com)  
[www.nevrax.com](http://www.nevrax.com)

#### TWISTED

Twisted Matrix Labs  
[www.twistedmatrix.com](http://www.twistedmatrix.com)



object that the server will load and use for game-state validation. The interface to the GSAPI proved clumsy; they give you a header file full of C function definitions and tell you to implement them. This is O.K., but it would be preferable to export a structure of function pointers, or implement a class with pure virtual members and export a factory.

In Terazona, sections are called regions, but they operate much like we described the Game State servers. One interesting thing to note is the fail-over support inherent to their design. Since entities (players or NPCs, generally) are ghosted to the necessary neighbor regions, when a region server fails, another server will instantiate the region and attempt to get entity state from servers that had it ghosted. If that fails, it will go back to the database. GSS servers can be brought up and down while the game is running, without restarting other components. Terazona does not have its own implementation of a landscape and does not do landscape validation for you.

Terazona uses the “NPC server as a trusted client” model. The downside of this model is that sometimes the most complicated logic exists in the NPC servers, and their scalability is left up to

the developer. The client-server protocol provided is based on TCP, which may be unacceptable for many applications, though Zona has promised a reliable UDP implementation. Because they only ask a small up-front fee in exchange for part of the subscription revenue, Zona’s business model is attractive to new MMOG developers. This has the extra benefit of increasing the support and library improvements developers will receive.

**Butterfly.net.** Butterfly.net is more of a total solution than Terazona, which is not necessarily a good thing. In fact they are proponents of a utility model that allows you to use their server clusters. As traffic goes up and down, those servers are shared with other products that are also using the utility model. Although very cost efficient, this prospect could be frightening to even the most experienced MMOG live teams. They don’t prevent you from putting the parts together yourself on your own systems.

The rules management system that the server uses to validate game state uses functions written in Python. This could greatly simplify game authoring and also helps with thread safety. The Python interpreter is built into the game server, so performance shouldn’t be a problem.


The Butterfly.net system can have a separate NPC server for every landscape section (which they call locales). If you have an NPC-heavy game, this should help with the distribution of CPU load. (The NPC server is not written in Python.) However, human intervention is required should these NPC servers or locale servers crash. Butterfly.net has no automated server management that can restart processes or bring up another from a pool of servers.

The suggested operating system is Linux with a DB2, Oracle, or Postgres database, but if you were required to use a different operating system or database, Butterfly.net claims that much of the system can be ported. Tools for porting terrain models from Maya or 3DS Max into Butterfly.net’s Landscape quad-tree are also provided.

**What would we prefer?** Although total solutions have their place, MMOGs

might be too young of a genre to support this strategy. A more open-ended toolbox approach is proving useful for graphics engines such as Criterion’s Renderware, a useful model to emulate on the server-side as well. This would mean a large collection of smaller functional libraries, separated by a well-documented API. Libraries of functions for messaging, persistence, cluster control, and terrain quad-trees could be combined into whatever system a developer might need.

In addition, neither system supports the concept of an automated generic pool of servers. This would entail every server being able to perform any function and would create a high amount of reliability with very little human intervention.

**Last word.** It’s very exciting that the massively multiplayer market is growing enough to support the creation of these types of solutions. MMOGs are fast becoming too complex and risky to develop for very much innovation to occur, and we are all happier if systems like Terazona and Butterfly.net help alleviate that. Still, neither has a well-known MMOG that has used their technology yet. Even so, both are strong contenders in this young market. 

Visit [www.gamasutra.com](http://www.gamasutra.com) for a longer version of this review.

## TERAZONA

### STATS

#### ZONA

Mountain View, Calif.  
(650) 964-1133  
[www.zona.net](http://www.zona.net)

#### PRICE

Depends on project and studio size.

### PROS

1. Portability.
2. Automated game state.
3. Flexible environment model (space, water, ground).

### CONS

1. Persistent data is stored in blob format.
2. Reliable UDP still in development.
3. Regions are statically defined.

## BUTTERFLY.NET

### STATS

#### BUTTERFLY.NET

Martinsburg, W. V.  
(304) 260-9520  
[www.butterfly.net](http://www.butterfly.net)

#### PRICE

Depends on project and studio size.

### PROS

1. Utility model.
2. Almost a complete solution.
3. Persistent data is stored in the clear.

### CONS

1. Utility model.
2. Lack of automated disaster recovery.
3. Locales are statically defined.

## Matt Toschlog's Descent into Outrage

Since 1986, Matt Toschlog has enjoyed a varied life within the game development industry. Starting his journey at Sublogic, Matt continued through Looking Glass Technologies, to Parallax Software (which he co-founded), and is now Studio Director for Outrage Games, a company he splintered off from Parallax and was president of until its recent acquisition by THQ. One of his most notable contributions to videogame history so far includes the DESCENT series.

With what looks to be an ongoing trend of big publishers and media conglomerates acquiring once-independent developers, we asked Matt — among other things — to give us an insight on what things are like after the Big Change.

**Game Developer.** Do you think the current trend of third-party developers being acquired by publishers bodes well or badly for the industry?

**Matt Toschlog.** It's a good thing in that it allows third-party developers to keep making games. As games and team sizes and budgets have grown, it's become increasingly hard for independent developers to survive. The stakes are too high, and publishers are hesitant to spend the money required to make a great game if they don't have more control over the process.

**GD.** How do you see your work changing from the days of Parallax to the present? Has working for THQ changed your work behavior in any way?

**MT.** The biggest change is that I don't program anymore. When we wrote DESCENT I was one of three principal programmers. Now I'm a full-time manager. Another major change is that our projects are much bigger. At the end of DESCENT we had eight people working on the game; our current project, ALTER ECHO, now has about 25. That means a lot more management and coordination, and a much less intimate environment.

At the time we were acquired, we'd been working on ALTER ECHO for THQ for about a year. When our other publisher ran into trouble, it was natural for THQ to step in. We were in the middle of our E3 crunch when the acquisition happened, and we just kept on working. We had to shift some people around when our other project was cancelled, but for most people, Outrage isn't much different now than before THQ came in.

**GD.** Do you think the creative process of making a game is affected by these acquisitions? Is there more paperwork, strategy, and playability planning going into the making of a game in a big company than in smaller ones, where things can be a lot more flex-



Outrage's Matt Toschlog.

ible as the game design and programming unfolds?

**MT.** I think the creative process is affected by the size of the games. If a company is going to spend 5 or 10 or 20 million dollars on a game, they're going to want to keep close tabs on where the money is going. You've got to be a lot more careful when you're spending that kind of money.

**GD.** What role do you see middleware playing in the future of game development?

**MT.** Middleware is starting to be a real factor. These days, most consumers don't care as much about technology. Developers realize that and want to focus on content; using stable development systems make that easier. We've always written our own engines here, and will probably continue to do so, but we're placing a high premium on writing reusable code.

**GD.** You served as chairman for the International Game Developer's Association (IGDA) for a couple of years. How do you see the organization's goals compared to when you came on board?


**MT.** I think the goals of the IGDA have remained pretty constant — to be the voice of the professional developer and to provide support to the people making games. What's changed — steadily, though sometimes slowly — is our ability to fulfill those roles. We've been building membership, establishing programs, making contacts, and generally creating a foundation. Many of us who have been with the IGDA for the last few years got involved because we had a vision of where the association could be down the road. We're still looking toward the future, but we've already come a long way.

**GD.** Do you miss those garage days when you first started creating DESCENT? Do you ever wish for a return to those simpler days?

**MT.** Oh, yeah. It was great being hungry and lean like that. Sometimes we talk about doing a GBA game to try and re-create that atmosphere.

**GD.** Looking back, what elements made the DESCENT series the success that it was? Do you think those elements came together by choice or chance?

**MT.** DESCENT offered the classic game mechanic of "shoot all the bad guys" in a way that no one had seen before. At the time, 3D games were still somewhat new, and DESCENT was the most 3D of them all. Piloting a ship through those crazy tunnels provided a challenge that rewarded those who could master it.

I think success depends a lot of chance. One thing I've learned is that no matter how hard you work and how well you plan, there's still a huge amount of luck that determines whether your game is fun or if the customers will care. If we really understood how to make hit games, then every game would be a hit. 

# Clustering

**F**or the past two months I've been discussing profiling. I started out by decrying the lack of good profiling tools for games; I talked about the difference between a "batch profiler," like VTune or the CodeWarrior profiler, and an "interactive profiler," like the ones we usually build into our games; I then proceeded to make a big wish list of features that a good interactive profiler might provide.

Chief among those features was the ability to abstract the program's timing measurements into higher-level "behaviors" and to do some analysis regarding the frequency and consistency of these behaviors. Modern games don't exhibit one consistent pattern of resource usage. At one time they might be fill-rate limited by the graphics card, at another they'll be slowed down by an abundance of pathfinding and physics. A tool that helps us visualize these patterns would be valuable.

## Finding a Clustering Method

Let's look at Table 1, which I introduced in last month's column ("Interactive Profiling, Part 2: Behavior Analysis," January 2003). Here we see three behaviors: behavior A is rendering-heavy, B is physics-heavy, and C is AI-heavy. Imagine the numbers in the table came from a profiling run of a game in progress; as each frame is processed, we get a new set of timing numbers. Now suppose that the next frame is measured, producing the results: **rendering** 72%, **physics\_sim** 17%, **ai\_pathfind** 11%. Those numbers are very much like behavior A. Even though the numbers do not match exactly, we would expect the profiling tool to classify this frame as exhibiting behavior A.

That's not hard to implement when you have a predefined list of behaviors like Table 1. It's harder when you start from scratch with no preconceived notions of program behavior. In

TABLE 1. The CPU usage profiles for three different behaviors A, B, and C, measuring time spent in three different program sections: "rendering," "physics\_sim," and "ai\_pathfind."

Behavior:	A	B	C
rendering	75%	25%	25%
physics_sim	15%	65%	15%
ai_pathfind	10%	10%	60%

general, we are faced with a clustering problem: given a bunch of vectors containing timing data, we want to arrange the vectors into groups, where the centroid of each group can be used as a summary.

Clustering is a common task in computer science and engineering applications, and many algorithms have been developed, for example so-called *k*-means clustering. Here's how *k*-means works: First, randomly partition your input into a set of initial clusters. Find the centroid of each cluster. These clusters obviously won't be perfect yet, so to fix that problem, you iterate over each input vector and reassign it to the cluster whose centroid is nearest. Now recompute the centroids. Repeat this reassignment process until the results converge, and you perform an entire pass without changing anything.

The name "*k*-means" indicates that there are some number of clusters (*k* of them), and that we are representing each cluster by its center (by the mean of the input values).

Unfortunately, *k*-means and most other clustering methods are batch algorithms. They require you to have all the vectors available at once and to access those vectors randomly. Since I want an interactive profiler, which can classify behaviors on the fly during a single program run, these algorithms are unsuitable.

## Nonbatch Clustering

There's a special term for clustering algorithms that can handle streaming input: they are called "online clustering algorithms" For example, there's an "online *k*-means."

Online *k*-means works like this: Start with *k* cluster centers arbitrarily distributed through space. For each input vector you want to classify, find the closest cluster point, classify the vector into that cluster, and move the cluster point closer to that vector. (See "Radial Basis Function Learning" in For More Information.) Online *k*-means really is an incremental version of batch *k*-means, but making the algorithm incremental is a big change, so the explanations of the two versions sound fairly different.

How do you decide how many clusters are necessary to represent your data stream accurately? There are some heuristics that involve removing cluster points when you have too many (some



**JONATHAN BLOW** | Jonathan Blow ([jon@number-none.com](mailto:jon@number-none.com)) really likes the way bedrooms in Korea have heated floors. You just walk around in your socks and it's all warm.



of them are lying idle, never attracting input) and adding cluster points when you have too few (when the radius of space attributed to a particular cluster point is too large). Unfortunately, these heuristics all require some kind of a priori knowledge of the data: you need to know how many samples is the minimum for considering a point to represent a valid cluster, or how spatially large a good cluster is likely to be. In other words, you need a good guess about the duration and scale of your data.

But we don't want to have to guess at that stuff in order to get a good profile. Really, we want the computer to figure out appropriate values from context. And those values may need to change. If we are running our game for five minutes and see some frames that spend 70 percent rendering, 75 percent rendering, or 80 percent rendering, we probably want to draw distinctions between these and call them three different behaviors. But then if we walk into another room in the game and we start seeing a lot of behaviors like B and C from Table 1, then perhaps all of the previous frames should be categorized as A.

Clearly, some kind of hierarchical clustering is necessary. It's probably best to let the user decide at which detail level to view the abstractions; we should maintain a tree of behaviors from which the user selects a particular view. Unfortunately, the typical methods, such as hierarchical agglomerative clustering, tend to be data-intensive batch processes.

## Visualization

There's one other problem we need to solve for effective visualization of our profiling data. In a full-sized game, there may be a few hundred different subareas of the program for which we get profiling data. In clustering terms, we're clustering data points that have a few hundred dimensions each. Suppose we manage to do this, and we get a result with 15 different behavior clusters. Those clusters are still points in a high-dimensional space. We can list them, like in Table 1, but we'd like better methods of visualization. Maybe we want to project these clusters somehow onto a 2D chart and have the clusters that are most alike placed near each other on that chart. Maybe the two dimensions of the chart can even mean something, related to the two biggest ways in which the behaviors tend to vary.

There are some well-known methods for projecting multidimensional data onto a 2-plane. These methods tend to fall into two categories: ad hoc, producing poor results; and complicated, slow, and scary. There's another alternative, though: a hybrid kind of algorithm that clusters and projects simultaneously, and does a good job of both.

## The Self-Organizing Map

In this month's sample code, I used this hybrid algorithm, the Kohonen self-organizing map (SOM) to perform the classification of CPU behavior. The SOM is an array of points in the input space; all the points are initialized to random values. For each incoming frame of profiling data, we find the closest point in the

SOM. We then move that closest-match point toward the position represented by the incoming frame, just like online *k*-means. But the SOM does something extra: in addition to giving the clusters positions in the input space, it treats them as having positions in another space also. The clusters have fixed positions along a regular grid, in what I will call "neighbor space." Whenever you move a cluster center toward an input in input space, you also look up its neighbors in neighbor space and move those neighbors through input space in the same direction, thought not by as much.

This neighbor space is usually implemented just by storing the cluster data in a 2D array and by iterating across nearby cells in that array. You can think of this as applying a "move filter" to the array, where the filter kernel is high in the center but tapers off with an increasing radius.

After many such iterations, the clusters in input space will converge, just like *k*-means. The size of the move filter shrinks over time as the clusters become settled, until it becomes a point. The "self-organizing" part of SOM represents the idea that, because we applied that move filter in neighborhood space, we were constantly influencing neighboring array cells to take on similar values. So after we are done processing our inputs, we not only get cluster centers, but the clusters are arranged in the 2D grid according to similarity. We can generate a 2D display where similar clusters are drawn near each other, which can help us understand the data.

This is a quick-and-dirty description of the SOM, with some simplifications. For reasons of isotropy, like we saw when looking at networking ("Transmitting Vectors," July 2002), it's usually better to use a hexagonal tiling than a rectangular grid. Also, the SOM grid can consist of any number of dimensions, but two dimensions is the most common choice, since a map with more dimensions is harder to visualize. Finally, initialization of the SOM is best done using some attributes of the training data, since a random spread in a high-dimensional space is likely to produce points that are not near anything in your data set. In this case, the only way array elements are used is when they get dragged into use by a neighbor, so it takes a while before the SOM is working at reasonable effectiveness.

For some references that explain self-organizing maps in greater detail, see For More Information.

## Adapting the SOM to Real-Time Tasks

The SOM is a batch algorithm for two reasons. For one, to produce good results, the algorithm wants a randomly ordered sampling of the data. Imagine first presenting all your samples of behavior A, then presenting an equal number of samples of behavior B, which happens to land in a cluster neighboring A, so the movement filter for B wipes out most of A — this process will not produce desirable results.

The second reason the SOM is a batch algorithm is that it wants to have a global notion of time. Early in training, the

movement filter is very large; over time it shrinks, until toward the end of training it's almost a point. But with live input data, we don't know how long we will be running, and we may wish to run indefinitely. So I made some modifications to the SOM algorithm to adapt it for real-time data. These modifications are somewhat questionable, since I have not analyzed them deeply, and I can already think of some ways they may be problematic. But to a first approximation they work O.K.

## Shrinking the Filter

Rather than using a global time, I introduced the idea of “local convergence,” which is a value between 0 and 1. The filter size centered around any given array cell depends on that cell's local convergence: a convergence of 0 means a big filter, 1 means a small filter.

Whenever a point in the SOM is generally staying in the same place, its convergence value increases. If it starts getting yanked someplace else, its convergence value decreases. I use the IIR filters I described in “Toward Better Scripting, Part 1” (October 2002) to track the trend of a point's motion. If many training steps confirm the positions of some region of the map, that region will no longer feel the need to upset its neighbors. But if an area of the map keeps being upset, its “local time” will go backward and it will jostle its neighbors in a bid to create more space for the competing values.

This “hardening of the map” does to some extent alleviate the “B wipes out A” problem but does not solve it. To solve it, I would implement a hierarchical series of maps, with maps that change quickly, feeding data into maps that change more slowly. I haven't had time to try this yet, but I plan to soon.

One further problem is that there's no way for the SOM to grow. To achieve reasonable results, you need to construct it with enough initial nodes. But I believe hierarchical maps would solve this problem as well.

## Initialization

I wanted a good way to initialize the SOM. As I mentioned earlier, random initialization is not so good. The way it's usually done for batch data is to perform principal component analysis on all the input, find the two non-axis-aligned dimensions along which it has the greatest spread, and initialize the SOM array entries to cover that spread evenly. In other words, you find the covariance of the data with respect to the mean, then pick the two longest axes of the multidimensional ellipsoid. This process is just an  $n$ -dimensional version of what I demonstrated in “My Friend, The Covariance Body” (September 2002).

Since I don't want to wait for the whole batch of input data, I collect 10 frames' worth of “warm-up points” before the SOM kicks into training mode. Then, because I didn't want to write  $n$ -dimensional matrix decomposition code, I approximate the spread as follows: First I pick a random warm-up point, then I find the warm-up point that is furthest from it. I find the vector

between them, and that is the longest axis of the spread. Then I remove that dimension from the warm-up data by subtracting each point's projection along that axis. Then I choose another random point, find the warm-up point furthest from that one, and that is the second-longest axis of the spread. I then initialize the SOM entries to points evenly spaced across the plane defined by these two axes, centered on the mean. If the axes are degenerate within some numerical threshold, then I seed the SOM with random offsets from the mean.

## Sample Code

This month's sample code (available at [www.gdmag.com](http://www.gdmag.com)) is an updated version of the toy game world that I described in my December column (“Interactive Profiling, Part 1”). In addition to AI-heavy and entity-render-heavy behavior modes, there's now a sun in the sky. When you look toward the sun, ridiculous amounts of lens flare are drawn, bogging the system down.

As you move around the game world, you'll cause various behaviors that are mixtures of AI, entity, and lens flare. Whenever `mouseLook` is enabled so that you can move around, the game system takes each frame's profiling data and classifies it using an SOM. When `mouseLook` is disabled and the SOM is paused, you can move the mouse pointer over the SOM's display and look at the behaviors it has detected.

There are a number of visualizations enabled, which are explained in the README. These visualizations aren't as intuitive as I'd like, but after a bit of exposure you get used to them and can infer interesting relationships about the clusters by looking at colored diagrams.

## Future Work

So far this work has been pretty experimental, but I think it shows some promise. I am going to try some hierarchical versions of the SOM and online  $k$ -means and will report on their effectiveness in a future column. In the meantime, next month I'll go back to a more conventional topic: creating better-looking graphics. 🐉

### FOR MORE INFORMATION

Kohonen, Teuvo. “The Self-Organizing Map (SOM)”

[www.cis.hut.fi/projects/somtoolbox/theory/somalgorithm.shtml](http://www.cis.hut.fi/projects/somtoolbox/theory/somalgorithm.shtml)

Moore, Andrew W. “K-Means and Hierarchical Clustering”

[www.cs.cmu.edu/~awm/tutorials/kmeans09.pdf](http://www.cs.cmu.edu/~awm/tutorials/kmeans09.pdf)

Palmeri, Paolo. “Radial Basis Function Learning”

<http://miles.cnuce.cnr.it/~palmeri/datam/articles/okrnc.ps.gz>

Vesanto, Juha, Johan Kimberg, and others. “Enhancing SOM-based Data Visualization” [www.cis.hut.fi/projects/ide/publications/html/iizuka98](http://www.cis.hut.fi/projects/ide/publications/html/iizuka98)

# What's Wrong with Our Games? Part 2

I have heard many times, expressed in a variety of ways, that the success of a game lies firmly with its execution rather than in its design. It is a fact that many ideas and their initial design documents put forward what seems to be an interesting and exciting concept, but by the time the game finds itself on the shelf, it has turned into nothing more than a mediocre space-filler that will find a home in the bargain bins within the month.

I'm sure that any experienced developer reading my column could write me a list as long as my arm of the reasons why a finished game differs substantially from the original design spec and why, in general terms, evolution of ideas through the development process is always a good thing. Implementation, rather than initial ideas, usually accounts for the eventual quality of a game when it is released.

There is no area more relevant to the discussion of implementation than that of a game's visuals. The irrelevant language of a pitch document that promises a game "set in richly detailed fantasy world of epic proportions" has to trans-

late meaningfully onto the screen to have any chance of being even close to the truth. If you run such a document through a truth filter, many such sentences would read, "Set in a routinely realized fantasy cliché, made to appear large by repeating the same geometry over 100 times."

Continuing the theme I began in last month's column ("What's Wrong with Our Games? Part 1," January 2003), in which I covered design, animation, and

character creation, this month I'll look at some additional areas in game visuals that often lose something in the translation from page to screen.

## Lighting

One indication of how videogame technology is improving is the quality of the lighting available to designers. Unfortunately, many games still view lighting as number 138 on their list of things to pay close attention to, and as a result, the energy developers spend creating a world of detail and quality is wasted for many games. They pay little attention to lighting, or decide not to invest much time in their light-related technology. Nasty lighting can hammer even a well-constructed scene to death, whereas high-quality lighting will bring a scene to



**HAYDEN DUVALL** | Hayden started work in 1987, creating airbrushed artwork for the games industry. Over the next eight years, Hayden continued as a freelance artist and lectured in psychology at Perth College in Scotland. Hayden now lives in Bristol, England, with his wife, Leah, and their four children, where he is lead artist at *Confounding Factor*.

life. Attempting to re-create the real-world behavior of light in a 3D scene has led to a whole variety of lighting solutions over the years.

Initially, severe color-palette restrictions made the idea of lighting irrelevant, but as we moved through the 8-bit and then the 16-bit era, the simulation of lighting effects allowed our early 3D worlds to appear more solid. Once true 3D began to appear, it wasn't long before basic light sourcing was a common feature. Today, artists can apply just about any high-end lighting solution to some degree in a gaming context, and the question of which solution to use then depends on cost versus benefit and which platform you're dealing with.

Up until recently, the most advanced of these solutions have been exceptionally slow. Unless you had access to a render farm the size of Alaska, the massive amount of numbers being crunched manifested themselves in single-frame render times that were measured in weeks rather than minutes.

Artists now have the tools and the technology to do more than place colored point lights around a scene that they hope will do the job. The approximation of real light distribution is now more accessible within the limitations of a game world. Designers can now use texture baking to incorporate the subtleties of radiosity, global illumination, and lighting from high-dynamic-range images.

Texture baking has been around for a while, but the newest iteration of many high-end renderers now brings together the most advanced lighting solutions with the ability to render this light information to textures. Whether the light information is rendered into a separate layer to be added on top of more traditional texturing or added directly to the textures themselves, the results one can achieve are far more realistic.

The downside is that texture baking can destroy texture memory in no time flat. The price to pay for what is effectively light distribution information, customized for each surface, is that the cost to memory can be massive in scenes with a huge amount of surfaces. Condensing multiple light maps into more economic single textures, applying low-resolution light maps on top of regular texturing, and careful level design that takes these limitations into account can

*Heavy-handed application of just about anything, regardless of how nice it is, can ultimately do more harm than good.*

all make texture baking more usable. While the cost is high, the quality of the end result can be worth it.

## Textures

The easiest targets for game artists to pick on when examining the work of their peers are most probably the textures and the texturing. Many games now have extremely large worlds that can be explored in fairly close detail, and as a result, it is hard to find a game that has faultless texture work across its entirety.

That said, there are a few texture problems that appear time and again in games:

**Bad mapping.** Bad mapping can take many forms, some worse than others, and usually indicates either a lack of time and resources, or some level of apathy on the part of the offending artist.

Faced with a significantly large amount of geometry to texture in a given time, it is understandable that inconsistencies should appear. Texture streaking

from applying a planar map across faces that are nearly perpendicular to the UV plane is one example of careless or hurried mapping that looks very ugly.

Seams that show painfully obvious joins between mapped sections can also stand out, especially if the geometry is large, like a rock face for example. With many complex objects, especially organic shapes, it's all but impossible to map the entire surface with a continuous, seam-free texture, so some jiggery-pokery is often needed to reduce the appearance of problems.

Many designers overlook the option of placing joins in the texturing where they will be the least visible. This may seem like an obvious solution, but it's important to remember that "the least visible" means from the player's perspective, not that of the artist, so developers must play through the sections in question carefully to assess the least visible areas for these scenes.

Another crude but nonetheless worthwhile problem-masking technique is using textures that naturally minimize the appearance of seams. Such textures are those that are even in terms of detail, color, and contrast. Ideally, a system that obscures unavoidable seams, such as multitexturing, works better, as long as the artist has the time and the inclination to take advantage of this method.

**Specular and bump mapping.** Specular and bump mapping technologies are not new, but they are only now becoming practical in any real sense for mainstream gaming. Still, heavy-handed application of just about anything, regardless of how nice it is, can ultimately do more harm than good. Both bump mapping and specular effects have the ability to introduce additional and dynamically responsive material properties to surfaces within your game. But if they are applied with no concept of restraint, they will end up being intru-

sive gimmicks that turn a game's visuals into an unsuccessful mess.

Coupled with dynamic light and shadowing, bump mapping is a particularly useful way to achieve the appearance of fine surface details without using actual geometry. But many designers are drawn into making just about everything bumpy, or at least into adding bumps where they have no real need to be. Likewise, specular maps can seduce artists into finding ways to incorporate specular effects on every object they create, not a good idea in most circumstances.

Bottom line: Restraint is key, as the player's attention is drawn to the areas where the techniques I've just mentioned are used judiciously, rather than to excess.

**Straight from scan.** Any game that is attempting a certain level of realism is likely to use photographic sources in its texture generation. Problems arise when the initial scan or digital photo finds its way into the game without any significant processing. The end result of unprocessed images is usually textures that feel out of place or flat. As most textures need to be balanced for brightness, contrast, and other properties to make them consistent with the game world, neglecting to process images will usually make them stand out unnecessarily.

**Scaling gone wrong.** Another common problem arises when textures are completely out of proportion with the environment that they are supposed to be part of. This problem is especially common in larger areas, where a texture is likely to be repeated many times, and artists sometimes cut down the number of repeats by scaling a texture up.

The problem becomes most apparent when a character (or player, if it's third-person) is in close proximity to the offending texture, where it is possible to see that the heads of the nails in the wooden planks are actually the size of watermelons. This kind of scaling mismatch is easy enough to avoid, but it requires the artist concerned to pay close attention once again to the player's experience, as opposed to building and

texturing geometry in isolation.

**Environment mapping.** The use of an environment map, particularly in conjunction with other layers of texturing in order to give the appearance that a surface is reflective, has become one of the most abused effects of recent times. It is a toy that needs to be put back into its box and only brought out when it genuinely fits.

Especially bad offenders combine environment maps with such effects as worn and damaged metal surfaces, so that a material that should be relatively unreflective ends up looking like it has been coated with a very thick layer of varnish and buffed to a mirrorlike shine. Here again, restraint is key.

## Cutscenes

**T**he art of creating the perfect cutscene is a large enough topic to warrant several columns, but when asking others what they think most often goes wrong with cutscenes in a game, the almost universal answer is that they are too long.

The debate about what the role of the cutscene actually should be in a game has several contrasting viewpoints. Still, no matter how cutscenes are presented, whether in-engine, 100 percent prerendered, partially interactive, or whatever, the bottom line should always be: do they improve the player's experience, or do they just get in the way of enjoying the game?

This can be a difficult question to answer, as my idea of a great cutscene could be the opposite of what you are looking for. However, because lots of people feel that long cutscenes intrude on the experience of playing a game, it would be a good idea to look at ways of reducing cutscene length where possible without compromising story or atmosphere. There is a limit to how much information a player can retain from a cutscene, so if it does contain things they need to remember, then dragging it on will make that more difficult.

## Geometry

**N**ow that polygon counts are less of an issue than they were a few years ago, such problems as having to use five-sided cylinders or being forced to resort to triangular cross-sections everywhere don't exert the pressure they once did. Building environments that look good isn't suddenly a piece of cake just because we can be a little freer with the triangles, but the problems at present tend to focus more on such elements as texturing, lighting, or general design issues.

In this respect, it isn't the quality of the geometry that is a problem; it often has more to do with the variety of geometry used, particularly when it comes to interior spaces. It's very easy to decide that the corridors of a space station would, in reality, all be pretty much identical, so why not just take a section of geometry that you are happy with and duplicate it as many times as necessary? Likewise, the sewers under the castle are all made from the same brick, so why build 10 variations of the same thing?

Repetition is a great time saver, and when you use it with care it can be virtually invisible to the player. On the other hand, without enough unique geometry or some other form of landmarking, a player can get lost quite easily. Navigating an area may seem simple to the artist who created it, but often the modeler's perspective is completely different from the player's restricted viewpoint. For the player, figuring out how an area is laid out, even somewhere that is relatively uncomplicated, may prove to be quite a task.

Aside from the possibility of becoming disoriented, a player can also bore quickly if the visuals become overly monotonous. The artist has failed in his or her task if players switch off because they are tired of wandering around locations that all seem to be essentially the same thing.

In the end, we can't expect our games to be 100 percent perfect in every respect. But in the quest for greatness,

# Music That Won't Give You a Rash: Taking the Irritation out of Repetition

**N**o matter how much you love your favorite piece of music, if you listen to it enough times, the repetition becomes annoying.

In-game music is subject to this kind of repetition, so how do you avoid irritating listeners? Given that repeating elements is a fundamental part of composition, let's look at how the game's audio engine affects the fundamentals.

Repetition in the structure. Music is typically structured in patterns. For example:

A-B-A-B-C-B

denotes a typical rock-song structure, where A is the verse, B is the chorus, and C is the bridge. Notice that B actually takes up half the piece — this is done purposefully so by the end of the piece the listener should be able to hum the chorus.

Even though the point is to hammer home the chorus, there's a reason the song doesn't simply feature the chorus all the way through. The contrast of other musical sections is required to avoid boring the listener. Although there are examples of 20th century music that push the envelope in the realms of contrast and repetition, for the most part a good balance is required.

But what happens to this balance when a game's audio engine loops a piece of music? Our example structure would end up looking like this:

A-B-A-B-C-B — A-B-A-B-C-B — A-B-A-B-C-B — and so on.

In just three loops we've heard B nine times and we're already sick of it. However, we can remedy this situation by breaking some of the traditional rules of composition.

The trick is to write with the structural repetition reduced to the point where the piece needs to be heard many times in



order to "understand" it. An extreme example of such a structure might be:

A-B-C-D-E

A music professor would give such a piece an F, because in theory it would lack coherency. However, when the game engine repeats the piece over and over while you're killing monsters, the player eventually becomes familiar with all five parts. It would take considerably more repetitions than the rock-song example to get to know this piece, which is exactly what we want.

For most composers, it's quite hard to write something completely lacking in repetition. They have a deeply ingrained instinct to use repetition, so it's better to try easing into it rather than giving it up cold turkey. If we took the rock-song structure, wrote an additional D section, and varied one of the A sections to give:

A-B-A'-D-C-B

we'd have a tune that is much more loop-friendly without being completely incoherent.

**Repetition in the melody.** Just as the use of structure needs to be reconsidered in the context of in-game repetition, so does that of melody. A melody or theme is the most commonly remembered element of a piece of music. While the arrangement

and groove may be pleasing, the melody is what the listener actually remembers. In a game, this can be bad — every time that theme pops up, the player notices it.

Themes are built using a structure. John Williams' opening theme for Star Wars uses a single rhythmic motif repeated four times with a variation on the last repetition. It's brilliantly crafted and highly memorable — too memorable, in fact, to be heard many times in succession.

Many of the preceding suggestions regarding the structure of a piece of music can be applied to creating themes. For example, you could try increasing the variety in the motifs that make up the theme. However, we'd like to make the case for not using full themes at all in games. Rather than making a coherent musical sentence using short motifs, try leaving the pieces unglued, peppering your piece with smaller fragments instead. By varying these motifs so that you seldom hear the same one twice, you can achieve musical coherency without hitting the listener over the head repeatedly with a single attention-grabbing melody.

**The final mix.** The result of all this reduced repetition will be music that plays well over the course of playing a game but doesn't stand up very well to a single listen. So if you're presenting music tracks independently of the game, such as on a CD, you may need to create different versions of the pieces that incorporate a more usual level of repetition. We never promised that better game music was going to be easy. 🎧



**SIMON AMARASINGHAM** | *Simon is a composer/sound designer with dSonic ([www.dsonicaudio.com](http://www.dsonicaudio.com)), a company that creates music and audio specifically for the game industry. Kemal Amarasingham and Vincent Stefanelli, who are also founding partners of dSonic, were contributors and consultants for this article. dSonic's past credits include SYSTEM SHOCK 2, ARX FATALIS, and MASTERS OF ORION 3.*

# Good Points Don't Go Down

One of my favorite jobs as a freelance designer/producer is doing game/studio reviews for publishers. Working on a design on my own can be an isolating experience, so I relish the opportunity to get out in the world to visit a development studio and essentially do a tune-up, advising them on business plan, company or game team structure, and tools and techniques to manage their games to completion. But my favorite part of that experience is tinkering with the design, looking for ways to help them improve the game while minimizing production costs and risks.

I've found the growing body of game rules very helpful in this process. I often switch roles from sage to scholar as someone makes a suggestion about the game design, and I think, "Rule!" and jot it down. One such moment came during a meeting at Firetoad Software in Calgary, reminding me of one of the very first rules I learned in the arcade business.

**The Rule: Don't take away points or other hard-won possessions from the player.**

It's often tempting to design a penalty for the player to emphasize failure at a task or to discourage the player from attempting to do something in the game you don't like. But failing and being discouraged just aren't fun. There's always a way to turn it around and reward the player for success, or encourage them to do what you want.

**The Rule's domain.** This rule applies to all games but is particularly important where points are awarded, as in sports games.

**Rules that it trumps.** This rule should trump the temptation to take things away from the player for reasons of realism or to add excitement.

**Rules that it is trumped by.** Make villain and opponents evil to the player, not just



The game CHASE awards points for success rather than taking them away for failure.

the character. There are cases where it can be a good thing to take things away from the player — a villain in a game who is out to destroy the world is well and good, but when he takes away your character's best magic sword, he's really evil! The caveat is that in defeating (or at least challenging) that villain it becomes possible to win back whatever you have lost. This technique has been used effectively in games as far back as the thief in ZORK, or the temporary loss of coins or rings in many platform games, or as recently as Dexter's transformation in JAK & DAXTER.

**Examples and counterexamples.** Let's consider two frequent violations of this rule. First, we have racing games that use timers to deduct from the score when the player is slow to finish, sometimes to the extent that it is impossible — dare I say pointless? — to try to finish the race. Why not, as in the new Xbox game CHASE from I-Imagine, give a bonus when the player finishes early? Or use a countdown bonus timer that counts more slowly with time, so there's

always some bonus, but one that increases geometrically the faster the player finishes. Second, in some shooting games when the player hits a team member or non-combatant, they turn on him in anger, and points are deducted. It's just as easy, and more fun for the player, to award a bonus for *not* hitting them, or even award the player with extra help or gifts as appropriate from gratefully spared civilians.

**Tidbits from the e-mailbox.** Dave Grossman was one of many who responded to the "Godfather Paradox" (November 2002) column. He offers some rules that I'll paraphrase for brevity:

- Have a realistic budget: Don't assume it can be done for less than the original.
- Cut some good stuff: Make room for new innovation by resisting the temptation to rehash all the good bits of the original.
- Pretend it's not a sequel: Challenge yourself to think of it as an original game.

Like a teenager moving out of the house, says Dave, the sequel must find its own path. His credentials include DAY OF THE TENTACLE, an excellent example of his rules.

Finally, Ray Mazza, a graduate student at Carnegie Mellon University, suggests using popular musical themes from original games and building on them in the sequels, and similarly using some recognizable territory as a part of the new, larger world. That reminded me of how nicely DIABLO II did both of those things by including the town of Tristram along with its signature musical theme. 🎵



**NOAH FALSTEIN** | Noah is a 22-year veteran of the game industry. You can find a list of his credits and other information at [www.theinspiracy.com](http://www.theinspiracy.com). If you're an experienced game designer interested in contributing to The 400 Project, please e-mail Noah at [noah@theinspiracy.com](mailto:noah@theinspiracy.com) (include your game design background) for more information about how to submit rules.



**JOHN LALLY** | John is the animation technical director at Insomniac Games. In addition to RATCHET & CLANK, John has animated characters for SPYRO 2 and 3 and for several Squaresoft Games. He is currently working on Insomniac's newest Playstation 2 project and can be contacted at [jpl@insomniacgames.com](mailto:jpl@insomniacgames.com).



---

# Giving Life to RATCHET & CLANK

## Enabling Complex Character Animations by Streamlining Processes

**A**t first, we were thrilled. As character animators, we couldn't have asked for a better project. There were two heroes, dozens of enemies, scores of NPCs, and more than 100 character-driven cutscenes. Enthusiasm and artistic latitude made it all ours for the taking.

But staying true to our shared vision of RATCHET & CLANK meant that our digital actors needed to become more than mere cycling automatons. We regarded each character as an intermediary through which we could reach out to players and draw them deeper into our universe. This meant our characters needed to blend physically into their environments, emotionally into their situations, and expressively into our narrative. It was on these principles that we based both our objectives and our standard of success.

Our team acknowledged that a rift existed between the level of complexity we desired and the time we had scheduled to implement it. In order to surmount this obstacle, we developed several methods for using Maya, our artistic skills, and our time more effectively.

This article will discuss these methods both in terms of their functionality and their implementation. To this end, it will provide technical details on our testing practices, our MEL shortcuts, and our real-time animation procedures. Furthermore, it will explain how each of these methods saved us valuable production time, enabling us to achieve our artistic goals.

### Testing with Prototypes: Why and How

**P**art of achieving our goal of tying our characters closely to their environments and gameplay meant prototyping low-resolution versions of our characters and their respective animations. Like coalmine canaries, we sent proto-models into our new levels to nose out potential animation, programming, and design problems. We relied on prototyping throughout the course of our production as a means of refining a character's move set. This process of refinement was key to winnowing down unworkable ideas before animating a character's high-resolution incarnation.

As a rule, our prototypes emphasized function over style. And although we set the aesthetic threshold low, these previsualization models still needed to be built and animated accurately enough to function as valid test cases. For the animators, this meant that prototype characters needed to jump to their correct heights, attack to their design specifications, and run at their proper speeds.

Generally, we created prototypes using a character design sketch as a guide. These proto-characters were constructed with primitive objects and only roughly resembled their future incarnations, as you can see in Figure 1 (on page 32). Since previsualization models were so simple to construct, every animator could assist in building them, regardless of their modeling experience. Accuracy was required only in the representation

of the character's height, proportions, and posture.

For the most part, our prototypes had extremely simple skeletons: all geometric components were assigned to a single bone with no special deformation. Though such simplicity made for blocky-looking models, in practice our animators had all the flexibility they needed to test out a move set.

Animating our proto-characters was similar to sketching a traditional pencil test. Although animators were given a designer-approved move set, it was understood that animations needed only to be rendered into their roughest forms. One pass was often sufficient, as polish and overlap were unnecessary.

The areas where precision did count were timing, measurement, and interaction with other characters. As they have the greatest direct impact on gameplay, these attributes were considered critical to testing a new character's behavior accurately.

Timing has a major effect on both the readability of an animation and on gameplay. From a distance, a poorly timed idle can look muddy. An attack animation can be too slow to make an enemy a worthy opponent, or too fast to be registered. Emphasis or a lack thereof on just a few frames can make or break any animation, especially within the short cycles of the real-time universe we were creating. We discovered that by testing and fine-tuning our timings in the prototype stage, we could often avoid reworking polished animations on final characters.

Making sure that proto-characters adhered to design measurements was also important. For example, if the design document called for an enemy to attack at a range of 4 meters, animators would ensure that the prototype did exactly this. Designers could then get an accurate idea of whether an enemy traveled at the correct speed, was tuned to the appropriate difficulty, and was scaled appropriately in relation to the main characters.

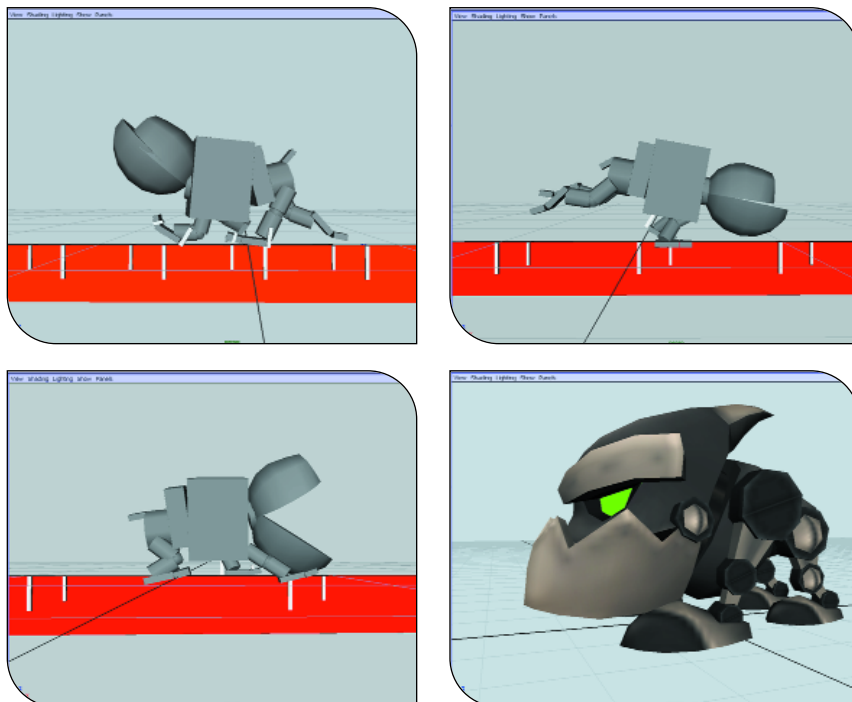
Prototyping also gave us a means of pretesting character behaviors and interactions. Whether it was with Ratchet or Clank, with the environment, or with another character, proto-models provided invaluable early glances at interactive behavior. For artists, programmers, and designers, previsualization served to telegraph character behaviors both in terms of their technical feasibility and their gameplay value.

Ultimately we found that our previsualization process was beneficial not just to animators but to our design and programming staff as well. It gave our programmers a head start on coding gameplay, while designers could test, tune, and ask for changes at a very early stage, allowing room for refinements.

Prototyping saved animators time and energy that otherwise would have been spent painstakingly modifying or redoing final multi-pass animations. It provided a relatively simple means for evaluating character behaviors with respect to their timing, specifications, and interactivity. Moreover, it provided our animators with a practice run, complete with feedback, before moving on to a high-resolution character (Figure 2).

## MEL Shortcuts: Automating Our Setups

**M**aya Embedded Language (MEL) scripts were essential for bridging the gap between the level of complexity we desired and the time we had scheduled to implement it. Through MEL scripts, we were able to streamline setup



FIGURES 1A–C. The Dog Charger prototype was used to pretest the final character's animations, including its walk, run, and attack. FIGURE 2 (bottom left). The final Dog Charger model.

operations, customize animation processes, and level our technological playing field.

Two such scripts (examined later in this article) allowed our team to take advantage of driven key functionality that otherwise would have been too cumbersome to animate or too tedious to rig by hand. Another tool enabled our artists, regardless of technical experience, to fit characters with IK systems automatically.

Most of our bipedal characters had leg setups like the one pictured in Figure 3. As seen in the hierarchy (Figure 4) our legs had standard hip, knee, and ankle joints, a heel joint, and two to three bones in the feet. (For clarity purposes, please note that we referred to our foot bones as “toes.”)

Our IK-rig consisted of three to four RP (Rotate Plane) IK-handles. These connected hip-to-ankle, ankle-to-toe, toe-to-toe and/or toe-to-null. All were configured into a hierarchy (Figure 5) that

specified relationships between the IK-handles, a set of locators, and several NURBS constraint objects.

Though relatively simple, setting this IK-system up by hand for every NPC, enemy, and prototype would have taken more time than we had. Moreover, we knew that this time would be better spent bringing our characters to life.

An actual tools programmer might scoff at the artist-authored MEL script we developed to make our leg chains. In the end, however, our “IK Setup Tool” reduced an hourlong technical chore to a simple task that took seconds. Furthermore, the script did not require setup expertise, and our relatively simple code could be customized and refined entirely from within the art department.

Using the IK Setup Tool (Figure 6) was a three-step process. First, an artist checked their characters' leg joint names against the tool's presets, making any necessary changes. Next, a scale factor for the constraint objects was entered,

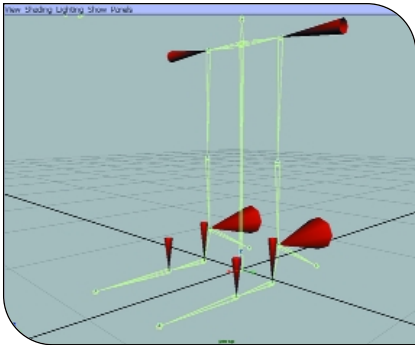


FIGURE 3. This leg setup was used for most bipedal characters, saving tedious hand-setups for IK systems for individual characters.

based loosely on a character's size. The artist then hit nine buttons in sequence. These buttons would auto-attach the IK handles and instantly build the constraint hierarchy.

## Dissecting the IK Setup Tool

MEL is a quirky and often inconsistent language. A good portion of the time we spent developing our IK Setup Tool was used to track down the proper commands for the tasks we needed to execute. Still, we managed to uncover the MEL commands we needed to actuate the core tasks of each of our nine tool buttons.

The first button's purpose was to place IK handles on a character's legs. It read the names of the bones from the top text fields by using the `textFieldGrp` command in its query (-q) mode. These string variables were then passed to the `ikHandle` command, which in turn created the IK handles.

The second button placed NURBS cones on a character's hip, ankle, and toe joints. These cones, created using MEL's `cone` command, were the primary constraint objects an animator would use to manipulate the legs. The `xform` command was used to query (-q) the positions of the leg bones and store them as variables. The `move` command then read these variables and moved the

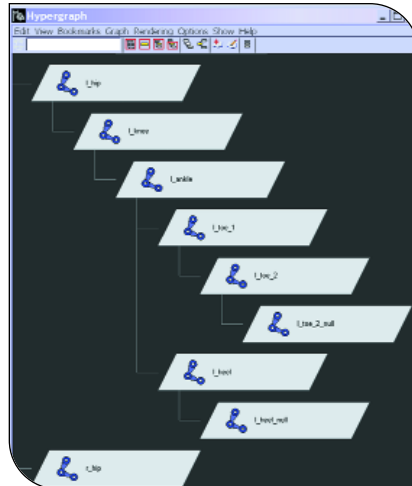


FIGURE 4. Standard hierarchy for a character's leg, as shown in the Hypergraph.



FIGURE 5. The leg constraint hierarchy viewed in the Hypergraph, showing connections between the IK handles, locator set, and NURBS constraint objects.

cones into place. Finally, MEL's `pointConstraint` locked the hip cones to the character's hips.

Pressing the third button called `CreateLocator` to place a pair of locators in the scene. Next, the `group` command grouped the locators to themselves. Then `xform` (-q) queried the positions of the character's knees, and `move` translated the

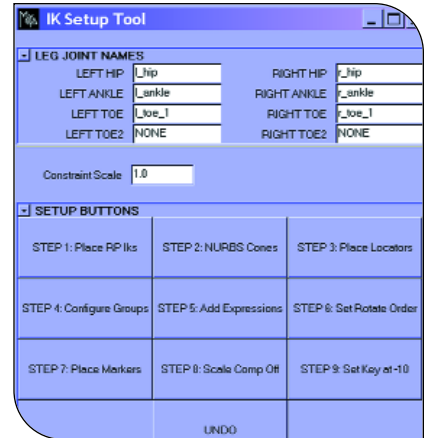


FIGURE 6. The IK Setup Tool streamlined repetitive, error-prone setup procedures and kept customization within the artists' hands.

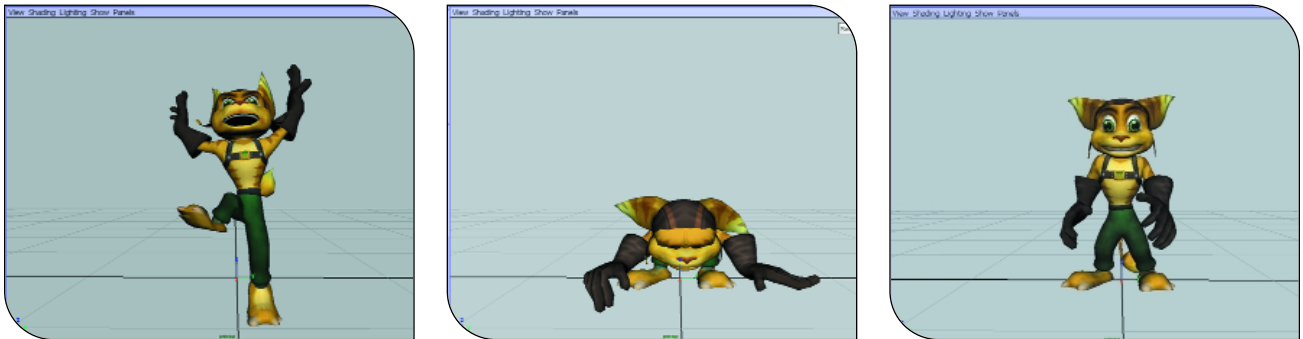
two new parent objects to the knee joints and the locators to positions in front of the knees.

Button number four configured the cones, locators, IK handles, parent groups, and constraints into a standardized hierarchy via the `parent` command. Again, the new groups were translated into place using `move`. New constraint relationships were created between the knee locators and main leg IK handles, and the new constraint hierarchy and the skeleton. These were implemented using the `poleVectorConstraint` and `scaleConstraint` commands, respectively.

Button five added several expressions to the scene, saving us data-entry drudgery. We added expression code for specifying both constraint and skeletal behavior using the `expression` command, allowing us to automate both the creation and the specifications of our setup expressions.

Number six altered the rotate order of the heel and toe NURBS cones from XYZ to YXZ using `setAttr`. We had previously determined that this rotate order produced the most reliable rotations in our quaint Z-up environment.

Buttons seven through nine performed some final housekeeping tasks. Button seven grouped custom rotation guides to



**FIGURES 7A–C.** Joint scaling and translation offered animators direct manipulation of poses and gave characters’ moves extra verve for the modest cost of a low-tech solution.

a character’s spine using the `polyCube` and `parent` commands. Button eight used `setAttr` to ensure Maya’s segment scale compensate was switched off for all of a character’s joints. Finally, button nine keyed a reference frame at `-10` on the character’s skeleton and constraint hierarchies using `setKeyframe`. Listing 1 (page 34) shows some of the MEL procedures we found most useful.

Automating this process with MEL both saved us time and eliminated the steps most prone to human error. Furthermore, by enabling any artist, regardless of their setup experience, to fit a prototype and/or character with a functioning IK system quickly, we alleviated bottlenecks. This conservation of both time and human resources saved energy that could then be devoted to artwork.

## Low-Tech Animation Solutions

The shortcuts and prototypes I’ve described so far shared a common purpose: to help us create better animation more efficiently. Both of these methods accomplished this by either by telegraphing problems or by saving time. Often, however, we would spurn a high-tech solution due to its specificity, inefficiency, and/or complexity. And still at other times, we embraced traditional CG taboos.

We consistently and repeatedly translated and scaled our characters’ bones. True, most of us learned on our grandmothers’ knees never to do that to a CG character. “Use your constraints,” she would say. “Rotate your bones if you must. But avoid scaling them, and don’t ever, *ever* let me catch you in a translation!” We all love our grandmothers, but we found that the tenets of traditional animation called for — nay, demanded — that we defy her.

The reason behind our disobedience was squash and stretch. We found that by scaling our joints, and especially by translating them, we could instill our animations with extra gravity and snap. Major translations often lasted only a couple of frames and, borrowing an idea from Disney, were “more felt than seen.”

Since we had no IK setups to speak of on the spines and arms of our characters, translating the bones in these body parts was quite simple. If needed, we could key the leg IK solvers “off” in order to

### LISTING 1. Some helpful MEL procedures.

```
// A method for querying a bone’s position in world space:
xform -query -worldSpace -translation my_joint_name;

// A method for querying the contents of a text field:
textFieldGrp -query -text my_text_field_name;

// A method for setting a keyframe at frame -10 on a hierarchy:
setKeyframe -time -10 -hierarchy below my_hierarchy_name;

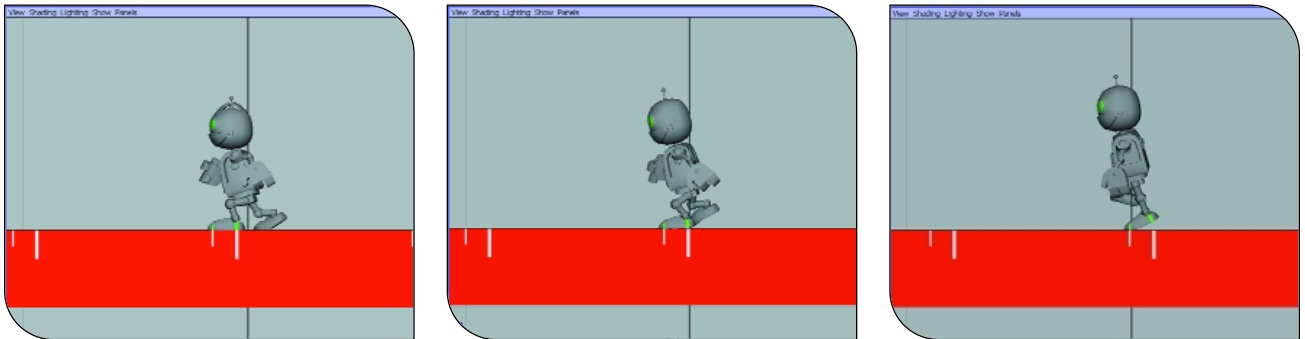
// Basic transformation methods: translation, rotation, and scaling:

// Moves an object to (0,0,5):
move -absolute 0 0 5 my_object_name;

// Rotates an object by 90 degrees on Z,
// relative to its current Rotation:
rotate -relative 0 0 90 my_object_name;

// Scales an object to 3 times its current size:
scale -relative 3 3 3 my_object_name;

// (Note: All flags are listed in their long forms.)
```



FIGURES 8A–C. The Walk Guide helped line up characters' feet on the ground properly every frame to minimize unattractive foot sliding.

manipulate these joints. Translation and scaling were effective across the board and worked wonders on anything from walks to attacks to facial animation (Figures 7a–c).

Requiring no additional setup, these low-tech solutions saved us time. Within limits, this method of animation provided animators with a direct, tactile, and expedient method of sculpting their characters' poses. Although unglamorous, this technique was as effective as any in terms of preserving our resources and improving our animations.

## Walks and the Walk Guide

**A**nother device we used to aid our animation was called the Walk Guide. We used this tool help our characters' feet stick to the ground during walk and run animations. Although foot slippage is commonly forgiven in the world of games, we hoped that by eliminating it we could add an extra dimension of believability to our characters' locomotion.

The Walk Guide was an elongated cube with many smaller cuboids attached to it. The smaller cuboids were identical to the polygonal markers on our characters' ankles and toes, which were grouped to their feet during setup.

By scaling a special parent node, the Guide's small cuboids could be adjusted to match a character's foot size. Scaling the large cuboid allowed an animator to accommodate for the character's stride

length. A set of constraints and locators ensured that as the stride length changed, the preset foot size remained constant.

Since our walk cycles were animated in place, we needed a way in which to simulate forward movement while keeping track of the positions of a character's feet. The solution was to animate the Walk Guide to the speed specified by the designer (2 meters per second, for example). Once the Walk Guide was moving at the proper speed and the small cuboids correctly scaled, an animator could begin working on the character's walk cycle.

The trick to using the Walk Guide to eliminate foot sliding was to keep the character's foot markers lined up with the small cuboids on the Guide. This applied for every frame in which the foot made contact with the ground (Figures 8a–c).

Upon a cycle's completion, a character could be put into a level and moved at its preset speed with little or no foot slippage. Additionally, programmers could scale the playback speed of the cycle relative to the character's velocity and still have the feet stay grounded.

There were several gameplay situations that were not as clean as the test case I just described; however, the Walk Guide did serve to plant our character's feet properly in most of our worlds. Once accustomed to the Guide, we animators found that using it benefited both our schedule and our artwork, as it kept track of the more technical aspects of locomotion for us.

## Making Faces: Artistic Reasons and Technical Details

**W**e knew from the start of developing RATCHET & CLANK that facial expression would be an important component not just to our cinematics but to our gameplay animations as well. Once again, we were faced with the dueling goals of animation depth and scheduling efficiency. We settled on two methods for making faces: one simple one for our enemies and one more complex for our heroes. Expressions exaggerated the idles, intensified the attacks, and sealed the deaths our of enemies and heroes alike.

When animating our enemies, we drew on a traditional animation dictum: A viewer of animation is usually drawn to a character's face, particularly to the eyes. Attention paid to a character's eyes and mouth was very important to making convincing actions, especially during our quick gameplay cycles.

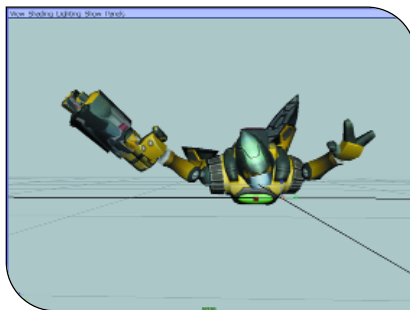
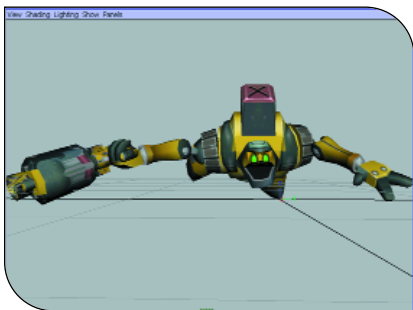
Most enemy characters had fairly simple face skeletons. However, these skeletons allowed for a high degree of manipulation of the eyes and mouth. Each eye had between two and four bones controlling its brow and lids. Mouths were generally simpler, using only one or two bones. In most cases, this setup gave us all the flexibility we needed to exaggerate the enemy's features and thus heighten the emotion of its actions (Figures 9a and 9b).

Our heroes' faces had a more sophisti-

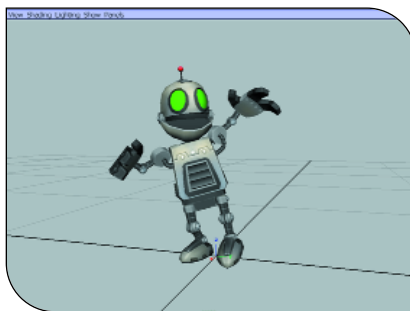
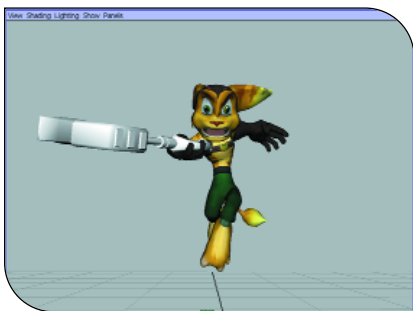
cated setup, which they shared with the NPCs. Though NPC faces were manipulated mostly in our cinematics, RATCHET & CLANK made heavy use of expression during gameplay, as well.

Like the enemy setups, hero and NPC faces were manipulated via their face joints. Unlike the enemies', these joints were animated through a driven key system instead of being transformed directly. Since they clocked more screen time, hero and NPC faces tended to have a far greater amount of bones — and hence expressive range — than their enemy counterparts.

Figures 10a and 10b show some of the range of expression Ratchet and Clank exhibit during gameplay. He smiles when excited, grimaces when he's hit, grits his teeth during combat, chatters them when he's cold, and drops his jaw when he dies. Clank's expressions change both while he's strapped to Ratchet's back and when he's played independently.



**FIGURES 9A & 9B.** With enemy face skeletons, less was more. Bone detail was reserved for the eyes and mouth to enable simple, exaggerated expressions. Here, during an in-game animation, the Robot Paratrooper's face reacts to being knocked down.



**FIGURES 10A–B.** Ratchet and Clank's gameplay facial animation system (also used for NPCs) needed more sophisticated setups than enemies' for the broader range of expression they were required to show during gameplay.

As I mentioned earlier, hero and NPC expressions were animated by combining preset driven key attributes via a MEL script slider interface. These presets allowed the animator to combine and create a wide array of facial expression without having to build them from scratch. Like color primaries, these attributes could be blended together to form new combinations.

About half of a character's 40 or so facial attributes were dedicated to producing a basic expression, either on all or on parts of the face. These basic expressions included anger, disgust, fear, happiness, sadness, and surprise, all of which would be easily recognizable to a player. More subtle attributes were dedicated to animating phonemes and controlling individual facial features. Unique and varied emotional ranges could then be achieved by combining expression, phoneme, and feature attributes together.

## Scripting Facial Presets

**A**ssigning facial presets to our characters cost us some setup time. However, we were able to optimize some of the processes with another MEL script. Like our other MEL tools, this script automated some of the tedious steps, allowing a setup artist to spend more time on the art of sculpting facial poses.

Facial presets were created in a separate animation file, where each expression, phoneme, or feature pose was stored as a separate keyframe. Upon completing this file, a character artist would use our MEL Driven Key Generator (Figure 11) to set the driven keys automatically for each pose.

The Driven Key Generator worked by comparing the transformations of the keyframed pose to those of a default. When the script registered that a channel had changed from the default, it would set a driven key on that channel based on its changed value. The script relied on MEL's arithmetic functions to identify value changes, and its `setAttr` and `setDrivenKeyframe` commands to activate the drivers. Listing 2 shows some of the Driven Key Generator's sample code.

The drivers for our facial animations were stored on a model called the Control Box, shown in Figure 12. This hierarchy of cubes served as a visual outline of facial attributes, and could also double as a second interface. For efficiency's sake, Ratchet, Clank and all of our NPC characters had identical Control Boxes, though Ratchet's had many more active drivers.

We found our automated setup method to be advantageous for three reasons. First, it saved a setup artist from having to manually identify and key bones, channels, and drivers. Second, it assigned driven keys to changed channels only, leaving any non-affected channels free for animators to keyframe. Finally, it circumvented Maya's built-in driven key interface, which we found to be cumbersome and even unreliable when simultaneously assigning multiple bones and channels to a driver.

Regardless of method, facial animation played a vital role in breathing life into our gameplay characters. Again, MEL was instrumental both in granting our artists access to an advanced Maya feature, and in optimizing our workflow. Whether a hero or an enemy, virtually every character personality in our game was strengthened through facial expressions. In turn, this enhanced interactions both with players as well as between the characters themselves.

## End of Cycle

Like all character-driven projects, RATCHET & CLANK presented our animation team with a unique set of artistic and technical challenges. Our artistic philosophy was built on the understanding that our characters were the instruments through which a player would experience our universe. We knew that in meeting these challenges, our puppets would transcend mere game space and become the entities that our players would identify with, vilify, and even personify.

However, this philosophy needed to be coupled with practical methodology if it was to see our project to its conclusion. From this necessity grew our testing practices, MEL shortcuts, and real-time animation procedures. Throughout production, these methods removed many of the barriers that would otherwise have obstructed the artistic efforts of our animators.

As the Insomniac team cycles into our next project, we continue to refine and expand upon the systems and procedures we developed during RATCHET & CLANK. Though our procedures continue to evolve, our underlying goals remain unchanged. For in the end, we can only prove a technology's worth by an audience's response to our characters. 🎮

### LISTING 2. Sample code from the Driven Key Generator.

```
// The "if" gate checks for changed X-Translation values
// between the Default and Posed frames.

if ($txa != $txb)
{
    // Sets the Driver Attribute and the Current Joint's
    // X-Translation to their Default Values;
    // Sets a Driven Key Frame for the Default Values.

    setAttr $atnm $dr0;
    setAttr ($current + ".tx") $txa;
    setDrivenKeyframe -currentDriver $atnm -attribute
        translateX $current;

    // Sets the Driver Attribute and the Current Joint's
    // X-Translation to their Posed Values;
    // Sets a Driven Key Frame for the Posed Values.

    setAttr $atnm $dr1;
    setAttr ($current + ".tx") $txb;
    setDrivenKeyframe -currentDriver $atnm -attribute
        translateX $current;

    // Prints command summary to the Script Editor for
    // easy reference.

    print ($current + ": TX has been keyed for slider
        value 0: " + $txa + " and slider value 10: " +
        $txb); print "\n";
}

// In this loop segment, $current is the current joint,
// and $atnm is the attribute name. $dr0 and $dr1 represent
// Default and Posed Driver values. $txa & $txb are the
// Default and Posed X-translation values, respectively.

// (Note: All flags are listed in their long forms.)
```

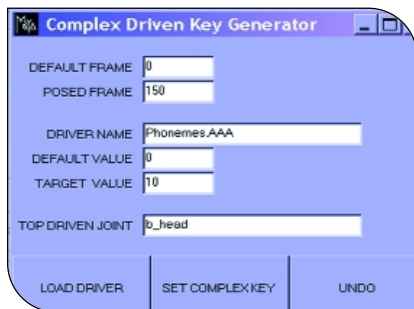


FIGURE 11. The Driven Key Generator analyzed a preset facial pose, compared it to a neutral pose, and assigned driven keys to the affected channels.

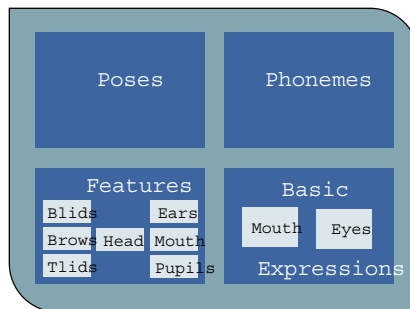


FIGURE 12. Each NPC and hero had its own Control Box on which its facial drivers were stored. Facial drivers were actually attributes of the Control Box's cubes.

### FOR MORE INFORMATION

#### BOOKS

Gary Fagin. *The Artist's Complete Guide to Facial Expression*. New York: Watson-Guptill Publications, 1990.

Frank Thomas and Ollie Johnson. *The Illusion of Life*. New York: Hyperion, 1981.

#### RECOMMENDED MAYA TRAINING

MEL Fundamentals (formerly MEL for Artists): Information available at [www.aliaswavefront.com](http://www.aliaswavefront.com). click on Maya Training under "Education"

# Inside the 2003

# Game Developers Conference

**T**he 17th Game Developers Conference convenes again in San Jose from March 4 to 8 this year, and conference organizers find themselves in a quandary similar to that faced by many of their attendees: overcoming sequelitis. Like many developers working in today's sequel-driven marketplace, their challenge lies in satisfying returning customers who expect certain features and experiences that made forerunners popular, while innovating and refining enough to attract new customers and add value for those returning.

**F**rom content spanning the different conference tracks to the extracurricular parties and events, the organizers, CMP Media's Gama Network (which also publishes *Game Developer*), and the GDC Advisory Board have focused their efforts on honing GDC into an indispensable event for game developers. The events market is still feeling the effects of 9/11 and the economic slowdown, and many development studios that used to spring for annual trips to GDC, E3, and Siggraph are now scaling back their budgets to include just one or two events for employees. Others make the trip on out-of-pocket expenses, hoping that glittering pitch demo or polished résumé will pay off big returns. Some critics contend the event has become too "corporate," leaving the needs of small independents, and even the event's own grassroots beginnings, behind.

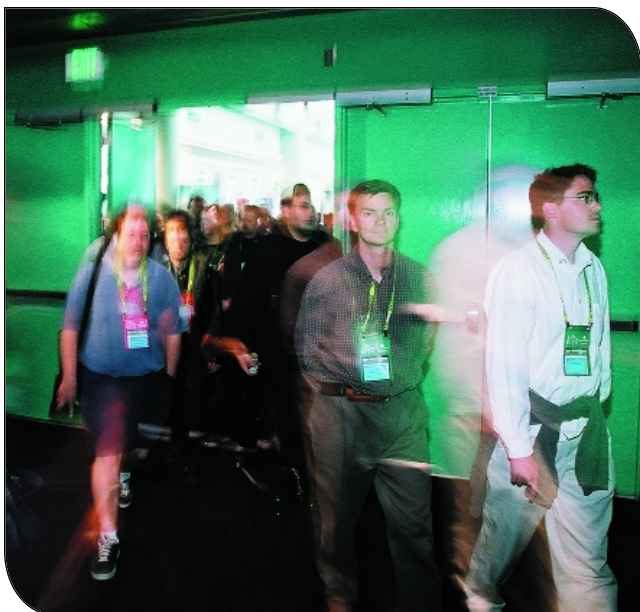
Taking into account the changing needs of a rapidly evolving industry, organizers are always looking at new opportunities to ratchet up the value of the event. The advisory board has made careful iterations in the content goals of the various conference tracks, while emerging markets are finding new prominence

through programs such as the two-day GDC Mobile event.

Creating a stronger sense of developer identity is important, too. The International Game Developers Association (which functions as an independent nonprofit under a management contract with the Gama Network) develops its own conference track focused on developer and community issues, and the IGDA returns with the third annual Game Developers Choice Awards, a program that aims to create meaningful peer recognition for game developers. Meanwhile, the Independent Games Festival, now in its fifth year, continues to groom itself into a showcase for innovative indie game projects.

Aside from the din of the GDC Expo floor, the blur of parties, and the impromptu encounters in the hallways, the heart of GDC remains in the more than 300 lectures, panels, and roundtables presented, and in the speakers who variously swim or sweat through their Powerpoint slides for the betterment of industry knowledge. We caught up with a few of this year's speakers to find out what they have in store for attendees and what they as attendees think of this ever-evolving event.





GDC attendees descend on the Expo floor as the doors open.

## ADAM SCHNITZER

Senior Artist, LucasArts

### Visual Arts Lecture: "How to Build a Better Cutscene"

**GD:** What will your session cover?

**Adam Schnitzer:** I'll talk about the reasons for doing cutscenes, how to plan your cutscenes, the importance of pre-visualization, and the ins and outs of how to transition from gameplay to cutscene and back again. I will also spend a little time talking about production methods that can streamline the cutscene creation process. Because my background is as a layout artist at Pixar, my perspective is that of someone who is very concerned with cinematic design, so much of my talk will focus on that as well.

**GD:** How do you see the role of cutscenes in games changing in the next few years?

**AS:** There are so many different styles of games, it's hard to predict with certainty what the role of cutscenes will be. But for a certain style of story-based game, they are indispensable. With game engines getting more and more sophisticated, and the consoles allowing us to create more filmlike environments, I can envision more immersive cutscenes and more seamless transitions in and out of cutscenes. With these higher-resolution in-game environments that are evolving, we will eventually be able to dispense with prerendered cutscenes altogether, but I would guess that that day is still five to 10 years off.

**GD:** The name of your session implies — fairly — that there is ample room for improvement in the quality of game cutscenes. Why do you think this area continues to vex developers?

**AS:** It's important to be very clear on the purpose of the

cutscene for your particular game. Cutscenes can do a lot to enhance gameplay, but gameplay always comes first. Having said that, I think there is a lot of ignorance of the fundamental principles of cinema in the game world. These are principles which were codified and haven't changed for the past 70 years or so. And when you go from gameplay to cutscene, you are stepping into the arena of cinematic storytelling. A greater awareness of cinematic structure and the power of the camera would help bring cutscenes to a higher level.

**GD:** How many years have you been going to GDC?

**AS:** This is going to be my first year at GDC.

## STUART ROCH

Executive Producer, Shiny Entertainment

### Production Lecture: "My MATRIX Experience: A Survival Guide to Working with Movie Licenses"

**GD:** What will your session cover?

**Stuart Roch:** While working on the MATRIX project, the film's interactive producer and I often talked about what a shame it would be if, once the project was completed, all the knowledge she and I had gained about the marriage of Hollywood and the gaming industry were to be lost. We actually felt a little bad, figuring that others in the business might be forced to learn through trial and error as we have had to these past couple years. So I hope to share all the lessons we have learned, so others in the industry can get a step up on a future licensed game. I'll cover everything from the initial deal all the way through the postproduction process.

**GD:** What are the communication challenges developers face when working with a movie studio on a licensed property?

**SR:** A number of communication problems can arise, from making sure the movie and game companies use the same terminology to describe issues, to coordinating schedules between two entertainment mediums whose pre- and postproduction schedules don't mesh very well. Perhaps the biggest communication problem that can arise is when a developer can communicate to the directors only by going through the studio office. This indirect communication channel can lead to delays and potential misinformation. In our case, the Wachowski brothers recognized this potential problem and set up communication channels directly between themselves and Shiny to make sure that accurate information and assets were flowing as efficiently as possible.

**GD:** Do you feel that your experience on the MATRIX project carries over well for future endeavors?

**AS:** Not only has the development process been easier than we expected due to the unique communication channels, but we've all learned a lot about how Hollywood approaches similar development problems, and we applied some of their solutions to our development process. Hollywood is a mature industry, and through my session, I'll be sharing some of their techniques which may be of help to other developers.

**GD:** What have been the most significant changes to GDC since you began attending?

**AS:** The most significant changes I've seen at GDC are its growth internationally and the fact that we have so many respected overseas speakers flying over to share their knowledge and experiences. There's also been an incredible growth in new talent looking to work in this industry. GDC gives those people a chance to

widen their perspective and really see what it's all about. They can meet the people that are making the games, they can meet some of the people they look up to, they can be inspired by the creativity of others, and they can take away concrete advice.

**GD:** What's your favorite event at GDC? What's your least favorite thing about GDC?

**AS:** I always really enjoy the lecture

sessions. Unlike other industry shows throughout the year, this is the one stop I make where I always have useful take-away from my industry peers. You never feel as though the time spent at GDC is time wasted. The thing that always seems to be a problem with GDC is that it inevitably coincides with some sort of crunch time on one of our projects.

**KATIE SALEN**

**Independent Game Designer**

**ERIC ZIMMERMAN**

**CEO, gameLab**

**Game Design Lecture: "Breaking the Rules of the Game"**

**GD:** What will your session cover?

**Katie Salen:** We are presenting a lecture on rule-breaking and its relevance for game design. Like it or not, rule twisting, bending, and breaking are part of games — but rule-breaking can be positive or negative, and game designers have a lot to learn from it. We will look at the formal, social, and cultural ways that players break rules and how rule-breaking can be integrated into the game design process.

**GD:** Why do players want to break the rules?

**KS:** Players break rules for all kinds of reasons, but they generally do so not to break or end the game. Instead, most creative forms of rule-breaking introduce excitement, variety, strategy, and fun into the game.

**GD:** What will your session cover? What can game designers learn from rule-breaking?

**KS:** Because games are systems, it is always possible for players to drive a wedge into it, bending the system into a new shape by breaking the rules. We challenge game designers to find ways of including this natural desire of players into their game designs. When given the right tools, players will transgress the rules of the game in pursuit of alternate forms of expression. How can this desire be enhanced or slowed, modified or transformed, by the design of the game itself? This is what we will explore in our talk.

**GD:** How many years have you been going to GDC? What has been the most significant change since you began attending?

**Eric Zimmerman:** My first GDC was 1998. It was the year in Long Beach

when the entire conference played a giant game of dart-gun ASSASSIN. The conference has lost that kind of wonderful folksiness, but that is inevitable as the industry grows. For the last two years at the GDC, gameLab has tried to bring back a bit of that spirit with our own massively multiplayer off-line games (BITE ME in 2001 and LEVIATHAN in 2002). Look for our new game this year in the IGDA booth.

**GD:** What's your favorite event at GDC?

**EZ:** The best moments at a conference always happen in the interstices between the organized talks, meetings, and parties. Surprise encounters between sessions, gab sessions at cheesy hotel bars, and late-night hotel-room game design debates are what make GDC so fantastic for me. I just wish there was more time.

#### JAKE SIMPSON

**Psychology Programmer, Maxis**

**Programming Lecture: "Animation System Implementation: What Works and What Doesn't"**

**GD:** What will your session cover?

**Jake Simpson:** My session involves discussing approaches to building state-of-the-art animation systems, heavy on the practicalities of what works and what doesn't, what's worth experimenting with and what's not. This information is drawn from experience in working with animation systems for third-person games (HERETIC II), and creating new animation systems for networked first-person games (Ghoul 2 for SOLDIER OF FORTUNE II and JEDI KNIGHT II) and for next-generation SIMS products.

**GD:** What's the single biggest takeaway you got from developing Raven's Ghoul system that you think can save developers time in planning animation system features?

**JS:** "Give the animators and coders more time to learn how to use the new technology." Game development being what it is, it usually takes two or three games before new technology matures enough for content creators to really know where the limits are and how to use it all in the most efficient manner. The earlier you get the tech in the hands of the content creators, the better-look-



United Game Artists' Tetsuya Mizuguchi demonstrates REZ during a game design lecture at last year's GDC.

ing and more spectacular the results will be. Great tech depends on artists and content creators building stuff that uses it to the best of its ability — you need everyone to be on the same page for it all to come together.

**GD:** How many years have you been going to GDC? What has been the most significant change since you began attending?

**JS:** I've been going for about four years now. From where I sit, the most significant thing is what the GDC Advisory Board is doing. They've made such an effort this year to try to be specific in what they present, in terms of actually getting value out of the lectures. I figure if I have to take notes at a lecture, then it's worth my time, and they seem to be going all out for this effect this year.

Also, every year I feel like we are starting to get more attendees from the rest of the world, notably the Japanese. We all have something to learn from people who have such mastery of our craft, and I for one applaud this international flavor.

**GD:** What's your favorite event at GDC? What's your least favorite thing about GDC?

**JS:** My favorite events are the informal get-togethers that happen after lectures and roundtables. Being able to pigeon-hole people with proven experience about a specific issue you may be having, or even just bending their ear about something that interests you, is great. It saves time, wasted research effort, and usually nets a suggestion you would never have had yourself.

Least favorite? Well, let's be honest, it's the hangover each morning. What's a GDC without some company-sponsored parties?

#### DAN SCHERLIS

**CEO, Etherplay**

**Business and Legal Lecture: "Doing Business with the Telecom Industry: Understanding Their Deal Terms, Culture, Rites, and Ritual"**

**GD:** What will your session cover?

**Dan Scherlis:** Many developers are fascinated with mobile games, for many good reasons. And many developers are having trouble getting attention from telecom types, not to mention getting deals and getting paid. I am not being glib in the session title; I do feel that the major issues are downright anthropological. The two industries speak and act differently.

**GD:** What has the telecom industry learned about game developers in the past year, and has it altered their approach to these new content creators at all?

**DS:** Telecom has been a motivated and attentive student of games and of game developers. In the last year I have heard far less talk of games as a commodity and more respect for the value of quality product. This is bringing about better deal terms.

That said, we are still of different worlds, and there is so little history of deal-making between us that we do not have much precedent for basic terms and structures of our deals.

**GD:** What's the single biggest gotcha that awaits game developers making their first foray into dealings with telecom companies?

**DS:** Communication — especially style of communication: Game developers use whiteboards; telecom types use Powerpoint.

Also, a developer should not assume that a telecom executive knows anything about games. If you offer an RTS with FPS action but RPG depth, with downloadable mods and ortho view, you will not be understood. If you give examples like WARCRAFT, HALF-LIFE, and FINAL FANTASY, you will continue to strike out. Test your pitch on a smart nongamer. Your parent or spouse will have too much of a clue. Try your dentist or accountant.

**GD:** There have been many losses, and much pain, amongst mobile-game startups. Where do you see an opportunity for profit from mobile game development?

**DS:** Mobile games are echoing the history of Internet games. We see many basic mobile games, which are hard to differentiate. Over the Internet, we saw many ad-supported games, few of which earned good money for their developers. The games that make big money online are fundamentally different: they are of premium quality, their designs exploit the network and include persistent social structures, and they have subscription economics. I believe that this description will apply to the most profitable mobile games. This is why several online-game veterans are being attracted to mobile platforms; we see an opportunity to build profitable communities, without the multi-million-dollar, multi-year development cycles.

The greatest challenge is to the designer. We need to design explicitly for this new medium, rather than trying to sell either shovelware or obvious but derivative and shallow distractions.

**GD:** How many years have you been going to GDC? What has been the most significant change since you began attending?

**DS:** I started attending GDC in 1992, I think, when I joined Papyrus. I've missed only one or two since then.

The most dramatic change has been the astounding growth. As the common complaint goes, this growth threatens the sense of community and collegiality that attracts many long-time attendees.

I can gripe with the best of them, but in fact I have been pleased by the degree to which the community survives. The San Jose Convention Center has been a chal-

lenge, many industry veterans stay home, and the sheer size is daunting, but I am always delighted by the many chance encounters with colleagues at GDC.

#### TOMMY TALLARICO

**President, Tommy Tallarico Studios  
Audio Panel: "Orchestral Panel"**

**GD:** Who's on the panel and what do you hope to cover?

**Tommy Tallarico:** We will have some of the best orchestral composers for the videogame industry on the panel: Jack Wall (MYST III), Clint Bajakian (of LucasArts fame), Jeremy Soule (HARRY POTTER), Bill Brown (RAINBOW SIX), and Dan Irish (producer of MYST III). We may also have a few special guests and surprises!

Last year's orchestra panel was really tailored toward composers. This year we felt it very important that the producers and designers know about live orchestral sessions as well, so we are tailoring it more for them. Budgets, production dos and don'ts, how to convince your boss, the real value of using live players, union vs. buyout, as well as other things, will all be covered in this panel. It is important for the project decision makers, milestone schedulers, and purse-string holders to find out how to go about getting live music in their projects.

**GD:** How do you think producers and audio professionals should best go about deciding when to use orchestral performances in games and when not to?

**TT:** Style of music and game genre is one big contributing factor. Not every game warrants live orchestra, but most would benefit greatly because of it. Publishers are looking to differentiate themselves to the consumer from a quality standpoint, and using a live orchestra can help do that. Live orchestras don't cost hundreds of thousands of dollars to produce, either — there are many different alternatives. You can't get a 50-piece live orchestra recorded for less than \$1,000 per minute of music.

**GD:** How many years have you been going to GDC? What has been the most significant change since you began attending?

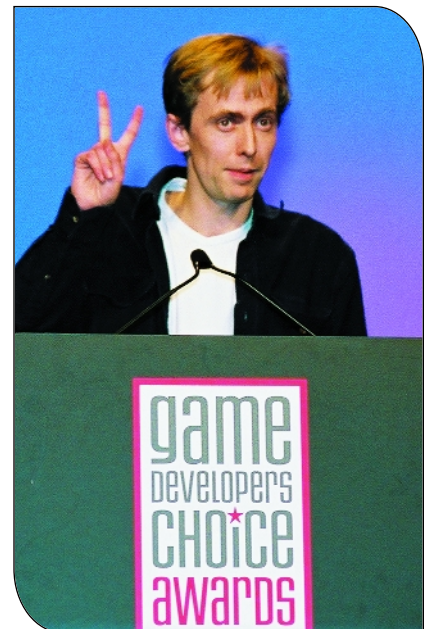
**TT:** I used to sneak in and just go to

the parties since 1993. Then I got smart and became a speaker. The biggest change besides the obvious sheer number of people would be the emergence of a huge audio community coming out to GDC. In the early years you could count the number of audio people on three hands. Since the Audio Pass was created, hundreds of interested audio professionals and nonprofessionals have come seeking knowledge and understanding of this crazy profession. In audio you have to deal with three important elements in production: creative, technical, and business. GDC is a place you can go to learn about all three.

**GD:** What's your favorite event at GDC? What's your least favorite thing about GDC?

**TT:** My absolute least favorite thing about GDC is trying to get a room every year. My favorite event is the Game Developers Choice Awards ceremony. It's really nice to see the developers get the recognition they deserve from their peers.

For the most up-to-date GDC 2003 information visit [www.gdconf.com](http://www.gdconf.com). 🎧



Lionhead's Richard Evans accepts last year's Game Developers Choice Award from the IGDA for Excellence in Programming, for his work on BLACK & WHITE's AI.

POSTMORTEM

*ian fischer and greg street*



---

# Developing Sequels: The Designer's Dilemma

## Ensemble Studios' AGE OF MYTHOLOGY

**O**ne is pretty hard. There are a lot of things to attempt and reject, a lot of mistakes to make, a lot of lessons to learn. Without a prior success (or even a prior failure) for comparison, much of your design relies on instinct. Without an experienced team, much of your schedule operates at dart-board-level accuracy. Figuring out both how to work around long-expected pieces that don't pan out and how to capitalize on unexpected miracles is a big part of the job. Mix these factors in with the usual chaos surrounding a game company on her maiden voyage and you have a situation often referred to generously as "challenging."

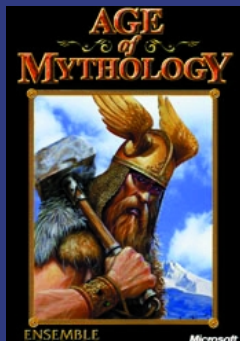
Two is easier, although you may not think so at the time. With your first game out in the wild, you're able to get real-world feedback on what worked and what didn't. You know more about your team and ideally have a familiar engine and tool set to work with, providing you with a much better idea of what's possible. Additionally, from the lazy designer perspective, half of your feature set is waiting for you at the start of the project — everything you ran out of time for on the first game.

Three is the end of the world. By this time you've amassed a good understanding of what people like about your games. Unfortunately, you also have fans who've played two titles in the series, plus a few expansions, and are starting to grumble

---

**IAN FISCHER** | Ian joined Ensemble Studios as a designer in 1997, just in time for the final stages of AGE OF EMPIRES. In his spare time he likes playing games and arguing. He can be reached at [ifischer@ensemblestudios.com](mailto:ifischer@ensemblestudios.com).

**GREG STREET** | Greg was a marine biologist up until he discovered AGE OF EMPIRES in 1998 and ended up joining Ensemble Studios to work as a game designer. Feel free to send your marine crustacean questions to [gstreet@ensemblestudios.com](mailto:gstreet@ensemblestudios.com).



GAME DATA

**PUBLISHER:** Microsoft  
**NUMBER OF FULL-TIME DEVELOPERS:** 50 total employees, 15 programmers  
**CONTRACTORS:** 10 quality-assurance contractors, no contract programmers  
**LENGTH OF DEVELOPMENT:** 30 months  
**RELEASE DATE:** October 31 2002  
**TARGET PLATFORM:** PC  
**DEVELOPMENT HARDWARE:** From Pentium 2, 300MHz, 64 MB RAM, TNT1 graphics cards to Pentium 4, 1.7 GHz, 2 GB storage, GeForce 4 graphics cards  
**DEVELOPMENT SOFTWARE:** MS Visual Studio 6, Source Safe, 3DS MAX 4.0, Photoshop  
**NOTABLE TECHNOLOGIES:** Granny, Bink  
**PROJECT SIZE:** 1,500,000 lines of code



for something different. At the same time, removal or alteration of any existing feature will be met with ranting e-mails, forum petitions, and overturned cars in your parking lot, so this is also the time when finding out that preserving everything the old games had becomes vital. The engine and tools you developed for the first game and advanced in the second are behind the technical curve by this time, so now you need to add developing and learning new ones to the to-do list. And, at some point, you're going to get a visit from a graduate of the this-trend-will-continue-forever school of projection who, armed with charts showing that title One moved a million copies and Two moved 3 million, will tell you Three should move 9 million copies.

This is where the design team was at the start of AGE OF MYTHOLOGY. The big question that haunted us was: Wow, what are we going to do to top AGE OF KINGS?

Ensemble Studios' projects are ambitious, and we ratchet up the ambition with each new project. As the scope of the project grew, the size of the team grew. We were developing a new engine and a new multiplayer online service at the same time that we were developing a new game, a game in which we wanted to incorporate new features, such as God Powers and myth units, and a more ambitious single-player campaign.

Rather than restate the all-too-common problems of having more ambition than resources, of having marketing push for content before it's ready, and of having personnel problems every company goes through from time to time, we find it more useful to focus on design aspects in this article. Designers are Ensemble's vision-bearers, but we don't get to just ram our ideas down everyone else's throats (as attractive as that power sounds at times). The designers must keep the project in scope, keep the artists and programmers from killing each other, and make sure feedback is heard without devolving the game into a design-by-committee project.

What Went Right

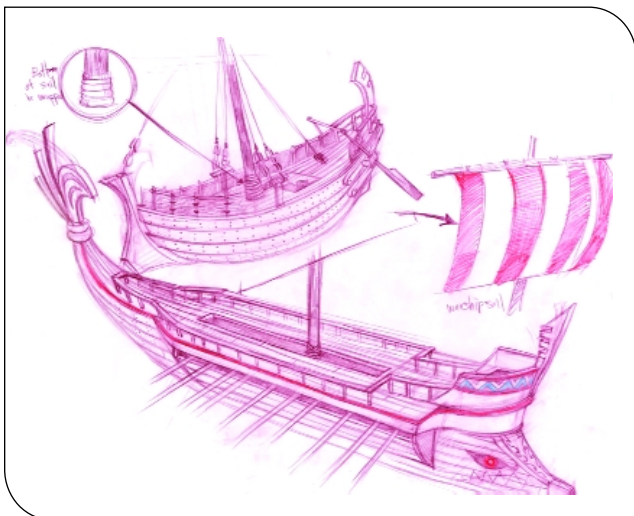
**1 • Iteration.** Ensemble's basic design process is to get the game playable early and then tweak it until it's fun. This applied to virtually every feature in the game. Some features changed a million times, and we were willing to abandon things that just didn't work, even when it was painful.

AGE OF MYTHOLOGY's God Power feature is a good example of this process in action. On paper, our initial concept of God Powers and Heroes sounded good: Heroes would have buttons on the interface to target God Powers wherever the selected Hero happened to be — simple enough.

Unfortunately, when we got the feature in the game and started playing with it, it was awful. Having to have a Hero in the place where you wanted a God Power devolved all combat tactics to selecting all your units and clicking on the enemy hero. This led to Heroes constantly getting killed, prompting comments like "The Heroes don't feel heroic." Additionally, with all God Powers targeted with your Hero, if you called down a meteor, it would land on his hand. It didn't damage him but it didn't look good.

We tried a new model where the Heroes built "lightning rods" for the God Powers (so players could kill something other than enemy Heroes to stop an opponent's God Power, and so that Heroes could get out of the way of their own God Powers). This wasn't fun. We tried another model where you could buy all of the God Powers with resources, like most everything else in the game. This wasn't fun. We tried a dozen more models and variants.

Finally, after a lot of trial and error, we hit on the model we shipped with. Heroes were divorced from God Powers and made the thing used to kill myth units (which feels decidedly heroic). God Powers were moved to the main interface, and we made them single-use only, which made them feel large and important and kept them from landing atop Heroes at every use.

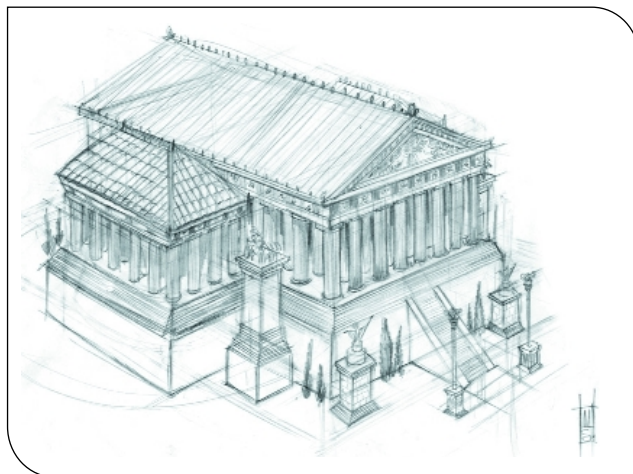


Units were first conceptualized to make sure that they both fit with the common art style, and also could be easily discerned from similar ones in the game.

Because God Powers were so important to the vision of the game, we couldn't just yank them from the title after the sixth or seventh different model didn't work. Instead, we just continued to try different systems, brainstorming and then implementing models. Because a new approach often required new code and new art before it could be evaluated, we ended up throwing a lot of work away to achieve our end result. But we did achieve an end result we're very happy with. Ours is emphatically not an efficient process, but it continues to work for us.

**2. Everyone play-tests.** It's amazing how many developers rely on outside testers to tell them if the game is fun or not. Outside feedback is vital in the later stages of a project, but if your entire game is designed by polling the fans or beta testers, you end up with a mushy game with no vision. At Ensemble, everyone play-tests the game at least once a week. This strategy keeps the team bought in to the game that's being developed. There are mandatory, assigned play-test times in the morning and multiple pickup games in the afternoons or evening.

We have found that these play-tests are instrumental in keeping the team informed on the state of the project, giving them ownership of the process, making sure bugs don't slip through the cracks, and figuring out when the gameplay is fun enough to ship. The earlier implementation of God Powers described in the previous section made sense until it got out in front of our co-workers, at which time a mob brandishing torches and pitchforks strolled into our office. If the designers had relied solely on our own instinct about the model, we likely would have shipped with it.



Ensemble Studios worked hard to capture the building detail that was a hallmark of AGE OF EMPIRES in the new 3D engine.

**3. Small meetings.** For AGE OF EMPIRES and to a lesser extent AGE OF KINGS, we kept the entire team involved in the high-level design. In one particularly long meeting for AGE OF KINGS we tried, as a company, to come up with a design for herd animals. Past a certain number of attendees, it became unmanageable to go around the room even once. So, as these meetings got longer, we tried to keep focus by including the various department leads and trusting them to relay the feedback of everyone on their respective teams. However, in a project the size of AGE OF MYTHOLOGY, even the leads' meetings could have a dozen attendees, making it harder to reach a consensus on any of the issues.

Eventually we refocused the leads' meetings on task management and progress reports and implemented a new series of meetings for design brainstorming with a core group of only four to five people, half of them designers. Features, such as the list of civilization bonuses, myth unit abilities, and God Powers, were all compiled in these meetings. When necessary, we took these meetings offsite to make sure we could get our business done without distractions. We found that e-mail wasn't efficient enough, and as busy as everyone was, impromptu meetings weren't always possible. We had to be formal about scheduling these conferences, which we ended up arbitrarily calling "small pets" (after "pet features," since we needed a name that didn't sound like we were excluding anyone). Because it was important to our process that everyone have a chance to give feedback, we would announce the decisions made in "small pets" meetings to the company at large. We heard people's concerns and ideas, incorporated any changes we thought necessary, and then implemented the design into the game. Everyone still had a voice, but ultimately we couldn't rely on a large group to come up with design implementations as fast as we needed them.



**4 ● Data-driven tools.** Developing data-driven tools early in the process was a strategy that really paid off for AGE OF MYTHOLOGY. It took a lot of programmer time away at the start of the project, when we were all anxious to begin playing the game, but the resource hit paid for itself when designers could implement new content without having to wake up the programmers all the time. For example, although we targeted 36 unique God Powers, some of these were implemented in similar ways behind the scenes. Once the programmers had implemented a God Power to switch units, we could turn enemy soldiers into pigs, or allied Pharaohs into the Son of Osiris, all through data.

There is, however, a potential downside to spending so much effort on tools. For one, instead of working on high-visibility game features, programmers spend their time on tools that players may not see. Second, you might be trading programmer time for designer time. Near the end of the project, the design team had all the tools they needed to implement some features but lacked the time to enter the changes.

**5 ● Focus on campaign.** The campaigns for AGE OF EMPIRES and AGE OF KINGS were fun but lackluster, largely completed by one or two designers. At the beginning of planning AGE OF MYTHOLOGY, we decided that the single-player campaign would be one of the game's big features on the back of the box. We hired several new content designers, invested a lot of time in custom animations for the in-game cinematics, and made two trips to Hollywood to work with professional voice actors instead of using local talent or (shudder) our own overacting.

We had never before attempted an epic, character-driven script, and we approached the task in epic fashion, appointing a story committee to review progress on the script. (In retrospect, trying to please so many people so early in the project was more trouble than it was worth.) Near the end of the project, the designers working on the campaign, often with the artists working on animations for the cinematics, met several times a day so they could all keep in touch on progress. The lead designer documented everything, ensuring changes



were made before testing the various scenarios again. It was a tremendous amount of work at the end of the project, when we could scarcely afford it. But the work paid off, and we delivered well-received single-player campaign.

## What Went Wrong

**1 ● Design drove too much.** Sure, the design department had all of these fancy tools, but in the end, designers ended up doing a lot of the work that a programmer might have been able to do faster than it took the programmer to develop the tool in the first place. We had early frustrations when specs didn't pan out as intended in the end product, coupled with the arrival of new people who had not worked with us on a title before. We compensated by heaping so much detail into specs that they were often not even read. We went so far as to provide descriptions for what artwork should be associated with the various icons in the scenario editor, and the various locations and states for those buttons.

Since all they were doing was connecting the dots on someone else's feature, new employees did not feel empowered. As a result of days spent writing things such as, "When you click this button, it should appear depressed until the user releases the mouse button, at which time it should revert to looking un-depressed; clicking the button in this manner should cause a sound to occur, the sound should be kind of like a twig snapping..." the design department took up drinking in the middle of the afternoon. In the future, we plan to keep design driving the process, but at a higher level. We will trust people at the implementation level to fill in the details.

**2 ● Scriptwriting n00bs.** We knew we wanted a script that had all the scope and drama of *The Iliad*, and we knew we wanted characters who could slap each other on the back, make fun of each other, and develop relationships over time. In short, we needed a big story with a lot of dialogue. While the designers had some writing experience in various forms, none of us had ever tackled a script like this. We also had not yet figured out how the in-game



High-polygon opening cinematics are demanded by the fans. This time around, Ensemble experimented with working with an outside contractor. Those experiences might make for an entire Postmortem by themselves.

cinematics would work, or how long we could make them without boring everyone.

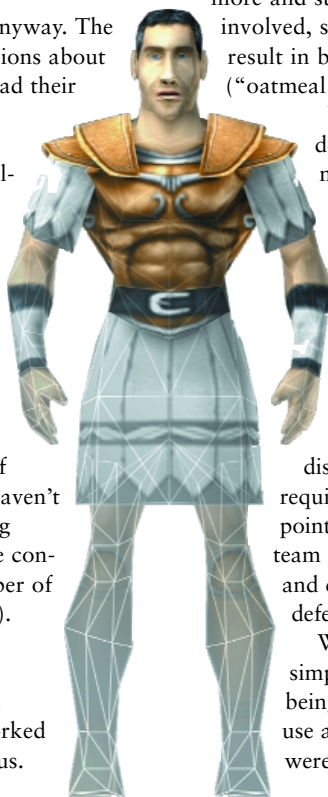
We had no idea what we were doing. We did it anyway. The result was a lot of revisions to satisfy different opinions about how a story or characters should work. Everyone had their favorite bit character or script fragment that was impossible to delete, and deleting anything threatened to collapse the intricate story line. We eventually had to take a step back and revise with a much smaller group of just two people. In the future we will keep early feedback to a smaller group, which we hope will get us closer to nailing the correct length, number of characters, and plot intricacies with fewer revisions.

Developing this sort of material at both a high level of quality and in something resembling an efficient manner demands some pain and experience that can only be gained by actually doing it. If you're planning on doing this sort of project and haven't done it before, double your estimates for everything related to the project, then halve your plans for the content (number of characters, lines of dialogue, number of scenarios, number of special art objects, and so on).

### 3 • Consensus is hard with large groups.

Consensus is the basis of the game design process at Ensemble. This company philosophy worked exceedingly well when there were only a dozen of us. Even when we grew to 20 or so people, getting everyone in a room (we usually all had lunch together) and hashing things out worked well. As the size of our team grew, however, it became increasingly less efficient to get everyone in a

room several times a week. Even worse, we stalemated a lot more and started to resort to compromises to placate all involved, so some of our design decisions began to result in bland design-by-committee game models (“oatmeal design,” in our parlance).



The models used for the in-game cinematics needed to match their low-poly equivalents that were seen during gameplay.

We settled on a strategy where the design department would gather everyone's feedback, mull it over, and then make the call. It was difficult changing our mode of communication in the middle of the project, and some of the squeakier wheels protested that there was a design black hole that swallowed up their feedback. This prompted us to redouble our efforts at documenting and communicating changes to the team, but this was an ever-widening gyre; for every e-mail you sent out explaining a decision, you got five replies that disagreed with various points of your logic and required a response. Eventually, we got to the point where we had to make a decision: the design team could either do the work we needed to do to and complete the game, or we could explain and defend every decision we made.

We came to think of our larger team size as simply a variable — it changes how we go about being consensus-based, not whether or not we use a consensus-based approach. Either way, we were committed to our consensus-based process.

Our plan at this stage was to formalize what eventually worked during the latter portions of AGE OF MYTHOLOGY with our “small pets” meetings. Our new projects are now built around small, nimble groups with rep-



One of the goals of AGE OF MYTHOLOGY was to design a beautiful, living world. Ensemble uses bright colors to make inviting landscapes they hope players will want to spend time in.



resentation from all of the disciplines. These groups have the ultimate decision-making authority but also the responsibility for gathering feedback from the entire team, explaining and defending decisions, and building a general consensus.

**4** ● **How different is “different”?** The well-known, inherent risk of sequels is that you need to keep what is popular about earlier products while still offering something new to justify the purchase of a new product. The AGE OF EMPIRES games are large and complex, and we knew we couldn’t take AGE OF KINGS and layer several new game features on top of it; we had to pull out some systems and change things to make a game that was “different.” While some of us were (or thought we were) clear on what “different” meant, there were many other definitions floating about. For some members of the team, different meant, “AGE OF KINGS, but the knights look like Minotaurs.” For other team members, different meant, “There are no units, and you control the game with your mind.” When a new feature didn’t work right away, the differences of opinion led to a lot of pressure to revert to the tried-and-true.

For example, we went through several variations of our population model. Earlier implementations lacked houses, which for some of the team was just too great a departure. The team quickly split into two camps, one that argued for even more change in the gameplay, and another that was scared that we were moving too far from the game our fans loved and expected. The fans are not particularly forgiving in this respect, and once the game was out, they applauded some of the new features while protesting about even the smallest features that were removed, such as choosing your player color.

Ultimately, there is no ideal solution to the problem of defining what is different; games today are too complex to be fully defined by a vision statement, so there will always be some degree of opinion to factor in. We plan to try to miti-

gate these disagreements in the future by attempting to answer the big questions such as “How different?” early on, and then keeping these guidelines in front of the team for the duration of the project.

**5** ● **Unfinished tools.** Because we were dealing with a new engine and there was no shared code from our previous titles, we made the right decision to reserve a lot of time early on to develop tools. Unfortunately, since the tools were for internal use only, it was easy to move people off of those tasks when other problems came up. Several tools were never completely finished, and the customers for those tools (typically designers) waited up until the final days of the development cycle to see if improvements of incomplete features could be added. While waiting for the tool, we hacked a lot of content in place, figuring that it was better to do some work than sit idle. A good example was the AI for computer opponents in scenarios, which came online very late, after the designers had hacked in fake AI using scenario triggers. This kind of work was difficult to unhack and left us with less time to iterate than we would have liked.

For the future, we (like everyone else in the game business) need to remember the value of tools and not skimp on their development time. We additionally need to pay particular attention to those tools that might ship with the game, such as the scenario editor, which was never completed to our satisfaction.

## Here to Four

**T**he good and bad of getting there aside, the end product of all this is our Three, AGE OF MYTHOLOGY. Everyone at Ensemble Studios is immensely proud to have contributed to this game and we’re now looking forward to the opportunity to figure out what type of beast Four will be.

At this early stage, we only have one question on the board:

# Narrative Games: Finding Another Side to the Story

**A**s developers, we place a disproportionate stake of our medium's identity in character-based narrative games. These games give us something that we can't get from the intellectual beta-wave exercise of puzzle games or the adrenaline-pumping simulation of sports games. While puzzle and sports games represent a hefty chunk of game sales and game players, you seldom see a polygonal quarterback, much less a falling brick, on a magazine cover. We want to reach players emotionally in the best way we know how; because stories holds a privileged place in our culture — in all cultures, in fact — we use these games to represent our craft to the outside world. As humans, we use stories to make sense of the world, and narrative games are how we achieve that goal in our medium.

If we don't make them relevant to wider audiences by increasing their variety and complexity, narrative games will ultimately hamper the evolution of games themselves. We must expand beyond the classic hero story to get these games past the status of geek recreation. It's not going to be easy, or pretty, but it's what we'll have to do if we want the game industry to grow up and achieve mainstream recognition as a form of art and human expression.

Console publishers and developers have long known that the majority of the potential game-buying population doesn't give a damn that your game has dynamic LOD, bump-mapped decals, and pushes 1.5 trillion multi-pass lit polygons per second. If we want a larger audience, we need to shift some focus from the visual aspects of game technology and focus on innovations in player experiences.

To accomplish such a shift in narrative games, we need to innovate on a number of different fronts:

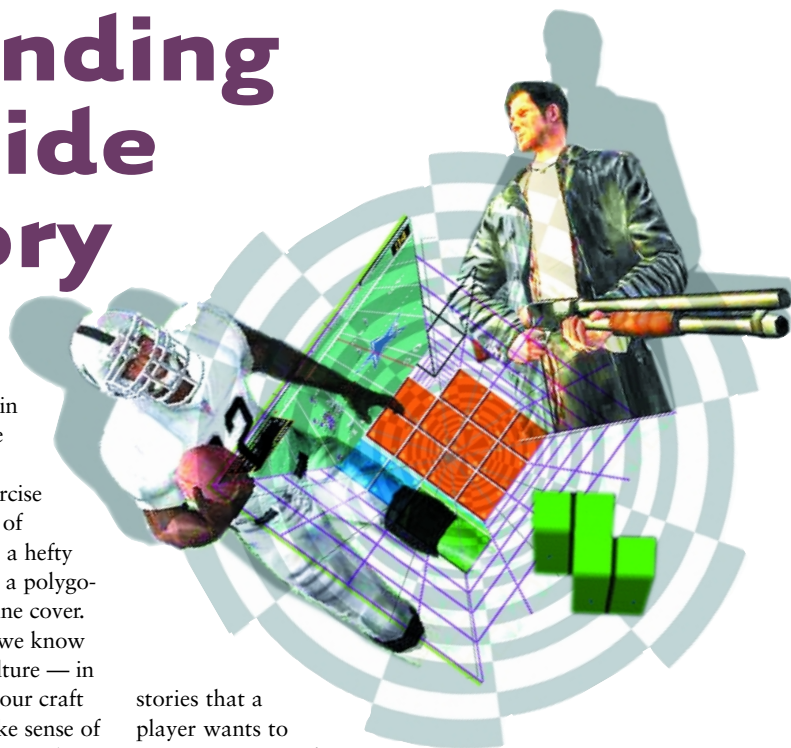
**Narrative variety.** Most story games are comparable to superhero comic books in terms of audience and themes. But why? Why not explore more varied subject matter? The majority of superhero comics and narrative games give us a chance to play with power. But that's not the only story there is to tell about being alive. We can innovate through our design and writing to explore other facets of existence; we can invent situations and

stories that a player wants to experience, even if they don't fulfill these obvious power fantasies.

This kind of innovation poses a bigger challenge than it sounds. Hero narratives depend on the triumph of the protagonist. Since we embody the story's protagonist rather than simply watching the story unfold, such games leave no room for tragedy or failure. Failure can be an effective emotional experience as a bystander, but as long as we have choice, we do not naturally choose to fail. We don't mind empathizing with a tragic character if we see the machinations of the universe working their ruin, beyond our control, but we can't stand to be that person ourselves and be powerless to prevent it. So we need new story mechanics that either allow the player to fail and still be satisfied, or other contexts for success that don't depend on an endless string of worlds-in-peril that only we can save.

**Multiple fully realized characters.** Most narrative games are plot driven, with no attention given to character subtlety. But this doesn't have to remain the case. Some games are already blazing compelling trails in this arena. For example, SEAMAN integrated the intimacy of a spoken interface with a nonheroic personal story arc. Player achievement in SEAMAN was on a much smaller scale but was as meaningful as saving the world. The day I accidentally killed my carefully nurtured two-month-old Seaman was probably the most emotionally charged moment I've had playing a game. On a technical level, until we have simulated personali-

*continued on page 63*



continued from page 64

ties complete with nonverbal communication and speech recognition, we at least need more robust AIs that can make realistic personal decisions in a wide variety of scenarios beyond simple fight-or-flight.

**Interface innovation.** Button and thumbstick controllers are especially good for fighting things but not so great at other interactions with anthropomorphic characters. Interface innovations in speech and face/gesture recognition, and the use of input methods such as music and text, all possess untapped potential to elevate and vary our emotional experiences in games.

**Emotional buy-in.** What makes us care? The size of the stakes? Not necessarily. What's really important is how much a given situation or problem relates to us personally. Everyone agrees that if "the universe" or "good" as we know it is destroyed, the situation would certainly be personally relevant. But smaller stakes can seem equally important if they are made sufficiently personal. One way to achieve this personal relevance is through a direct responsibility for another single creature. SEAMAN and earlier, simpler sim creatures were examples of this, and other recent games have successfully used this formula, such as BLACK & WHITE and ICO. More importantly, personalizing the responsibility means that

player responsibility becomes a facet of gameplay. In ICO you must actively keep Princess Yorda with you by pulling or coaxing her forward, and defend her from the constant threat of capture.

**Design for player expression.** As world builders, we can give the players tools and settings to inject themselves into the story, supporting a broader freedom within the constraints of the fictional setting. In GRAND THEFT AUTO 3, the best example to date of this kind of richly developed game world, players interact with the game world to write their own short story, while playing out the larger narrative. More often than not, it's those player-written stories that make the game memorable and generate excitement. That's why many players spurned GTA3's story mode entirely in favor of the open-ended play mode. To make better narrative games, we need to incorporate the lessons of systemic gameplay and integrate that play style into fiction arcs that aren't so easily discarded.

**Development processes that encourage innovation.** Games with characters and narrative are almost universally expensive. It's no coincidence so many indie games are puzzle games, card games, and space flight sims. Creating living, walking, talking beings is incredibly labor intensive. Large teams can saddle such risk because the potentials are large enough and because the genre is inher-

ently appealing to established developers. Publishers in turn like the fact that recognizable characters help sell games. But indie developers don't have the same resources and infrastructure to make revolutionary narrative games. Mods offer some potential, but availability of better and cheaper tools for enabling deep character development will lead to more innovation, sooner.

For years we've survived on easy interactivity, and what I'm proposing here is not the easy stuff. But our medium will not expand until we make more effort to tackle the hard and messy problems. Not all games made with more complicated technology are going to be masterpieces of the form — in fact they most certainly won't be, initially. The biggest ground-breakers should try first to entertain. Mass appeal nabs publishers and others to fund projects; risk can be incremental, and amortized. Little by little, our creative and artistic landscape will grow; only then will our *Anna Kareninas* and *Citizen Kanes* emerge. 🐉

---

**HEATHER KELLEY |** *Heather is a designer on THIEF 3 at Ion Storm Austin. She has a master's degree in radio-TV-film from the University of Texas and has contributed production and design work to three published titles during her first five years in the industry. You can excoriate her at [hkelley@ionstorm.com](mailto:hkelley@ionstorm.com).*

---