

SEPTEMBER 2005

game developer

THE LEADING GAME INDUSTRY MAGAZINE

» HEADS UP DISPLAY

PHONES WE LOVE FOR
THEIR PLAYING POWER

» STATE OF THE INDUSTRY

HOW GAMES FIT INTO
THE MOBILE AGENDA

» MOBILIZE YOUR CONTENT

PRACTICAL HINTS FOR
YOUR PORTING BLUES

MOBILE
ISSUE

MOBILE POSTMORTEM
BEFORE CRISIS:
FINAL FANTASY VII





**STILL USING
THAT OLD RAG-DOLL?**

**Do your characters walk, climb, fall down,
get-up and interact just like...well, you?**

They should!

- Unsurpassed cross-platform game physics with an integrated animation system and tool chain
- Enables next-generation character animation that blends physics and animation in real-time
- Includes HydraCore™ technology for multi-core and multi-threaded platforms
- Backed by the most experienced support in the middleware industry

Havok Complete™

Any genre. Any situation.

havok
www.havok.com



POSTMORTEM

28 MOBILE POSTMORTEM: BEFORE CRISIS: FINAL FANTASY VII

When Square Enix decided to take its classic FINAL FANTASY VII franchise onto mobile platforms, the developers quickly realized that working with established intellectual property needn't restrict the game style. The game that resulted uses the platform's camera and network functionality innovatively to weave a side-story to the classic RPG, but development of the title wasn't without its difficulties. From the stiff technical requirements to the game's initial popularity, the creators explain how they created a landmark mobile game in this month's postmortem.

By Kosei Ito

DEPARTMENTS

2 GAME PLAN *By Simon Carless*
Memphis Blues

4 HEADS UP DISPLAY
Mobile news, G-Phoria, Microsoft Meltdown wrap, and more.

7 SKUNK WORKS *By Mathew Kumar and Justin Lloyd*
Tira Jump Product Suite and SlickEdit 10

48 A THOUSAND WORDS
Capbara Games' PLANT LIFE

COVER ART COURTESY OF SQUARE ENIX

FEATURES

11 STATE OF THE INDUSTRY: MOBILE GAMES

Over the last few years, the North American mobile phone market has grown by leaps and bounds, and the mobile game industry now represents a rapidly growing revenue stream for the market leaders. But how is the interaction between developers, publishers, and carriers shifting as the market matures? In this state of the industry report, Paul Hyman investigates what the market holds for all the major stakeholders.

By Paul Hyman

19 BUSINESS LEVEL: THE TIPPING POINT

Each year, the line separating wireless and console/PC gaming blurs, as do the design fundamentals behind them. Is having cross platform design a step that all games can leverage in the future? Mike Yuen takes this concept as a starting point and discusses the connection between wireless and traditional gaming platforms.

By Mike Yuen

23 MOBILIZING CONTENT: PORTING GAMES FOR MOBILE DEVICES

Every cell phone used to play games is different, sometimes startlingly different in terms of hardware, and porting games for the mobile market is full of thorny but intriguing challenges for developers. This article looks at some of the common problems and possible solutions for those making multiple SKUs of mobile games effectively, from palette swapping to compression methods.

*By Mike Ying, Jacob Abrams, Vikas Gupta,
Bob Whiteman, and Kal Iyer*



11



23

COLUMNS

34 INNER PRODUCT *By Sean Barrett*
Immediate Mode GUIs

[PROGRAMMING]

38 PIXEL PUSHER *By Steve Theodore*
Eyes Right

[ART]

40 GAME SHUI *By Noah Falstein*
Don't Cell Out

[DESIGN]

41 AURAL FIXATION *By Alexander Brandon*
Audio Budgets

[AUDIO]

EDITORIAL

EDITOR

Simon Carless scarless@gdmag.com

MANAGING EDITOR

Jill Duffy jduffy@gdmag.com

ASSISTANT EDITOR

Brandon Sheffield bsheffield@gdmag.com

ART DIRECTOR

Cliff Scorsio cscorso@gdmag.com

INTERNS

Alan Bank, editorial & Rebecca Nickodemus, design

CONTRIBUTING EDITORS

Sean Barrett sbarrett@gdmag.com

Alexander Brandon abrandon@gdmag.com

Noah Falstein nfalstein@gdmag.com

Steve Theodore stheodore@gdmag.com

ADVISORY BOARD

Hal Barwood Designer-at-Large Dave Pottinger Ensemble Studios

Ellen Guon Beeman Monolith George Sanger Big Fat Inc.

Andy Gavin Naughty Dog Harvey Smith Midway

Joby Otero Luxoflux Paul Steed Microsoft

ADVERTISING SALES

NATIONAL SALES MANAGER

Afton Thatcher athatcher@cmp.com t: 415.947.6217

SENIOR ACCOUNT MANAGER, EAST COAST, EUROPE & EASTERN CANADA

Ayrien Machiran amachiran@cmp.com t: 415.947.6224

ACCOUNT MANAGER, SO. CALIF., SOUTH WEST, CONTRACTORS & MARKETPLACE

Susan Kirby skirby@cmp.com t: 415.947.6226

ACCOUNT MANAGER, NO. CALIF., NORTH WEST, ASIA & WESTERN CANADA

Nick Geist ngeist@cmp.com t: 415.947.6223

ACCOUNT MANAGER, GLOBAL RECRUITMENT/EDUCATION & TEXAS

Aaron Murawski amurawski@cmp.com t: 415.947.6227

ADVERTISING PRODUCTION

ADVERTISING PRODUCTION COORDINATOR Kevin Chanel

REPRINTS Julie Rapp jarapp@cmp.com t: 510.834.4752

GAME GROUP MARKETING

DIRECTOR OF BUSINESS STRATEGY Michele Maguire

DIRECTOR OF MARKETING Tara C. Gibb

MARKETING MANAGER Kathleen Cheng

CMP GAME GROUP

VP, GROUP PUBLISHER APPLIED TECHNOLOGIES Philip Chapnick

CONFERENCE DIRECTOR, GDC Jamil Moledina

ASSOCIATE CONFERENCE DIRECTOR, GDC Susan Marshall

EXECUTIVE WEB PRODUCER Peter Leahy

EDITOR, GAMASUTRA.COM Simon Carless

FEATURES EDITOR, GAMASUTRA.COM Quang Hong

CIRCULATION

CIRCULATION DIRECTOR Kevin Regan kregan@cmp.com

CIRCULATION ASSISTANT MANAGER Jessica Ward jward@cmp.com

CIRCULATION COORDINATOR Miguel Mendiola mmendiola@cmp.com

CIRCULATION ASSISTANT Michael Campbell mcampbell@cmp.com

CIRCULATION ASSISTANT Adrea Abidor aabidor@cmp.com

SUBSCRIPTION SERVICES

FOR INFORMATION, ORDER QUESTIONS, AND ADDRESS CHANGES

t: 800.250.2429 f: 847.763.9606 e: gamedeveloper@halldata.com

INTERNATIONAL LICENSING INFORMATION

Mario Salinas

msalinas@cmp.com t: 650.513.4234 f: 650.513.4482

CMP MEDIA MANAGEMENT

PRESIDENT AND CEO, Gary Marshall

EXECUTIVE VP & CFO, John Day

EXECUTIVE VP AND COO, Steve Weitzner

EXECUTIVE VP, CORPORATE SALES & MARKETING, Jeff Patterson

EXECUTIVE VP, HUMAN RESOURCES, Leah Landro

CHIEF INFORMATION OFFICER, Mike Mikos

SENIOR VP, OPERATIONS, Bill Amstutz

SENIOR VP, CMP INTEGRATED MARKETING SOLUTIONS, Joseph Braue

SENIOR VP AND GENERAL COUNSEL, Sandra Grayson

SENIOR VP, COMMUNICATIONS, Alexandra Raine

SENIOR VP, CORPORATE MARKETING, Kate Spellman

VP, GROUP DIRECTOR OF INTERNET BUSINESS, Mike Azzara

CORPORATE DIRECTOR, AUDIENCE DEVELOPMENT, Shannon Aronson

CORPORATE DIRECTOR, CREATIVE SERVICES, Kellie Ferris

CORPORATE DIRECTOR, PUBLISHING SERVICES, Marie Myers

CORPORATE DIRECTOR, AUDIENCE DEVELOPMENT, Michael Zane

PRESIDENT, CHANNEL GROUP, Robert Faletta

PRESIDENT CMP ENTERTAINMENT MEDIA, Tony Keefe

PRESIDENT CMP HEALTHCARE MEDIA, Vicki Masseria

VP GROUP PUBLISHER ELECTRONICS, Paul Miller

VP GROUP PUBLISHER ENTERPRISE, Fritz Nelson

VP, GROUP PUBLISHER SOFTWARE DEVELOPMENT MEDIA, Peter Westerman



Game Developer is BPA approved



United Business Media



MEMPHIS BLUES

A COUPLE OF YEARS AGO, THE CONCEPT OF DOING

an entire issue of *Game Developer* on mobile phone game development might have been met with a little skepticism from some quarters.

But today, with major console and handheld game publishers streaming into the mobile space, and a rapidly stratifying set of mobile-only developers and publishers who are producing increasingly interesting, competitive, and handsome products, it's high time that all of us sat down and had a proper look at what the mobile market has to offer.

Firstly, when looking for a postmortem for this issue, there was one particular flagship game we were itching to feature, and as you can see from our cover, the makers of *BEFORE CRISIS: FINAL FANTASY VII* at Square Enix in Tokyo were able to accommodate us and write a detailed account of how they made this wholly separate side story to *FINAL FANTASY VII*.

From innovative use of mobile phone cameras, to intriguing ways of leveraging network features, this action-strategy game strived to be different. But naturally, in using complex hardware and building off such a well-loved franchise, there were possible pitfalls as well as massive advantages, which the team details (pg. 28).

CAN YOU HEAR ME NOW?

For those not working directly in the mobile game industry, the most confusing question is probably the simplest: What exactly is the focus, extent, and growth pattern of the North American mobile game market? Fortunately, Paul Hyman has dug up some answers in his "State of the Industry: Mobile Games" article (pg. 11).

Through a series of interviews with many of the major players in the development/publishing and mobile carrier arenas, Hyman's reporting provides a rounded picture of how the mobile game market has evolved, and how it will continue to evolve.

HARD TO PORT, CAPTAIN

There are compelling technical problems to be solved on mobile platforms as well—in particular, the issues related to how you get one game to play on what are, effectively, scores of individually different hardware platforms. The complex issues related to this are exactly what Michael Ying, Jacob Abrams, Vikas Gupta, Bob Whiteman, and Kal Iyer try to tackle in their article

on mobile game porting (pg. 23). From problems with restricted heap sizes, all the way to differing J2ME implementations, there are a host of interesting, unique problems to be addressed, and this article helps to do just that.

BIZ LEVEL UP

Of course, we're also featuring all our traditional columns for the issue, and whether you'd like to know more about computer-generated eyes or new tactics regarding effective estimation for in-game audio costing, our regular columnists should be referred to immediately.

Plus, we've got another special Business Level column, this time by Mike Yuen of Qualcomm (pg. 19), talking about possible mobile and console/PC game convergence, and both a mobile game-themed Thousand Words art piece and a Heads Up Display that takes a look at the latest mobile handsets and their suitability for game-related key-pounding.

TELEVISION PERSONALITIES

Finally, we'd like to bid a fond farewell to our Inner Product columnist Sean Barrett. Sean has been invaluable at analyzing a multitude of code issues relevant to our industry, and we're very sorry to see him go, even if his figures, diagrams, listings, and tables make our underpowered journalistic heads ache.

Despite Mr. Barrett's departure, the Inner Product column won't be going anywhere. If you have feedback regarding what you'd like to see covered in the column in the future, or, indeed, comments and suggestions on what the other columnists have been up to, contact us at the usual address: editors@gdmag.com.

Now, if you'll excuse me, the allure of mobile gaming has me firmly in its thrall, and I'm off to play one of those terminally addictive PopCap-licensed games on a mobile handset we tested for the Heads Up Display feature. Mmm, PopCap.

S!

Simon Carless, editor

Adding a new dimension to mobile gaming has never been this easy.

```
figure = new Figure("/res/myfigure.mbac");  
Texture tex = new Texture("/res/mytexture.bmp", true);  
figure.setTexture(tex);  
...  
action = new ActionTable("/res/walking.mtra");  
...  
figure.setPosture(action, 0, frame);  
frame+=action.getNumFrames()/50;  
if(frame>=action.getNumFrames()){  
    frame=0; }try{  
    g3d.bind(g);  
    g3d.renderFigure(figure, 0, 0, layout, effect);  
    g3d.flush();  
}catch(Exception e){}  
  
<<robot.jpg>> <<sphere.jpg>>
```



It's easy to make your 3D models come alive in Sony Ericsson mobile phones with Mascot Capsule Micro3D v3. For tips, tricks, code and case studies, go to www.SonyEricsson.com/developer/java3d.

With mobile plug-ins for Mascot Capsule v3 and JSR-184 (m3g) you can now use your favorite tools, like 3D Studio Max, Maya and Lighwave, to add exciting new dimensions to mobile games. And at Sony Ericsson Developer World you'll find everything you need to get up to speed in mobile Java™ 3D development. From technical documentation and the popular J2ME SDK with support for full 3D emulation and on-device debugging, to responsive tech support.

Once you've realized the possibilities, Sony Ericsson Developer World can assist you all the way from mind to market. No one is more committed to mobile Java 3D than Sony Ericsson, and our unparalleled line of Java 3D technology-enabled phones is just one proof of that.

But Java 3D doesn't just have the power to move the mobile gaming experience into new dimensions, it also has the power to energize your business. So to find out what mobile Java 3D can do for your games and your business, go to www.SonyEricsson.com/developer/java3d – your starting point in mobile Java development.



Sony Ericsson

www.SonyEricsson.com/developer

AWARDS IN A STATE OF G-PHORIA

THE G-PHORIA AWARDS SHOW, HELD ON JULY 27

and televised August 9, was the third such annual event for cable TV channel G4, and marked a turning point in the entertainment value of game-related award shows.

The event focused more on presentation and spent less time on the actual games or the pomp and circumstance of award-giving. This largely made for a more enjoyable show, while still clearly tied to the games it represented, with host Wilmer Valderrama (of *That '70s Show*) arriving on the scene attached to a giant Katamari.

The awards presentation generally eschewed the traditionally stilted awards show humor by virtue of the casual, standing room-only atmosphere and by having some genuinely intriguing B-list celebrity presenters, ranging from Tommy Chong to William Shatner.

Shatner's presentation of the Legend Award to Ralph Baer was certainly a crowd pleaser, even if a large portion of the audience didn't know who Magnavox creator Baer was.

Live musical performances from industrial-meets-*Stomp!* duo BabyLand, The Bravery [the lead singer of which proclaimed his favorite game to be *BURGERTIME* ("It kept me warm on many a lonely night")], and The Black Eyed Peas spiced up the night. The award nominees and winners (listed on G4.com) wound up being quite sensible, given that they were largely player- and audience-chosen—a fact which also accounts for the relative *HALO 2* and *GOD OF WAR* dominance.

It may not have had the most readily-made street credibility with the development community, but in terms of putting on a show, this was one of the more successful attempts witnessed. Given that the



William Shatner presents Ralph Baer with the Legend Award.

broadcast featured the games in greater prominence than the live taping, it seems as though the G-Phoria awards might now be close to the right balance of mainstream entertainment and game content, in the quickly-expanding arena of game awards shows.

—Brandon Sheffield

PHOTO BY BRANDON SHEFFIELD

EA EXPANDS MOBILE OFFERINGS

ELECTRONIC ARTS HAS ANNOUNCED BOTH THE signing of two major new deals with mobile phone carriers Sprint and Verizon and an agreement to publish games that will see the company's titles released on a wider range of mobile handsets than ever before in the U.S., as part of an aggressive expansion plan into the mobile marketplace.

In the Sprint deal, the carrier will offer mobile versions of many of EA's titles, including *TIGER WOODS PGA TOUR 06*, as well as mobile-only titles

from the company's mobile subsidiary Pogo.com. The games will be available in the Game Lobby by Sprint, which can be accessed from both the fixed Internet and the mobile Web.

The Verizon deal will work along similar lines. Verizon Wireless customers with VCast-enabled phones will be able to download 3D games, including *NEED FOR SPEED UNDERGROUND 2*, which will launch in 3D exclusively on VCast.

An inter-related deal announced within a similar timeframe was the agreement for EA to become a

publisher of wireless games developed for Qualcomm's BREW solution. EA's roster of BREW games will feature popular titles including *MADDEN NFL 2006*, *THE SIMS 2 MOBILE*, *TIGER WOODS PGA TOUR 06*, as well as *POPPIT!*, *TURBO 21*, and *TRI PEAKS SOLITAIRE* from Pogo.com.

In a keynote speech at the Game Developers Conference Mobile earlier this year, EA Mobile general manager John Batter suggested that between 15 and 20 existing EA franchises would make their move to mobile formats this year, as EA took more firm control of its mobile content following a previously license-heavy approach.

—David Jenkins and Simon Carless



In Vista, Microsoft puts the user's games at the forefront, along with other commonly used applications.

MICROSOFT MELTS INTO VISTA

COUPLING WITH THE TENTH anniversary of DirectX, Microsoft hosted its annual Meltdown conference in Seattle July 25–27. The agenda leaned heavily toward discussing the upcoming features and capabilities of Windows Vista (née Longhorn), whose beta version is now available to select developers. The event was attended by approximately 500 professionals.

Mike Morhaime, president of Blizzard Entertainment, gave a keynote that was, in essence, a

bullet-pointed postmortem of the company's tremendously successful MMO title, *WORLD OF WARCRAFT*. In discussing the game's production, Morhaime advised other developers that "a test site is not optional," adding later that "testing is everyone's job." He also advocated that developers prepare to grow their games by thinking long-term and staying flexible, and pointed out "planning for migration," mentioning *WORLD OF WARCRAFT*'s move into foreign markets such

as Korea and China, as a practical example of how to maximize profits.

Other topics discussed in sessions included the importance of the move towards making 64-bit games; parental control features for electronic entertainment; techniques for specific applications of shaders; and DirectX's future.

For sessions in which Microsoft officials held the podium, a few key game-related points seemed to emerge. *One*: According to the

GAME YOUR PHONE UP

SINCE UPGRADING OR TESTING OUT PHONES IS NOT

always the top priority for a busy game developer, we're showcasing some of the latest handsets from the biggest manufacturers of mobile phones for the American market. We specifically report how easy it is to reach the gaming interface and how titles play on the phones, both from an ease of use and control scheme point of view.

—Brandon Sheffield and Simon Carless



NOKIA N-GAGE QD ▲

SPECS: 176x208 screen in 4,096 colors, 3.4MB memory, custom N-Gage-compatible games in cartridge form.

PROS

- Games controlled via a proper game pad layout, and control is natural and largely uninhibited.
- N-Gage exclusive games easy to access via MMC.
- Some impressive use of multiplayer N-Gage Arena capabilities.

CONS

- Technologically lags a little behind today's latest 3D-powered handsets.



LG VX8000

SPECS: 176x220 screen in 262,144 colors and an additional 128 x 160 external screen in 262,144 colors, 128MB memory, 1.3 megapixel camera.

PROS

- Notably sharp and large LCD screen.
- A well-defined "disc" controller for playable results.
- Also runs cutting-edge VCast titles.

CONS

- Game section several clicks away from the main menu.



SONY ERICSSON K750i ▶

SPECS: 176x220 screen in 262,144 colors, 34MB memory, Memory Stick Duo support, 2 megapixel camera, radio, MP3 player.

PROS

- Extremely intuitive interface for easily accessing games and other content.
- Centrally located control pad for game playing.
- Good performance on 3D Java games.

CONS

- Some possible control pad responsiveness and accidental click-down issues during gameplay.

MOTOROLA E815 ▶

SPECS: 176x220 screen in 262,144 colors and an additional 96 x 64 external screen in 4,096 colors, 40MB memory, 1.3 megapixel camera.

PROS

- Simple but effective games access.
- Central "disc"-controller very precise, action button well-separated.
- Available VCast games extremely sophisticated.

CONS

- Spread out controls somewhat less conducive to arcade-style twitch gaming, like the LG.



company's research, PC users spend about 18 percent of their total time playing games (including installed casual games like SOLITAIRE, but excluding Web browser based ones). *Two:* Game playing is the third most popular application for PC users, behind using the Internet and email. *Three:* The market for games, especially casual and online games, will increase over time. (In 2004, for example, online game revenues reached \$2 billion, up from \$695 million in 1995, according to Rick Wickham, director of business development and strategy for Microsoft Windows Gaming and Graphics.)

It's clear that Microsoft is planning to upgrade the visibility of gaming in its next Windows operation system—to this end, Microsoft's Vista will put games at the forefront of the user's plate, right next to "My Documents" and other popular, one-click subsets (see the image).

With game profits growing as fast as an expectant mother, it's no wonder that the company is pushing so hard to put games at the forefront of Vista. If Microsoft can get the timing just right, Vista may be the company's first offspring born with a silver joystick in its mouth.

—Jill Duffy

CALENDAR

Accelerating Change: Artificial Intelligence Amplification

Stanford University
Palo Alto, Calif.
September 16–18
Cost: \$150–\$600

www.accelerating.org

Tokyo Game Show 2005

Nippon Convention Center
Makuhari, Chiba, Japan
September 16–17, 2005
Cost: 1,000–1,200 yen

<http://tgs.cesa.or.jp/english>

Games for Health Conference

University of Maryland,
School of Medicine
Baltimore, Md.
September 22–23, 2005
Cost: \$300–500

www.gamesforhealth.org

Indie Games Conference

Mallard Banquet Hall
Eugene, Ore.
October 7–9, 2005
Cost: \$195–\$250

www.indiegamescon.com

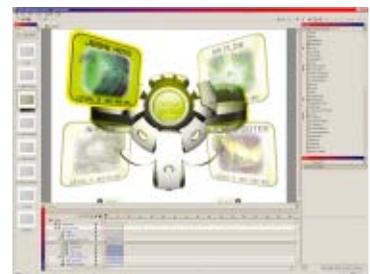
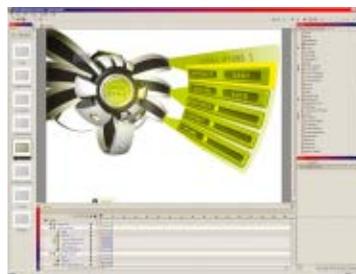
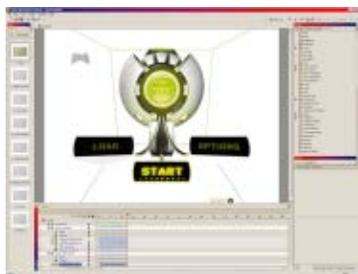
ANARK GAMEFACE

Build great UIs in half the time



Anark Gameface, the first professional solution for creating 3D game user interfaces and front ends. Cut your development time in half by building a bullet proof UI pipeline.

Now an authorized middleware tool for the Xbox 360.



Who wins?

- **Programmers:** Code the pipeline once and never touch the UI again.
- **Artist:** Use a professional authoring tool with real workflow.
- **Producers:** Save money and reduce schedule risk.

Anark Gameface Bundle Includes

- **Anark Studio:** Proven interactive 3D authoring tool
- **Anark Format SDK:** All the tools you need to import Anark UI projects into your existing game engine.

For more information go to www.anark.com/gameface or call 866.705.1010 x102.

TIRA WIRELESS' TIRA JUMP PRODUCT SUITE

By Mathew Kumar

TIRA JUMP PRODUCT SUITE



Tira Wireless Inc.

4110 Yonge St.
Suite 606
Toronto, Ontario
M2P 2B7
416.642.8472
www.tirawireless.com

PRICE

\$100 per user per month, plus \$50-\$300 transaction fee per port.

SYSTEM REQUIREMENTS

PC: 2GHz Pentium 4, 1GB RAM, high speed Internet connection, Windows 98/2000/XP.

PROS

1. A fully functioning suite that fits all the possible needs of a porting team.
2. Automated system decreases porting time significantly and nearly error free when used on good source code.
3. Porting team doesn't have to rely on programming team due to solid documentation and no direct modification of source code.

CONS

1. Still requires physical handsets throughout nearly all testing, as manufacturer- and Tira-provided emulators are often poor.
2. Requires users to learn a new Java Library and be familiar with Aspect Oriented Programming techniques.
3. Projects have to be programmed with Jump in mind to avoid major errors in porting.

WHILE SIMPLE JAVA-BASED MOBILE

games can be programmed quickly and without major outlay, there are hundreds of different Java-enabled phones currently available, each with wildly differing specifications: screen sizes, memory capabilities, key placements, and so forth. To take advantage of the mobile market, game developers need to be able to port their games to as many handsets possible. Until now, this process has been extremely costly and time consuming, as you had to program each game specifically for each different handset.

This is where Tira Wireless' Jump Product Suite steps in. The product lets game developers and publishers run their own porting teams (rather than relying on outsourced service providers, though this auxiliary service is also provided by Tira) using a largely automated system to quickly and efficiently port games to hundreds of handsets in far less time than it would have taken to port them manually.

JUMP DEVELOPER DESKTOP

The meat of the Jump Product Suite is the Developer Desktop, integrated with the Jump Transformation Engine. The Jump Developer Desktop takes the form of a plug-in for the popular open source Java development environment Eclipse. As part of Eclipse, the Desktop is quick and easy to learn to use for anyone with prior knowledge of Java Development Environments.

Due to the automation inherent in the suite, the porting cycle is fairly rigid. To maximize the usefulness of the system, Tira Wireless recommends that developers use three reference builds—the master versions of the software to be ported—although users are able to choose how many reference builds they want to start with. The reference build platforms currently supported include the Nokia Series 40 and 60. Using a reference build, the product can be ported to handsets that most closely match the required capabilities in these handsets. This function is semi-automated, as each port is itself a separate project. Say, for example, you have a game that you want to port to a high-end phone, a handset

loosely similar to Nokia's Series 60. Jump would create for you, on your computer, a configuration file outlining the changes needed to transform this application. The files (the configuration file and the JAR/JAD) are sent to Tira Wireless' servers where the Jump Transformation Engine converts the game to the new handset using the combined target device and channel plug-ins. The revised build is returned to the desktop quickly. Then, using the automatically selected emulator (provided by the device manufacturers, or a generic emulator provided by Tira), the newly created Java files are tested for compatibility.

While the Transformation Engine works well, it has, like all automated systems, a susceptibility to human error. Unless the code fed to it is as generic as possible, the automated ports can be poor, greatly increasing the amount of time required for porting. With the limited resources of the current generation of handsets, teams may have to alter their programming ethos to allow for a smooth porting experience. To many programmers dedicated to squeezing the most out of a platform, this is a major flaw. But the increased portability of clean code makes up for the performance lost from reference builds.

Flaws caused by automated porting can take any form—including severe crashes. However, they mostly comprise graphical glitches and sound errors. The Jump Product Suite is designed so that the porting team never has to alter the reference build source code. But for many projects, errors can be circumvented simply by altering the XML rules the Transformation Engine makes reference to during the port.

For projects with deeper issues, the build is optimized using so-called "adjustment files," which alter the byte code as it's compiled on the Jump servers. Though the



Tira Jump provides several phone emulators for testing the porting of games.

Jump system does decrease the number of original source files held by the versioning software, they are modified by a large number of adjustment files for each handset (similar in concept to Aspects in Aspect Oriented Programming). The porting team does need to learn the new adjustment Java library—a library of pointcuts to sections of code and the alterations possible at such points. Porting engineers therefore still need to be well-versed in Java to debug the provided code—but now they don't have to rely on the programming team (unlike in manual porting), particularly due to the solid documentation Tira provides to assist them.

During the adjustment phase, the project will need to be tested repeatedly. Tira provides a whole range of official manufacturer-approved emulators, along with its own generic emulators for phones whose manufacturers have not provided them. The generic emulators may be the weakest part of the suite. For a popular handset, such as British service provider O2's X4 third-generation handset, the emulated version exhibits errors such as refresh problems and graphical glitches that aren't found on the real handset. Some working ports don't even boot successfully in the emulator—they just freeze. Due to the likelihood that the porting team won't have all physical handsets available to them at all times, this is a severe limitation.

Fortunately, for teams that do have physical handsets available for testing, the Jump Product Suite can cut porting

SLICKEDIT 10



SlickEdit
3000 Aerial Center Parkway Suite 120 Morrisville, NC 27560 919.473.0070 www.slickedit.com

PRICE
From \$284 (Windows) to \$799 (multi-platform).

PLATFORM SPACE REQUIRED
Windows XP, 2000, NT, Me, 98, 150MB disk space.
Linux Kernel 2.4, 250MB disk space.
AIX 5, 300MB disk space.
HP-UX 11 and higher, 400MB disk space.
IRIX 6.5 and higher, 300MB disk space.
Solaris SPARC 7 and higher, 300MB disk space.
Mac OS X 10.3 and higher, 150MB disk space.

C++ REFACTORING REQUIREMENTS
One of the following C++ compilers is required for C++ Refactoring:
Microsoft Visual C++ 6.
Microsoft Visual Studio .NET 2002.
Microsoft Visual Studio .NET 2003.
GNU C++ 3.x (for Linux and UNIX platforms).

PROS

1. Language and project support are second to none.
2. Mature and stable code editing and project environment.
3. Good for console development environments.

CONS

1. Prohibitively priced.
2. Unfamiliarity with new tools prevents changing from current environment.
3. Complexity of product.

time dramatically, even with difficult spaghetti code.

JUMP WORKFLOW MANAGER

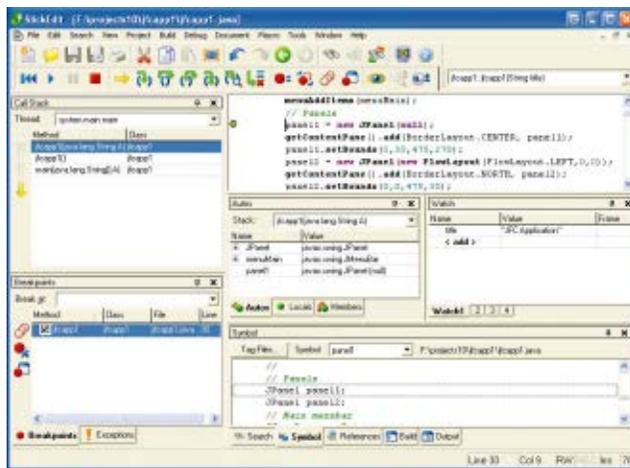
The Jump Product Suite also includes the Jump Workflow Manager, a service hosted by Tira that provides tracking, management, and control for the porting process, including secure versioning control and storage. This option is appreciated by some, I'm sure, but is less useful to developers who already have a system implemented for versioning control and local storage system. Yet, for developers who work with remotely located teams or small teams without a strong system already in place, the Workflow Manager is a functional inclusion.

The Tira Knowledge Base is also handy for some. It's a database of known fixes for common porting errors and handset issues, and while it might at first seem to be a boon, the database is confusing to navigate, and some topics are poorly explained.

Tira does provide a great deal of help in the form of manuals and tutorials for beginning users, making the set-up and learning of the system a painless procedure. The availability early on of such thorough documentation makes it particularly disappointing when dealing with advanced issues later, times when you may find yourself trouble-shooting on your own. However, the Knowledge Base is continuously updated, so should become more useful as Tira's customers start to encounter more advanced issues.

EXTENDING MOBILITY

The Tira Jump Product Suite is a system that seemingly revolutionizes the wireless games market, allowing even the smallest development teams to take



SlickEdit 10 is one of the most versatile code editors currently available.

control of porting their games to the widest possible audience. The suite is hampered by the limitations of the emulators provided by the device manufacturers and Tira. Therefore, developers, realistically, should plan to run tests on physical devices throughout the entire porting cycle. While Tira Jump offers the ability to port to hundreds of handsets, small teams might not have the resources available to maintain an ever-growing collection of mobile phones. Even if you're relying on outsourced testing, the low quality of the Tira-provided emulators can leave the porting team with an impossible job.

Despite the system's various issues, once a porting team has learned how to use it, the Tira Jump Product Suite is an excellent and powerful tool for any studio looking to take personal control over its entire wireless games output. While smaller teams will be better suited by outsourcing the entire process, teams with the resources to handle a large library of handsets—and staff with good programming fundamentals—will reap the benefits.

MATHEW KUMAR is a graduate of Computer Games Technology at the University of Paisley, Scotland, and is now a freelance journalist in Toronto, Canada. Email him at mkumar@gdmag.com.

SLICKEDIT INC.'S SLICKEDIT V10
By Justin Lloyd

Since version 8, source code editor SlickEdit has enhanced integration with native operating systems: Under Microsoft Windows, SlickEdit installs itself to the Microsoft standard "Program Files" directory and also makes use of "My Documents" to store all preferences and tag files—tag files are constructed by SlickEdit. Previously, preferences and tag files were stored under the main SlickEdit directory making backup,

migration, and multi-user desktops difficult to manage. Now with version 10, SlickEdit continues to meet and exceed user expectations.

The use of user directories also eases the sharing of tag files between developers on teams. You know precisely where to find your tag files and no longer need to hunt for them in program and project directories. And with subsequent updates, it's this level of detail that makes SlickEdit as useful as it is.

The GUI has undergone a partial overhaul so that it falls more in line with Microsoft and Apple user interface guidelines. Enhancements include dockable panes, auto-hide utility windows after use, and grouping of utility windows onto a tabbed pane. SlickEdit is also multi-monitor aware—I can't imagine trying to work productively on a single monitor workstation anymore. The multi-monitor support is some of the best I've seen, preventing text editing windows from splitting across a monitor when first opened, and being able to set "full screen editing" on more than just your primary monitor.

SlickEdit's biggest new feature for game developers has to be C++ refactoring, introduced in v9 and improved greatly in v10. SlickEdit is one of the few editors that offer the ability. There are so many improvements in this area it's now actually usable functionality and no longer just a checkmark on a marketing brochure. If you couple

refactoring with customizable code beautifiers, you have a decent tool to hammer your source code into something readable.

PROJECT SUPPORT

SlickEdit isn't an editor or environment aimed purely at people building Microsoft Windows applications. It's designed for developers who work with multiple languages, project types, and data files on a daily basis, making it ideal for teams who have to ship titles for multiple platforms that include embedded scripting languages.

With support for 47 languages and myriad project types, I'm sure SlickEdit has weak support for some of them, but I have yet to find those. The handling of the major project types, Microsoft Visual Studio (MSVS) 2003 and 2005, ANT, and a number of others I have tried over the years, has proved to me that it can replace your integrated development environment (IDE) of choice for everyday usage.

If you're moving from a one-platform, single solution IDE (such as MSVS), I suggest that you sit down with the tutorials for SlickEdit, as the application does possess a different nomenclature and workflow from what you're probably used to. There's mental translation from one environment to another until the new one becomes familiar enough.

SlickEdit has supported the major source code repository systems for a number of years—for example, CVS and Perforce—and now in v10, Subversion. My company is currently transitioning from Perforce to Subversion for content management and it's making a big difference for us.

I'm in the midst of developing games in Java, so I am pleased to see that SlickEdit v10 supports Java 2 Micro Edition and Standard Edition (J2ME and J2SE). This feature alone lets me debug the database driven web application and the J2ME and J2SE games inside of my IDE. Finally, I'm moving away from multiple DOS boxes and MSVS. SlickEdit has extensive support for ANT and JUnit—both of which I use for J2ME games—that integrate flawlessly into the IDE. Double-click compilation errors output by ANT or JUnit and SlickEdit jumps straight to the problem line every time.

As a mostly embedded systems and console developer (and now J2ME), I occasionally have to lower myself to distasteful tasks like building web pages for the company site. Utilizing ASP.NET, C# or VB.NET, XML, Javascript, and HTML—all of these languages quite often buried in the same source file—I have to build some complex web application on the back-end that will communicate with the game front-end, or a utility that pulls data from a FAQ directly into a forum post. Until SlickEdit came along, there was no editor that I was aware of capable of handling the errors, syntax coloring, and proper formatting of a source file containing more than one language.

SlickEdit supports multi-session debugging, allowing you to debug multiple applications, for example, a game and a web server application that the game communicates with, from a single instance of SlickEdit, which is useful when you are creating a J2ME application with a back-end web application.

A LOT OF FEATURES IN ONE BOX

Very few IDEs provide a comprehensive and flexible source code compare-and-merge system, so I've always used third-party diff utilities to perform these actions on my source code. The problem with third-party utilities is that they use a different interface, different keyboard layout for navigation, and so forth. SlickEdit makes use of Diffzilla, integrated completely in to the SlickEdit environment so that you never have to step out of your IDE to perform a merge or source tree compare. Diffzilla uses all the key bindings and keyboard emulation that you have set and makes use of the SlickEdit editor too, so you aren't lumbered with a crippled diff utility editor.

Navigation through your projects has been enhanced with new syntax searching, improved tagging of source files—especially those with embedded languages, such as VBScript or Perl inside of regular source files—and better symbol analysis when navigating by function definition and reference.

One tip: you should ensure that you're running the latest patch, 10.0.2 as of press time, as it fixes dozens of small, irritating bugs. I ran into two minor UI

bugs during my review of version 10 that were completely fixed post-patch. SlickEdit isn't buggy by any means—any application this complex will no doubt exhibit a few latent bugs. I just happened to exercise the program through some of the more esoteric features while reviewing. You probably wouldn't find these bugs in a year's worth of use.

Perhaps I'm just too used to the slick way that MMORPGs and Microsoft Windows handle patching, but when a patch is available I expect the application to just handle it all for me after I tell it I want the patch. SlickEdit isn't the only application on the market guilty of adding friction to the user experience through the UI when it comes to patching. SlickEdit can readily check for updates, but that's all it does. What would you do if you took your car in for repair and after a brief inspection the mechanic told you, "Yup, it's broken. Here's the tools, the replacement parts, and the service manual. Have fun." I expect more than a link that opens a new browser window to the support area of the company's web site, leaving me to determine the application version, specific operating system release, and patch release I require.

POWER TO YOUR TOOLBOX

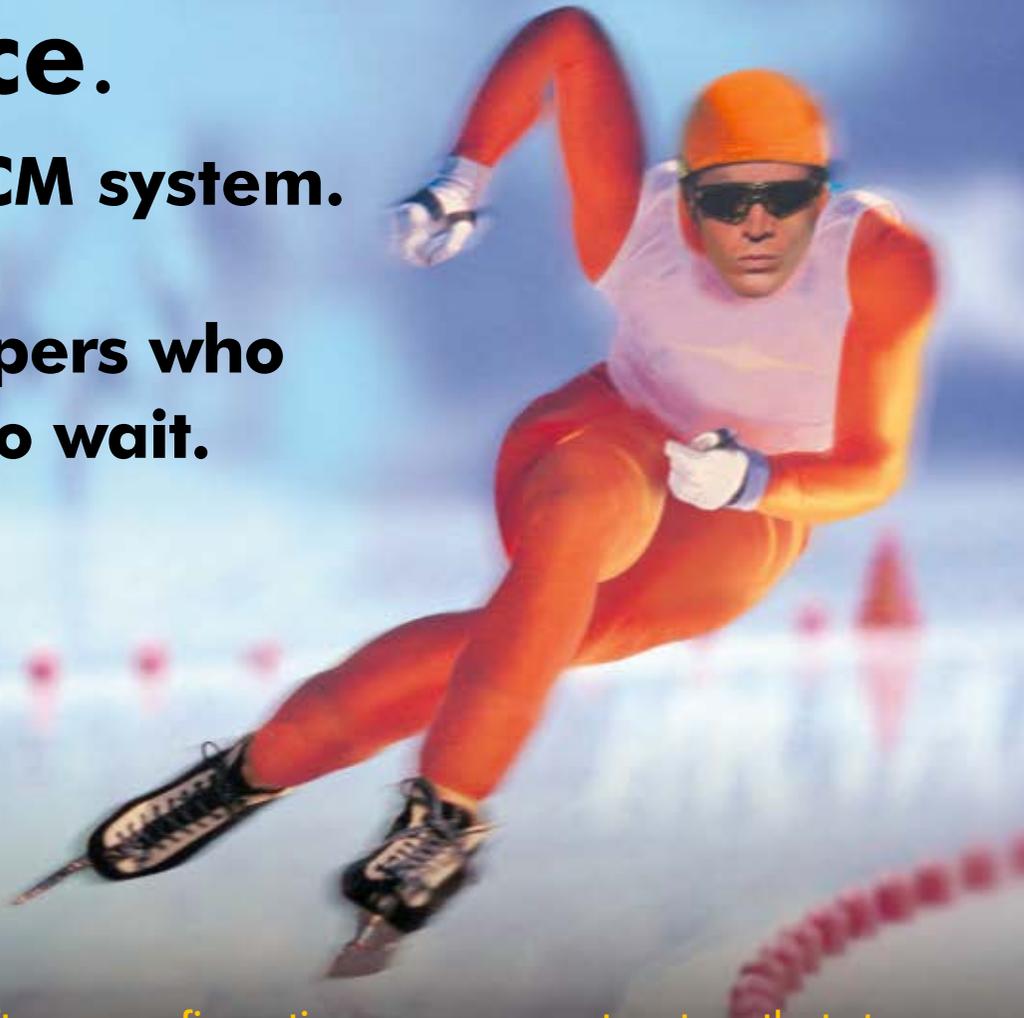
You should consider SlickEdit as another powerful IDE to add to your toolbox. You can use SlickEdit to replace the old standby, MSVS, or you can use it to supplement and enhance your current set of tools. Asking developers who are deeply entrenched in the MSVS camp to switch editors is difficult, but by sticking to only one environment—and ignoring SlickEdit—you're passing up an immense opportunity to use one of the best editing and development environments on the market. ❌

JUSTIN LLOYD is director of business development for *Infinite Monkey Factory*, a game development studio based in Los Angeles. He has worked with software and hardware for 27 years, 20 of them specifically on video games. Email him at jllloyd@gdmag.com.

Perforce.

The *fast* SCM system.

For developers who don't like to wait.



Tired of using a software configuration management system that stops you from checking in your digital assets? Perforce SCM is different: fast and powerful, elegant and clean. Perforce works at your speed.

[Fast]

[Scalable]

[Distributed]

Perforce's lock on performance rests firmly on three pillars of design. A carefully keyed relational database ensures a rapid response time for small operations plus high throughput when the requests get big - millions of files big. An efficient streaming network protocol minimizes the effects of latency and maximizes the benefits of bandwidth. And an intelligent, server-centric data model keeps both the database and network performing at top speed.

It's your call. Do you want to work, or do you want to wait?

PERFORCE
SOFTWARE

Download a free copy of Perforce, no questions asked, from www.perforce.com. Free technical support is available throughout your evaluation.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

STATE OF THE INDUSTRY: MOBILE GAMES

MOBILE GAME DEVELOPERS ARE TESTING the waters with more graphics-intensive 3D games. Phone carriers are installing third-generation (3G) wireless networks capable of transmitting bigger games at higher speeds.

Meanwhile, as the industry rushes to more and more high-tech titles, the Top 10 most popular games on Verizon Wireless still include very popular—but rather low-tech—games: PAC-MAN, Ms. PAC-MAN, FROGGER, bowling, two versions of TETRIS, and two poker games.

The stronghold of so-called casual mobile games has not gone unnoticed by developers and carriers. Indeed, there seems to be a certain maturation in the mobile games sector of 2005 compared to last year, a recognition that while there is a segment of consumers that appreciates, say, a huge movie like *The Lord of the Rings* stuffed into a game that plays on a 2x2-inch screen, the majority

PAUL HYMAN was editor-in-chief of CMP Media's GamePower and currently writes a weekly column on the game industry for The Hollywood Reporter. He's covered gaming for over a dozen years. Email him at phyman@gdmag.com

STATE OF THE INDUSTRY: MOBILE GAMES



David Gosen of I-Play

of gamers seem to prefer a good three-minute round of TEXAS HOLD'EM.

Most observers say that casual games are so popular on phones because people tend to be on the go and only have minutes—not hours—to play them, and because the control pad for phone games (the keypad buttons) isn't as conducive to gameplay as the dedicated controllers used for console or PC games.

"There will always be those people who want to play console-style games and [games based on] big licenses," says David Gosen, COO of London-based mobile game developer I-Play (which changed its name from Digital Bridges earlier this year). "But we're finding that it's the

casual gaming—what we call 'one-thumb gaming'—that's taken hold of the mobile audience. We're talking about games that are challenging but easy to access. If gaming is a three-course meal, then mobile gaming is the snack. You dip in, you dip out."

He cites, for example, a game distributed by I-Play called SKIPPING STONE that will be released in this year's third quarter.

"The object of the game is to skim a stone across a pond," says Gosen, "and every time the stone touches the water, you have to press one button on the keypad to keep it bouncing. It's certainly not GRAND THEFT AUTO, but it's incredibly compelling and, whether you're a hardcore gamer or my grandma, it's going to hook you."



I-Play's MARIA SHARAPOVA TENNIS

NO LONGER AN EASY TICKET TO RIDE

Last year, smaller developers were enthusiastic about the ease with which they were able to enter the mobile market. The cost of admission was comparatively low and smaller teams were appropriate for the bite-sized games that cost between about \$300,000 and \$500,000 to build compared to the more than \$10 million it takes to build a console or PC game.

But all that's changing.

"We're not talking doom and gloom here," says Mike Yuen, director of Qualcomm's

Gaming Group, which oversees the BREW platform. "It's just that some of the cottage industry starts to go away when a sector begins to grow up, when money starts to be made, and when the space undergoes lots of merger-and-acquisition activity. Earlier, anyone could get in and build something, but no longer."

Indeed, the two mobile carriers that do the most business in terms of volume of games downloaded—Verizon Wireless (30 percent market share) and Sprint (nearly 20 percent market share)—both say they are being much more selective about their "developer partners."

A close third in the market is Cingular, with its total market in

the fourth quarter of 2004 at just over 18 million games downloaded and paid for, according to the report "The IDC U.S. Wireless Carrier 4Q '04 Datametrics Quarterly View."

Verizon Wireless has about 350 games listed on its cell phone "deck" (or menu) and is currently focusing on raising the overall quality of its games by selecting "only the branded, highest quality games," according to Alex Bloom, Verizon Wireless' director of content and programming.

"We aren't limiting the number of developers we see or the specific developers we'll work with," Bloom explains. "That said, we are urging developers to understand our catalog and to know enough not to bring us a 17th baseball game. We tend to get so many submissions of the same style of game. I mean, we already have BEJEWELED; we don't need a game called 'Animal Farm' where all the jewels have been replaced by barnyard animals."

Sprint, on the other hand, says it is consciously trimming back both the number of games it carries and the developers and publishers with whom it deals.

"We had about 250 games and we've come down a bit to about 200. And we've seen our sales go up," says Jason Ford, general manager of games and entertainment at Sprint. "There are other carriers that have 500 games, and I don't think that's a positive thing. It's not hard to do; we could have thousands of games if we put our mind to it. But what's the point? You can only show nine game titles at a time on the menu on a mobile phone screen, and how many screens do you want to force your customers to wade through? Besides, how many versions of TETRIS do they want anyway?"

Ford notes that Sprint is approached by hundreds of developers, and he suspects that, as the market matures, it won't be able to sustain that many players.

"I remember AT&T initially signed on 65 partners and quickly realized that it had 40 too many," he says. "We already had a



Jason Ford of Sprint

So real it renders fear.

Idea:

Create the most gripping and realistic stealth action game on the market.

Realized:

Ubisoft™ modeled and animated the realistic characters and backgrounds of *Tom Clancy's Splinter Cell® Chaos Theory™* with Autodesk's 3ds Max to build on one of the most popular series ever. 3ds Max's work-horse capability helped Ubisoft stay on top of their grueling production schedule and garner a 9.9 out of 10 by *Official Xbox Magazine*. To learn how Autodesk software can help you realize your ideas to compete and win, visit autodesk.com/3dsmax



STATE OF THE INDUSTRY: MOBILE GAMES



Walt Disney Internet Group's KINGDOM HEARTS is distributed through Verizon's VCast network.

few that we dropped from our direct partnership list, and we currently deal directly with 20 developers, which seems about right. I mean, if we can get 200 games from 20 developers, or 200 from 100 developers, which do you think is more advantageous for us?"

GOING TO MARKET

As the developer-carrier relationship has matured, there's been a realization that the phone companies can't continue to bear the full brunt of marketing new games, observes

Schellely Olhava, a games market analyst at research firm IDC.

"Until recently, developers never had the budget to publicize their tiny mobile games and left that to the carriers," she explains. "On the other hand, from the carrier's perspective, games are a drop in the bucket of all their revenues; they have so many other things they're trying to push—like data usage and ringtones and reminding customers that it's time to replace their handsets—that games can't be their primary focus."

But the market is growing so quickly that, for the first time, some developers are putting significant promotional funds behind their new titles.

"We just launched MARIA SHARAPOVA TENNIS in Europe and will be releasing it in the U.S. in time for the U.S. Open," says I-Play's Gosen. "We put significant promotional and advertising spending behind it."

He declined to discuss his marketing budget but says it's the result of the industry "reaching a critical mass where it now makes sense for a developer to put some serious market support into driving awareness of a game. There are now something like 35 million game-enabled handsets in the U.S. and by the end of this calendar year, there will be in

excess of 100 million. If the leading content providers don't do this, if they rely on the carriers who have so many other things to focus on, this market won't continue to grow."

Similarly, in Calabasas Hills, Calif., at THQ Wireless, the developer is integrating the marketing of its mobile games with that of the console versions created by its parent, THQ.

"So you're seeing ads for, say, the Xbox, PlayStation 2, and PC versions of DESTROY ALL HUMANS!, and each one also refers to the wireless version," says Jeff Nuzzi, director of global marketing.

THQ Wireless also partnered with carrier Cingular on a marketing

campaign for its STAR WARS mobile games, particularly in TV advertising that featured the movie character Chewbacca trying to record a cell phone ringtone.

"It was the perfect opportunity to meld our marketing campaigns," notes Nuzzi. "But this was an unusual event. The challenge is that the carriers still control the fat part of the value chain. They still own the customer in so many ways."

Sprint's Ford begs to differ.

"Do you know who gets the lion's share of the revenue from the games? The publishers. If they feel like adjusting the split by a few points, it would probably pay for a lot of advertising on our side," he says. "There are millions of dollars in gross revenues from games—although I can't be more specific about that—but we could put together a wonderful TV campaign about playing games on your phone with a piece of that."

YOU HAVE A LICENSE FOR THAT?

When gamers have a choice of 200–350 games on their cell phones, a good recognizable title does wonders to catch their eye, says David Linsalata, research analyst at IDC Mobile Devices. It's also one of the factors that carriers use when they determine how high up on the phone's deck to place a game.

"Deck placement is very, very important to the success of a mobile game," says Linsalata. "Brand names and other games that the carrier is trying to push tend to rise to the top, and that puts a lot of weight behind that game," which is one reason why developer THQ Wireless is a big believer in licensing.

"You have 18 characters on the deck to set expectations for the consumer when they're selecting games. There's no box, they can't see screenshots. The name of the game is everything," explains Nuzzi.

Seattle-based developer Reaxion, for example, realizing the importance of a mobile game title, licensed the name "The Longest Yard" from Paramount Pictures, which released a movie by the same title, for its latest mobile football game.

"If they had created an unbranded football game, their opportunity for sales would probably have been a lot less," observes Verizon Wireless' Bloom. "I mean, how many football games already exist on the handset?"

Similarly, THQ Wireless will take a license like *Star Wars* and create a whole suite of games around it: a side-scrolling action game, a puzzle game, a fighting game, a flying game, even a *Magic 8-Ball* kind of game called ASK YODA.

"We don't try to recreate the movie experience or even the console game or PC game experience," says Nuzzi. "We take a very genre-driven approach. If you're a Star Wars fan, we have



Schellely Olhava of research firm IDC



Jamdat's DOWNTOWN TEXAS HOLD'EM



Mike Yuen of Qualcomm

something that will satisfy you, regardless what sort of game genre you enjoy.”

While the big trend in 2004 was to brand everything, this year the mad rush to snatch up brands has slowed.

“Last year, if you had a branded game, you could probably get on the deck,” says Qualcomm’s Yuen, “even if it was just an average game. You’ll see that happening less and less this year and next because gamers have caught on that a licensed name doesn’t necessarily equate with great gameplay.”

At Sprint, Jason Ford says he’s cracking down; a licensed game that’s no fun just doesn’t cut it. “I don’t want to drive people to something that’s not fun,” he says. “It’s different from retail. If you buy a lousy game from Best Buy, you’re not going to hold it against the store. But, if you buy a lousy game from us [the phone service carrier], people are going to wonder why we offered it to them—and they may not want to come back and buy anything else. That’s why my primary concern is to provide a great gaming experience.”

OH WHAT GAMES WE PLAY

Providing a great gameplay experience may have less and less to do with trying to jam a big console game into a 2x2-inch screen and more with taking advantage of what a mobile phone can do that other platforms can’t.

“The big opportunity from my perspective is to recognize that the cell phone is a different device and that there ought to be games that can only be played in a mobile environment,” says Lewis Ward, senior research analyst, wireless and mobile communications, at IDC. “Perhaps a game that uses the phone’s GPS capabilities or one that uses voice or instant messaging, for example.”

“People are comfortable with using phones to interact with each other,” Ward adds, “and I suspect we’ll be seeing a move toward community-based gaming. Live head-to-head play is coming but, in the meantime, it’s more about leader boards and being the best PAC-MAN player and having a blast doing turn-

based games with your buddy, perhaps to determine who has the fastest time around a lap.”

In Sprint’s Game Lobby, for example, which is managed by La Jolla, Calif.-based M7 Networks, more than 500,000 gamers have signed up to meet, rate titles, receive notices about new games, compare high scores, and see what are the most popular downloads at Sprint.

For the first week of May, they were—as with the list of most popular games on Verizon Wireless—mostly casual games: TETRIS, FAMILY FEUD, 2FAST 2FURIOUS, Ms. PAC-MAN, SCRABBLE, PAC-MAN, GALAGA, WORLD POKER TOUR, BEJEWELLED, and JAMDAT BOWLING 2.

“These are the kinds of games that the gaming community seems to enjoy since the greatest number of players considers themselves to be casual gamers,” says Sprint’s Ford. “But what’s funny is that, while they call themselves ‘casual gamers,’ they’re playing for hours upon hours. When someone plays 14 hours a week, is that a casual gamer? I think we need to coin a term for people who are playing these simple games but are hooked on them and are playing them for insane amounts of time.”



Blue Lava/Jamdat’s TETRIS



THQ Wireless’ NBA ALL-STAR 3 POINT SHOOTOUT

LIFESTUDIO:HEAD® 2.7 SDK

MULTIPLATFORM SDK SOLUTION NOW AVAILABLE!
LEADING FACIAL ANIMATION TECHNOLOGY AT THE CUTTING EDGE!

To find out more about our licensing policy, please contact sales@lifemi.com.

- Run-time control using API
- Automated lip-synch
- SDK supports PC PlayStation® 2 and Xbox®

Lifemode Interactive
www.facialanimation.com
www.lifemi.com

*PlayStation® is a registered trademark of Sony Computer Entertainment Inc.
 *Xbox® is a registered trademark of Microsoft Corp.
 *LifeStudio:HEAD® is a registered trademark of Lifemode Interactive Ltd



THQ's *Ask Yoda* plays much like *Magic-8 Ball*.

However, Verizon Wireless' Bloom says the jury is still out on multiplayer gaming. "We find that multiplayer does pretty well, but perhaps not as well as we might have expected given the community aspect of the device. Part of the reason, I suspect, is the user interface; it's sometimes difficult on a mobile device to both play the game and communicate by text or chat with other players. There are just so many buttons you can press at the same time."

HERE COMES 3D GAMING

With so many mobile gamers playing classic and casual 2D games, one might ask why a carrier like Verizon installed a faster—and more expensive—3G network to handle the 3D games that are typically 1.5MB in size compared to their few hundred kilobyte 2D brethren.

Sprint, which doesn't have a 3G line, wonders the same thing. "It's not yet clear whether people will be willing to pay extra or buy more games just because the graphics are 3D—and how much more?" Ford asks. "The challenge to the carrier is to make

money on them since the games cost three times more to build than a 2D game. Will the carrier get three times as much in sales or charge three times as much? I think that's what everyone is struggling with right now."

But officials at Verizon say the company is "well ahead of our internal forecast" on its VCast 3D games, which can only be downloaded by a VCast phone from its new EV-DO 3G broadband network. The carrier is charging customers a monthly flat rate of \$15 just to access the network; the cost of the game is extra.

"We're not putting out any figures on VCast yet," says Verizon's Bloom, "because the service only started Feb. 1. Besides, we're pretty famous for not putting out a lot of metrics."

According to Verizon's web site, the hottest 3D games include *S.W.A.T.: The Movie 3D Game*, *Tony Hawk's Pro Skater: 3D Mobile Edition*, *Ghost Recon Jungle Storm*, *Final Fantasy VII Snowboarding*, *The Incredibles 3D*, and *Jamdat Bowling 3D*.

As for the future of 3D mobile games, IDC's Ward says, "they are a big part of where things are going," but, at the moment, "they are only a small part of the current market"—which is why today's developers are concentrating mainly on 2D.

"We need to stay focused on the core 2D market today," says I-Play's Gosen. "Gamers don't have 3D and we need to grow the market with what our customers have today while still laying the

foundation for 3D tomorrow." And so, of the 30 to 35 titles that I-Play builds annually, only 15 to 20 percent are 3D.

Similarly, at THQ Wireless, Nuzzi predicts that it will be another 12 to 18 months before 3D games are at the "top of the deck." He adds, "We're still in the early-adopter stage now and those devices that can support today's 3D games aren't out there yet." He says that THQ is currently working on several 3D games but they have not yet been announced.

WHO REALLY CARES?

Suddenly, big game publishers are starting to care about mobile. Perhaps it's because they've seen some of the metrics. According to In-Stat/MDR, revenue from mobile gaming in the U.S. is expected to increase from \$91.3 million in 2003 to approximately \$1.8 billion, or 4.4 percent of all wireless data revenue, by 2009.

It's no wonder that, this year, Electronic Arts announced that it intends to go mobile.

"When we talked to EA in 2001, they said they thought they'd wait out the first round," reports Qualcomm's Yuen. "But now, they think the time is right. It will be interesting to see what effect that has on the few holdouts, like Take-Two and Vivendi, and the big console guys like Sony, Microsoft, and Nintendo."

The more interesting question: Exactly how important is mobile gaming to the phone carriers, especially when games reportedly produce only 2 or 3 percent of their overall average rate per user?

"I hate to say this, but I don't think gaming is as sexy as it once was," says Sprint's Ford. "Maybe that's because games have been around for awhile; this is year three for Java applications on the main carriers. So customers are starting to look at other things that are sexier—like video clips. Games are a great revenue stream for us, but they aren't necessarily the cutting edge anymore."

According to Ford, the top three reasons why a customer chooses Sprint service are price, coverage, and handsets. "Games don't come anywhere near those three," he adds, "and that's why our advertising emphasizes those three areas. All the carriers try to differentiate themselves to acquire new customers and most everybody has games. I mean, who doesn't have *TETRIS*? So it's a real challenge for us to tell the public that they need to come to Sprint because we've got *TETRIS*. But we do have exclusives on certain ringtones and streaming music videos; those are the kind of things that show that we're innovative."

But if games don't lure customers to mobile carriers, they're often what make the customers stick around.

"They come to us because they want the coolest technology which might be the latest video ringtones, but then we're also able to give them that extra frosting on the cake—which is the best games in the industry," notes Ford. "And maybe we're even able to keep them here longer because, heck, they buy a few games and get hooked on them. That's what makes the games so important to us." ❖



Bandai's *3D Field Golf*

A drop of liquid

is calculated for wind velocity, adjusted for gravity and accelerated as it nears the ground below. It hits, splatters and droplets travel in multiple directions. Repeat a thousand times.

[precisely calculated while your battle rages in the foreground.]




ageia
Make it real.

AGEIA's physX™ SDK provides the most powerful and advanced toolset for creating dynamic physical environments. With unmatched support for collision detection, fluids and advanced multi-threading for tomorrow's massively parallel architectures, your games will feature the next level in immersive game play. AGEIA's physX technology has you covered for today's software physics effects and tomorrow's game play with powerful hardware-based physics. Learn why more than 60 game development studios use AGEIA's physX technology.

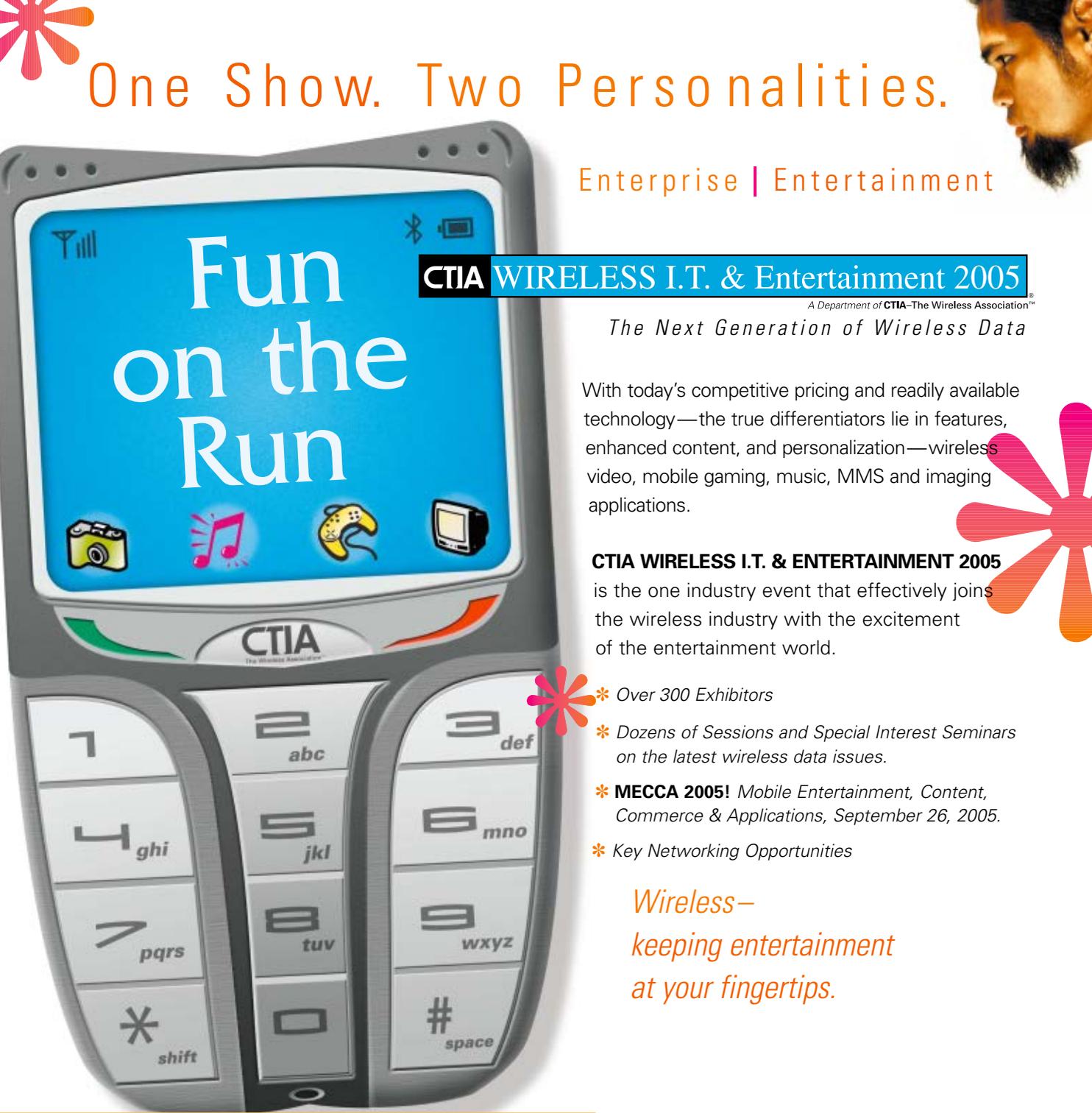
Visit us at www.ageia.com to learn more.



All logos and trademarks are the property of their respective owner

One Show. Two Personalities.

Enterprise | Entertainment



CTIA WIRELESS I.T. & Entertainment 2005

A Department of CTIA-The Wireless Association®

The Next Generation of Wireless Data

With today's competitive pricing and readily available technology—the true differentiators lie in features, enhanced content, and personalization—wireless video, mobile gaming, music, MMS and imaging applications.

CTIA WIRELESS I.T. & ENTERTAINMENT 2005

is the one industry event that effectively joins the wireless industry with the excitement of the entertainment world.

- * Over 300 Exhibitors
- * Dozens of Sessions and Special Interest Seminars on the latest wireless data issues.
- * **MECCA 2005!** Mobile Entertainment, Content, Commerce & Applications, September 26, 2005.
- * Key Networking Opportunities

*Wireless—
keeping entertainment
at your fingertips.*

KEYNOTES



Keynote Host:
Steve Largent
President/CEO
CTIA-The Wireless
Association™



Edgar Bronfman, Jr.
Chairman & CEO
Warner Music Group



Suzan DelBene
Corporate Vice President
of Marketing, Mobile and
Embedded Devices Division
Microsoft Corp.



Rob Glaser
Chairman & CEO
RealNetworks, Inc.



Joel Katz
Co-Managing Shareholder,
Atlanta Office Chair
Greenberg Traurig, LLP



Sean Maloney
Executive Vice President,
General Manager,
Mobility Group
Intel Corporation

*Stay tuned for more exciting
Keynote announcements...*

September 27–29, 2005

Moscone West | San Francisco, CA

www.ctia.org

Register today @ www.ctia.org



MIKE YUEN

» BUSINESS LEVEL

THE TIPPING POINT

THE CONVERGENCE OF **WIRELESS** AND **CONSOLE/PC** GAME DESIGN

MOBILE PHONES HAVE COME A LONG

way. Gone but not forgotten are the early handsets that were heavy, cumbersome, and shaped like bricks. While the external form factor and industrial design of phones have certainly improved rapidly, the underlying mobile phone functionality has increased with an exponential progression in performance capabilities.

The worldwide prospects for mobile games are equally impressive. Emerging markets such as China and India are just beginning to explore opportunities in wireless gaming, both as enormous markets of potential mobile gamers and as hotbeds of game development.

UNTAPPED POTENTIAL

With the potential to reach so many subscribers worldwide, industry players within the mobile gaming value chain are faced with a multitude of opportunities. This is especially true for wireless game publishers and developers. It's clear that the mobile game industry is set to explode, but what many in the traditional game industry currently fail to realize is the massive untapped

potential for integrated crossover game designs that can drive the expansion of traditional gaming experiences to new levels of innovation.

Current wireless game designs are isolated islands of content and design that offer low levels of interaction with other traditional gaming platforms. This has led to limited exploitation of cross-marketing and promotional opportunities and a failure to explore innovative designs that extend the gaming worlds of the traditional PC and console experience. As the wireless gaming industry continues to grow and the underlying platform capabilities rapidly increase, the opportunity to leverage the unique features of the mobile phone to create a new gaming experience will provide the industry with a rich palette of innovative design possibilities as well as new recurring digital revenue streams.

THE TIPPING POINT IN WIRELESS GAMING

Although less than 30 years old, the game industry has exploded as an entertainment medium and vehicle for reaching an ever-expanding audience. Throughout its brief history, the gaming industry has witnessed several key moments in software design that have propelled it to a higher level of success and exposure. Seen as quintessential

"tipping points," the creation of four major games—PAC-MAN, SUPER MARIO BROS., TETRIS, and DOOM—serve as game industry milestones that not only fueled the creativity of millions of game developers worldwide, but led to the success of their respective hardware platforms.

With the confluence of the increase in hardware capabilities, high-speed data network access, and the growth of the wireless user base, some are arguing that the mobile phone may very well represent the next platform to enable a tipping point in game design. As a game platform that has built-in wireless voice and data connectivity, inherent mobility, location based services, and an always-on presence, the mobile phone is poised to drive design innovation within the gaming industry on an unforeseen level.

Cross platform design represents a new way of innovating, driving the simultaneous development of mutually supported features across different platforms (game console, PC, and wireless). As a glue that could potentially bind game platforms and gaming worlds, the wireless phone could enable the next tipping point in game design, resulting in new, recurring revenue streams and expanding the reach of gaming as a form of entertainment. The tipping point in wireless gaming would usher in a

MIKE YUEN is director of Qualcomm's Gaming Group, which oversees the BREW platform. Email him at myuen@qmag.com.



Callaway[®]
GOLF



Callaway Golf Mobile Tour

Superscape/GWE 2D & 3D

Play like a pro on the real TROON course, earn the right to play with high performance Callaway Golf clubs. If you're good enough take on real players in a real 4 course tournament... Let's see if you've got what it takes to be the best and top the leaderboard!



Take Your Game...
Anywhere.

Callaway Tour features the following TROON Courses:

- CONCORD
- LEXINGTON
- PINNACLE
- MONUMENT



TROON GOLF[®]



GLOBAL
WIRELESS
ENTERTAINMENT



published by
SUPERSCAPE

These mobile applications have been produced with Callaway Golf by Global Wireless Entertainment, Inc., and published by eMobiency, Inc. and Superscape, Inc. Callaway Golf, the Chevron device, Big Bertha, Heavenwood, i TRAX, Mobile Caddie and Callaway Golf Mobile Tour are registered trademarks and trademarks of the Callaway Golf Company. Global Wireless Entertainment 3148 Midway Drive, Suite 200 San Diego, CA 92110 is an Official Licensee of Callaway Golf Company.

© 2005 Global Wireless Entertainment. All rights reserved.

BUSINESS LEVEL

whole new approach to game design that unifies multiple platforms by leveraging the business and creative opportunities that convergence brings.

Tomorrow's wireless game designs will bridge various platforms with wireless capabilities. Today's high-speed connectivity combined with the distinctive aspects of mobility provides players with a unique extension of platform game worlds, integrating content and gameplay across platforms rather than merely branding wireless games.

DESIGN EFFECTS

A successful cross platform game design will likely utilize one or more of the following design effects.

Mobile DNA effect. If a game design has a mobile DNA effect, it means that the design makes use of the fact that each mobile phone has a unique phone number.

Trailer effect. A trailer effect means that a mobile game generates future excitement and incentive about a console or PC version of the same game. Using the mobile game as a teaser for and constant reminder of the primary game, players are excited for daily, weekly, and monthly play at home. The concept is similar to how a movie trailer entices people to see a new movie or how a television commercial tries to get people to watch upcoming weekly programs or special shows. Persistent worlds that exist in massively multiplayer online environments are perfect for this type of effect.

Season premiere effect. A game design that serializes monthly content culminating with the annual launch of the next major console/PC release of a franchise could be said to have a season premiere effect. A mobile version can be used to help build momentum for the launch of the next major release of a traditional title.

Frequent flyer effect. A game design that has a supporting and contextually relevant affinity/loyalty points program can serve to lock in user commitment and increase switching costs. This affinity program would also integrate across a publisher's traditional titles, web site, and marketing and merchandising programs. Users would be allowed to redeem points for discounted or complete purchases of additional games and content.

Amazon associates effect. A game design that increases viral word of mouth and sales via super distribution might be said to display

the Amazon associates effect. An example of this might be the ability for users to sell mobile games and content while getting a revenue share or other attending benefits of actual sales generated due to their individual efforts.

Reciprocation effect. A game design that uses two-way integration to strengthen brand and game loyalty could be said to show a reciprocation effect. A user can win benefits in a wireless game that can be used in the console/PC version—he or she can unlock unique weapons, special characters, moves, modes, levels, rooms, or currency only by purchasing, playing, and advancing within the wireless version. Similarly, a player can win benefits in the console/PC game that can be used on the phone. They can win wallpapers, screensavers, ring tones, UI skins/themes, and so forth, only by purchasing, playing, and advancing within the console/PC version.

Games that utilize any of the above design effects will certainly have to have strong designs in all of the various platform versions so that gameplay is balanced and fair.

With the global wireless market expected to add an average of 186 million new subscribers each year, resulting in a total population of more than 2 billion by 2007 (according to In-Stat/MDR), the gaming industry is poised for continued explosive growth if it can find a way to successfully tap into and leverage this installed base. Cross platform design with the mobile phone acting as the centerpiece is an answer.

PURPOSEFUL TIPPING

The next tipping point in gaming will not happen by accident. It will stem from the creativity of a handful of pioneering game publishers and developers eager to embrace and usher in the next generation of game design and innovation.

With a potential wireless installed base 20 times the size of the user base of the original PlayStation, who knows how far cross platform design in gaming would drive the industry forward. Furthermore, success with a true cross-platform design would be measured by not only a pure revenue-generation standpoint, but also by an advancement in game design. Achieving this tipping point can drive creativity and innovation within the industry, resulting in a whole new generation of games for consumers to experience and enjoy. ❖

-  EDUCATION
-  GOVERNMENT
-  HEALTH
-  MILITARY
-  CORPORATE
-  FIRST RESPONDERS
-  SCIENCE

SERIOUS GAMES SUMMIT

WASHINGTON D.C.
OCT 31-NOV 1, 2005 **D.C.**

HAZMAT: HORIZON - Entertainment Technology
Center at Carnegie Mellon University.

REGISTER BY
SEPTEMBER 21, 2005 AT
WWW.SERIOUSGAMESSUMMIT.COM
AND SAVE \$200 ON YOUR PASS

Use Priority Code PEMAXW when registering.

○ THE SERIOUS GAMES SUMMIT D.C. IS THE ONLY EVENT that shows developers and project administrators how non-entertainment games are creating new opportunities.

- Discover cost-effective, customizable, and engaging teaching methods
- Apply current skills to tap into new business and revenue streams
- Make valuable industry contacts and catch up with peers

**DON'T MISS THE ONLY EVENT DEDICATED TO SERIOUS GAMES!
REGISTER TODAY AT SERIOUSGAMESSUMMIT.COM**

game developer

DIGITAL EDITION

THE GAME INDUSTRY'S LEADING SOURCE OF INFORMATION
FOR DEVELOPING NEXT GENERATION GAMES



**PAY 70% LESS THAN
INTERNATIONAL PRINT
SUBSCRIPTION**

Game Developer: Digital Edition features all of the timely game industry news, articles, and reviews found in Game Developer magazine, delivered in a faster, more convenient digital package.

Download archived editions and access the newest content the moment it becomes available.

**SUBSCRIBE TODAY AT
WWW.GDMAG.COM/DIGITAL**

MOBILIZING CONTENT

Porting games for mobile devices

» THE BUZZ AROUND MOBILE GAMING AT THIS YEAR'S GAME

Developers Conference and E3 was loud for a reason: mobile phone gaming has arrived. There are now more than 300 million game-capable phones worldwide with mobile game revenues estimated at \$2.6 billion this year and growing. Game publishers are flocking to this profitable space.

Compared to traditional video games, however, mobile games present some daunting development challenges. Mobile phones support games written in multiple environments, including BREW and J2ME. Wireless carriers add more complications by imposing their own set of standards and requirements. While there are less than 10 current platforms for traditional entertainment software, phone-game developers must support hundreds of different handsets.

If you factor in the 100 or so carriers worldwide all clamoring for great content, there are now myriad combinations of underlying technologies, network interfaces, button layouts, screen

resolutions, and sound requirements across all the possible platforms for a mobile game. It's no wonder that porting a game to the majority of devices available today is a difficult, time-consuming, and expensive task. Mobile game publishers need to have flexibility, creativity, and experience in order to succeed.

SO MANY PHONES, SO LITTLE TIME

Device fragmentation remains the biggest challenge in the mobile space, which is what makes porting these games to the devices so difficult. To have exceptional porting capabilities is crucial. The rapid improvement of mobile phones in terms of graphics, memory capacity, and screen size has created a wide gap in performance between the newest phones and older, less expensive ones. The challenge for a mobile game developer is to create a title that will utilize the features of a high-end phone while still providing a compelling experience to the user of a low-end phone.

Screen sizes vary wildly, ranging from small (96x65 pixels) to

MICHAEL YING, JACOB ABRAMS, VIKAS GUPTA, BOB WHITEMAN, and KAL IYER are senior members of game publisher Glu Mobile's technical team. You can contact Michael and his co-authors via mying@gdmag.com.

MOBILIZING CONTENT



FIGURE 1 It's important to be able to adjust a game's art according to each phone's screen size. For example, the AQUA TEEN HUNGER FORCE mobile phone game is adaptable for different screen sizes: 176x220 pixels (left) and 128x144 pixels (right).

240x320 pixels. Memory heap restrictions can be as generous as 6MB or as constrained as 200K. Even binary size can be restricted, with some low-end handsets only allowing a maximum of 64K for code and resources. Each manufacturer uses a different implementation of J2ME or the BREW API, introducing the possibility of non-standard behavior from each device. Developers must discover and account for each of these idiosyncrasies.

The only way a mobile game publisher can ensure a successful revenue stream is by targeting a large number of different devices. The best handsets in terms of capability are usually more expensive and, therefore, have low sales volume, meaning there aren't many of them in use. By far, the most popular handsets on the market are those that carriers offer at heavily discounted prices or for free. These tend to be older handsets with less than stellar performance. But because they have such a high installed base, these low-grade phones represent a huge chunk of potential revenue. To support multiple handsets, languages, and carriers, publishers must produce an incredible number of versions of each game. Glu Mobile, for instance, has released more than 300 versions of Atari's DRIV3R, targeting a wide variety of languages and carriers worldwide.

SURMOUNTING SPECIFIC PROBLEMS

When designing games for mobile phones, the graphics must be flexible and modular enough to accommodate minute differences in each phone's variables, like screen size, aspect ratio, and orientation. Glu handles this challenge of accommodating variables by organizing a game's art into one of three categories: global, core, or resolution-specific.

Global art constitutes the most general pieces, such as small widgets, icons, and other art that remains the same across all SKUs. Core art contains game elements like sprites and tiles, and generally requires about three sets to support the full range of screen resolutions. Resolution-specific art consists of full screen art, which, while visually compelling, is time-consuming

to resize for each different screen. Our company also organizes art into modular art sets, and avoids the use of full screen art. Even elements that appear to be full screen art can use modular design to minimize art changes between handsets.

A porting team needs to be flexible and competent enough to rewrite code and create new layouts to deal with both landscape and portrait oriented screens. User interface elements such as heads up displays must be modular enough to display well at the bottom of a portrait screen and on the right side of a landscape screen.

For Cartoon Network's game AQUA TEEN HUNGER FORCE, Glu designed its graphics to be as modular and flexible as possible. One binary accommodates both screen sizes, shown in Figure 1. The bottom heads up display comprises images for the speed meter and the angle meter. As varying screen sizes change (that is, widen or narrow), the code pads the heads up display with more black space in between the two elements. The heads up display uses system draw calls to add the red color, indicating speed as well as the current angle. A side-scrolling game like AQUA TEEN HUNGER FORCE animates large background images in one direction or another to indicate movement. Different devices share the same background art sets without any compromise in game quality. By composing the display and other parts of the game with screen-agnostic drawing, we can target many different devices with no code or art changes. We targeted all devices for AQUA TEEN HUNGER FORCE with only three sets of core art and zero full-screen assets.

LOWER YOUR HEAP

Mobile game developers face a difficult challenge in restricted heap sizes. Low-end handsets may have as little as 200K of heap. Developers can tailor a game for these heap-constrained handsets by cutting features. Another way to pare a game's heap requirements down to fit on a low-end handset is to reduce the bit depth of images and the number of frames in



FIGURE 2 These graffiti airbrush animation frames are a good example of using palette swapping for animated effects without needing many different animated frames.

animated sprites. This technique lessens the burden of images that are loaded in memory; careful memory management ensures that images are dynamically loaded in and out of the heap as needed.

Heap limitation is not the only hardware-based variable. In fact, developers must account for large variance in graphical and math processing power, too. Some handsets perform poorly in math, in drawing to the screen, or in some cases, perform poorly in both aspects. Developers must minimize the math and drawing requirements of their application. Code must repaint sprites constantly as they move across the screen, while background tiles or heads up displays need repainting less often. In particularly graphic-intensive applications, developers can use damage rectangles to indicate the areas of the screen that need to be repainted. For some BREW devices, Glu engineers reduce drawing by painting to the screen only once every two ticks.

In addition, the type of image drawn to a BREW device's screen can be very important. When dealing with particularly slow phones, images using transparency cause slowdown because of the processing that's required to paint the transparent areas of the image to the screen. By replacing transparent elements of a bitmap image with colors that blend into the background, BREW developers significantly speed up the drawing operation without affecting the look and feel of the game.

Some J2ME devices restrict the size of Java binaries to 64K or less. Given the growing complexity of code and art in mobile games today, binary size restrictions require that developers do everything possible to save space. Glu approaches these problems from several angles: the design and creation of art assets themselves, storage of art and data assets, and finally, delivery of those assets.

By designing art thoughtfully, artists can help developers minimize the amount of space devoted to art while still maintaining vivid images and visual effects. Indexed images utilize a color table, or palette, to determine what colors to use in a given image. By swapping color palettes of an image set, a developer can add a new color scheme for a set of sprites with between 30 and 50 bytes of additional data. With palette swapping, developers add polish to games with minimal increase in binary size.

But don't mistake palette swapping for a mere color changing effect. Through ingenious use of palette swapping, game art transforms from flat backgrounds to living animation. Glu's mobile version of MARC ECKO'S GETTING UP uses graffiti art, which appears on screen with an "airbrush" animated effect (see Figure 2). Glu's art team reduced the art assets required for the game by 60 percent when using palette swapping instead of separate images.

When creating images for palette swapping, an artist creates each version of an image with the same size palette. The order of colors in the palette is essential to making sure

the right colors change when switching from palette to palette. A developer keeps one image intact as originally created and extracts the raw palette data for each of the remaining images into binary data files. These extracted palettes take up less than 50 bytes of space. Code should read the data as raw binary data, allowing a developer to read in a separate palette file and overwrite the existing palette on the image. This Java code snippet creates a portable network graphics (PNG) image object using a separate palette (see Listing 1, page 26).

Approximately 200 bytes of overhead are required for each individual file added to a Java Archive (or JAR), used as the binary format for J2ME. With art and data assets numbering into the hundreds, 200 bytes of overhead per file can balloon into an unacceptable amount of wasted space. To avoid this problem, developers can pack smaller art and data assets into a larger binary file. This larger file incurs the penalty of 200 bytes only once, saving valuable space.

Glu focuses on compressing files more efficiently as well; J2ME applications use PNG images. We use open source tools like pngcrush to strip extraneous information from each image and further compress the image data. Because PNG supports several types of compression, Glu developed tools that compress an image using whatever compression method yields the smallest image.

After exhausting these two useful techniques, a developer may still need an extra few kilobytes of space to fit in a killer feature for a game. For this situation, Glu uses resource downloads to reduce the size of the binary even further. At first run, a Glu game that uses resource downloads will contact Glu's servers to get any resources that it does not already have in the JAR and save them in device storage. For the puzzle game ZUMA, Glu's code binary size reached nearly 60K. However, because resource downloading was available, ZUMA's quality on 64K was comparable to builds created for devices with unlimited binary size.

Device-specific bugs and idiosyncrasies add even more complexity to the life of a mobile game developer. Even the same device with different firmware can exhibit different behavior. Developers use animated sprites and tile sets stored as a strip of images side by side. A 10x10 pixel sprite with 20 frames of animation would be stored in a 10x200 pixel strip. However, on older Sony Ericsson T616 phones, loading images larger than the screen resolution of 128x127 pixels will cause a game to crash. Later firmware versions of the T616 improved slightly, allowing images up to 256 pixels square before crashing. Engineers targeting those older phones rearrange or split these strips to prevent crashes on all T616 phones.

J2ME implementations vary from phone to phone, and the

virtual machine running on a device dictates the use of image strips or individual images for each frame of animation. When drawing image strips to a screen, J2ME code uses clipping to select the right image to paint to the screen. Through profiling and other benchmarking, Glu engineers discovered that some devices painted very slowly when clipping paint operations. By splitting images into their individual frames, the frames per second drawing performance improved by 50 percent on those devices.

CLASSING MOBILES BY TYPE

Experience is invaluable when working with the staggering number of devices present in the mobile phone landscape. Developers acquire this experience through close examination of the capabilities of each device. Glu maintains a large database of information for each handset it supports. The database tracks idiosyncratic behavior, performance, and other attributes of the handsets. As manufacturers release handsets, engineers must test and exercise each one to learn this information.

With each new port, the existing knowledge base improves the porting process. Though the number of devices available on the market is growing quickly, similar devices can be grouped together to increase efficiency of porting. By creating families of devices and tailoring game code to adapt to small differences in devices, Glu produces a number of SKUs with a smaller effort, increasing efficiency of porting without sacrificing quality. Glu groups these families based on any number of attributes—screen size, heap size, manufacturer, or device bugs. Carriers also favor this style of binary consolidation because it eases administration of submitted titles.

Ideally, a game should work just as well on a low-end phone as on a high-end phone, but that's not always possible. Developers can prioritize a list of core features that are essential to a game

and use that to decide which of them, if any, to remove so that a game will work on low-end handsets.

Experience aids this process, and seasoned mobile game developers know what features will work best on specific handsets. The longer you work on mobile games, the more intuition you gain for knowing which features to add or strip from a given SKU. A good philosophy to follow when adapting different versions of a game is to maintain the same core gameplay for all phones, adding (or subtracting) polish—not integral elements.

When porting Atari's DRIV3R, Glu made the design decision to reduce the number of cities and vehicles for the low-end version rather than sacrifice the animation and feel of the driving aspect of the game. The high-end version includes three cities and five vehicles, whereas the low-end version only includes one city and three vehicles. The number of levels in the game stayed the same, and the story elements needed minimal changes between the high and low-end.

PHONE-SPECIFIC EDITIONS

Manufacturer relationships are just as important as carrier relationships. By taking special advantage of manufacturer APIs, a publisher can create showcase titles for a given device. The Kyocera KX2 phone is a unique folding device with a rocker button. Users can still see the screen and navigate it when it's fully closed. Glu developed a special version of FIVE-CARD DRAW POKER that allows full control of the game while the phone is closed.

With Nokia's 3220 handset, Glu developed a version of Hasbro's popular game, *Simon*, that used the unusual LEDs on the handset to present a gameplay experience much like the original game.

While these enhancements may affect sales only minimally, they can indirectly benefit the publisher. By showing the willingness to create showcase titles in a short timeframe, publishers gain the trust and favor of both carriers and manufacturers. As a benefit of its close working relationship with handset manufacturers, Glu has access to prototype phones, giving them the ability to port titles to these handsets prior to launch and the opportunity to preload games onto phones because of good carrier and manufacturer relationships.

GROWTH AHEAD

While developing and publishing games for phones is riddled with complexities and nuances completely unique to the platform, the market for mobile games is enormous, with revenues in the U.S. alone expected to grow to \$1.5 billion by 2008. Publishers that focus on flexible game designs and creative tools make porting a relatively painless and efficient process. With experienced teams, good relationships with carriers (and plenty of caffeine), mobile game publishers can thrive and succeed. ❖

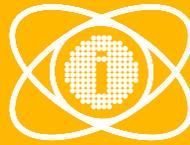
LISTING 1

```
byte[] raw;
byte[] png;
byte[] plt;

// read in image data as a raw byte array
// read in palette data as a raw byte array
raw = <read raw image data>;
plt = <read raw palette data>;

// Grab original PNG and new palette, then create a new image
png = new byte[raw.Length];
System.arraycopy(raw, 0, png, 0, png.Length);
System.arraycopy(plt, 0, png, 41, plt.Length);

Image img = Image.createImage(png, 0, png.Length);
```



INDEPENDENT
GAMES FESTIVAL

REWARDING INNOVATION IN INDEPENDENT GAMES

THE 8TH ANNUAL

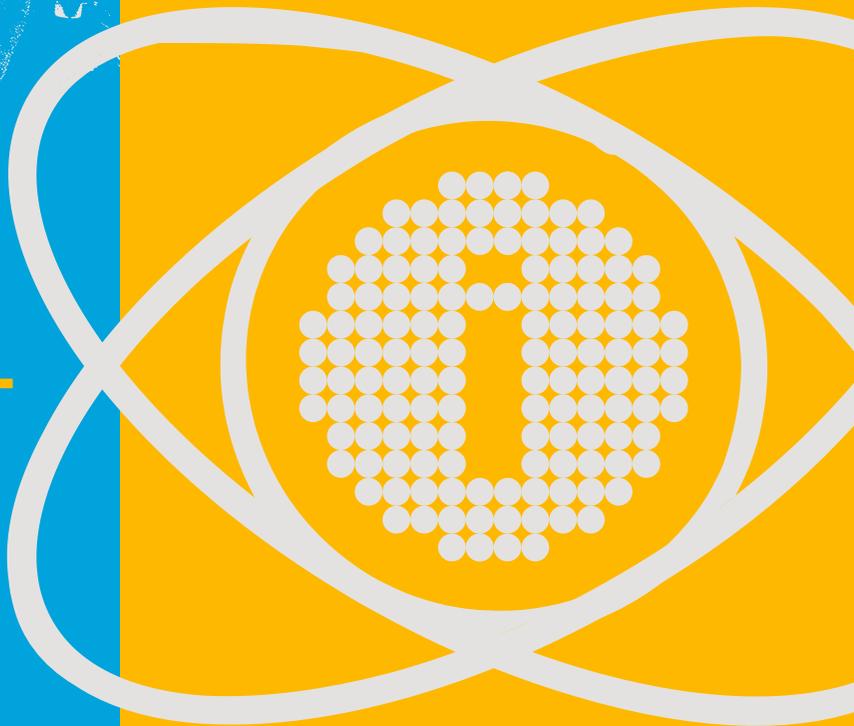
INDEPENDENT GAMES FESTIVAL

MARCH 20-24, 2006

SAN JOSE, CA

WWW.IGF.COM

The stage is being set for the eighth annual Independent Games Festival—the most respected event celebrating the indie game developer. Here's your chance to see if you have what it takes to bring home an award in 2006!



OPEN CALL FOR ENTRIES

IGF COMPETITION

DEADLINE FOR SUBMISSIONS: SEPTEMBER 6, 2005

Open to all independent game development teams, the IGF Competition offers over \$35,000 in cash and prizes.

IGF MODDING COMPETITION

DEADLINE FOR SUBMISSIONS: OCTOBER, 2005

New Competition! Non-commercial mod teams compete in one of four categories for a chance to take home a \$2500 cash prize.

IGF STUDENT COMPETITION

DEADLINE FOR SUBMISSIONS: NOVEMBER 15, 2005

A competition for games developed by full-time students.

FOR INFORMATION ON CATEGORY DESCRIPTIONS,
ENTRY FORMS, AND ELIGIBILITY RULES GO TO

WWW.IGF.COM



CMP

United Business Media

MOBILE POSTMORTEM

BEFORE CRISIS:

FINAL FANTASY VII

KOSEI ITO joined Square Co., Ltd. in 1997. After stints in the marketing and online departments, he started up Square's first mobile business in 2002, when he oversaw the management and production of content for mobile phones. Now, as vice president and producer of mobile products at Square Enix, he is in charge of North American mobile business. Email him at kito@gdmag.com.

THE DEVELOPMENT OF BEFORE

CRISIS: FINAL FANTASY VII started almost three years ago at Square Enix Japan as part of a wide ranging group of software called the COMPILATION OF FINAL FANTASY VII, based on the best-selling 1998 PlayStation role-playing game. These new games, all tied into the same storyline, include a number of titles across multiple platforms, including consoles, mobile, and a computer-generated movie.

Late one night, Tetsuya Nomura, concept and character designer for BEFORE CRISIS, was in the development room and wondered, "Can we make an action RPG that utilizes the mobile phone network?" (Nomura also designed specific characters from FINAL FANTASY VII, VIII, X, and X-2, and directed the KINGDOM HEARTS series and the *Final Fantasy VII: Advent Children* movie.)

When BEFORE CRISIS was in its infancy, initial discussions revolved around one basic question: "What kind of game would be fun to play on a mobile phone?" First, we had to decide to make it an action RPG, and only then did Nomura come up with a story concept that would utilize the worldview of FINAL FANTASY VII, but set it six years prior to the original and make the protagonists the Turks, the enemy in FINAL FANTASY VII.

After establishing the game world, Hajime Tabata, who joined later as the game's director, set a clear direction for the game design. Tabata wanted it to incorporate the cool nature of the Turks as an elite intelligence unit. This was the initial base of development for BEFORE CRISIS.

The game wound up integrating all of these concepts, becoming a network-based action RPG, the first of its kind, developed exclusively for mobile phones. We aimed to utilize the unique potential of mobile handsets, creating an experience



GAME DATA



BEFORE CRISIS FINAL FANTASY VII

RELEASE DATE

September 2004 (Japan)
2006 (North America)

PUBLISHER

Square Enix, Co., Ltd.

DEVELOPER

Square Enix, Co., Ltd.

PLATFORMS

NTT DoCoMo FOMA 900 series

NUMBER OF DEVELOPERS

18 (core team)
15 (contractors)

DEVELOPMENT TIME

12 months



POSTMORTEM



BEFORE CRISIS: FINAL FANTASY VII uses prerendered backgrounds, to mimic the feel of the PlayStation original.



In-game battles are fought in real time.

not possible through any other medium, by using the phone's camera, network capabilities, and portability to its advantage.

After approximately a year of development by 11 staff members of a mobile team based in Square Enix's Tokyo offices, at the end of August 2004, the beta version of *BEFORE CRISIS* was revealed. The game may very well have been the first mobile phone content to ever undergo a beta test, and it was received with wild popularity—1.6 million accesses on the first day. Proper service started in Japan in September 2004 for NTT DoCoMo's 3G FOMA 900 series of mobile phones, and continues to this day with strong user support and several major upgrades.

Although the mobile development team did most of the work on *BEFORE CRISIS*, many original *FINAL FANTASY VII* staff members helped to supervise the project. Even though mobile phone hardware specs at the time were nowhere near those of the PlayStation, much care went into the production so that we did not lose the overall *FINAL FANTASY VII* worldview while using limited technology.

Since we were working on a mobile application project, there were no special tools in terms of the development environment. If we had to name one thing that was special, it would be the "pre-rendering" technique—the same we used for the original *FINAL FANTASY VII*—that was applied to all the graphics in order to recreate some of the look and feel of the original. For *BEFORE CRISIS*, we created 3D graphics in Maya, then converted these into 2D. Although this adds an extra step in development, the result allows the user to somewhat ignore the limitations of mobile phone graphics.

WHAT WENT RIGHT

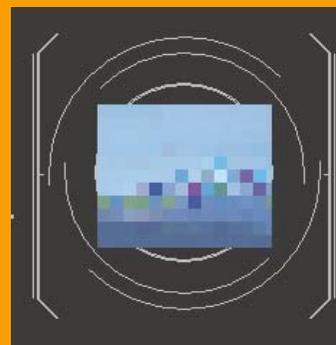
1 INTEGRATING CONSOLE AND MOBILE USER BASES. It's clear that the mobile phone is constantly evolving as a game platform. But when compared to game consoles, phones still have a long way to go before they will be recognized as a viable platform by hardcore consumers. On the other hand, in terms of dissemination, phones win over consoles hands down. That's why we integrated the game into our *COMPILATION OF FINAL FANTASY VII* plan: the platform actually presented a chance to unite diverse players.

We have the core gamers who would be drawn to *BEFORE CRISIS* simply by virtue of its being related to *FINAL FANTASY VII*. The new game gave us an excellent chance to pique the interest of hardcore gamers (who tend to think of mobile games as trivial) and potentially introduce them to a new platform. However, there are more casual consumers who own a mobile phone but not a game console. Let's say these users, previously totally unaccounted for by the game industry, tried *BEFORE CRISIS* just to kill time. If they are interested and enjoy the game, we may be able to persuade these users to take an interest in console games because *BEFORE CRISIS* extends and precedes a popular console title.

Most important in this rollout of multiple *FINAL FANTASY VII* titles is the fact these games are not the same across all of the platforms. Although they share a common worldview, they are all essentially different games. Each part of the *Compilation of FINAL FANTASY VII* takes advantage of the hardware for which it is

designed. In other words, the quality of each game (or movie) is maximized for its respective hardware.

Due to this differentiation, we needed to delve deep into the possibilities of mobile game styles to make sure that *BEFORE CRISIS* would take full advantage of every major feature of its platform hardware. Even though the phone is a device with limited specifications, we were able to realize *FINAL FANTASY VII*'s worldview as well as successfully expand it into a new world.



In-game analysis of a picture taken with the phone, to be converted into magic.

2 UTILIZING THE CAMERA. Even in the early stages of *BEFORE CRISIS*'s development, phones with cameras were already practically the norm in Japan. Additionally, an application programming interface for operating the camera through applications had just come out. What we needed to do was figure out a way to use the camera easily within the game. Could we use the camera's ability to recognize color?

Since every *Materia* (the source of magic in the *FINAL FANTASY VII* world) has an attribute, the user can take a picture with the camera, then have the application analyze the color of the picture, determine an attribute based on that color, and ultimately generate *Materia* for in-game use.

Take a picture of the fire burning in the fireplace and you are able to use fire magic. Photograph a cup of brown coffee and you earn meteor magic.

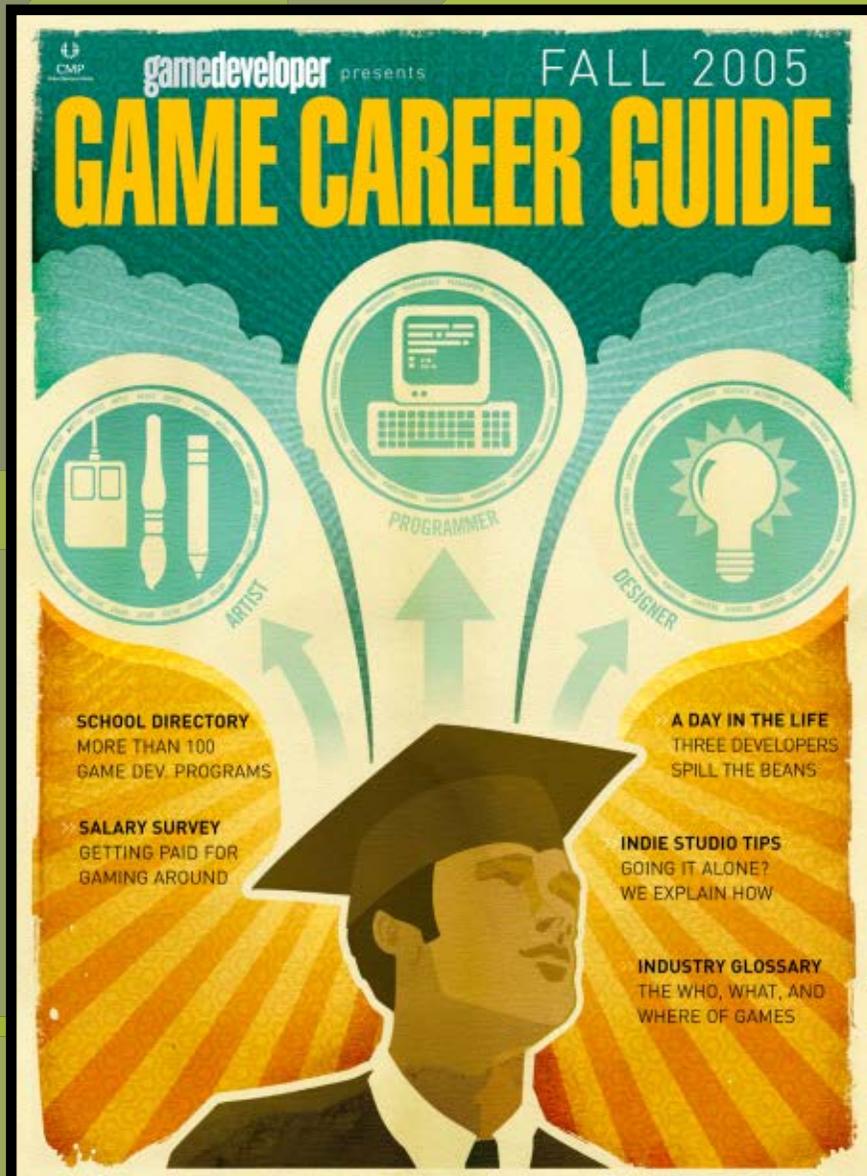
Anything in sight can be converted to magic, although combining what's real and what's virtual within a game has to be approached with care so as not to spoil the worldview. Creating magic through pictures is the stuff of fantasies in and of itself, so we believe we were able to offer a fresh experience through this integration.



Though intended for quick play, *BEFORE CRISIS* still has a heavy emphasis on story.

3 NETWORK COOPERATIVE PLAY. When it comes to network-based RPGs, the massively multiplayer online (MMO) is the most popular form and the most obvious choice for those playing games with others. Could we play a MMORPG on our phones? The answer for us, at this point, was no. Plus, we already have *FINAL FANTASY XI* for people who want to play an MMORPG version of our franchise, and it was decided that *BEFORE CRISIS* should be something unique.

WHICH WAY IS YOUR CAREER HEADED?



Brought to you by the editors of *Game Developer* magazine, this year's *Game Career Guide 2005* is packed with everything aspiring game developers need to get started in the game industry!

This 64-page special issue provides practical career advice, including:

- Real-world career advice on how to make it in the business
- Professional resources and salary survey
- Insider information and experience from industry veterans

Purchase a digital edition
of the *Game Career Guide*
online at
gamecareerguide.com



POSTMORTEM

Phones are not meant for playing games for long periods of time in the first place, as they are multi-use devices, primarily for talking and messaging, which may take precedence over game playing. With this in mind, we decided to figure out a more casual way of bringing a community feel to a mobile game. The network-based gameplay in *BEFORE CRISIS* can be described as loose-connection cooperative.

One of the systems that shows how *BEFORE CRISIS* is a loose connection cooperative is the Materia Support system, which enables users to send camera-generated Materia to other users. A player who needs assistance, usually in battle, sends a request to the network. Whoever receives the request doesn't even have to have the application running; the request comes by mail, and if accepted, the application runs automatically. The helper, the receiver of the message, can then send Materia with a few simple keystrokes and immediately close the application, making the whole process enjoyable and quick. The user who receives the support can then use magic without expending magic points, or depending on the combination, can summon beasts. This feature is handy when users are in a pinch. The system facilitates the participation of the helper by not requiring them to be currently playing, and also enables the requester to receive enormous support in the midst of a heated game.

4 REVIVAL. Another example of the networking ability is the Rescue Mission system. When your game is over, the Rescue Mission system allows users to either end their game and restart the mission from the beginning, or wait for an ally to rescue them, thereby letting them continue playing without receiving a penalty. Again, the helper need not be playing at the time and can receive incentive points, depending on the content of the rescue mission.

This system releases the users from the traditional crutch of online play, where cooperative playing required users to be online simultaneously. Now, each player has the chance to help another at his or her own leisure. This was our solution to create MMO-like cooperation, while still fitting in with the typical mobile lifestyle, keeping in mind the quick and concise ways that phones are used everyday.

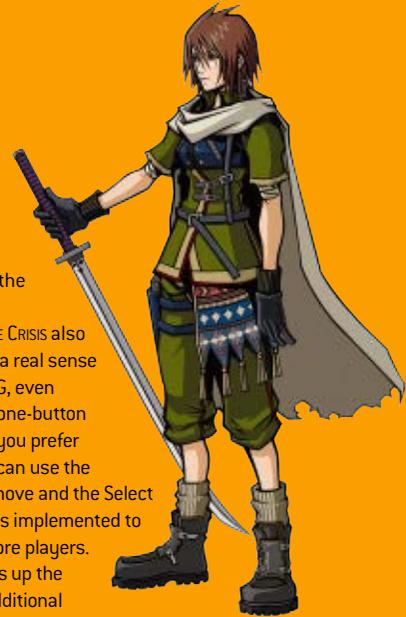


The Rescue Mission system in action: one character rescues another from prison.

5 DEALING INTELLIGENTLY WITH BUTTON LAYOUT. Because of the issues inherent with the operability of mobile devices (button layout), making an action RPG on a phone is no easy task. *BEFORE CRISIS* took this problem to task by automating basic action RPG maneuvers by default. An encounter with an enemy immediately puts you in battle mode. While in battle mode, holding Select enables you to attack the enemy closest to you with your current weapon. This simple operation is the crux of the battle gameplay. We didn't want to turn off casual users with complicated maneuvers, so the gameplay

was designed around the concept of simplicity.

The graphics of *BEFORE CRISIS* also provide the player with a real sense of playing an action RPG, even through it's basically a one-button operation. Of course, if you prefer manual operation, you can use the directional buttons to move and the Select key to attack, which was implemented to satisfy the more hardcore players. Playing this way speeds up the mission and leads to additional bonuses such as an increased reward upon completion. A lot of thought has gone into the operability of the cell phone as a game device, and we feel that the result is a level of playability that can adapt to a wide range of users.



WHAT WENT WRONG

1 DIFFICULTIES WITH SERVER CAPACITY. Although we are now able to provide stable service, in the beginning it was difficult to predict the servers' capacities, resulting in instances when some users had a tough time connecting. Square Enix loves undertaking new endeavors, but regarding mobile, it was difficult to get an accurate idea of how much reinforcement the servers needed, since there was no precedent. There were 1.6 million accesses on the first day of the beta test, which ultimately resulted in a server crash.

We had set up the servers based on previous experience with the mobile-ported version of *FINAL FANTASY I* for mobile, which also saw extremely high access. But the response to *BEFORE CRISIS* was so overwhelming that it caused the servers to crash in an instant. It was hard work trying to keep the service alive while maintaining the servers, but through painstaking analysis of the accesses, we continued to carefully but quickly tune up the finer settings, bringing back stability at a relatively early stage.

2 LIMITATIONS TO PHONES AS A DEVICE. The NTT DoCoMo 900 i-mode series of phones, for which *BEFORE CRISIS* was specifically designed, are great terminals for playing games, as we discovered when porting *FINAL FANTASY I*. But after several updates, we quickly ran into the phone's limitations of JAR (Java archive, or program space) at 100KB and SPD (data space) at 400KB. As much as we try to deal with the users' needs, the application's speed, including imaging, eventually hit a plateau.

Because the data space is re-writable, it's possible to add content by judiciously cutting out and compressing graphical data, but we have to keep in mind that we need to limit the transmission of data. The program space, however, is not re-writable, putting us in the difficult position of having to handle spec changes; for example, increasing program complexity within the 100KB capacity without being able to alter existing code. Some extreme fine-tuning was necessary before release.

To this day, even with the latest terminal devices, the



The BEFORE CRISIS: FINAL FANTASY VII core

spec changes; for example, increasing program complexity within the 100KB capacity without being able to alter existing code. Some extreme fine-tuning was necessary before release.

To this day, even with the latest terminal devices, the development staff must constantly think about presentation and features to lessen the stress of limited capacity and speed, much like we did in the days of the Famicom. This is done to offset issues that we couldn't resolve on the programming side.

3 OPERABILITY. As mentioned, the phone interface was never designed for playing games, so no matter how much we wrack our brains, sometimes the shape or layout of the buttons makes the game too difficult to play. Although the operability of BEFORE CRISIS was designed with this in mind, it can still be a difficult experience depending on your playing style.

For example, the player can avert attacks by pressing the key opposite to the direction from which you are attacked. Here, we've made adjustments to the margin of timing for the aversion to be activated to accommodate the potential difficulties in pressing the proper button. Although these efforts will continue to be made on the programming level, some fundamental issues of operability remain that can't be resolved through software alone.

4 CONTINUING ISSUES WITH DIFFERING MODELS.

Depending on the manufacturer, phone models may differ even within the same series. Even if a company makes a newer model, the internal specs sometimes change slightly. Every time a new model is released, the sound has to be almost completely remade based on the new sound module and speakers. Minor programming changes must be made to the visuals depending on the model's particular quirks. And the whole quality assurance process must be repeated, too.

For BEFORE CRISIS, we used the experience we had gained from the FINAL FANTASY I mobile port, and stripped down as much of the music and sound effects as possible in early stages of development. Even so, it's still extremely troublesome to have to remake all the sounds for every new phone that hits the market. Imagine 10 new phone models in a single year, and then having to



transplant the application for every one of those models. That's the kind of painstaking effort that continues to this day. Even though the applications are powerful enough to provide rich textures, issues unique to the mobile phone as a device still remain, and the difficulty with inter-model transplants will most likely continue to haunt developers for the foreseeable future.

5 SETTING DIFFICULTY LEVELS. When distributing new missions online, one very difficult issue in game design involves setting the right level of difficulty. Difficulty can be raised easily enough, but considering that mobile phone users are mostly casual gamers, the challenge is to judge how to make it just right. BEFORE CRISIS takes advantage of its online connection to continuously adjust the overall fun factor based on the player's current environment. This is done by making adjustments in enemy difficulty levels, experience points, and the effects of your attacks. Still, at the basic level, there are a lot of new specs and additional features to worry about, so a lot of time and effort goes into achieving the right balance to satisfy all users.

THE PLATFORM PICKLE

Starting development on a completely new game for any platform is hard work that requires extreme tenacity and concentration. This predicament is compounded when the platform has limited specifications and complex porting issues. However, it's precisely because of the difficulties we had to face with this very limited device that we gained some very valuable experience.

Making the game was not a quantitative issue of adding this or subtracting that, but rather a qualitative issue, requiring us to maintain our focus on making the best game possible within these constraints from the very beginning, which is the essence of game development. We hope to offer our users new gaming experiences, building on the know-how we have gained through developing BEFORE CRISIS. With the game's success in Japan behind us, we look forward to launching the U.S. version in 2006





IMMEDIATE MODE GUIS

THIS IS MY FINAL INNER PRODUCT

column. As much as I've enjoyed writing this column, I need to get back to working on games without the specter of a looming monthly article deadline.

Over the past year I've used this column to explore some technically challenging problems and to demonstrate the value of understanding what's really going on "under the hood."

Although I've looked mainly at quantifiable issues, once in a while, I've offered unquantified suggestions for reducing development complexity, such as my column on cooperative multi-threading ("Opening Doors," September 2004). This month I'm going to do that again, this time looking at implementing graphical user interfaces (GUIs).

GUIs are used in two ways within games: for in-game end-user interfaces, and for development tools like level editors. There may be other tools that could use GUIs, but programmers rarely want to add GUI support to their tools since it's such a pain; if we could reduce the painfulness, we might see significant improvements in development workflow.

The modern GUI was developed at Xerox PARC in the 1970s, hand-in-hand with the object-oriented language Smalltalk. The widgets of a GUI and the objects of an object-oriented language have become inextricably linked in the minds of programmers. GUI widgets are often considered proof of the benefits of object-oriented programming. However, a few years ago, my friend Casey Muratori developed a new, non-object-oriented approach to GUI programming that upsets the classic paradigm, an approach

he calls "immediate mode GUI." I started using it myself, and I've never gone back.

IMMEDIATE VS. RETAINED MODE

In graphics programming, we distinguish between retained and immediate mode style interfaces. In retained mode, we give specific objects to the library and make the library responsible for drawing them. We get back handles for the objects and use them to update positions and other properties, but in the end we just say "Draw()" and the library does it. In immediate mode, we may describe specific object properties to the library (textures, meshes), but for every frame we tell the library (from scratch) what all the visible objects are.

Generally for games, immediate mode APIs have been more successful than retained mode APIs. Retained mode interfaces require you to keep track of extra identifiers (the retained mode object handles) and copy information back and forth between the game and the renderer (such as object locations). Creating custom rendering modes requires using callbacks, complicating control flow. The more games want to do leading-edge rendering or particularly non-standard behaviors, the more valuable the immediate mode interface is. Of course, some code somewhere has to keep track of the objects. In a game engine, there will be something internal that is roughly like a retained-mode interface. Often, though, the game combines its notion of "game objects" and its notion of "render objects," whereas with a true retained-mode API, the renderer keeps its own copy of the objects, and the two must be kept synchronized. Even if a game implements its own full-fledged retained-mode graphics library, it's customized to the needs of the game and then built upon one or more platforms' immediate-mode APIs.

Traditional GUIs use retained mode. You create widgets, which the GUI library keeps track of for you. As you change the value of a variable, you copy it into the GUI widget, and copy it back out as the library changes it. If the user pushes a button, either the button triggers a callback to app code, or a piece of app code polls the button and reacts.

The need to copy data in and out of the widgets is exemplified by the model-view-controller (MVC) paradigm of GUI programming, originally formulated to describe programming simulations in Smalltalk. In the MVC, the model is part of the simulation; a view is a GUI widget that displays some state of the simulation; and a controller is a GUI widget that changes some state of the simulation. The model must know about all the views that are displaying it, so it can update them when the model changes. Changes must go through the official interface—directly changing a variable prevents the views from updating.

Although the MVC paradigm makes sense, it can be viewed as a premature optimization that occurred for historical reasons. To illustrate, let me describe an analogous scenario.

EDGE-TRIGGERED CACHING

Suppose we have a game with an inventory system in which objects can be placed in containers, and containers can be placed in other containers. The player can carry some objects, so we consider the player object itself to be a container. Objects have weight, and we might want to query to determine how much weight the player is carrying.

A traditional implementation of this has been to make each container store its current total weight. Any time an object is added or removed from the container, we update its weight. If the weight of an object is changed, we update the weight of its parent container as well. As a consequence, when an object is added to

a container, we update the weight of the container, which requires us to update the parent container of that container, etc., all the way up to the player. After each operation that might change the player's weight, we've incrementally updated that weight in a very efficient manner. If querying the player's weight happens more often than changing it (as is typically the case), this method is much more efficient.

Unfortunately, it's also bug-prone. It's easy to add some code that changes something's weight but does the incremental update incorrectly. Luckily, we can switch to a simpler approach: write a routine that recurses through all the objects in a container and recomputes the total weight for that container. Now, any time an object's weight changes, the "recompute from scratch" code will run. We still have to add this call everywhere we did before, but it's more mindless; just call "recompute()" without having to think about how to optimally compute it. This is less efficient than the previous approach, but it's less likely to have bugs.

If we have CPU speed to spare, we can get rid of all the bugs. Every time we need to know how much weight the player is carrying, we can recursively traverse the player's contents and add everything up. We can eliminate all the "when things change, update the cache" sort of code and just "brute force" it. This isn't a hack, though. It's simple code. It's less code. It's more maintainable. And it's actually more flexible, since we don't have to know when changes occur.

Essentially, we've switched from "edge-triggered" code—noticing when things change and propagating those changes—to an "always fully compute" style, which simplifies things a lot.

IMMEDIATE MODE GUI

The point of the above example is to show how it compares to retained-mode GUI (RMGUI). In RMGUIs, we notice when our model data changes and we update the appropriate widgets. If we change something—say, we want to disable a widget—we notice when it's time to disable the widget, and call a disable function. If the condition for whether the widget should be active or not is [A or (B and C)], we have to notice changes to any of those three variables and determine whether to enable or disable it. We might even save ourselves some complexity by making a simple function `crazyWidgetComputeEnabled` and call it whenever A, B, or C change, rather than try to compute the exact update locally.

All of this is, in some sense, a premature

optimization over the brute-force "do it every time" approach. If we have some notion of "every frame," as in a game or animation, we could simply call `crazyWidgetComputeEnabled` every frame and just not worry about it.

The primary reason that RMGUIs work in edge-triggered fashion is because they were necessarily optimized for the original era they were created in: to allow for minimal screen redraw, repainting only changed widgets. On a 68000-based Macintosh, this method was mandatory. But in a 3D game UI, the screen is redrawn from scratch every frame anyway. Moore's Law has radically changed the balance of what optimizations are necessary, making this edge-triggering essentially premature.

We're calling every widget every frame (to draw it), and I suggest that maybe we should just set the enabled state of a widget every frame, rather than try to edge-trigger it. We need to update the variable in the widget whenever it changes; maybe we should just copy it in and out every frame. If we're going to do that, our lives actually get simpler if we combine all those functions into a single operation, at which point we can just move the code that decides what widgets to update and draw from the library into our application. That is, we can switch to an immediate-mode interface.

Listing 1 shows a very simple set of widgets written for a hypothetical RMGUI system. Often, code like this won't be written explicitly; instead, some kind of data-driven system will be used. However, under the hood, that data-driven system eventually calls exactly these functions with exactly these parameters. (For simplicity's sake, I've omitted things like the widget screen locations, which could be explicit parameters to the functions; here I'm assuming automatic layout.)

What would an immediate mode GUI look like? The most important thing to remember is that immediate mode doesn't require us to create and destroy the objects involved. Instead, we simply describe them from scratch every frame, which doesn't actually result in more code. It's more like we run the "create" code every frame. The big

LISTING 1

```
void createWS(void)
{
    my_ws = new WidgetSet;
    my_ws += CreateButton ("Do It!", ID_WS_DOIT);
    my_ws += CreateSliderFloat("alpha", ID_WS_ALPHA, 0,1);
    my_ws += CreateSliderInt ("size", ID_WS_SIZE , 10,20);
    setChildFloat(my_ws, ID_WS_ALPHA, alpha);
    setChildInt (my_ws, ID_WS_SIZE , size);
    setCallback (my_ws, callbackWS);
}

void callbackWS(int id)
{
    switch (id) {
        case ID_WS_DOIT: doIt(); break;
        case ID_WS_ALPHA: alpha = getChildFloat(my_ws, id); break;
        case ID_WS_SIZE: size = getChildInt( my_ws, id); break;
    }
}

void deleteWS(void)
{
    delete my_ws;
}
```

Creating and updating a simple set of widgets for a retained-mode GUI. Not shown are calls to `setChildFloat()` and `setChildInt()` when the variables `alpha` and `size` change.

LISTING 2

```
void tickWS(void)
{
    if (doButton("Do It!", ID_WS_DOIT))
        doIt();
    doSliderFloat("alpha", 0,1, &alpha);
    doSliderInt ("size", 10,20, &size);
}
```

Displaying and interacting with the same widgets from Listing 1, but using an immediate mode UI.

advantage to ImGui is that we don't have to synchronize data between a create, an update, and a delete. In fact, we only ever need to write one function call per widget.

A plausible ImGui implementation of the same widget set is shown in Listing 2. Rather than have the library traverse an RMGUI widget hierarchy dispatching events, the application traverses all the widgets every frame, and the widget-

Sample code

<http://silverspaceship.com/inner/imgui>

Casey's ImGui forum

<https://mollyrocket.com/forums/viewforum.php?f=10>

Raymond Chen's Chinese/English dictionary tutorial

<http://blogs.msdn.com/oldnewthing/archive/2005/06/15/429338.aspx>

THE INNER PRODUCT

processing functions individually process the current frame's events and draw the widget in an appropriate state, returning true if the widget "acted" this frame.

Although this example might seem unfair because it's a best case for ImGui, it's still a crucial example. You could always layer a retained-mode GUI library atop the ImGui interface and use that for most widgets (although I don't). But any arbitrary code you're writing can still go ahead and toss in an extra immediate-mode widget, as simple as those shown in Listing 2, and it will still interoperate with your RMGUI widgets.

In the example code, the ImGui passes in the addresses of the variables to update, and the library updates them directly with no copying back and forth. It's possible for RMGUIs to do this as well, although few do (GLUI, a GUI library for OpenGL, is the only example I know of). But even if an RMGUI offered this feature, it still wouldn't be as effective as ImGui. The RMGUI widget is attached to a single variable. In ImGui, you can write `doSliderFloat("alpha", 0, 1, &debug_obj->alpha)`

and it doesn't matter how `debug_obj` changes. The ImGui slider will always be editing the current one. Put an "if (`debug_obj != NULL`)" in front of it and it's robust and does just what you want.

Yet another advantage of ImGui is this: By avoiding creating widgets at all, it can avoid the cost of creating those widgets. If you try to browse a list of one million items with an RMGUI system that has to create one million widgets, you may find your machine grinding. Since ImGui never has to create anything, it's much more tolerable—assuming you write some code to only traverse the items that are actually on screen. Some RMGUI techniques, such as owner drawing, can approximate this effect (see Chen in Resources), but generally not when the contents are themselves actual widgets.

IMPLEMENTING IMGUI

The initial implementation of immediate mode GUI is subtle and requires care, but once you're familiar with the principles you'll find it's actually easier to create custom widgets and behaviors than with RMGUI.

The starting place for implementing ImGui is to make it stateless, leaving all state on the client side of the API. For example, the client specifies the size and location of the widgets. This might sound horrible; in a traditional RMGUI, each widget has a lot of state. In turns out, ImGuis need a lot less. Table 1 shows a breakdown of where the state from an RMGUI lives in two possible ImGui systems.

Listing 2 illustrates how "app data" (like the variables) and "configuration data" (like button labels) live app-side and are explicitly passed-in. ImGuis don't need to store callbacks or explicitly represent traversal information as state because the app is responsible for traversing all widgets and calling their update-and-draw functions.

The only other interesting case in Table 1 is for interactive state, which refers to the state in widgets that is only used when the user is interacting with them—for example when a user clicks left-down on a button, until she releases the button and its effect occurs. This information might include flags for whether the widget is currently being interacted with, what important things have happened, timestamps for when the user first clicked on it, and so on.

The sneaky observation is that the user can actually only interact with one UI widget at a time. Instead of having each widget carry around a flag for whether it's being interacted with, the library

can keep a "global variable" which identifies which widget is active (being interacted with). Also, any other interaction-only state can be stored globally in the library, rather than per-widget. If a text-edit widget receives a pointer to the "app string" it's supposed to edit, it copies the string into a global buffer and manipulates that, all while it's being edited. It's copied over the app string when the user presses enter or else is discarded when the user presses escape; but either way, only one text-edit widget can be active at a time, so a global variable suffices.

The most subtle aspect of ImGui arises with this notion of the active widget. Every widget still needs a unique identifier so that, from one frame to the next, the library can tell which widget the user is interacting with. You may not see many identifiers like `IM_WS_ALPHA` in Listing 2, but unique identifiers are still there. As well as using the pointer (for example `&alpha`) to directly read and write the variable, this ImGui library actually uses the address of the pointer itself as the unique identifier. In fact, rather than passing `IM_WS_DOIT` into `doButton` in Listing 2, an actual application would most likely pass in `{void*} doit`—not so it would be called, but merely as a unique identifier. Most ImGui widgets can make use of a simple identifier like this, but occasionally, a little more state is required.

As long as we have unique identifiers, it's actually possible for the ImGui library to store data for us behind the scenes, if it's data we don't need to access. This is how the sophisticated ImGui in Table 1 can store UI data like scrollbar state and presentation data like fade state for a widget that fades out as it becomes inactive. The app provides a unique identifier for the scrollbar, and the library associates the appropriate data with that identifier (for example, with a hash table).

A complete description on implementing even the simple ImGui would take too much space. I invite you instead to watch Casey's video lecture mirrored on my web site, and look at the sample code I've posted.

I'd like to thank Casey Muratori, Jonathan Blow, and Atman Binstock for feedback on this article, some of which you can read on Casey's web site forum. ❖

Table 1: Immediate-mode GUI state management

WIDGET STATE	NAIVE IMGUI	SOPHISTICATED IMGUI
traversal information	-	-
callback pointers	-	-
configuration	app	app
interactive state	library global	library global
app data	app	app
UI data	app	library
layout	app	library
presentation	app	library

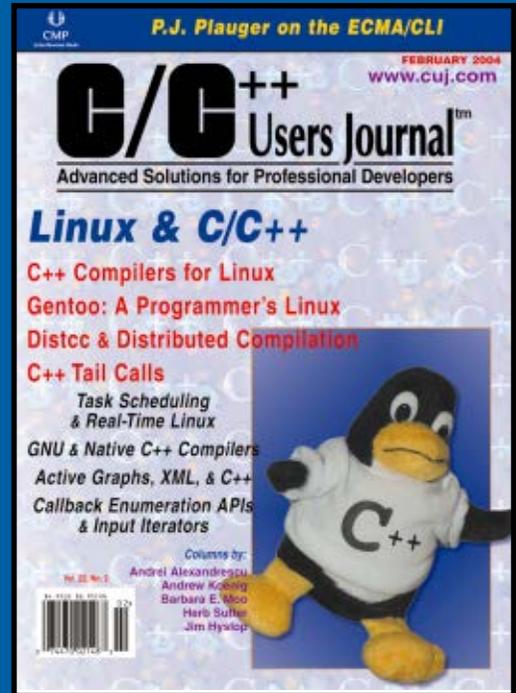
Immediate-mode GUI systems need to store less state than retained-mode GUI systems. Per-widget state from a retained-mode system can be stored in two plausible immediate-mode systems. Traversal information means things like parent and child widgets. Configuration refers to data that is effectively read-only. Interactive state refers to state that's only valid while the user is interacting with the widget. App data are user-manipulable quantities that reflect existing data in the app. UI data are user-manipulable quantities that the app doesn't care about. Layout is the location and/or size of a widget. Presentation refers to special "flashy" display, like animation or widget glow effects.

BUILD WITH THE BEST!

C/C++ Users Journal gives you the tightest, fastest code possible from experts in C/C++ programming.

Every month you'll get features such as:

- Internet Programming
- Object-Oriented Programming
- Embedded Systems
- Graphics
- C# and .NET
- And more...



Subscribe Today 12 issues for **\$19⁹⁵**

C/C++ Users Journal[™]
Advanced Solutions for Professional Developers

VISIT US AT: www.cuj.com/sub/

Discount KeyCode 2CDT

Canada/Mexico Subscriptions: 1 year - \$40 / Outside North America: 1 year - \$60



STEVE THEODORE

PIXEL PUSHER

EYES RIGHT

LAST MONTH WE TALKED ABOUT THE basics of building realistic eyes for animated characters. This month we're going to look at the second part of bringing life to the eyes: animation.

STAYING FOCUSED

Before we talk about the subtleties of expression in the eyes, let's quickly review the mechanics of setting up eyes for animation.

Nowadays, most facial rigs manage this with some kind of an eye tracking mechanism (for example, a pair of aim constraints in Maya, or look-at controllers in 3DS Max). Naturally, this type of setup makes it very simple to fix a character's gaze firmly in the 3D space of a scene and allows you to track moving targets with perfect accuracy.

More importantly, good eye tracking creates a realistic convergence between the eyes. Our brains analyze the angle of that convergence to supply the critical third dimension in our perception of 3D space—it's the main clue we use to judge distance. Because eye tracking is so important to our own perceptions, we are also very good at reading it in the eyes of others too. Audiences are very sensitive to the implied focal distance in a character's eyes. Many animated characters that look great in stills become unsettling to watch in motion when we have a better sense of what the character is—or ought to be—focusing on. As you can see in Figure 1, too much or too little convergence can produce unsettling effects.

The biggest drawback to eye tracking rigs is, ironically, that they're a bit too realistic for some kinds of scenes. Tracking rigs do a great job if your

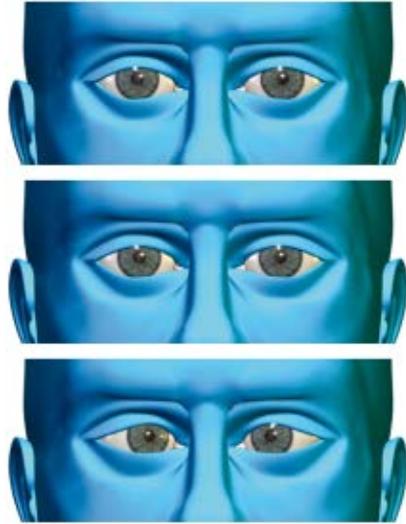


FIGURE 1 The top set of eyes are parallel in an unfocused, 1,000-yard stare. In the middle image, the character's eyes are focused at a distance of six feet. A model with nearly crossed eyes (bottom image) focuses at a point about 12 inches from the face.

character is looking directly at something or tracking a moving object—but often, characters aren't really looking at anything at all. Much of the time, a character's gaze is more important as an indicator of thought or emotion than as a radar-like target-tracking device.

When your character needs to gaze off into the distance in a wistful reverie, it's a huge hassle to have to whisk that eye target off to the far uncharted regions of your scene. One handy workaround is to use a separate eye target for each eye and then group them together. By scaling the group, you can increase the distance at which the eye lines appear to converge, allowing you to fake long distance gazes without sending your target object off into outer space. If you use this trick often, you should add an aim constraint to the target group itself so that the group is always aligned to the head, which keeps focal distance from appearing to shift as the target moves around the head.

Another problem with eye tracking rigs occurs when a character stops focusing on one object and switches to a different one. These moves are extremely quick. The muscles of the eye are by far the

fastest in the body and so the eyes don't really focus on anything at all during that brief transition. Moreover, many people drop their gaze downward as they switch from one focal point to another, almost as if the body were acknowledging that the eyes are between jobs. The combination of very high speed and non-linear movement makes it hard to key shifts of gaze convincingly, although fortunately, most people cover these fast shifts with a blink or a half-blink so you can often cheat your way around this problem.

If your scenes deal mostly with the character's facial expression and emotions and less with the careful tracking of targets, you ought to at least consider using forward kinematics (FK) controls on the eyes instead of a tracking rig. This is especially true for close ups in which the audience will get a really good look at the character's eyes and won't necessarily see the target. If you do go with an FK rig, you don't want to forego the all-important convergence effect, so you'll need to add a simple method for simulating it. The best method is to use two controls—a two-axis eye direction object that establishes the basic direction of the gaze and a convergence control with an expression

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



FIGURE 2 The televised Nixon-Kennedy debate in 1960 had a memorable moment when Nixon's eyes betrayed his true inner emotions despite his carefully controlled voice and facial expressions.

that rotates both eyes inward to create the illusion of focal distance. Otherwise you'll constantly struggle to keep the eyes from getting out of sync.

Sometimes a character's behavior makes it tough to keep FK eyes on target. If, for example, you have a character who does a lot of head gestures while talking to someone, and you hate counter-animating the eyes against the head movement, you might want set up the eye direction control in world space rather than the space of the head. Of course, this won't be a good idea if the entire character is turning during the scene!

DON'T BE AFRAID TO CHEAT

Regardless of which rig you use, the basic rules for good eye animation are pretty simple. The first and most important rule is "don't be afraid to cheat."

When fiddling with an eye tracking rig, it's very easy to get caught up in the niggling details. It's important to remember that the only real test is what works for the shot you're putting together. Actors in films and TV, who don't have to worry about a roving camera, frequently mis-direct their gaze in order to create particular emotional or visual effects—it's called "cheating the eyelines,"

and it's done all the time. Naturally, if you're building an animation that will be seen in the round you can't cheat too blatantly, but in the end the performance is the only thing that matters.

DON'T FORGET TO BLINK

The second rule of good eye acting is "don't forget to blink."

Few artists would try to build a detailed character without at least some attention to blinks, but often we assume that because blinking is involuntary, it's essentially random.

Blinks are anything but random. Walter Murch's brilliant essay on film editing, "The Blink of an Eye," likens blinking to the action of a film editor, outlining the natural separations between sequences of thoughts and actions. Good eye animation should always observe the rule that any important change in the character's thoughts will be marked by a blink. [This is also why fast changes in look direction are covered by blinks—the change of gaze is almost always a change in the character's thinking.]

But not every blink is significant. Random blinks act like windshield wipers for the eyes. They happen every few seconds, unless overridden by intense

concentration. Random blinks occur between three and six seconds apart. When we're speaking, blinks come more rapidly, every two and a half to three seconds. In stressful situations, blinks may happen as often as once every second or two, a classic subliminal indication of nervousness and defensiveness. It's possible to control blink frequency with careful concentration, but inevitably the nervousness bursts through in the form of double and triple or even quadruple blinks; psychologists refer to this as the "Nixon effect" (see Figure 2) after Richard Nixon's famously awkward appearance in the first televised presidential debate in 1960.

DON'T STARE

The third rule to remember when animating eyes is "don't stare." Even when we think we're just looking at something, our eyes are in fact constantly roving around the object we're considering (see Figure 3). Because the eyes are so quick, the actual movements are effectively invisible at typical game frame rates, so the small movements (or saccades) resemble the darting of a hummingbird: short periods of hovering punctuated by instantaneous shifts to a new position.

Saccades usually focus on important details. For example, when looking at someone's face, most of the saccades

CONTINUED ON PG 42

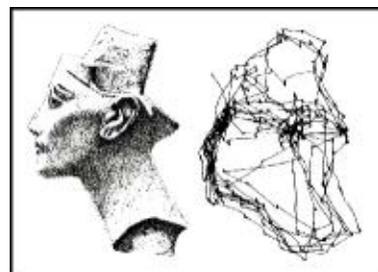


FIGURE 3 Active eyes never rest on one spot for long. A viewer looked at the image on the left for a few moments; the plots on the right show points where the viewer's eyes fixated.



NOAH FALSTEIN

» GAME SHUI

DON'T CELL OUT

THE LYRICS AT RIGHT ARE FROM A

little-known song written by a singer/songwriter who went back to teaching math after a very successful start in music in the 1950s and '60s. He was a man who kept his principles, and his words are a good warning for game developers today.

For several years now, game conferences have been abuzz with speculation on how the cell phone game market is the next big opportunity for our industry. Occasionally, a counter-argument arises that says, "No, it's China—that's the next big opportunity," and then both sides happily agree that games on Chinese cell phones are surely a great opportunity.

But I know a lot of people who have lost time and money trying to make cell phone games pay, and although a few have been able to eke out a business, most have found the market to be brutal and competitive or even just plain frustrating. I have yet to work in China, so I'll set aside that subject for a future date to examine some of the reasons the mobile game market is so tough for developers.

FRAG-FEST

The biggest problem with the mobile market is fragmentation. Design is tough enough when you have a specific set of platform capabilities, but the cell phone market is fragmented even beyond the wildest nightmares of PC developers. The hardware is disjointed in many ways: screen resolution, color depth, processor speed and capability, RAM and long-term storage ... all of these vary wildly from phone to phone (see "Mobilizing Content," page 23).

Selling out is easy to do.

It's not so hard to find a buyer for you.

When money talks, you're under its spell.

Ah, but whaddya have when there's nothing left to sell?

—Tom Lehrer, "Selling Out," 1973

Even when a phone has a useful hardware capability—like Bluetooth connectivity, a camera, or GPS—there's no assurance that that component will be available to the game software. In my April 2004 column ("Have Cell Phone, Will Play"), I suggested that some of these then-new capabilities might be good for games, but that was naive.

I've since learned that even if a developer is willing and interested to make a game for one specific model of phone to exploit its capabilities, another type of fragmentation—that of the many cellular carriers who provide distribution of games—means that many of those carriers won't even accept a game unless it can run on nearly all phones. So that means that your killer idea for a geo-caching game using GPS may not be picked up by any major carrier.

HOPE AFTER ALL

From a design standpoint, I can suggest one possible solution. Go for quality, not quantity.

I believe that a really stellar, breakthrough concept on a mobile game could transform the market, motivating people to switch carriers, or at least switch to a new phone with new capabilities, if the killer app was fun enough.

Phones seem to be built for viral marketing. You can potentially make a game that's so much fun, players convince their friends to buy it. Or the game can subsist through word-of-mouth advertising when complete strangers ask why you're staring at your phone and chuckling with glee. The communications aspects of phones, their near-universal

availability, their increasing power and decreasing cost—all of these things are reason for hope for mobile games.

THE KILLER APP IS OUT THERE

The bottom line is this: Don't sell out. Giving into a mobile carrier's insistence on making a game work on the lowest common denominator of phones may get you on the download list, but it won't necessarily make you rich, now that you've watered down your game to fit the slowest, clunkiest, black-and-green handset. Instead, stick by your principles.

Easy to say, but hard to do, right? And yet, there are people out there finding a way. Not everyone interested in cell phone games has a mercenary mentality. I've recently dealt with several companies that impressed me with their methods. The serious games company Morphonix aims to make cell phone games designed to teach teens about how the brain works, and is using government grants to fund the work so they're not dependent on big deals with carriers to get the games made.

And that's just one company with one approach. There are thousands of them worldwide, and soon enough, someone will create a true killer app.

So don't go for the easy big bucks—keep your integrity. Let's give Tom Lehrer the last word:

It's so nice to have integrity

I'll tell you why:

If you really have integrity

It means your price is very high! ❖

NOAH FALSTEIN is a 25-year veteran of the game industry. His web site, www.theinspiracy.com, has a description of The 400 Project, the basis for these columns. Also at that site is a list of the game design rules collected so far and tips on how to use them. Email him at nfalstein@gdmag.com.



ALEXANDER BRANDON

✦ AURAL FIXATION

AUDIO BUDGETS

WHEN WE AUDIO FOLK ARE ENGAGING IN

the more fun side of game audio, we're sitting around the campfire strumming Martin Backpacker guitars, fudging with Kyma and a Korg Kaoss Pad on the laptop, and figuring out how to squeeze a thousand sounds into 32K of memory. Yet, there's a harsher reality emerging in recent days: our budgets.

Without a keen understanding of what really goes into an audio budget, content creators and directors alike will encounter some nasty pitfalls in today's game development environment. More and more, game budgets seem to encounter cost overruns, and while I can't make an analysis of an entire game budget in one column, let's take a look at some techniques that can help your audio budget stay on target.

VARIABLES

What are the variables that make an audio budget go out of control? There are a few simple formulas I follow to avoid over or under estimating a budget (three of which I detail for you in this column).

Estimating the audio budget is the responsibility of audio leads, managers, and producers, but estimation also plays a role in a content creator's pitching process. A high-level initial plan of many music contractors (I say music contractors because most of the demos and solicitations I get are from composers) is to look at a publisher's annual revenue and charge accordingly. But this is only part of the picture.

THE BIRTH OF SOUND

Music. The industry standard (IS) rate for approved minutes of original music is \$1,000–\$1,200 per minute. "Approved"

means with a reasonable amount of revision (if you revise something more than five times, you're wasting money). When devising your budget, you need to estimate the number of minutes required (MR) for in-game original music. Each game has its own needs, but fortunately we're mostly beyond the old tradition of having one minute of looped original music per game level.

The standard MR these days is around 2–3 minutes of music per level, but there are always ways to spend less and get zero repetition with an adaptive soundtrack. [See "Streaming by Design," December 2004.]

Padding (P) is incidental music that you add depending on the project genre. Racing games, puzzle games, and fighting games have less need for padding than other genres. Role-playing games usually require the most padding because they have the most gameplay time and the greatest need to avoid aural repetition. Here's the formula I use for figuring the estimated music cost:

ISxMR+P=estimated music cost.

SOUND EFFECTIVENESS

SFX. I find a good way to estimate is based on sound type. If you break your sound asset list into categories, it will help you identify which sounds are easy to produce, and which are more complex.

This equation can be used for as many sound types as you need, and so I urge whoever is estimating costs to use this for every sound the game design will require. A top notch driving game will have a lot of complex sound effects, but that complexity may vary from a single engine loop to a complex engine loop with piston sounds as well as the hum of the engine block, which varies cost.

Knowing the number of sound effects required (FXR) and their production rates (PR)—which range from \$5 per sound effect for something like footsteps (simple effect, or SE), to \$50 or more per sound effect for a good machine gun sound (complex effect, or CE)—will help boil down a budget to

something more realistic.

The formula I use to estimate my sound effects cost looks like this:

(FXR x PR(SE))+(FXR x PR(CE))= estimated sound effect cost.

[Note that (SE) is a signifier rather than a multiplicative.]

INTELLIVOICE

Voice over. The industry standard rate for a four-hour session with either a Screen Actors Guild or American Federation of Television and Radio Artists-based union member is \$759 as of press time.

Good non-union rates are around \$500 for a four-hour session. Yes, there is a place for non-union talent in games, but remember that in most cases you can only use union or non-union, not both. Count on about 80–100 lines an hour (LH) with a good director, and 50 lines an hour or less for ADR [automated dialogue replacement, which is the process of re-recording dialogue once video has already been created].

ADR is what you really want to avoid since some actors can nail it and some take a lot more time. Regardless of time, it is a very painful process, and often it is more economical to re-mocap and re-lipsync, depending on the length of the scene. At this point, you have your line count (LC).

Finally, will you outsource (OS) the editing of the dialogue? Will you outsource the directing? If so, you're looking at around \$2–\$3 per line for editing and \$3–\$6 per line for directing. It is usually less expensive to do this in-house when possible.

A useful formula for configuring your estimated voice over cost is:

LC/LHxIS(+OS)= estimated voice over cost.

FORMULA CONTEXT

These formulas won't necessarily get you a completely accurate estimate. You have to use these tools throughout the



GRAN TURISMO 4 squeals with complex, yet effective sound effects.

ALEXANDER BRANDON has been involved with game audio since 1994 and is currently the audio manager at Midway in San Diego, Calif. You can email him at abrandon@gdmag.com.

project, keeping track of what you're spending, while also following a few more guidelines.

Read the design document thoroughly. If you are lucky enough to have an accurate, regularly updated design document, reading it carefully will allow you to estimate every possible area for sound, voice over, or music that you will need, outside of the developer's basic asset requirements.

Integration. Just as in art, integrating is an important part of the game audio process. If you're with a big firm like Electronic Arts or Sony, you have in-house engineers that can do this for you. But if you aren't, chances are a designer or programmer will be integrating your audio, and you're entering a potential cost overrun situation. Audio designers are less expensive than programmers (and often designers as well), and they'll integrate the audio faster and with better quality. I don't have a formula for integrating because each

game has different individual integration requirements, but this climate is changing with a slow but steady global movement towards standardized audio integration tools. Just be sure you have a solid pipeline for getting sounds into your game. (See "The Line of Quality Part III: Integration," May 2005.)

Be careful with licensing. There are many hazards and benefits involved in licensing, but using celebrity talent for a game is the first thing that can balloon an audio budget. Often, using a single star for the lead voice over role will cost as much or more than the rest of the actors' fees combined. Ask hard questions of the marketing departments and producers: will use of a star really help sell units? Historically, stars have not helped games sell units unless the game is based around that star, and even then, the gameplay and license are the main selling points.

Understand the goals from the top down. Most developers have two primary goals: make a fun game and spend the

least possible amount of money. The second goal is a hard one to take in, but understanding it is important because it is an integral part of generating profit, namely the money that developers and publishers use to stay alive and grow.

Each publisher and developer has a different way of doing this, but history tells us that intelligently concentrating on the "fun" goal first and the "money" goal second generates a hit. Two simple examples are *WORLD OF WARCRAFT* and later iterations of the *GRAND THEFT AUTO III* franchise, but remember, I said "intelligently." There have also been a lot of failures using that goal focus.

THE SOUND OF GAME

With all of this in mind you're well on your way to spending less while achieving more with your game audio. If you use this information from the first days of pre-production, you'll be able to more intelligently estimate and stay on or under budget. ❖

CONTINUED FROM PG 39

will center on the eyes and the mouth. However some of them are in less important locations and a few are essentially random. In most cases you can simply cycle between a few obvious locations, but a small amount of randomness can also add a lot of life to the otherwise static act of simply looking.

The timing of saccadic movements is very important to establishing a character's attitude. For example, if a character is making "eye contact" with the camera, leaving out saccades altogether gives a creepy impression of unhealthy intensity or aggression. Too many saccades, on the other hand, creates an impression of anxiety or diffidence. In most circumstances, saccadic jumps happen four or five times per second; however, if the viewer is paying careful attention, they may slow down to once a second or so. In conversation, saccades are rarer still. Someone speaking will shift his or her gaze around once every 1.8 seconds, while someone listening may fixate on one spot for as long as 2.5 seconds. Of

course, there are a number of other factors that influence eye movements as well, so use your best judgment and treat these numbers as general guidelines only.

SPACING OUT

The last rule for eye animation covers what to do when your character is spacing out, and isn't actually focused on anything.

Defocusing—basically, staring into space for a moment—happens quite a lot when characters are thinking or remembering, rather than attending to the present moment. You probably know the age-old cartoon convention that says characters look up and to the left when trying to remember something (of course, it also says they stick out their tongue at the same time).

What you may not know is that there's an entire school of psychology devoted to parsing the mysteries of random look directions. Advocates of neuro-linguistic processing theory (NLP) believe that eyes can be read almost like status displays while the mind is engaged.

When the mind is focusing on sound, the eyes drift downward; when it's pondering images, they tend up. When attempting to remember, the eyes look left, but when inventing new sounds or images the eyes tend right (this is why many books on body language suggest that people look to the right when they are about to lie).

Bear in mind that NLP theory is pretty controversial among psychologists, so it shouldn't be treated as gospel. But it can be a useful source of ideas when you're working out a facial performance. For more information on NLP, check out www.nlpinfo.com.

EYES ALIVE

Obviously, it's hard to capture the essence of a great animation performance in words. The tips and rules covered here are only the building blocks for a successful job. The heart of the matter still depends on the fundamentals: timing, pose, and the animator's intuition. Like any set of art commandments, these rules are made to be discarded once they're mastered. ❖

radicalgetstalent

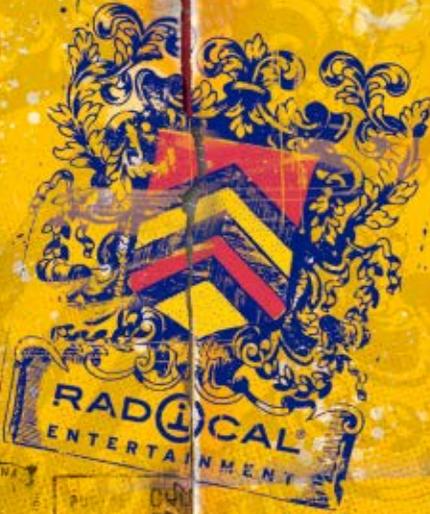
Radical Entertainment is seeking talent in the following fields:

- **Game Programming & Technical Development**
Including Technical Directors, Technical Project Managers and experienced game programmers
- **Game Art Development**
Including Art Directors, Art Production Managers, Lead Specialists in Environments, Characters, FX, Animation etc
- **Game Design**
Including Creative Directors, Lead Designers, World Builders and Scripters

e-mail: jobs@radical.ca with resume

online: www.radical.ca

call: 1-866-radjobs



play work live breathe games play

8th Floor, 369 Terminal Avenue, Vancouver, BC, Canada V6A 4C4
Radical Entertainment is a developer of interactive games for all major platforms.
Radical Entertainment is a Vivendi Universal Games company.
© Radical Entertainment. All rights reserved.



Eugene, OR

BACKBONE



Emeryville, CA
Vancouver, BC



Newport Beach, CA



imaginengine™

Boston, MA

JOIN OUR TEAM



Visit www.f9e.com
for current job listings





Student artwork...Vizdev K.Nguyen



deliver shroom pizza **or** design shroom land?

80% of our graduates are working in the art and design industry

SCHOOL OF ANIMATION & VISUAL EFFECTS

2D Animation, 3D Modeling,
3D Animation, Character Animation,
Background Painting, Games,
Storyboard & VFX/Compositing

ACADEMY *of* ART UNIVERSITY

FOUNDED IN SAN FRANCISCO 1929
BY ARTISTS FOR ARTISTS



1.800.544.ARTS | www.academyart.edu FOR FALL-CLASSES START SEPTEMBER 1

79 New Montgomery St., San Francisco, CA 94105 • Nationally Accredited by ACICS, NASAD, FIDER (BFA-IAD), NAAAB - Candidate Status (M-ARCH)

REGISTER NOW

VFS™

VANCOUVER FILM SCHOOL
WHERE RESULTS MATTER



GAME DESIGN

Work Hard, Play Harder

Siggraph
Los Angeles Convention Center
August 2 - 4, 2005
Visit us at **Booth 2166**

Come and see award-winning student work and meet with a VFS representative. Join us for **TechTalk** Wednesday, August 3 at 10am on the show floor.

For detailed program information call **1.800.661.4101** or visit **vfs.com**.

Tired of your day job?

The Centre for Distance Education offers diploma programs in New Media, entirely online.

Get a Diploma in: **3D Animation**

Character Animation

Video Game Art & Illustration

Digital Imaging

Digital Publishing

Web Page Design

Get with the program. Get in the game.



CENTRE FOR
Distance Education www.cd-ed.com
1.866.446.5898

The Centre for Distance Education is a member of the Private Career Colleges Association of Nova Scotia and of the National Association of Career Colleges. Financial assistance may be available to those who qualify. International applicants welcome. Campus is entirely virtual.

"Let's just go drop a car..."

crane ->

car ->

CHAKRASOUND
GAME AUDIO YOU CAN FEEL

Levon Louis
about 10' away with a microphone ->

Pushing game audio forward, one sound at a time.
www.chakrasound.com

to advertise in
**Creative Careers and
Get Educated**
call **Aaron Murawski** at
(415) 947-6227,
or email to:
amurawski@cmp.com

to advertise in
Marketplace
call **Susan Kirby** at
(415) 947-6226,
or email to:
skirby@cmp.com

FLY THROUGH YOUR CREATION

FREEDOM SPEED COMFORT

MEET SPACEPILOT™

A bird can't fly with one wing. Why use just a mouse when you can put both hands into your creation. SpacePilot™, the Intelligent Motion Controller, is a giant leap forward in intelligent interface devices for professionals. Featuring adaptive sensing technology that automatically gives you the functions you want, when you want them and an array of extendable keys that put unlimited functions within a fingertip's reach.

Master your 3D world today – learn more at: www.3dconnexion.com/GDfly

3Dconnexion
A LOGITECH COMPANY

CORRECTIONS: In the June/July 2005 issue, the cover image was created by Sach Steffel of Backbone Entertainment. In addition, in the Career Guide issue, Academy of Art University was misnamed. We regret the errors.

ADVERTISER INDEX

COMPANY NAME	PAGE
3D Connexion	46
Academy of Art University	44
Ageia Technology/Novodex	17
Akamai Technologies, Inc	C3
Anark	6
Autodesk	13
Centre for Distance Education	45
Chakrasound	46
CTIA	18
C/C++ Users Journal	37
Full Sail Real World Education	47
Havok	C2
Lifemode Interactive	15
Perforce Software	10
RAD Game Tools	C4
Radical Entertainment	43
Sony Ericsson Mobile	3
Superscape	20
The Collective, Inc.	43
University of Advancing Technology	47
Vancouver Film School	45

Game Developer (ISSN 1073-922X) is published monthly by CMP Media LLC, 600 Harrison St., 6th Fl., San Francisco, CA 94107, (415) 947-6000. Please direct advertising and editorial inquiries to this address. Canadian Registered for GST as CMP Media LLC, GST No. R13288078, Customer No. 2116057, Agreement No. 40011901. **SUBSCRIPTION RATES:** Subscription rate for the U.S. is \$49.95 for twelve issues. Countries outside the U.S. must be prepaid in U.S. funds drawn on a U.S. bank or via credit card. Canada/Mexico: \$69.95; all other countries: \$99.95 (issues shipped via air delivery). Periodical postage paid at San Francisco, CA and additional mailing offices. **POSTMASTER:** Send address changes to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. **CUSTOMER SERVICE:** For subscription orders and changes of address, call toll-free in the U.S. (800) 250-2429 or fax (847) 647-5972. All other countries call (1) (847) 647-5928 or fax (1) (847) 647-5972. Send payments to *Game Developer*, P.O. Box 1274, Skokie, IL 60076-8274. For back issues write to *Game Developer*, 1601 W. 23rd St. Ste. 200, Lawrence, KS 66046-2703. Call toll-free in the U.S./Canada (800) 444-4881 or fax (785) 841-2624. All other countries call (1) (785) 841-1631 or fax (1) (785) 841-2624. Please remember to indicate *Game Developer* on any correspondence. All content, copyright *Game Developer* magazine/CMP Media, LLC, unless otherwise indicated. Don't steal any of it.

FULL SAIL
Real World Education

Game Design & Development
Bachelor of Science
Degree Program

Associate of Science &
Bachelor of Science
Degree Programs

school of

- ▶ Computer Animation
- ▶ Digital Media
- ▶ Entertainment Business
- Film
- ▶ Game Design & Development
- Recording Arts
- Show Production & Touring

800.226.7625
fullsail.com

3300 University Boulevard
Winter Park, FL 32792

Financial aid available to those who qualify
Job placement assistance
Accredited College, ACCSCT

© 2004 Full Sail, Inc. All rights reserved. The terms "Full Sail," "Full Sail Real World Education," and the Full Sail logo are either registered service marks or service marks of Full Sail, Inc.

The Future

Brought To You
By Inspiration
—And Coffee



www.uat.edu or 800.658.5744

Available online or on-campus

GAME DESIGN	VIDEO PRODUCTION
DIGITAL ANIMATION	NETWORK SECURITY
ARTIFICIAL LIFE	SOFTWARE ENGINEERING
MULTIMEDIA	TECHNOLOGY MANAGEMENT
WEB DESIGN	GAME PROGRAMMING



CAPYBARA GAMES'
PLANT LIFE

Concept artist: Anthony Chan
Pixel artists: Nathan Vella and
Mike Nguyen



PLANT LIFE





>80%

25 of the top 30 media & entertainment companies trust Akamai.

They trust Akamai to deliver more than content. To deliver billions of downloads daily and give users what they want when they want it. Why? Because Akamai provides instant scalability to handle spikes in traffic, delivering positive user experiences and increased customer loyalty.

Start learning more and get your copy of *Online Games: 10 Tips to Increase Play and Profitability*.
Call 888-340-4252 or visit www.akamai.com/gaming



GRANNY^{3D}

Has all the tools!

The World's Finest Run-Time Animation Engine

Silky-smooth b-spline playback • Multi-animation blends • Seamless transitions • Per-bone masking and feathering • Real-time IK • Flexible clocking • Motion extraction and prediction • Built-in animation L.O.D.

Fast, Accurate Normal-map Generation

The results are so good, we even used our low-res demo model for this ad! That's our real-time Granny you're looking at. Her high-resolution counterpart that weighs in at over ten times the triangle count.

Blazingly fast • Tool-integrated • Generates tangent-space, object-space, and displacement maps • Copy high-res textures onto low-res models with any UV mapping

Fast Mesh Deformation

Highly optimized CPU vertex deformers • Tangent space deformation • Generate GPU-compatible hardware-skinable vertex buffers • Full support for multiple vertex streams

Powerful Exporters

Export from MAX, Maya and Lightwave • Mesh, animation and texture data • Advanced b-spline fitting and reduction • Full material graphs • Text and binary annotation • Animated custom attributes

Model & Animation Previewing

Preview animations on any model • View transitions • Inspect all exported data and annotation • View mesh tangent spaces • Check texture-map assignment • Overlay bone structures

Source Code Included

Full run-time engine source code included • Clean, cross-platform design • Modular and independently reusable components

For more information:

425.893.4300

www.radgametools.com



©

THE BEST IN GAME DEVELOPMENT TECHNOLOGY



Bink
Video
Technology



Miles
Sound
System



Pixomatic
Software
Renderer