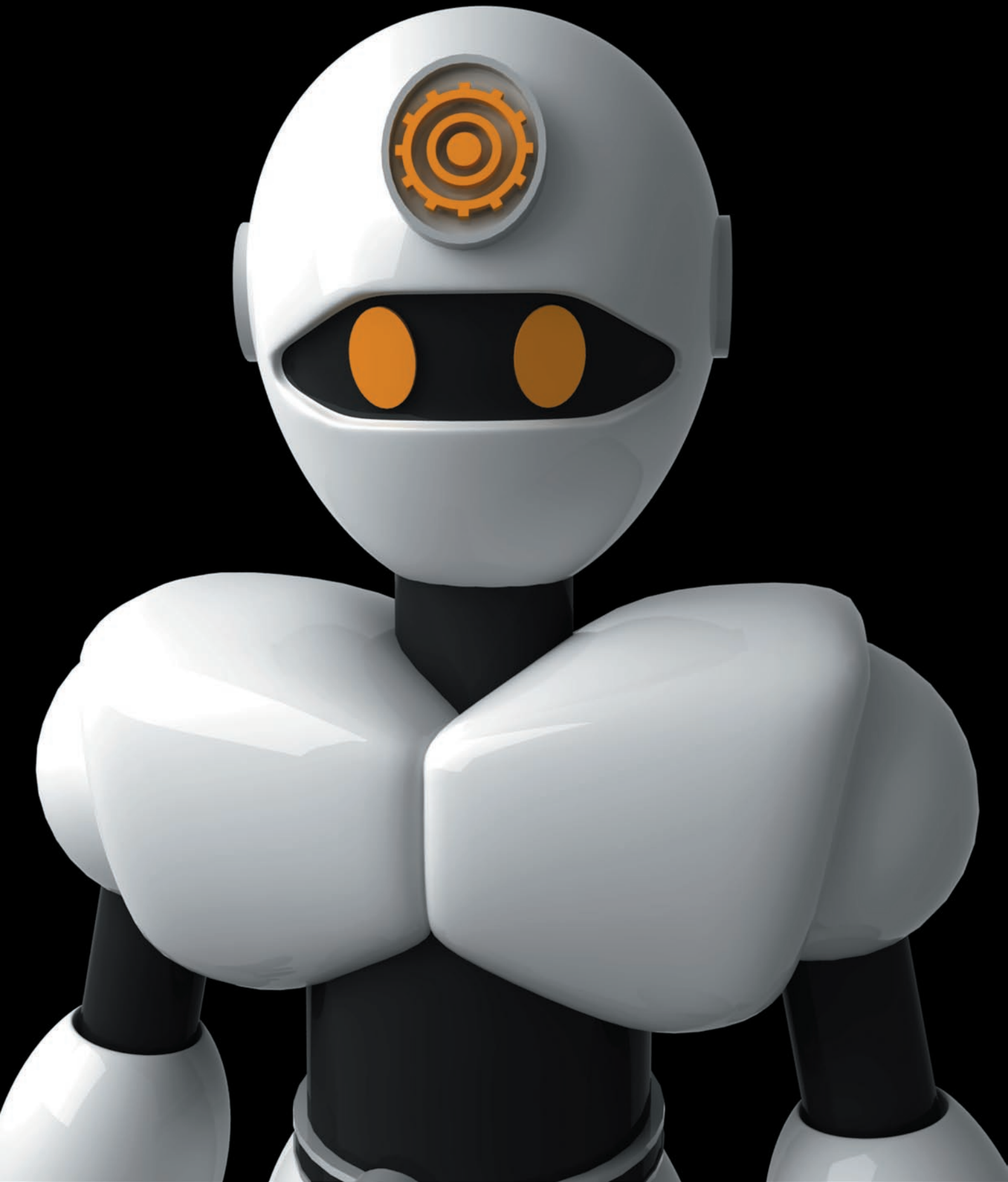
SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

APRIL 2007

DEV.MAG

CREATE • DEVELOP • EXPERIENCE



REGULARS

FEATURE

SPOTLIGHT

OPINION

REVIEW

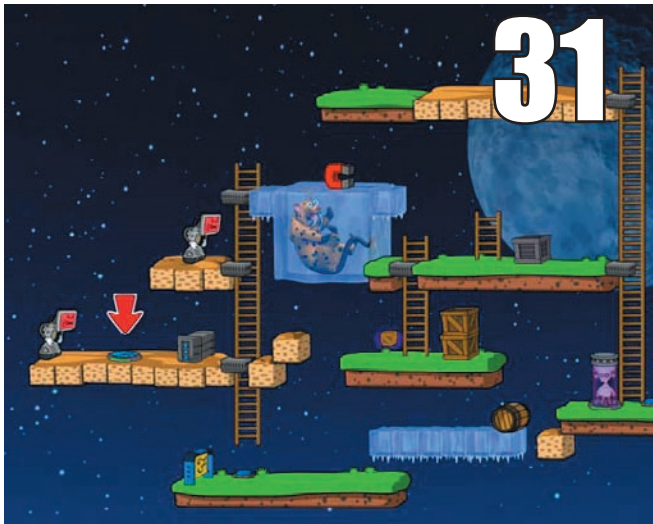
DESIGN

PROJECTS

TECH

HISTORY

TAIL PIECE



There are times when I can veritably say that I love this job. This is one of those times. For the past couple of months, we've been waiting with bated breath for the green light on several major pieces of news, one of them being Game.Dev's deal with Mindset (www.mindset.co.za/learn/) that involves a whopping R10 000 sponsorship for Comp 15, the second such prize competition that Game.Dev has had the fortune to host. Imagine sitting on that for a few months and not telling anybody about it?

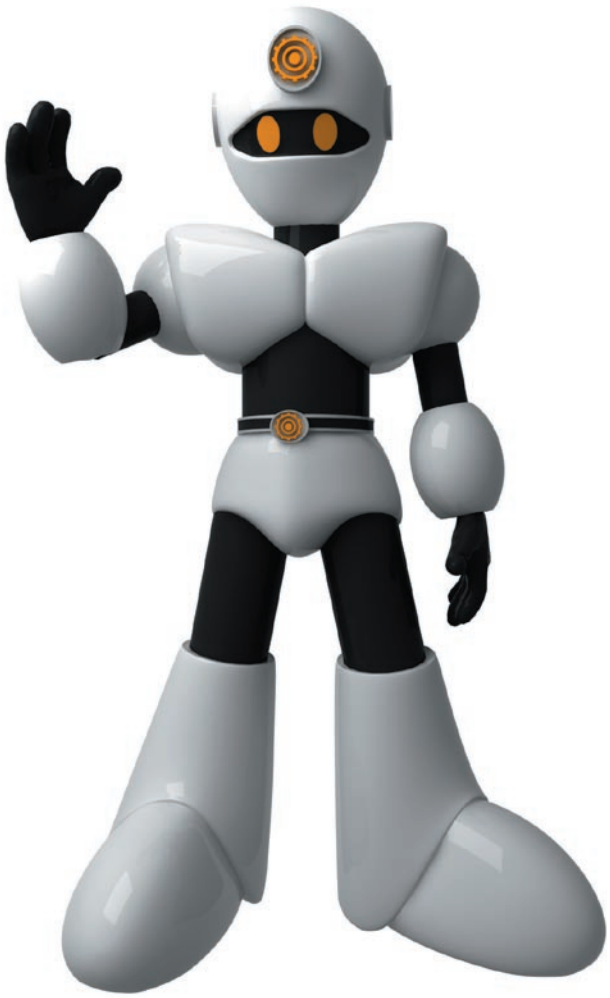
Because this competition is a biggie, we've naturally splashed reminders about it all over our pages (HINT: THIS IS ONE OF THEM) to make sure that people get the message and start participating. If you're a South African citizen, resident or passport holder, you should high-tail it over to the Game.Dev website (www.gamedotdev.co.za) and enter your own creation. Don't be shy – prizes have been reserved especially for new entrants, so developers across the board will have a shot at the money.

Competition aside, Mindset have also decided to distribute Dev.Mag and Game.Dev advertising over their datacast service to schools all over South Africa. This is a major step and hopefully the word will be spread much more quickly concerning the joys and benefits of local game development. Did I hear "TV appearances", anyone?

Concerning the magazine itself, I've already fallen in love with this issue, and I hope you do as well. Featured this month is PascalGameDevelopment.com, an amazing website with a great crew who have shown the world that not only can you make games in Pascal/Delphi, but that you can also be a huge success in the process. Turn a few pages and check it out. Our Tailpiece this month is also something well worth checking out – it was a surprising little entry that snuck onto my proverbial desk before deadline and genuinely gripped me from beginning to end (being the arrogant sod that I am, I consider my own opinion of a piece to be worth a fair bit).

Cramming all of the announcements, news points and massive features into this edition has also had another profound effect: Issue 13 is officially the biggest Dev.Mag edition released to date, meaning that you're getting even more bang for your buck. Many hands gave up their fingers to count the pages we've got filled with awesome stuff right now, so maybe it's about time you stopped reading my waffle and got to what really matters – enjoying the rest of the mag!

Editor
Rodain " Nandrew " Joubert



OOOH, WHAT'S THIS?
Say hi to our new mascot, DB (pronounced "deebee"). DB is our resident Dev.Bot, and you'll find him popping up quite a bit from now on. He's smart, he's sassy and he serves as a neat replacement for Zelda on the editorial page. As it so happens, most of DB's circuitry is committed to the logistics of game development (how convenient!), making him a grand addition to the Dev.Mag crew.

This month's opinion columnists:
Tarryn "Azimuth" van der Byl
Gareth "Garson" Gibson
Simon "Tr00jg" de la Rouviere

DEVMAG
CREATE DEVELOP EXPERIENCE

EDITOR
Rodain " Nandrew " Joubert

DEPUTY EDITOR
Claudio " Chippit " de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNER
Brandon " Cyberninja " Rajkumar

MARKETING
Bernard " Mushi Mushi " Boshoff
Andre " Fengol " Odendaal

CARTOONIST
Paul " Higushi " Myburgh

WRITERS
Simon " Tr00jg " de la Rouviere
Ricky " Insomniac " Abell
William " Cairnswm" Cairns
Danny " Dislekcia " Day
Andre " Fengol " Odendaal
Heinrich " Himmler " Rall
Matt " Flint " Benic
Luke " Coolhand " Lamothe
Stefan "?rman" van der Vyver
Gareth "Gazza_N" Wilcock

WEBSITE DESIGNER
Robbie " Squid " Fraser

WEBSITE
www.devmag.org.za

EMAIL
devmag@gmail.com

This magazine is a project of the South African Game.Dev community. Visit us at: **www.gamedotdev.co.za**.

Programming for kids

http://scratch.mit.edu/

It's not too difficult to come across tools that promise "ease of use" and "no programming experience necessary", but it's entirely different to find such a tool geared specifically at kids. Scratch is a promising (and free) programming tool that introduces children to the idea of making games or animated stories using standard programming techniques, albeit cleverly disguised by a kid-friendly interface. The tool has become very popular, and there's nothing stopping grown-ups from using it too!

R10 000 up for grabs!

http://www.gamedotdev.co.za/

Game.Dev's Comp 15 commences on the first day of June, this time with sponsoring from Mindset Learn amounting to R10 000. Entrants will have two months to create their game based on an educational premise, and the top three games (as well as the best new entrant) will be eligible for cash prizes. Check out the website, the Game.Dev forums and further on in this issue of Dev.Mag for further details.

The Everyman and the Action Hero

http://www.gamasutra.com/view/feature/1420/the_everyman_and_the_action_hero_.php

Gamasutra has published a feature on character design which takes a look at just what players are keen for in this day and age of Greek warriors, trash-talking street gangsters and infinitely powerful planet-eaters. The article takes a critical look at some of the important ideas surrounding a game character, including how free the player feels with it, how much the player relates to it and how much the player desires to be like it. Games such as Psychonauts and Planescape: Torment are examined, and the article is overall a very enjoyable read.

One game, one day ...

http://www.gameinaday.net/

For those speed-freaks out there, Gamedev.net advises that you could do a lot worse than heading on over to Game In A Day (GID), the monthly competition which has developers creating games in -- you guessed it -- 24 hours or less. The next GID is taking place on the 16 / 17 June, so get your thinking caps ready and see what you can get out of the experience! For more info, check out the website.

GAMES, DEVELOPMENT AND PASCAL

The website that has them all

Pascal Game Development, or PGD for short, is a community site for game developers and programmers to come together, share ideas and new tricks, and just enjoy the fun of making a game. It's just like any other game community, except for one difference - the programming language of choice is Pascal. Members love making games and they love programming in Pascal, it's just that simple. The PGD site is currently run by both Dominique 'savage' Louis and Jason 'WILL' McMillen. We interviewed Dom for this edition of the mag, while Jason was gracious enough to give us an overview of what PGD was all about. Here's what he had to say.

"Before there was PGD, there wasn't too much of a large port of call for a Pascal game programmer to go to for information for his or her native code language. There were a few places here and there, such as the (now defunct) Turbo site run by Michael 'turbo' Wilson, and DelphiGamer ran by Dominique Louis. But these weren't too much to go as they were small and usually run by only one or two people. Another missing factor was forums.

It was 2002 and a component suite called DelphiX for DirectX-based components was all the rage. That November, a gentleman named Ian Ashbury aka BlueCat, created a forum site called DGDev. This was probably what was needed to start bringing everyone together to meet and discuss ideas. It really helped turn what was so fragmented back then into the growing community in the years after. Information about new game and graphics libraries started to reach others where feedback could be had. Also new projects and project sites stemmed from the people that were sharing ideas in the forums. It was only six months later that I became a member, myself.

At the end of 2004 I had become very much involved in the activities at DGDev. I also became a 'news reporter' for the DelphiGamer site to help out there. DelphiGamer had turned into one of the two major news sites in the greater Delphi game programmer community, the other being, of course, the Turbo site at GameDev.net. There were newsgroups on Yahoo and other places, but I was never big on those so I couldn't really tell you how successful they were, only that they existed.

Also around that time, the team working on the Free Pascal compiler had been doing some seriously extensive work and it became more and more of a viable alternative to Delphi. It had a bit to go before version 2.0, but it was worthy of questioning why we called it the 'Delphi game programming' community back then. Borland's commercial name butcher-

ing aside, that's what it was called back then. I was never quite happy with how nomadic all of the Delphi-based sites were and I had come up with the idea of merging what content they had into one big Pascal community site. I proposed a concept I had worked out myself to the top 4 'Delphi' sites at the time. DelphiGamer and DGDev eventually joined on to what would be a merging of all the sites that would take part. The result would be one big super site for all aspects of Pascal game programming and development.

After a long year of coding and organizing the Pascal Game Development site was eventually launched in January of 2005!

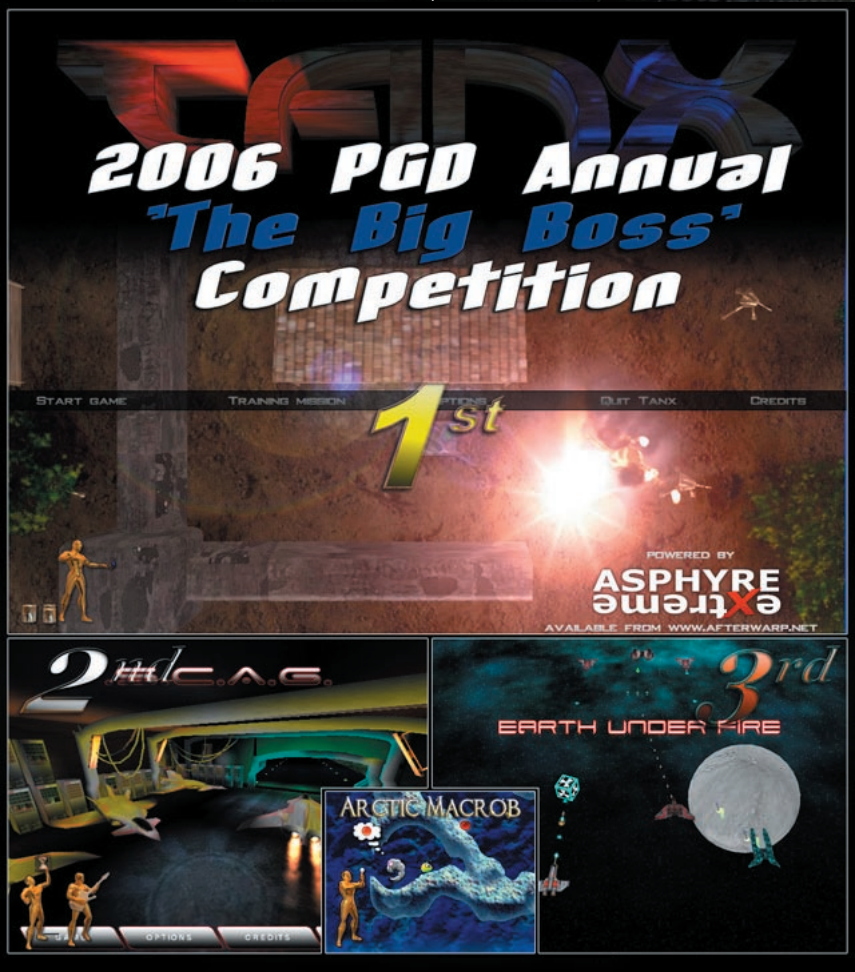
Why Pascal?

The popularity of the Pascal and Object Pascal languages has receded much over the years since C had become the default language for schools and the industry. Despite that, there has actually been a rise in its use over the last few years, specifically its OOP variant, Object Pascal. Borland played a huge role in keeping the language alive with its Delphi compiler and IDE. Another notable group is the people that have worked on the Free Pascal and Lazarus tools, a cross-platform compiler and IDE respectively. These great tools along with RemObjects Software's Chrome for .NET, are the Pascal programmer's most commonly used tools for making games.

Now most of these tools were not around when the PGD community started out, but I like to think that the very reason they exist today is due to the love of the language.

The PGD Annual Competition

The 'PGD Annual' is a yearly event in which teams from all over the world are tasked to create a game using Pascal. Any publicly available library or game engine is allowed, but the core of the game engine must be made in Pascal or Object Pascal. Each year the organisers of the competition choose a gameplay theme, select a panel of judges, write up rules, gather sponsors, and everything else that's involved.



It's a really exciting time for the site. Each year, participants, sponsorship, site traffic, seem to grow. Final entries are submitted and scored, then judged. The top 3 are then awarded with prizes provided by the sponsors. This year the sponsors have been really generous with over $6,000 in US dollars.

The coolest thing about the PGD Annual competition is that each theme is about gameplay rather than some tacky season or holiday as others have done. What kind of game you choose to make is entirely up to you so long as it contains the piece of gameplay that is asked for in the rules. This makes for a wide variety of games at the end.

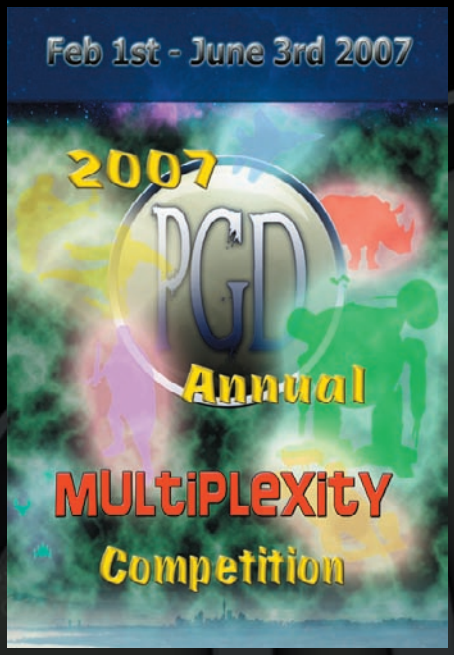
This year's theme, dubbed 'Multiplexity', is about combining 2 or more game genres together. We colour coded a chart with different game genres, and allowed teams to choose as many as they liked, so long as at least 2 colours were represented.

In 2005, 'Dogfight' was chosen as the theme where you had to go one-on-one with anything from people to tanks to spaceships. Eleven final entries were sent in.

In 2006, 'The Big Boss' was that year's theme, where the game had to be broken down into levels or stages with bosses you had to face at the end of each and one big boss to fight at the end. It was a very successful event that year with 20 final entries.

PGD at the IGF

Before last year's PGD Annual, Dom and I wanted to do something special for the first place prize. It was supposed to be a bigger competition and we really wanted to up the stakes. I was thinking of something along the lines of a publishing deal of sorts, but we



couldn't find anyone willing to do the deed for us. It was then Dom's idea to flip the entry fee for the Independent Game Festival's main competition. The idea worked out perfectly and we had our amazing prize all set.

The 1st place winner that year, Dirk Nordhusen in Germany, was more then happy to take his game 'TANX' to the 2007 IGF competition. His game won out over the rest and was generally very well received by all PGD site members. Since the IGF is a bigger event than the PGD Annual, Dirk wanted to put a little more into his game to be able to compete with those that would have likely been into development for years longer than he was. So I had offered to join to help him with story and design, and we started to work on the transformation of 'TANX' into 'Iron Strike: Stormfront' with only four months to go.

Working with Dirk on his game was a great experience. He is a really talented guy and has a lot of energy that you see come out in his games. We were able to really accomplish a lot in a short amount of time, especially considering that we both already had full time jobs.

We just barely managed to get the game completed, despite some remaining bugs, and submitted to the IGF's FTP servers. Alas, we did not place in the preliminaries. Dirk has since vowed to revisit his game engine at a later date and fix a lot of the bugs before releasing to the public.

More Glory To Be Found

It's not only the PGD Annual winners that have found their way into the professional arena. Others have gone their way into commercial development. Steve 'Sly' Wilson has been around since 'the old days' and he works for Krome Studios, the company that makes Spiro and other console based games. Dom has done some work on a few games too; 'Hero X' and 'Siege of Avalon', both of which were made with Delphi. The latter was one of the first 'episodic content' based commercial games and its engine is now open source. Michiel and Frank of NecroSOFT are now under apprenticeship at Team 6. Abra Academy, created by a member called 'wagenheimer' is a puzzle game that was recently published by Big Fish Games. The authors have stuck a deal to make a sequel and are also looking at the possibility of porting it to both Mac and Linux. Eric Behm who runs Magic Storm has released a few shareware games and is working on his latest, 'The Immolate'. Mars Miner, a Bomber-man inspired game, is another recent shareware release by another member called 'RetroStyleGames'.

Plus a whole cast of others... Maarten Kronberger, Jeremy Darling, Mike Weiring, Dave Kerr, Christina Warne, Christen Fihl, the list goes on.

And Now Some More Cool Stuff

Obviously commercial gain isn't the only thing on the minds of Pascal game developers. A lot of these people simply enjoy making fun games and creative developer tools. Both Free Pascal and Lazarus are open source, with a rather large following onto their own.

Other such projects include translated API headers, game media, and networking libraries such as JEDI-SDL, DelphiX, Asphyre, GLScene, Omega, Phoenix, S2DL, Clootie's DirectX headers and Noeska's OpenAL headers.

There are even game engine projects such as the LEAF game engine by Relfos and the Rage 3D Game engine from SULACO. Or if you prefer there is the open sourced Quake 2 engine translated to compile under Delphi.

Some of the coolest new stuff coming around the corner these days has been the gaming platform support projects. Francesco Lombardi has done extensive work on adding support for the GameBoy Advance and Nintendo DS to the Free Pascal compiler for a project called 'FPC 4 GBA' in the last year, and almost half of all hardware for both platforms is now supported. Due to his work it is now a main-stay target platform for the compiler. There is now a project inspired by 'FPC 4 GBA' to add support for the GP2X handheld gaming platform that uses Linux as its core OS.

Another recent platform to receive such treatment is Microsoft's XNA. Dominique Louis and William Cairns have been working away with RemObject Software at adding full support for the gaming framework into the Chrome compiler. Several demos have been put together and results are quite promising so far.

Games, Oh the Games

Of all the publicly released games you can find, which are almost countless, just a few of the most recent and stunning ones include a world domination game called 'Projekt W' by Sascha Willems, an adventure-puzzle game by Dirk Nordhusen called 'Valgard's Fate', and the Genesis Engine next gen FPS game engine by Luuk van Venrooij, which has some of the most impressive 'real world' graphics I've seen in any PC game.

'Last Dawn' is a zombie survival platform game created for an Australian demo contest. It won 5th place. 'Uber Chess' was a university class

project by Michiel 'NecroDOME' Gijbels that turned into a visually amazing chess game. And let's not forget 'Guns Reloaded' by Jason Farmer of Cerebral Bicycle Co., a revamped version of his original artillery game, 'Guns'.

There are so many games that I can go on practically forever about them.

So What's So Special About PGD Anyways?

The first thing that comes to mind of the benefits of a site like this is the forums. Not only has the site been run by a mature group of people, but everyone is generally friendly and helpful. It's a common thing to help one another at the site. Probably the most prominent forum is the 'My Projects' forum, where members can post about the latest and greatest projects that they are working on. It can be anything from a game to a development tool, or from a commercial game to an open source one.

We also have a great directory on the site called the PGD Library. It's a sizeable catalogue of articles that can be found on external sites related to a specific subject like graphics, networking, 3D animation, music design, games industry, postmortems, online magazine issues (like this one!) and many other game development related topics. Of course, it grows all the time and is worth checking out from time to time. Then there are our own staff published articles, which are always a great read.

Then of course there is the News section where we'll report all new goings-on related to Pascal, game projects, other game competitions, or game development and the game industry at large. We like to cover a wide range of content so that we not only cover the basics of making a game with Pascal, but can find what other things would obviously be required in the making of that game as well. The rule of thumb is inclusive rather than exclusive.

You should know that not all PGD members are programmers. Some are also artists and musicians. Although we don't have a huge number of them, they do exist here and come back to contribute from time to time.

On a personal note, I have been a part of the site and the community's growth for the last 4 years and it's been a blast. I can't think of another community on the net that I'd want to have had this kind of an experience with. I love making games and I love the Object Pascal language. Without having done these 2 things on my downtime from the military, I'd likely have snapped out of boredom at some point. I'd like to extend a huge thanks to everyone that had a hand in the growth of PGD over the years, no matter how big or small. This site isn't just a place for us Pascal programmers to swarm, but a community of creative people that has a passion for making great games."

Pascal Game Development website:
www.PascalGameDevelopment.com

German Delphi / OpenGL community:
www.delphigl.com

Free Pascal website:
www.freepascal.org

Lazarus website:
lazarus.freepascal.org

CodeGear website:
www.codegear.com

RemObjects Software website:
www.remobjects.com

A CHAT WITH DOMINIQUE LOUIS

Site co-manager: PascalGameDevelopment.com

What made you personally get into game development using Pascal?

I wrote my first game in C way back in 1986. It was a side scrolling shooter using ASCII characters on the DOS prompt. The enemy was '<', bullets were '-' and you were 'X', which, to me, was an X-Wing fighter. Though this worked, it was quite laborious getting it all working for such a simple game. I then learned C++ and object-orientated programming. Then in 1995 I heard about a RAD IDE that was coded in itself. That is when my love affair with Delphi began. v1.0 worked on Windows 3.11. Naturally, since I already had a love of playing games and had dabbled in writing games I looked around to see if there were any Delphi game programming resources and I don't remember finding any. When Delphi 2.x came out for Win32 I was totally hooked. To me, Delphi was – and I think still is – the most productive Win32 development tool (though C# is catching up fast). I have always been more interested in getting things done rather worrying about the internals/low levels. So for me Delphi was ideal - it was a tool that allowed you to work at the level or complexity that was required at that moment. I eventually found Goodberry's game dev site (http://www.geocities.com/goodberry/) and shortly afterward the delphigames yahoo group (http://tech.groups.yahoo.com/group/delphigames/) was created. I've been using Delphi and variations thereof, like FreePascal, ever since for various game related projects.

Not only does Dominique program, but he plays the piano too! Is there nothing these Pascal types can't do?

What prompted the decision for a site geared specifically at Pascal developers?

Before PGD was conceived, Dean "technomage" Ellis and I created a site called www.DelphiGamer.com, which we wrote from scratch using PHP. It was designed to be a news site for the Delphi/Pascal game development community. Our vision was always to show that Delphi was just as capable as C/C++ of producing quality games. Commercial games like Age of Wonders (1 & 2), Siege of Avalon and Hero X were all proof that Delphi-developed game were marketable. Actually, Hero X is still available from the Atari site (http://www.atari.com/us/games/herox/pc). The DG site went well until Real Life encroached on our ability to post news

regularly. Meanwhile other Borland/Pascal sites had also sprung up, like turbo (no longer active) and more specialist sites like Delphi3D (http://www.delphi3d.net/ - but no longer maintained) and Sulaco (http://www.sulaco.co.za/ - started by the late/great Jan Horn and now maintained by Maarten "McCLaw" Kronberger). Shortly afterwards, the DGDev forums were formed and Jason "WILL" McMillen became one of DelphiGamer's news posters. After some discussion about the fracturing of the Delphi Game community we decided to merge the DGDev forums with DelphiGamer. PGD was formed and went live in January 2005, the idea being that we would become the central point for all things Pascal and Game Development related.

What appeal would PGD have for game developers in general?

PGD is mainly geared to cater for game developers who have a love of using Pascal as their language of choice. We discuss all aspects of game development and even talk about porting games to Nintendo's Game Boy and the GP2X games console.

Do you guys cater for beginners?

We certainly cater for beginners and always welcome new blood to the fold. Developers of all abilities are welcome. We can all learn from the gurus and the better developers can always give something back to the community and teach the newbies. I think overall we are all a fairly friendly sort of crowd. If there are any issues we talk about it and try to resolve them.

How does a site like yours market itself, especially in the "early days"?

We have mainly marketed ourselves by contacting more general game sites and letting them know what sort of events we have on. Site like GameDev.net or DevMasters and the like. We also post messages about our competitions on CodeGear newsgroups or MSDN site if anyone asks about Pascal related questions.

What sort of obstacles have there been?

The main obstacle has always been that most game developers see Pascal as not being equal to C/C++. Our aim all along has been to show that this it a falsehood. As mentioned earlier, there are commercial games out there that were written using Delphi - there was also a game project to port the Quake 2 source code to Delphi (http://sourceforge.net/projects/quake2delphi) which showed that the engine worked just as well as the original C code. The only other obstacle is that 99% of SDKs only target C/C++ as developers, so this usually involves a team of Pascal coders porting the bindings over. Like Alexey "Clootie" Barkovoy (http://www.clootie.ru) has done for DirectX. This last issue will probably become a moot point once more game developers use XNA and the managed DirectX API and they can use any managed language.

What has been your greatest success to date?

My greatest success was working on Siege of Avalon (a 6-chapter episodic content RPG)

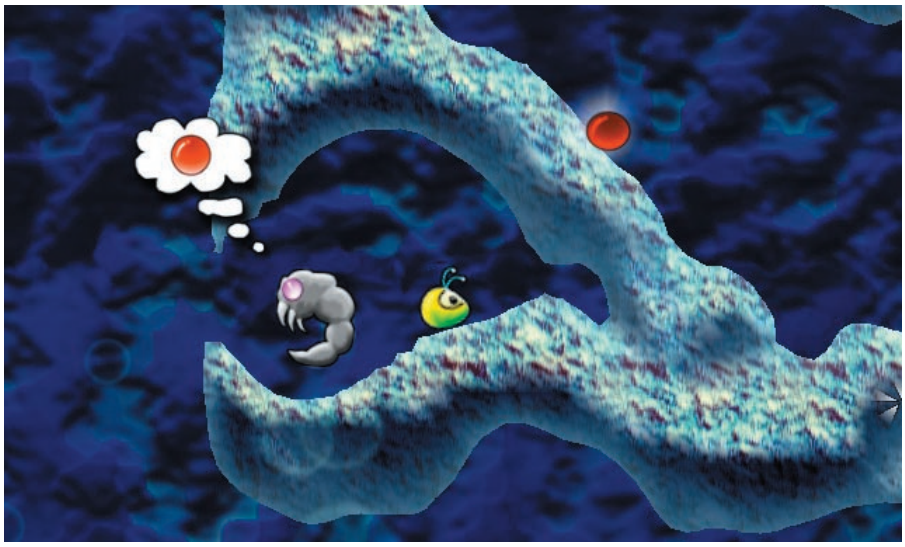
and Hero X as a remote developer. Seeing my finished game on a store shelf was one of the greatest buzzes in my life. I happened to be travelling through Singapore at the time and walked past a software shop and there on the rack was a copy of Hero X. Of course I had to buy it. I am also proud of the work I have done on the JEDI-SDL. They are Pascal headers that talk to the excellent SDL libraries. So far I have seen it work on Win32, Linux, MacOS X and recently some of the PGD guys got it working on Game Boy Advance and the GP2X console. If I can see it working on a Wii, PSP and PS3, then my dream will be complete. I'm also working on porting SoA, now that it has been open sourced, over to SDL so that it can be played on more platforms. It's a very under-appreciated game and had some features which only recently started appearing in 3D games. Its depth of story is second to none.

Can you tell us about the latest PGD competition? And what about PGD's plans for the future?

The latest PGD competition (it's an annual competition) is called Multiplexity. The idea was that we wanted to push the entrants to write something innovative that incorporated multiple genres and complexity. After Jason and I did much brain storming, I thought that Multiplexity, as an invented word, fitted the bill. Hence the name. Our aim was also to get entrants to try to work to certain deliverable deadlines to simulate the idea of delivering to a publisher. The winner of this year's competition will also have their entry fee into the IGF (www.igf.com) paid for, amongst other prizes. Some of our sponsors have been with us since our first competition 3 years ago. I've had a fairly good relationship with CodeGear (formerly known as Borland) so when I asked them about the possibility of sponsoring the competition, they said yes. We also approached a few other companies in the hope that they would help out with prizes. A few said no or never go back to us, but quite a few said yes, as you can see.

Next year, we plan to revamp our competition structure in the hope of attracting even more entrants and, of course, more sponsors. We will be partnering with DelphiGamer (www.DelphiGamer.com) and www.SavageSoftware.com.au in an effort to provide a showcase site as well as a distribution channel for Delphi/Pascal developed games to be downloaded, played and bought. The next year will be very interesting indeed.

NANDREW



Quick Questions ... 3 ... 2 ... 1 ... Go!

How many games have you made?

I have worked on 2 commercial games, namely Siege of Avalon and Hero X, using Delphi.

Any games you'd recommend from the PGD community?

In my humble opinion, the PGD community has produced some excellent games over the years. In fact, just jump over to the main PGD competition page and decide for yourself.

What famous cartoon character would you compare yourself to?

The cartoon character I most identify with is Foghorn Leghorn. I think I'm always dishing out advice when people least need it and giving them the wrong advice when they do need it.

What are you playing at the moment?

Right now I'm playing Spellforce 2.

STOP PRESS

MAKE A GAME AND GET YOUR PIECE OF R10 000

Here at Game.Dev, we believe that games are more than just entertainment or a silly fad — games can (and will) change the world and make a difference. Mindset Learn are keen on doing just that, making a difference. So they're challenging us – and you – to produce the next generation of entertaining and meaningful games. That challenge takes the form of:

R5000 for first place,
R2500 for second place,
R1500 for third place and
R1000 for the best new entrant.

So all you have to do now to get a shot at that prize money is put together a game that's fun and focuses on a concept you enjoyed or found challenging at school. You don't need to produce a fully fledged "educational" experience, in fact we're looking for quite the opposite: fun games that are enjoyable because they're well-designed and entertaining, you just happen to be playing within rules or a setting that match something you've learned. Think "guerrilla learning" and you're on the right track!

The competition begins on 1 June 2007, and the deadline for entries is 31 July 2007. That's two months that you have to make the ultimate game and win some big money!

For more details, head on over to the Game.Dev website at http://www.gamedotdev.co.za

game·dev

It`s what you wish you were doing.

"UNIQUE ISN'T AWESOME"

One of 2006's juggernaut successes was Iron Lore's Titan Quest. A click 'n' slash monster mash that saw players doing little more than scuttling around a variety of locales, killing stuff, and collecting stuff to help them kill more stuff, TQ was the embodiment of "bog-standard action RPG". For all intents and purposes, it was Diablo with harpies and Corinthian armour. And that's why everyone loved it.

There's an old maxim that sagely admonishes, "If it ain't broke, don't fix it", and this is readily applicable when it comes to game development. There's no need to reinvent the wheel when, with its rugged rotary design conveniently and purposefully engineered for isobilateral locomotion, it's already quite capable of rolling along without further assistance. Similarly, there are certain fundamental concepts in game design that just work, e.g. "shooting stuff is cool", "collecting stuff is cool", and "villainous mutated purple tentacles with despotic ambitions are cool", and the aspiring developer would do well to remember this. The reality is that the vast majority of gamers really just want more of the same.

Look before you think before you code

So you've decided to make a game. Fantastic. Is there already something else out there that - even remotely - resembles what you have in mind? Chances are, there is. So get your claws on a copy of it, and tinker with it. Think about it. What is it about this game that really engages you? What's a total buzz-kill? Make a note of your conclusions, and plot your way forwards from there.

Let's go back to our previous example. The beating heart of any action RPG is its item collection. Players might wax indignant about levelling, character development, narrative arcs, and other "integral" elements, but they'd be filthy liars. All they're really interested in is the next cool item they're going to trip over, and that's what keeps them click-click-clicking away, red-eyed and raw-fingered, into the wee hours. Now Titan Quest wasn't the only potential usurper to Diablo's long-held gilded crown. Before TQ, there was Sacred. But Sacred just didn't cut the proverbial yellow condiment. Why? The items were rubbish. Ergo, if you're going to develop an action RPG, the collectible stuff had better be the top-drawer shizniz. Trying to hinge your action RPG's playability on an ever-evolving hairstyle, on the other hand, is quite likely to fail wretchedly.

Hello Dolly

"Imitation," says the wise old owl, "is the sincerest form of flattery." We might adapt this to say, "Imitation is the surest form of solid gameplay." This isn't to say that originality is without merit (we're all on wriggling tenterhooks for Spore, after all), but there's nothing whatsoever wrong with simply reproducing something that's gone before - especially when you're just getting started in this bewildering game development business, and your head's a whirligig of nebulous ideas. Titan Quest is an entirely unabashed Diablo clone, and has proved to be an immensely entertaining romp not only because you get to punch centaurs, but because it's comfortingly jump-in-and-play familiar, and all awash in softly dappled shades of nostalgia.

AZIMUTH

The Dev.Mag staff does not expressly support or agree with the views of guest columnists in this section.
In fact, the Dev.Mag staff does not expressly support or agree with the views of Dev.Mag staff writers in this section.
It's a crazy world, isn't it?

"BACK TO BASICS"

When you are a programmer you sometimes forget the basics, and I don't mean the Basic language. I'm talking about the basic assignments you make in your code. I don't like theory much myself, so I seldom come across anything - much less read it - that handles the basics of a programming language.

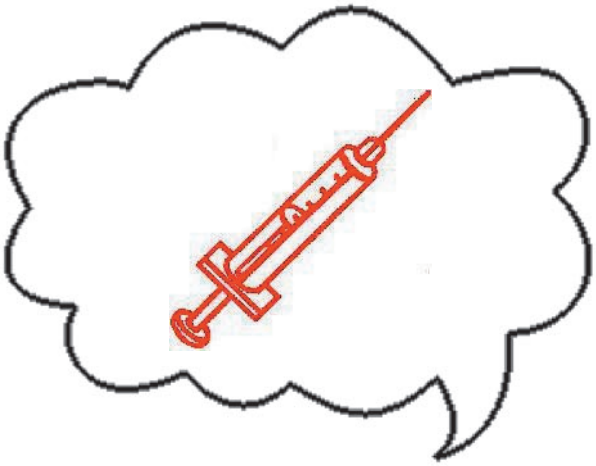
To my surprise, I found out about a month ago, after almost five years coding in Java, that when you assign a name to a new object you actually only create the object independent of its name, where the name is only a reference (pointer) to said object. This of course has lasting consequences if you did not know that. For example, if you take that object and assign it another variable name, you will not copy the object but only have two differently named pointers (of the same class) pointing to exactly the same object. Just thinking of how not being knowledgeable of the basics may have an effect on a large program is enough to hang my head in shame at my error. This of course brings me to the next step: Manual execution. It's important to be able to execute a program in your head, which consists of less than 100 written lines. If you can't do it, then train yourself to do so. Not only will this help you understand what you are coding, it will also enable you to code more efficiently, effectively and more timely the more you think like the machine you're coding on.

This all of course brings me down to my last piece of advice. Know what you are coding before you code! I remember Computer Science class back in High School started with these Weiner-Or(?) Diagrams or something, I remember hating them because they were just damn stupid, writing out everything you want to code into pseudo-code. That is not what I want you to do. I just want you people to know what you want your program to do, and if it is a big program pull out that writing block and write down what you want to do. Not only will this help you understand what you are coding, but it prevents staring at the screen or coding nonsense. Of course, only put the basics down as well, all the additions that you want to make to your program can be added in later, except if it is fundamental to your program.

I know that all these new IDEs offer incredible debugging support, but please try and not rely on it, as it will only make you a better programmer if you don't. In the end, do not let the limit of your programming skill, due to lack of understanding in the basics, influence your program's outcome, but rather the limit of your imagination.

GARSON

The Dev.Mag staff does not expressly support or agree with the views of guest columnists in this section.
In fact, the Dev.Mag staff does not expressly support or agree with the views of Dev.Mag staff writers in this section.
It's a crazy world, isn't it?



"ADDICTIVENESS = BAD"

Yes, you heard it right. In fact, after reading a thread on people suggesting how they would change the gaming industry, a mobile developer, Flint from Smallfry Mobile, said that he would let go of the term "addictiveness" when marketing a game.

If you market your game as "addictive", you might lose potential customers. You might be baffled by this statement, but it has its merits.

We, as long time gamers/developers find that a game with an addictive quality is obviously a good thing, but if you look it through the eyes of the general public, it can be detrimental. The public still see "addictive" as something negative, something that you become obsessed about, something that you can't put down. The obvious connotation is drug abuse.

I think it still depends on which market you are aiming for. The bulk of the mobile/casual market is clearly not hardcore gamers, so they would still see "addictive" as a bad quality of a game.

This doesn't say that your game does not need to be addictive. Heck, if a game can be addictive, it should be by all means, just be careful when marketing your game to a certain market as "addictive".

What is the moral of the story? Be careful what labels you give your game when aiming for a certain market.

TROOJG

If you fancy saying something about game development and have enough faith in your writing ability to do so, feel free to submit content to our monthly opinions section.
Send your work, 400-500 words in length, with the subject title "Opinion Submission" to devmag@gmail.com, and you may just wind up in our pages. Please note that we reserve the right to decide what material is suitable to publish in the magazine.

The Dev.Mag staff does not expressly support or agree with the views of guest columnists in this section.
In fact, the Dev.Mag staff does not expressly support or agree with the views of Dev.Mag staff writers in this section.
It's a crazy world, isn't it?

BOWMASTER PRELUDE

Developer:

Lost Vectors

Year of creation:

2006

Website:

www.lostvectors.com

Genre:

Action

Flash games are commonly frowned upon in the game development crowd, mostly due to their low average quality. Few exist that require more than a modicum of thought to play, or feature gameplay with any decent amount of depth. But there are exceptions.

Bowmaster Prelude is one such exception.

At heart, Bowmaster runs under a basic capture-the-flag premise. The enemy will dispatch waves of units in an attempt to capture the flag from your castle and return it to their own. You can complete levels either by capturing the enemy's, or by defending your own against all waves of onslaught. As an elite archer, it is your charge to help secure your victory using your bow as well as numerous other skills that you may purchase with the gold earned from your exploits. Skills include arrows imbued with fire, as well as skills that allow you to summon meteors, storms, or your very own units that can capture the enemy's flag. Over time and with use your skills will gradually become more powerful, gaining the ability to destroy enemies with a single shot or even more than one at a time.

The game also features a novel way in which to vary game difficulty. Rather than creating more enemies to fight, or making them harder to defeat, the game changes how you aim your bow. At the easiest level, you simply click where you want your arrow to go, and your Hero will fire at that point exactly. At higher levels, you will need to adjust for gravity. The highest levels work in a method that simulates pulling back the actual bowstring and requires some practice to master.

The game does little to vary the experience, though, and - especially since this is a more casually paced game - some gamers may become bored with it. There are few enemies, and once all their weaknesses are discerned, the game only really varies in landscape, and that is little more than a subtle change.

However, most gamers should be able to find considerable enjoyment in Bowmaster, no matter how monotonous it may become. Perhaps it will hold you enough for you to complete it, perhaps not. What is certain, however, is that the experience is worth the time you spend on it.

CHIPPIT

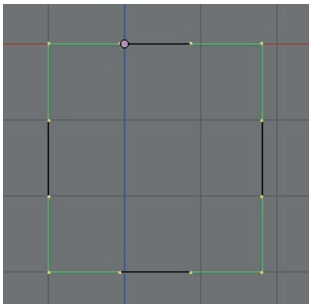
BLENDER INTERMEDIATE TUTORIAL: Modeling with curves

We all meet things that challenge us, and mostly, if we are not forced to face it, we shy away from those things. Modeling in 3D with CURVES, or creating designs in Photoshop/ Gimp/ Paint Shop with CURVES is one of those challenging things.

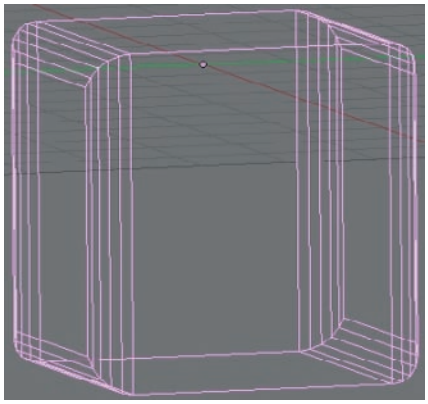
This tutorial deals with CURVES, so you'll need to make the decision before starting out that you will be willing to learn a new concept that will frustrate you at first, but save you huge amounts of time later on.

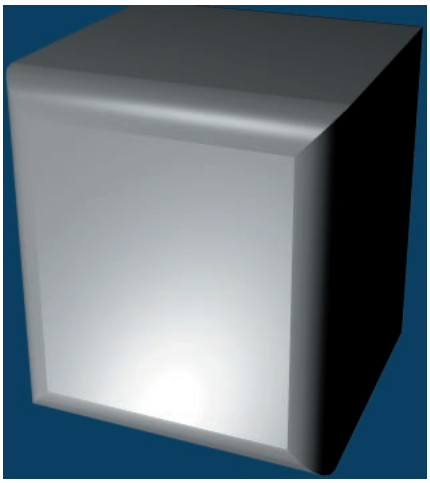
What makes 3D curves good to work with? Simply put, you can use a single line to draw the outline of a shape, and ask your software to make that shape into a 3D object. Not only that, but you can demand smooth, rounded edges (bevels) without modeling them.

Example: With a curve (yeah, I know it's a straight line, but see, the line is a "curve" object), I draw a square ...



Using buttons built into Blender, I make a cube with rounded, nicely beveled edges...





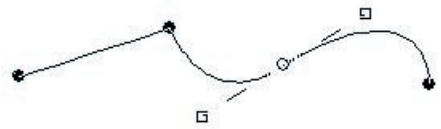
Now, you may challenge me by asking "How is this different from modeling the cube with a mesh?"

Well, I haven't changed the basic shape that I drew. I can adjust thickness and depth of the bevel in a non-destructive manner. Also, I can adjust my initial shape, and the 3D shape will adjust with it.

This tutorial will teach you how to:
1.Use Curves
2.Build Text
3.Build game screen assets

I'm going to keep this simple:

A curve is a line, connected by points. These points can be changed to either point straight at the next point, or draw a curved line to the next point. The "handles" attached to the point allows you to change the shape.

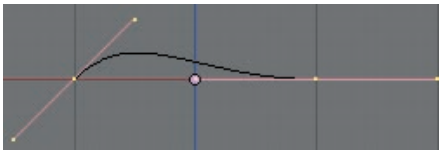


Make sense?

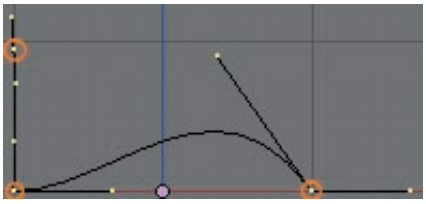
Have a look at what a letter "P" might look like: (Image prepared in The Gimp)

Can you see that it would consist of straight as well as curved sections?

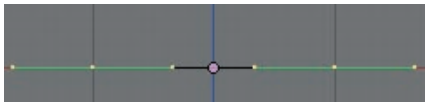
Before we do it in Blender, you need to learn how to change the manner in which a point connects to the next point. The direction of the handles determines the shape of the connecting line, and the type of handle is indicated by colour – either pink, black or green:



Aligned: Shortcut HKEY



Free: Shortcut HKEY

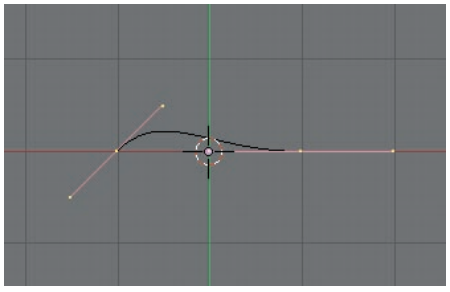


Vector: Shortcut VKEY

Remember that you can always refer to the Blenderwiki (wiki.blender.org) for additional tutorials and information.

So, let's start modeling in Blender 3D
Start a new Blender file.
Select all objects (AKEY).
Delete all (XKEY)
Change to Top View (NUMPAD7KEY)
Press SPACEBAR and select
Add --> Curve --> Bezier Curve

You should see something like this:



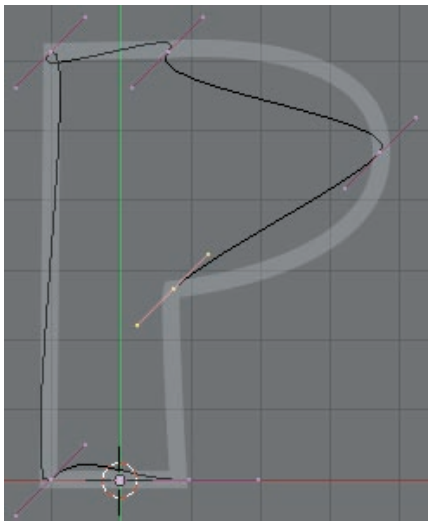
Make sure that you are in wireframe shading mode. Now, let's build a capital letter "P".

Select the point on the left (RMB on the middle dot). Press EKEY and then YKEY. This will extrude the point and constrain its movement vertically on your screen.

If at this stage the extruded point does not appear to move upwards or downwards with a mouse movement, you probably didn't add the curve from the Top View. In this case, start again.

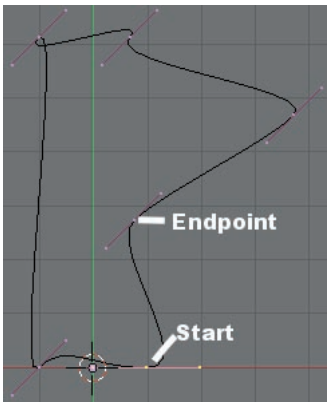
Now, complete the outline of the capital letter "P" by extruding the point on the curve. For now, don't worry about the type of handle (don't worry about what the curve looks like).

Work until your curve looks like this: (I have overlayed a vague "P" outline in GIMP to indicate the basic shape for you.)



We need to close the shape of the "P". RMB the start point on the curve, making sure that you now have the two endpoints selected, then press CKEY.

This action closes the curve ends, making a closed shape.

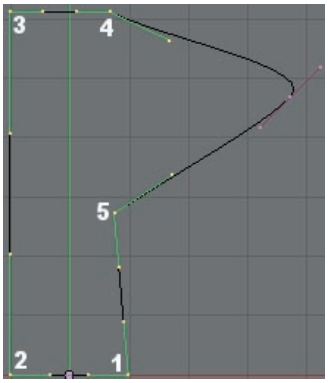


Now, let's analyze the shape of the "P", to determine what types of handles we should define.

Firstly, we know that there are several straight edges on the curve. So, we select (RMB + SHIFTKEY) the points that define those straight edges, and assign their handles accordingly.

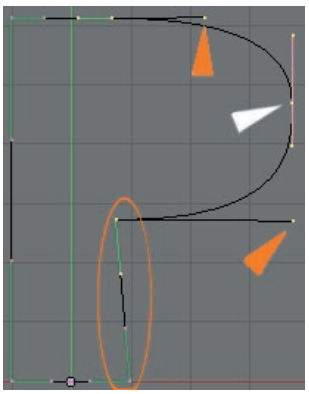
Select the points as shown in this picture, and press VKEY.

Please take note of what this key does: it changes the effect of the handles so that the line runs absolutely straight between two points.

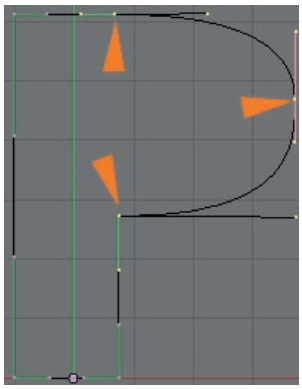


To correct the rounded aspect of the "P", grab (GKEY) one of the handles of the point indicated by the white arrow. Move the handle so that the pink line is vertical. You will see the curve changing shape. Now grab and move, one at a time, the handles indicated by the orange arrows. You'll see the handle changing colour from green to black automatically, indicating that the type of handle has changed from vector (straight) alignment to free alignment.

The orange circle indicates that one of my straight lines is skew. I correct this by moving the bottom point (at the edge of the image) to the left, so that the line is pulled straight.

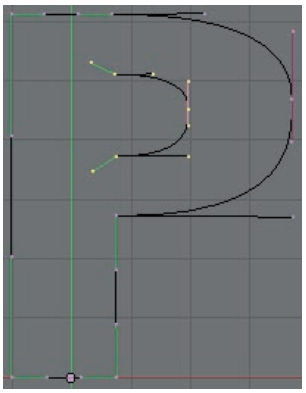


We need to create the inside of the letter. Select the point as shown, then follow the shortcut keys closely.



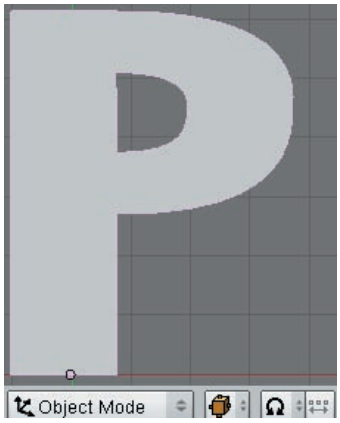
RMB + SHIFTKEY (select the three points)
SHIFTKEY + DKEY (copy/ duplicate)
SKEY (scale)

Copy the points and scale them to a size smaller than the initial round edge. Move the copied points into position with the mouse. Select the two open ends and connect them with the CKEY.



When you have closed the ends, you can switch to shaded view with ZKEY to see the result. Exit Edit mode so that you are in Object mode.

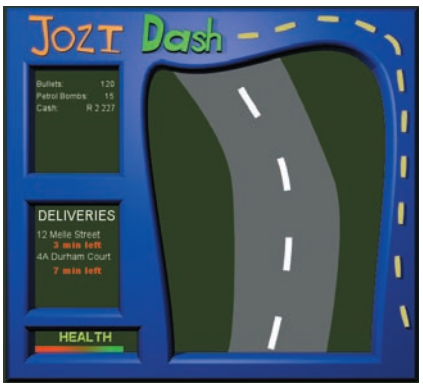
Go to the Object's edit buttons (F9KEY). Rotate your object so that you can see it from an angle. Locate the curve editing buttons as indicated by the arrow.



Use the "Extrude" button to give the object some depth. Then add some "Bevel Depth". "BevResol" will smooth the bevel. If your object grows too much in size from adding Bevel Depth, use "Width" to scale it back down.

Well done. Now see if you can understand how I built the following curves. I decided to build a gaming interface for a game called "JOZI Dash" (shown below).

By drawing a 2D curve I can shape the screen to my taste, render it with a transparent background, then import it into my game application and start adding graphics and programming.

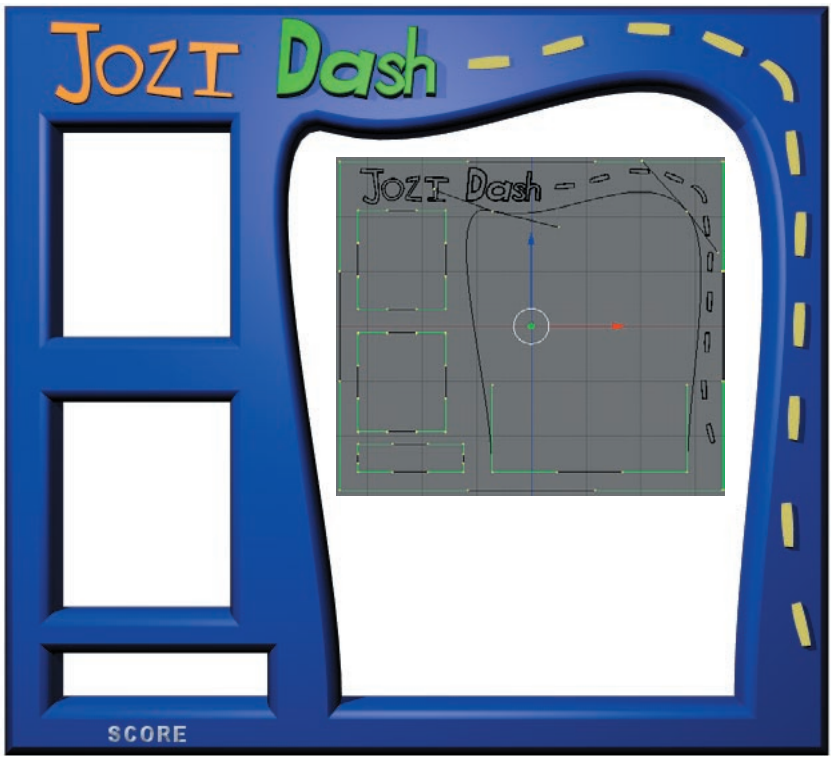


If I should feel that anything won't fit, I can always go back to my 3D app and adjust the curves for a new render.

You can always load images in the background and draw your curves on top of them. This is the best way to approach building logos and text according to specific fonts.This technique is detailed in the Blender manual that you can find on the internet at www.blender.org.

There we go. I hope that this tutorial sparked some interest with you, and that it may offer you some very nice solutions in the future.

?rman



```
<LINK REL=stylesheet TYPE="text/css
<META HTTP-EQUIV="Content-Type" CO
<META NAME="GENERATOR" CONTENT="Mo
<META NAME="DESCRIPTION" CONTENT="
<META NAME="KEYWORDS" CONTENT="jav
<SCRIPT language="JavaScript">
```

HTML FOR NOOBS - PART 3: Good Web Design

Now that you know the basics of building a webpage, it's time to understand the most important part. Everyone hates a badly designed web page so pay attention.

What to do:

Text

- Background should not interrupt the text
- Text is big enough to read, but not too big
- Columns of text are narrower than in a book to make reading easier on the screen or else nobody will read it

Navigation

- Navigation buttons and bars are easy to understand and use, keep it simple
- Navigation is consistent throughout web site.
- Navigation buttons and bars provide the visitor with a clue as to where they are, and what page of the site they are currently on.
- A large site has an index or site map.

Links

- Link colors coordinate with page colours. Basic colour combination, people!
- Links are underlined so they are instantly clear to the visitor.

General Design

- Pages download quickly
- Pages should generally have some content immediately displayed on the screen. A banner that is 600px high is going to obscure the content, nobody likes scrolling down.
- Break up large paragraphs of text.
- Every web page in the site looks like it belongs to the same site. There are repetitive elements that carry throughout the pages. (Keep it consistent)

What not to do:

Backgrounds

- Default gray color
- Color combinations of text and background that make the text hard to read
- Busy, distracting backgrounds that make the text hard to read

Text

- Text crowding against the left edge
- Paragraphs of type in all capitals
- Paragraphs of type in bold
- Paragraphs of type in italic
- Paragraphs of type in all caps, bold, and italic all at once
- Underlined text that is not a link

Links

- Default blue links
- Blue link borders around graphics
- Links that are not clear about where they will take you
- Links in body that distracts readers and sends them off to remote, useless pages
- Text links that are not underlined so you don't know they are links
- Dead links (links that don't work anymore)

Graphics

- Meaningless or useless graphics
- Thumbnail images that are nearly as large as the full-sized images they link to
- Missing graphics, especially missing graphics with no alternate labels
- Tables
- Borders turned on in tables
- Tables used as design elements, especially with extra large borders

Blinking and animations

- Anything that blinks, especially text
- "Under construction" signs
- Animations that never stop

Junk

- Counters on pages, who cares?
- Too many little pictures of meaningless awards on the first page
- Frame scroll bars in the middle of a page

Navigation

- Unclear navigation or overly complex navigation
- Complicated frames, too many frames, unnecessary scroll bars in frames
- Orphan pages (no links back to where they came from, no identification)
- Useless page titles that don't explain what the page is about

General Design

- Frames that make you scroll sideways
- No focal point on the page
- Too many focal points on the page
- Navigation buttons as the only visual interest, especially when they're large
- Cluttered, not enough alignment of elements
- Lack of contrast

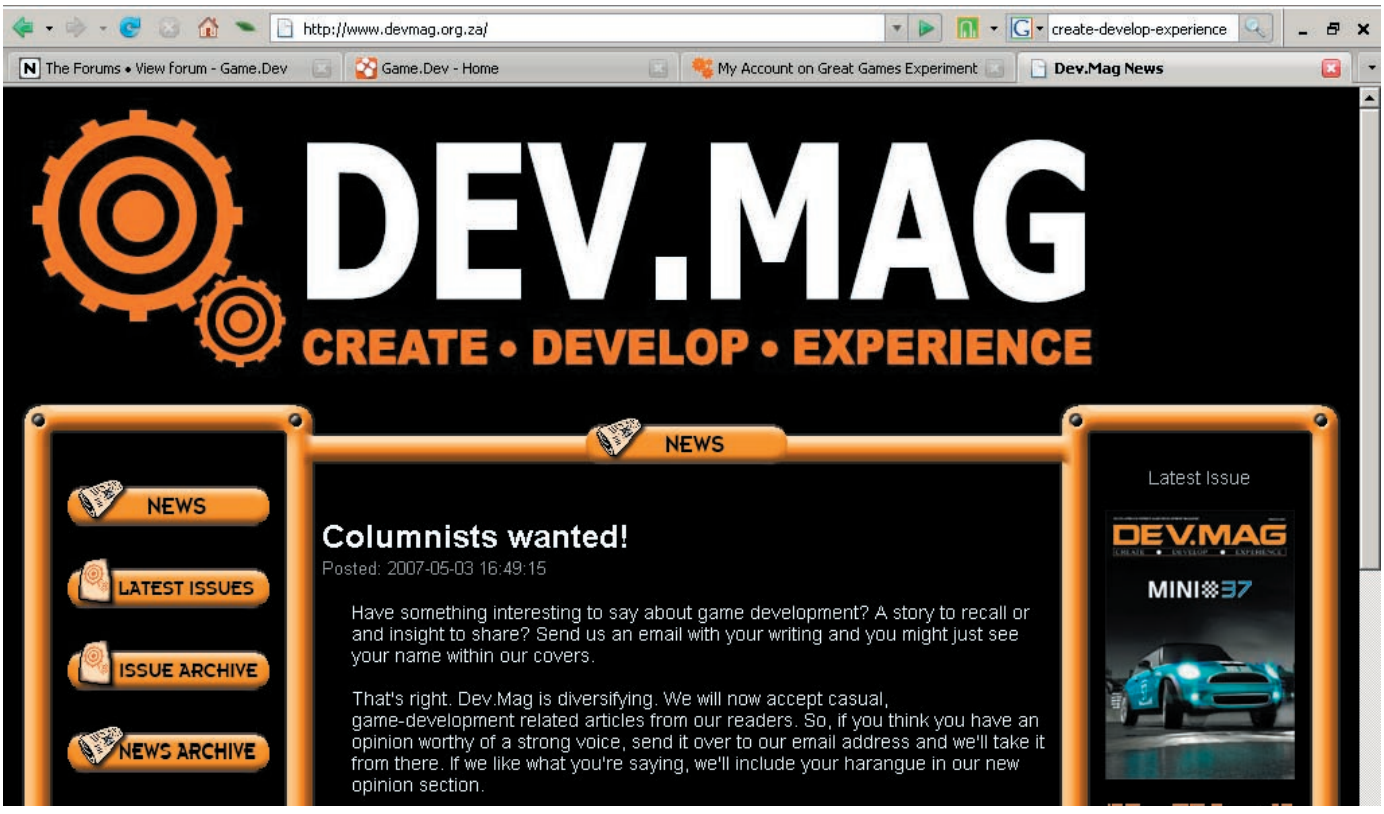
Right, if you didn't bother to read the rest of this, just remember the number one rule: NEVER BREAK THE HORIZONTAL SCROLL. Viewers should never have to scroll horizontally.

You should also know some things about links and images. First, links must always have a title. This is the little bit of text that pops up when you mouse over the link. The title should tell the user what is at the other end of the link. One of your HREF parameters is the Title="" parameter. Secondly, images should always have an alternate text especially link images. Alternate text can be added using the Alt="" parameter in an IMG tag.

Also, all web browsers work differently. Make sure you test your page in different ones. Internet Explorer in particular tends to display pages rather erratically.

Well, that about concludes the series on HTML and beginners' website creation, now go out there and make your own web pages!

SQUID



ROACH TOASTER 2: PICKING OUT THE BUGS PART 4

After the unexpected success of Roach Toaster 1, I knew I had something special to work with, which is obviously the reason why I chose to work on Roach Toaster 2. Right at the beginning I drew up a list of goals and expectations. Here they are:

Goals:

I want to show the world my unique game.
I want to sell a maximum of 20 (yes, only 20) copies from my own site.
I want to learn the ropes of selling and marketing online shareware games.

You might realise that a release date is missing from my goals. Initially I expected beta to hit the world in December 2006. As of writing, I still haven't released the beta. I also expected to have the gold version ready by March 30.

Expectations:

I expect to have a sure-fire hit in the gameplay department.
I expect it to sell quite reasonably, but I do not at all know whether it will be runaway success or not.

The expectations tie in with the fact that the market for this kind of game is still small. Although it is small, it is growing substantially. The market I am referring to is indie turn-based strategy games, and turn-based strategy games as a whole. The indie turn-based strategy market is pretty much dominated by the likes of Galactic Civilization and Lux-type games. A few years ago Oasis made a big splash and even took away the grand prize at IGF (Independent Games Festival)!

So yeah, in the end, I do not really know what to expect. As of writing, the open-beta is practically ready!

For more info on Roach Toaster 2: Big City, head on to http://www.shotbeakgames.za.net

TROOJG

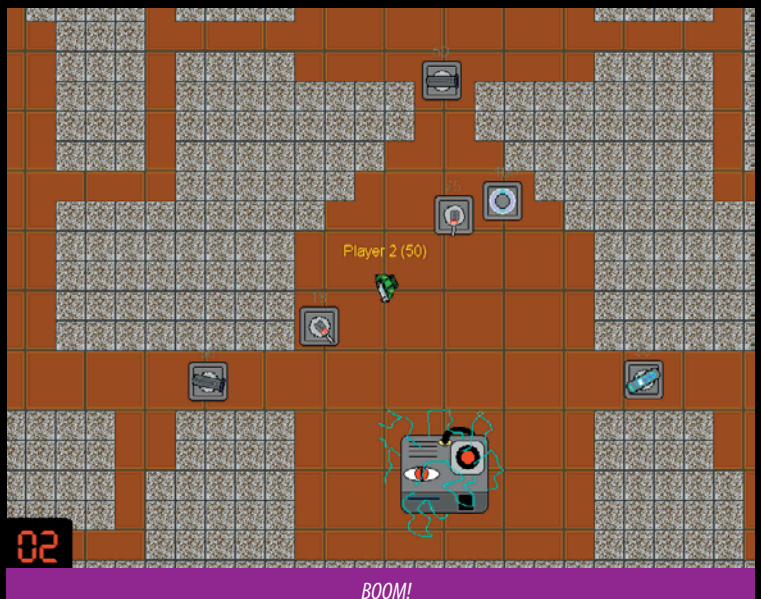
POSTMORTEM: Overseer Assault

Overseer Assault is a freak of nature. Now that statement may seem harsh, especially coming from the game's creator, but it's true - Overseer Assault is a game that shouldn't work, but through some dark and twisted means just does. One need only look at the description to understand: OA is a hybrid top-down shooter/turn-based strategy game. It also happens to be the first game that I ever made. Ambitious? Absolutely. Flawlessly executed? Not entirely...This is an overview of the Making of Overseer Assault, and new developers would do well to heed the painful lessons contained herein.

An Overseer fights off a dual-pronged attack.

...And ne'er the twain shall meet

Overseer Assault was conceived for Game.Dev Competition 12, where the aim was to build a multiplayer game that could be played on a single computer. I wanted to build a game that simultaneously pitted players against each other in the opposing roles of attacker and defender, as a way for both to exercise their tactical skills from two separate viewpoints and play styles. The original plan was to have a real-time split-screen game with both players fully mobile, one equipped with powerful offensive weaponry in order to kill the other, who was loaded out with cunning snares and traps to deter the pursuer. This idea quickly evolved into OA as it is now - turn-based with a fully mobile, powerfully armed attacker tasked with destroying a stationary opponent with the ability to place defensive turrets to protect itself. Initially I dismissed the whole idea as unworkable. It was an ambitious project for a newbie who had only ever coded in VB6, and I was only just starting to get the hang of programming in Game Maker (the framework/IDE used to build OA). Nonetheless, an annoying little voice in the back of my head urged me on, and the result was... Interesting. The moral: think big!



BOOM!

Seeking Balance (What went right)

One thing that I realized from the outset was that OA would have to be meticulously balanced if it was going to work, and to my great joy balance was one of the main areas of praise for the game. Now, I wish I could say that I sat down for hours on end mapping the interrelationships between weapons and sides on a spreadsheet, but I put far less effort into balance than one may think. This was due to my core design philosophy: no one side should overpower the other. By keeping this "mantra" in mind when conceiving the weapons and game mechanics I was able to easily implement the balancing with very little effort (even for last-minute additions). In fact, I spent less time tweaking the game balance than anything else, and the two sides still worked together beautifully. The lesson is

simple: determining and keeping to a core philosophy/philosophies for your gameplay in the design phase can reap great rewards when it comes to actually building the game!

Aaaagh! They're everywhere!
(What went very, very wrong)

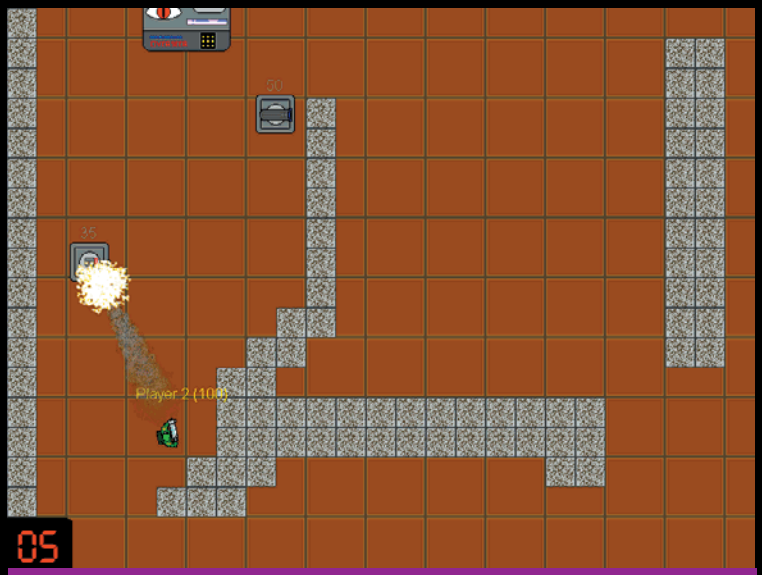
As some may be aware, the original version of OA was overrun with bugs. It was suggested that this was due to the scope of the game but, alas, I knew better… The attention that I paid to the game mechanics wasn't duplicated when it came to the code. Code was unplanned and hacked out at the spur of the moment with the sole aim of getting the game to function as I had initially designed it, and with a complete disregard for any future modifications.

Well, the game did work, until I tried to change it! A good example was how the code for all the turrets was lumped into a single object, something that would give any self-respecting programmer a seizure. The code ended up being so convoluted that adding a single new weapon or turret to the mix was an exercise in misery and frustration, and resulted in some truly game-obliterating bugs due to the single-object structure of the turrets. Eventually I was so lost in my own code that I considered trashing the whole lot and starting from scratch, but with a competition deadline looming it would be certain suicide. So I cleaned up what I could and submitted it with bated breath.

OA ended up winning second place in the comp due to its concept and balancing, but it became a shallow victory when the judges mentioned the bugs. My own lazy coding practices had killed a potentially great game.

Redemption is nigh!

Fortunately, Competition 13 came along, and I resolved to turn OA into something I could be proud of. The aim of Competition 13 was, quite conveniently, to polish an existing game to the point that it could be sold. Out came the pen and paper: OA's code was ripped apart and, where necessary, rewritten. Turrets were given a fully



One hot rocket coming right up ...

The almighty firebomb wipes out another bank of turrets.

Object Oriented makeover, which in itself killed about half the game's bugs in one fell swoop and made the game much easier to modify. The code that handled timing and player turns was refined, the graphics were spruced up, and sloppy design was rectified. In short, the game was rebuilt to play the way it should have. The code still isn't stellar, but compared to the original version it is much improved, and the game finally plays as beautifully as I intended.

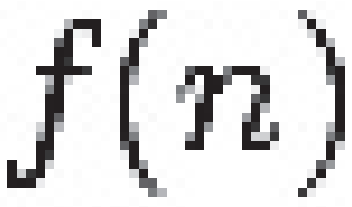
In the end, OA turned out a game that I could be proud of, and ended up winning Comp 13, which was as gratifying as it was surprising. I also learned why a game's code structure needs to be designed solidly from the outset, a lesson that I'd like to shove down the throats of all the other newbie game devvers reading this. As for OA, it's a concept I'd like to expand on sometime in the future. It was mentioned that the game could do with a graphical overhaul, and I'm looking into that.

GAZZA_N

At the time of this issue's release, the most recent version of Overseer Assault could be downloaded at www.gamedev.za.net/filecloset/download.php?id=8

CODING ETIQUETTE: FUNCTION USAGE

Functions, as defined in high level languages such as C and C++ (along with their respective counterparts in various other programming and scripting languages), are an integral part of programming games in today's world. Even in the "old days" of programming games in pure assembler, or even the "not-so-old days" of using BASIC, the functionality of functions has been conveyed through the use of various label and jump style commands. Today, it is actually next to impossible to create even the most simple of computer games without separating your code into various functions, merely due to the fact that the logic behind computer games is not linear in nature and needs to "jump around" in order to perform various operations based on a multitude of potential conditions.



The process of breaking up your code into what usually amounts to hundreds if not thousands or even tens of thousands of smaller scale code blocks contained within functions is not something that should be taken lightly if you wish to keep your code neat and maintainable. The challenge to programmers here is that in order for your functions to add to (and not detract from) other portions of your coding style (such as commenting and naming conventions), then the proper creation and use of functions is something that needs to be done discriminatingly.

Knowing when to turn a section of code into a function is actually a very easy thing, and in most cases the only thing preventing programmers from doing it when they should be is their own laziness. At the core of function creation there exists three very simple rules that are designed to help programmers decide when to place a block of code into function. If these straightforward guidelines can be adhered to, then in all likelihood your code become not only more organised, but also more maintainable. The first rule of function creation is usually the most difficult one to interpret. It specifies that any block of code that performs a specific or defined task should be isolated into its own function. Now, if that rule was to be taken too literally it would actually mean that every line of code should be separated into its own function as every line of code does have a specific task. Instead, the rule is designed to be applied in a more broad manner whereby sections of code that are responsible for achieving something more special than arbitrary data assignment or simple arithmetic are targeted for function creation. Therefore, you wouldn't place a simple addition operation into its own function, but you may create a function for a more complex sequence of mathematical operations such as some form of matrix manipulation or a collision intersection test.

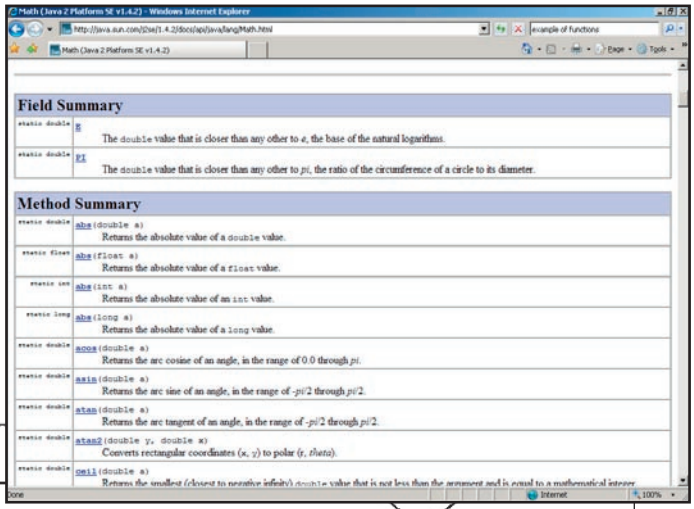
The second rule of function creation is pretty straightforward. It specifies that any block of code that performs a task considered to be generic enough to be used more than once in your program should be separated into its own function. This is the one rule out of the three that programmers do tend to follow on a consistent basis as it appeals to the lazy nature of the average programmer: "Why write code more than once if I can instead write a function and then access it multiple times?". The inherent laziness of the programmer is finally correct for once as it is

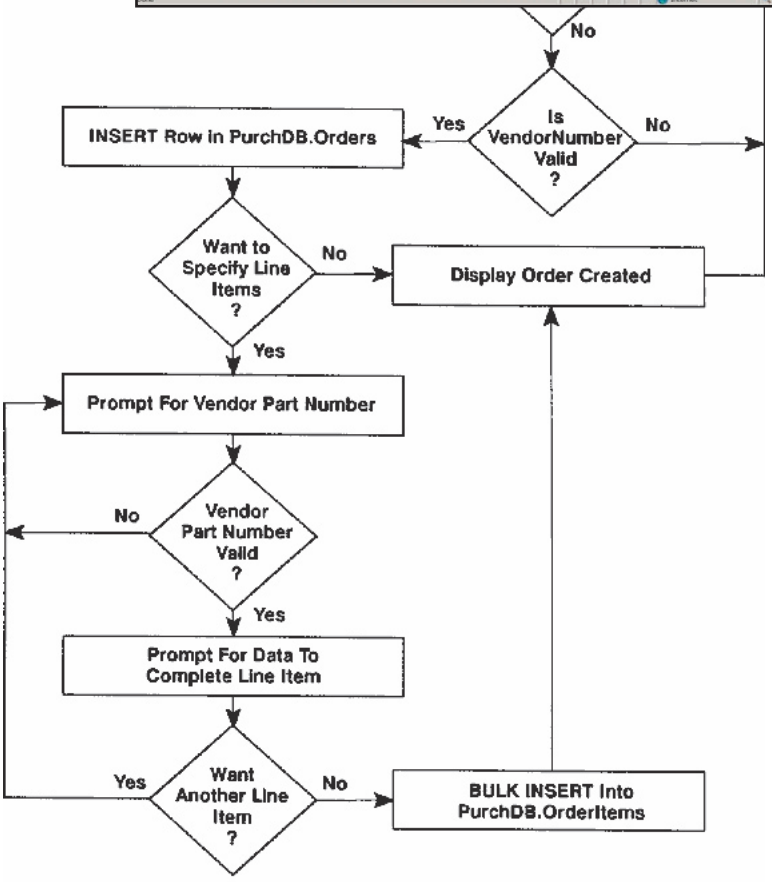
most definitely a preferred practice for programmers to reuse whatever code they can instead of wasting time redoing logic again and again.

The third and final rule of function creation is not only the most difficult one to follow, but it is probably the most controversial one as well. It specifies that all functions should be limited to roughly a single computer screen in length. The reasoning behind this is that if code inside of a function exceeds this length, then you are obviously not following the first rule of function creation; that there are blocks of code within your overly long function that can be turned into smaller functions of their own. The effect of limiting the length of your individual functions is that it forces them to be more readable and understandable, as whatever complexities you may be creating within your code are forced into being broken down into smaller problems which are easier for other people to digest.

Most computer programmers generally view functions as merely tools to be used for making the end result of a program easier to accomplish. However, the proper use of functions when taking programming etiquette into consideration can achieve not only that, but can also allow for functions to be used as a tool for making code much more understandable and maintainable.

COOLHAND





THE HISTORY OF I-IMAGINE

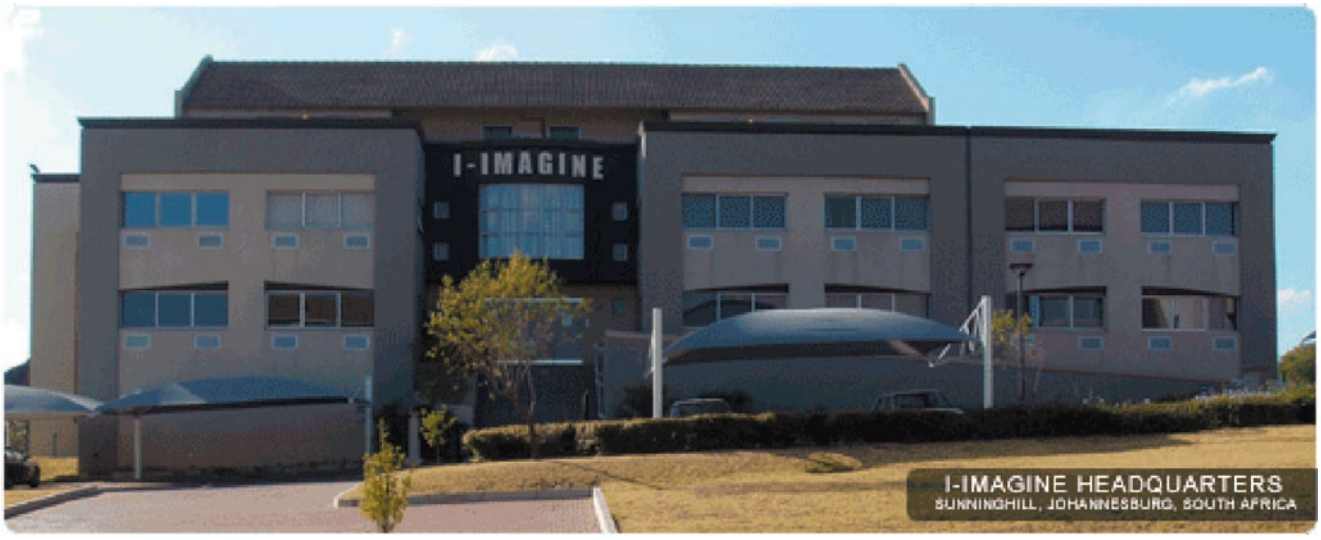
Part 4: Renong House

I'm pretty sure that not many of you have had the pleasure of experiencing the 18 hour non-stop flight between New York and Johannesburg. For those of you who haven't, re-read the previous sentence and replace "pleasure" with "misfortune", and add "from hell" directly after "flight" if you would like have an idea as to what it really feels like. Actually, I could fill an entire separate article just describing how those 18 hours feel more like 3 ½ years, and how I no longer fear what may await me in the afterlife. But I digress...

When my plane finally did touch down on the tarmac I managed to make my way through immigration with surprisingly little difficulty... I think it was most likely because the guy on duty was taken off guard by the fact that someone was actually moving to South Africa for work. Regardless, once that final remaining hurdle had been passed I met up with Dan and officially became the second employee of I-Imagine (after Dan). I was the first person to arrive from overseas and I still remember how full of awe and excitement I was when, on my first night here, I walked out into the middle of Sandton Square for dinner. All of the people, the bright lights, and that stupid little fountain thing shooting water came together to form a sign, telling me that I had reached a very important point in my life.

My isolation at I-Imagine ended quite quickly as the next day saw the arrival of Hoang Pham, who was an artist (formerly of Volition) brought on board by Felix. There were actually supposed to be two artists from Volition coming to join us, but the second one ended up using the offer from I-Imagine to negotiate a better deal for himself at Volition. A few days later, Dave Gierok (a programmer from Looking Glass) and his wife Stacia arrived, and about a week after them, Matt Keele and his wife Andrea showed up and brought I-Imagine it's first case of drama. The two of them were attempting to bring their cat Oscar (I can't believe I actually remember that stupid cat's name...) into South Africa with them, but they ran into a problem when they landed as they hadn't completed the necessary forms correctly. I seem to remember something about a forged veterinarian's signature, but I can't remember for sure so I'll just tell you that a few connections and a bottle of J&B later everything managed to get sorted out, and leave it at that...

Now for those of you who don't know, I-Imagine's offices along with the living quarters for all of us, were in a pretty large house in the Sandhurst suburb of Sandton. The place was so big that it even had a name, "Renong House", that was spelled out in big brass lettering outside of the front gate. It had six bedrooms, a guest house, swimming pool, jacuzzi, steam bath, sauna, bar, giant lounge which we turned into a the-



I-IMAGINE HEADQUARTERS
SUNNINGHILL, JOHANNESBURG, SOUTH AFRICA

atre, gas braai, maids quarters to sleep six, and about four hectares of garden. The Malaysians actually used Renong House as their Embassy just prior to us moving in, and they kept a flock of peacocks and a herd of sheep in the garden. That is how flipping huge the place was! As well, on top of all of these "residential fixtures", it also had two large open plan office rooms which we converted into our development studio (one for the artists, one for the programmers). All in all, we couldn't have asked for a more perfect place to use a base of operations for a company comprised of mostly overseas individuals.

The first month or so in South Africa was pretty much an extended holiday for the group of us. It was actually quite a while before we really got down to doing any game development whatsoever as not only were all of us foreigners trying to get settled into a new country, but I-Imagine as a company also had a lot of little things to sort out before we could properly get down to business. We needed computers, software, a server, networking connections, Internet, office furniture, and most importantly, an idea for us to use for our first game. So once the settling in period come to an end and our offices were all fit and ready for work to begin, the group of us (minus Felix and Kenny Kupis who weren't planning on arriving until the end of October) sat down and spent many brainstorming sessions trying to come up with an idea for our first game.

It was quickly apparent that we all liked the idea of doing a 3rd person action/adventure title as there really weren't many good ones available, and we thought that we would be able to exploit that hole in the market. However, we were worried about having to develop the technology required for doing good character animations and as a result, we decided to play it safe and not go down that road. So, once we had come to the conclusion that the direction of our first title needed to be dictated by the technology that our small team would be able to realistically develop, we decided that our best bet would be to make a racing/driving game. With that part of the decision making process settled upon, the next step was to come up with a game idea for us to put this technology to use on.

This was easily the most difficult part of our game concept brainstorming. We didn't want to merely do a normal racing game due to the fact that we would certainly just get lost in the crowd of the dozens of racing titles released every year. It wasn't easy to to come up with something original to do in this overly saturated genre, but after throwing around a lot of potential concepts, we came up with the idea of somehow incorporating stunt driving into our title as it was something fresh and fun that hadn't been done before as the main component of a driving game. Once we were sold on the idea of stunt driving, we then decided that what better way would there be to incorporate stunt driving into a game than to have the player take on the role of an actual stunt driver performing the various exhilarating chases and crashes required for Hollywood action blockbusters? It all seemed to come together quite naturally for us at the end of our discussions, and the entire team was quite excited about working on a game that was pretty much one of a kind. Little did we know however that this was on the the beginning of what would be a three year journey of highs and lows on the way to completing "Chase: Hollywood Stunt Driver"...

COOLHAND

THE HISTORY OF TWILYT / CELESTIAL

One of the first recorded South African game development studios was Celestial, led by Travis Bulford. Their first commercial game was a platformer called Toxic Bunny back in 1996. It had you controlling a crazy… well, bunny. The gameplay was similar in style to Megagames' (an overseas studio) Jazz Jackrabbit. Some argued that the level design was flawed compared to Jazz Jackrabbit, but it is a matter of taste. The game interacted more with the environment, unlike Jazz Jackrabbit. It went on to sell in the region of 150 000 copies.

After Toxic Bunny they released Tainted in 1999. Tainted had slightly flawed gameplay. It was too hard for the player to get into the game. It also had slightly sub-par graphics. As a result of the above, it was not received too well with the press. With Tainted they learned a lot in terms of what works and does not.

Soon after the release of Tainted in 1999, the developers of Celestial changed name in December of that year to Twilyt productions.

Asking where the name came from, Twilyt responded that they work mostly during the twilight hours of the day.

Twilyt consisted of 13 members that started their development juices by developing a game called Silver Rain in conjunction with NAG magazine. It was a simplistic 3D game designed just to educate eager developers about the way it works. They wrote the game development articles for South Africa's premier gaming magazine.

Twilyt Productions' first commercial release would in fact have been Zulu-war, a strategy game taking you back in South Africa's bloody history and placing you in command of the Zulu or Boere. Alongside Zulu-war they were also developing Anglo-Boer war, a similar title with the English fighting the Boere instead of the Zulu.

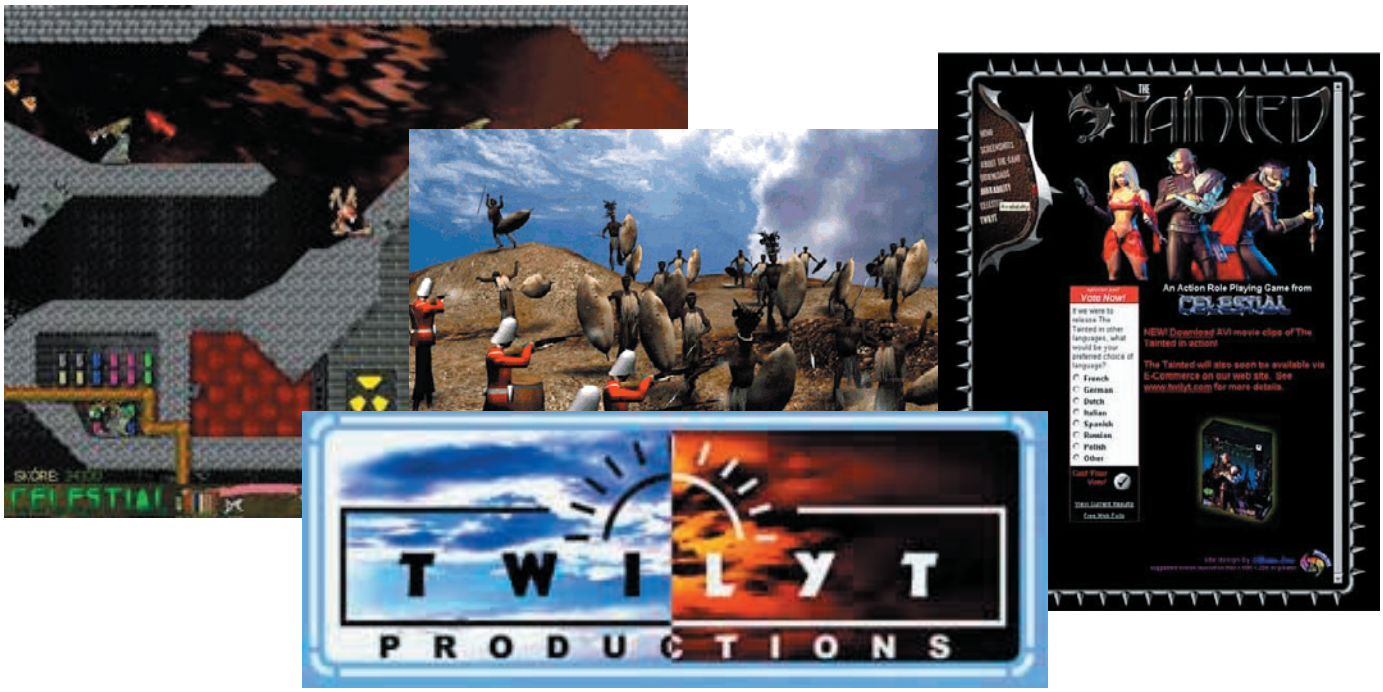
Apart from these 2 titles they also planned a game called Tincan's Colouring-in book. This title is a self-explanatory educational/children's game. Toxic Bunny 2 for the X-box was also planned.

Not one of these games saw the light of day.

Travis Bulford was the only developer and his money went into the funding of Twilyt. Due to lack of other funding, Twilyt eventually lost steam and had to close its doors in 2001.

Although it's no longer around, Twilyt serves as a bastion of hope to other SA developers out there, showing that it has long been a possibility to develop games in good 'ol South Africa!

TROOJG



CHOOSING A GAME

AN ADVENTURE INTO SHAREWARE

There are lots, hundreds, maybe even thousands of new shareware titles available at the moment. How and why some games are selling better than others is a question regularly asked by many independent game developers, myself included. A few weekends ago, in the spirit of "Market Research", I spent some time with my kids buying a few shareware games.

I have three children: a daughter of 10, and two sons aged 8 and 6. My daughter is pretty typical of a 10 year old girl - she loves her dolls and the colour pink! My eldest son is growing up quite quickly and is in the "Cowboys and Indians" stage of life where fighting games are cool, while my youngest son loves art and engineering, and is happiest when painting the cars skins in Track Mania or just scribbling in Tux Paint. All three kids play Chess and have an appreciation for games that involve thinking instead of just pure reflexes. I myself like strategy and building games, my favourite games of all time being Zeus and Age of Empires, with Civilisation a close 3rd.

The original idea was that we would enter an online shareware site, each of us was to choose two games that we would download, and everyone would play all the games and then be allowed to choose one game that we would buy. In addition I already had a few games that I had downloaded and we could look at them as well. I also included a few free games that I had already downloaded.

In reality we ended up choosing a few more games than planned. We downloaded most of the games from www.BigFishGames.com but a few from www.alawar.com and www.Popcap.com. In general we looked at the Kids categories when choosing games, the kids chose games that I though pretty typical for their age and sex, girly fashion games for my daughter, fighting style games for my eldest son and puzzle games for my youngest son. My youngest son also chose a very kiddies game that contained multiple mini games such as racing, colouring in, etc.

The games we downloaded:

- Peggle
- Fizzball
- Lego Le Chic Boutique
- Spongebob Squarepants Obstacle Odyssey (the only 3D game downloaded)
- The Wonderful Wizard of Oz
- Virtual Villagers
- Theseus, return of the hero
- Diego's Dinosaur Game
- Professor Fizzwizzle

Brief reviews of the games:

Peggle:
A simple shooting game where the player shoots a ball into a field of Pegs and blocks trying to remove all orange items. Some pegs have special abilities, and the game and all the levels are very pleasantly themed.

Fizzball:
The evolution of Arkanoid – control a ball using Arkanoid-style controls but where the ball is limited to what blocks it destroys based on its size, while eating as many blocks as it can in order to grow. Very cartoon-style graphics.

Lego Le Chic Boutique:
A Lego branded game, with absolutely no Lego effects. Solve a match-three puzzle to score points with which to modify a shop environment selling jewellery. As items are removed you build the stock levels in the shop. Between each level you can spend points on new things for the shop, attracting more customers.

Spongebob Squarepants Obstacle Odyssey:
The only 3D game we downloaded. Along with Lego Le Chic Boutique, it shows how copyrighted material is making an impact on the casual game market. Basically a simple 3D game

where you direct Spongebob to solve puzzles on platforms in space.

The Wonderful Wizard of Oz:
A Bejewelled clone along with the story of the Wizard of Oz. Included in the puzzle game are munchkins that must be released as the match-threes are complete, as well as a yellow brick road that must be cleared to finish a level. Releasing five munchkins grants you a diagonal move that can be very very handy.

Virtual Villagers:
Downloaded for myself really. Give general directions to a collection of villagers stranded on an island. They breed, get food and carry on their lives even when you are not watching. "Solo online multiplayer game" is the best description I can think of for it.

Theseus, return of the Hero:
A top-down isometric shooter with hordes of evil looking enemies. Many options to allow the player to choose the weapons they like as well as pickups such as ammo. This game seemed to have a very high level of difficulty for a new player.

Diego's Dinosaur Adventure:
A collection of mini games aimed at very small kids, including racing, picking up eggs and directing dinosaurs out of a maze.

Professor Fizzwizzle:
A puzzle game where the player needs to direct professor Fizzle Wizzle to get from point A to point B. Various traps and obstacles are in the way, and need to be solved. Each level is neatly created to look like something such as a giant fish or hippo that adds some extra visual appeal to the game.

Each game has its own style, each being very different from the others. Peggle's graphics and graphical effects are exceptional, while Le Chic Boutique keeps to very simple graphics that are just functional in the scope of the game. Fizzball's cartoonish style is very appealing. The graphical style, however, had little impact on what the kids thought of the game.

A few of the games have special graphic effects – the zoom in, and particle trails of the balls in Peggle are absolutely fantastic, while other games have virtually no special effects, even when they could very easily have been added. Spongebob Squarepants Obstacle Odyssey had nothing to show the player that they had reached their goal, but just transitioned into the menu. Special effects, while nice and do add to the polished feel of a game, are not absolutely necessary to making the game appealing.

I personally am very bad at identifying good and bad sound in a game. I will not play a game where I can't turn off the music, and often play a game with my speakers off. The only game where the sound effects really had an impact

on me was Peggle, where the ping sound as the ball hit a peg changes to make it more exciting each time. Also the different tracks to show that you had finished a level were really noteworthy.

My youngest son is still at the age where he plays games with very basic controls – he does not yet have the understanding of how to build bases in an RTS, or cause and effect of needing food to buy new tech levels. In addition he will often play the simplest game over and over again as long as there is something else to discover each time. One thing that fascinates me when watching my kids play games is the way they will click on everything on the screen to see if it does something. This sort of replayability is different from solving the same puzzles over and over again. None of the games downloaded included this sort of effect at all and therefore missed the younger child market.

Some games are endlessly replayable. Le Chic Boutique allows you 10 days in each of about 10 cities to build your jewellery shop. Every few levels you get new items to sell, and new items to buy for your shop to really make it stylish. Without playing through all the levels (about 100), you will never get to see everything that's available in the game. Other games do the same thing over and over again with new special effects or a new look to screen. Peggle was fun, it had great graphics, and the best sound, but it was boring. The demo was quite long enough, and after an hour of watching a little ball bounce around the screen there was just nothing more offered from a replayability point of view.

The games ranged widely in difficulty. Other than Theseus, Return of the Hero all the games were chosen from the kids section. Theseus was way too difficult for my son, while on the other extreme Diego's Dinosaur Adventure was too simple even for my 6 year old son. Le Chic Boutique seems to have a very interesting method of managing game difficulty to player skill (a negative feedback loop) by bringing in customers based on the way the shop looks, which is impacted by how well you do - thus ensuring the number of customers matches with the player's puzzle-solving skills.

A key aspect to the enjoyment my kids got out of the games was the non game aspects available in the games. Le Chic Boutique works because of the non game aspects – build a shop, stock it, and sell the products to customers. I have also seen in other games my kids play that the game itself is not necessarily why they play a game, but the environment that the game is in is just as important. For example, in the Track Mania games they purposefully drive off the track and spend hours driving around in the world just to find cool places and things to do. At the same time the additional activities must not just be reading pages of text. In The Wonderful Wizard of Oz, each level and stage is based on excerpts from the book but displayed as a full page of text to read. My youngest son at 6 can't read yet, so these pages were just time wasters for him and resulted in him losing interest in the game.

For my daughter, the theme was very important – she chose her download games because of their theme. Theme was less important for the boys, who rather used the description of the game to choose their titles – however, I do think that the Spongebob Squarepants branding had a big impact. Many games don't fit into

any defined theme. Peggle, for example, isn't themed, but it does seem easier for the kids to associate with a game if it is themed properly.

Only a few games kept my kids busy for their full trial period, and each of these games had their own reason for captivating the kids. Le Chic Boutique captured my daughter's imagination because of its theme and colour scheme. As soon as she had worked through the trial period she wanted to buy the game. The only competition for Le Chic Boutique in her mind was Fizzball, and in reality it was a distant second. My eldest son spent a lot of time on Spongebob Squarepants Obstacle Odyssey, but in reality I think it was the 3D interface that caught his attention rather than the theme or game play itself. My youngest son didn't spend more than ten minutes or so on any game except Diego's Dinosaur Adventure, where he played each option once and then he had enough of the game. The other game he spent some time with was Peggle, but I think it was more from a physics ('see how the ball' bounces) point of view than the actual game itself.

While the game with the best production quality (polish) was clearly Peggle, not one of my children gave it a thought when choosing a game to buy. The games we bought were: Lego Le Chic Boutique (The best "girly" game, but also the game with a lot of replayability and it had the right colours to appeal to my daughter) and Spongebob Squarepants Obstacle Odyssey (I think the 3D view appealed to my son's sense of 'Modern game', but he hasn't played it much since we bought it). My youngest son didn't find a game he liked and went back to some of the game demos from a local PC magazine.

The payment process was in some cases OK, while in other cases a pain. Buying the games from www.BigFishGames.com worked fine, but the same credit card was rejected by www.Popcap.com as expired. As the games were to be used by all the kids they needed to be installed on all our PCs at home. Initially I thought this would be a problem but the second installation happened fine with a new license being sent by email quite quickly from www.BigFishGames.com. It seems that it is now normal for the games to register themselves over the internet with their home site; luckily I have ADSL at home to enable this connection.

So what should you do with this information? Good question. This covers the experience with three kids. All three kids know computers well, spend a number of hours each day playing a variety of computer games, and as such are actually pretty experienced gamers (for their age). To me the whole process showed me that every aspect of a game is important, but often to different people/markets. If these kids fall into the target market for your game you may find the insights here very relevant, and if not I hope it goes to show the sort of things that, as a game designer/developer, you need to keep in mind when creating a game.

My feeling at the end of the whole process was "Make games for girls, not boys", as boys are heavily influenced by the 'AAA' titles they play while girls are more attracted to the theme of the game and in truth are less interested in the fancy special effects and graphics. Rather spend time making a game that plays well, has lots of options and is themed in a very girly fashion, covered all over in pretty pink colours. At least at the end of this I know my daughter will be hooked on my game!

CAIRNSWM

If you're interested in checking out any of the games featured in this article, try the following links:

PopCap Games:
http://www.popcap.com/

Alawar Games:
http://www.alawar.com/

Big Fish Games:
http://www.bigfishgames.com/