DEV.MAG

CREATE • DEVELOP • EXPERIENCE

SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE • ISSUE 22 • JUNE 2008

A DAY TO REMEMBER

We interview one of South Africa's most prolific game development evangelists

INSIDE:

DEV.MAG TIME MACHINE • TRILBY: ART OF THEFT REVIEWED • PROCEDURAL GENERATION MADE EASY • DESIGN DOCUMENTS: TECHNIQUES AND VALUES • LATEST NEWS • MUCH MORE ...

CONTENTS

REGULARS

FEATURES

REVIEWS

TUTORIALS

DESIGN

TAILPIECE

ED.PAGE

EDITOR
Rodain "Nandrew" Joubert

DEPUTY EDITOR
Claudio "Chippit" de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNER
Brandon "Cyberninja" Rajkumar

WRITERS
Simon "Tr00jg" de la Rouviere
Ricky "Insomniac" Abell
William "Cairnswm" Cairns
Bernard "Mushi Mushi" Boshoff
Danny "Dislekcia" Day
Andre "Fengol" Odendaal
Luke "Coolhand" Lamothe
Rishal "UntouchableOne" Hurbans
James "NightTimeHornets" Etherington-Smith
Gareth "Gazza_N" Wilcock
Sven "FuzzYspo0N" Bergstrom
Kyle "SkinkLizzard" van Duffelen

WEBSITE ADMIN
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

You admire the layout. You are in awe of the layout. You are amazed by the layout. Reward us with hugs and candy.

DEV.MAG

Hello again, dear readers!

Time flies by quickly, it seems. I hope you haven't grown too impatient waiting for this edition of the magazine to come out – as you can see, we've been rather busy with some things and it has slowed our progress considerably.

This edition has been especially prettified for three reasons: one, we've been hankering for a makeover for a while, and it's always fun putting on a new skin. Two, this mag needed to be presented as a journalistic project to some sinister authority, meaning that some polishing up was required on the looks front. Three, this shall be my final Dev.Mag issue as your loyal and ever-humble editor, and I have personally put in the hours to make sure that the last edition under my control leaves a good impression on you, the reader.

I'd like to start by allaying any fears: while I may have fallen casualty to real-world commitments, the magazine itself will most certainly endure in my absence. Our dear Dep-Ed, Claudio "Chippit" de Sa, has accepted the mantle and is already making preparations to re-organise the magazine and get the next issue ready. From what I hear, he's being assisted by a crack squad of journalists who have been called in especially to help him make his mark, so there's bound to be some exciting things happening in the near future.

Given that this is possibly my final soapbox opportunity for the magazine, I'm going to hit a little bit of self-indulgence and wish you all a proper goodbye. I've never really been all that good with farewells (who can honestly say that they practice for such things?) but I want to say that it's been a great privilege to serve as editor over the years. Dev.Mag has grown from a simple concept to a fully-fledged reality, and new opportunities make themselves known with each passing month, helped along by a drastically swelling reader base and one of the most dedicated crews I have ever had the fortune of working with. As I walk away from the proverbial seat of power, I hope that I can allow myself some small credit for helping all of this come to pass.

Beloved reader, thank you for taking the road with us this far. I hope that you continue to enjoy the magazine in my absence and may you fully reap the benefits of the fresh hands that are now taking the helm. Your support has been invaluable – the audience is what we end up writing for, after all, and without the readers ... well, there wouldn't be much of a magazine to talk about!

I salute you, and I salute the hard-working staff of Dev.Mag. Goodbye all, and happy game devving. ;)

RODAIN "NANDREW" JOUBERT
EDITOR

NETBRIEFS

Comp 19: Playing with Death!

http://forums.tidemedia.co.za/nag/showthread.php?t=5633

Most games stop when your character's life meter hits zero. Game.Dev's nineteenth competition encourages participants to challenge this standardised view of gaming and create a project which blurs the boundaries of life and death, playing about with the latter to create a new and unique gaming experience. If you're a South African citizen and want to have a go at this comp, high-tail your way to the forums and get something going for the July 1st deadline!

TIG database

http://db.tigsource.com/

Recently, Derek Yu over at TIGsource decided to update the affiliated Independent Games Database, throwing in filters, doohickeys, whatchamacallits and a load of extra games, bringing the database count to a little over 200 hand-crafted indie games. It's a humble collection, but still really neat to browse through if you're looking for some high-quality indie gems to try out.

MochiAds

http://www.mochiads.com/

An interesting website which offers developers a chance to earn money with their online Flash games, using an in-game ad system that allows the game's creator to get buckazoids on a per-play basis. If you're an experienced Flash developer and are craving a bit of cash, have a look here and see if the system appeals to you.

MMOG business models

http://www.gamasutra.com/view/feature/3688/mmog_business_models_cancel_that_.php

For the business-oriented readers out there, Gamasutra offers a feature comparing the various types of online gaming business models. It looks in particular at the subscription vs microtransaction debate, examining the latter's benefits in light of the former's slackening hold on marketing minds. Interviews with SOE, Three Rings and EA representatives claim to shed light on the matter.

DEV.MAG PRESENTS ...

A DAY TO REMEMBER

Danny Day (known in the online realm as dislekcia) will be a name familiar to most who have dwelled amidst the Game.Dev community of South Africa. Everyone's favourite mentor has now taken a step towards realising the dream of a booming South African game development environment by starting his own company, QCF design. Writer James "NightTimeHornets" Etherington-Smith goes to learn a little more about the man who was instrumental in the creation of the Game.Dev community.

DM: Mind telling us a little about yourself?

DANNY: My name is Danny Day. I'm 27 years old. I own QCF Design, an independent game development studio. I run Game.Dev, a non-profit community of South African game developers. I've lectured on games at UNISA, given talks all over the country and consulted on local and international initiatives on growth and innovation in Information and Communications Technology in South Africa. Generally I just answer a ton of questions and try to get people's enthusiasm channelled in ways that will show them results and keep them going.

DM: Tell us about the Game.Dev community and the service it provides, for the benefit of newcomers.

DANNY: I've never really considered Game.Dev as a service, but I guess we are… If you want to find out about game development, have an idea that you'd like feedback on, want help learning how to make your own games, have a skill that you'd like to offer other game developers or simply have a ton of experience that you'd like to share, Game.Dev is the place to go.

DM: How does the Game.Dev community function?

DANNY: I spend a ton of time online, as do the rest of the Game.Dev regulars, answering questions on our forums, solving problems and giving feedback on games. Every two months I come up with a competition concept that I feel will grow skills and direct interesting discussions, run the competition for a month and then spend an inordinate amount of time judging the entries that come out of it. There are also the physical events like workshops, development LANs and the hugeness that is rAge every year. The main goal is to raise awareness and visibility of game development as both an art form and viable solution to some of our problems across the country. It's idealistic, but it seems to be working so far.

DM: When did you first become interested in game design and development?

DANNY: If I look back, I've always been 'designing games,' from drawing mazes by hand for other kids in primary school, to being the one that always comes up with new things to play during break. If there was any kind of interactivity to something, I wanted to get in there and make my own.

DM: What piqued this interest?

DANNY: One day my dad brought home one of those ancient folding-keyboard workstations and started writing his own games on it. He had all these theories about how people learn and what environments they learn best under, so my sister and I became guinea pigs. We got very, very good at mastermind and mathematics playing Mastery and MathComp respectively. To my dad, the games were just a means to an end, but I was hooked on them after that, always looking for the next game that would challenge me to learn or understand something new. I never really cared for programming until someone showed me LOGO. I got hooked on its instant results and never really looked back.

Danny travels all over South Africa to spread the word of game development. One of the highlights in his calendar is the annual rAge expo where he maintains a Game.Dev stand offering talks, workshops and get-togethers for keen developers.

DM: Tell us about QCF.

DANNY: QCF officially started life as Squirrel Cube Software in November 2007. The original name was chosen out of desperation and ended up changing pretty quickly because it would confuse people… QCF stands for Quarter Circle Forward, which is usually a special move in most games, hence QCF + Design being a "design special move." Corny, but us gamers like that sort of thing.

DM: What inspired you to start QCF?

DANNY: Mainly the fact that I'd been earning a living without a 'real job' for over a year just doing game design consulting projects. I realised that I was turning down job offers and it was finally time to start my own company to bring a couple of products I was working on to market. I've always admired the indie studios out there, but it was only once the whole casual games market exploded and digital distribution really took off and Game.Dev literally grew out of nothing that I thought I could manage to earn a living doing what I love.

DM: What are your goals and aims with QCF?

DANNY: I want to build games that people enjoy, make a bit of a mark in the world with them. Hence my pushing for innovation and learning in games. Of course, I need to eat along the way and have somewhere to stay… I guess the dream success stage of QCF would be to have it be big and successful enough to enable the people working in it to be auteur and experiment with the games they make without fear of financial ruin.

DM: Have you achieved any of these so far?

DANNY: Well, QCF is putting food on my table! Plus it's only been around for about 6 months and I'm already going to start working on a personal project instead of a title for a client. That's a big deal in the industry, if it works out I'll have another intellectual property that hasn't used any publisher money which we can then take to market.

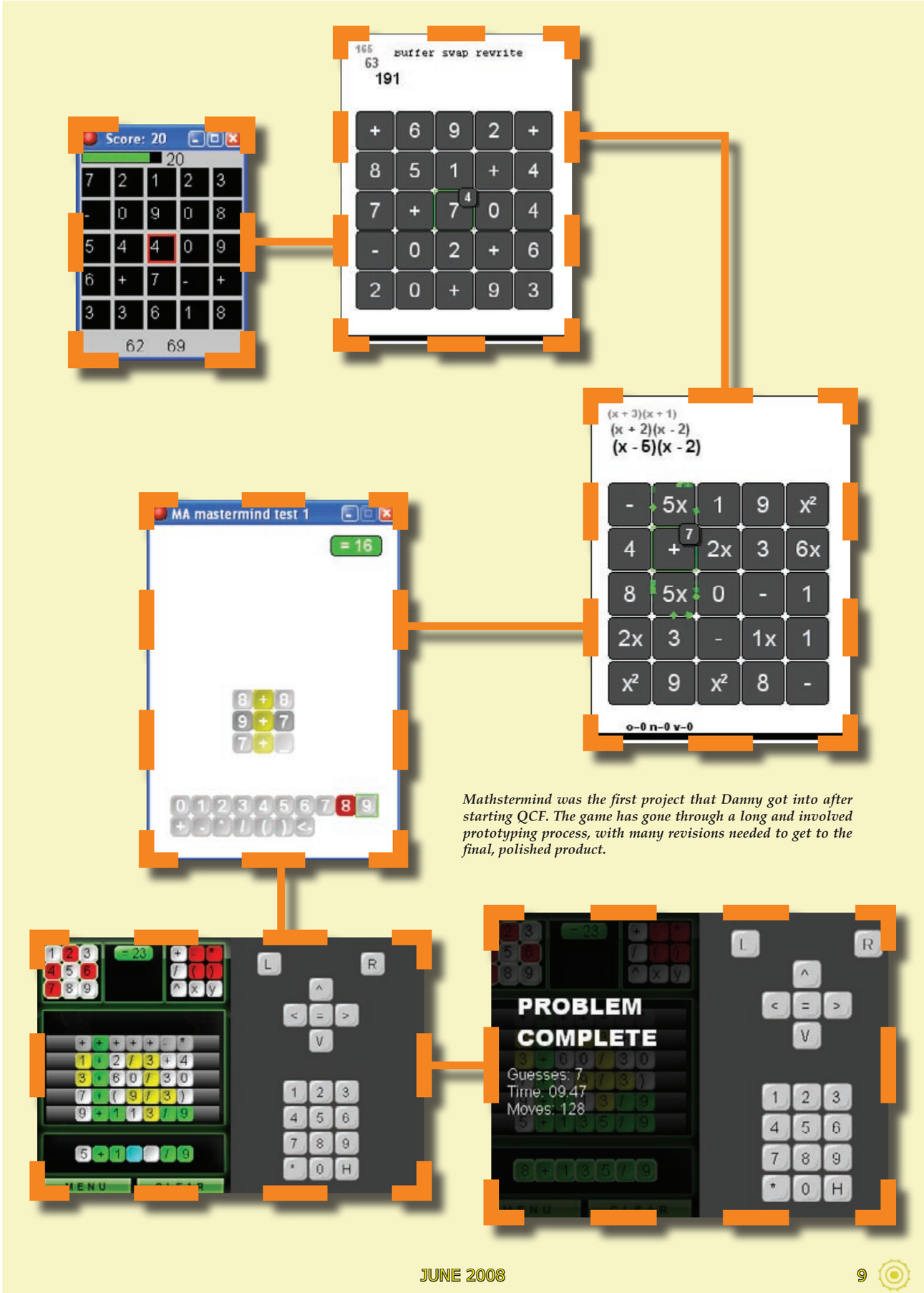
DM: Have you released any games?

DANNY: We've just completed QCF's first game, a cellphone game called MathsterMind. We've managed to retain the intellectual property rights and so will begin the process of looking for publishers and regional distributors soon.

DM: Do you have any partners in QCF? Any employees?

DANNY: No partners, the company is mainly me at the moment. I'm building a team up though. MathsterMind employed Robbie "Squid" Fraser as a designer and programmer as well as Brandon "Cyberninja"

"I WANT TO BUILD GAMES THAT PEOPLE ENJOY, MAKE A BIT OF A MARK IN THE WORLD WITH THEM"



Mathstermind was the first project that Danny got into after starting QCF. The game has gone through a long and involved prototyping process, with many revisions needed to get to the final, polished product.

Rajkumar as a graphics artist. MathsterMind evolved out of Robbie's Math Attack game, which made Mindset bring the project to us. In general, I don't see any problems growing QCF from here, there's all the talent from Game.Dev available.

DM: Any information on current projects that you can share?

DANNY: I have a couple of casual games I spend the odd few days here and there working on, Drawkanoid and Dream Catcher, both do something unique that I need to stop blabbing about and get out there, lest they stop being unique in a month or two…

Our next big project is likely to be a self-funded experiment for Microsoft's Dream Build Play competition – which is an excellent way to get international press and attention. You just have to make a great game. No pressure, y'know? There's also the marketing of our current properties. We need to find Mathstermind distribution partners as well as start selling a visualisation tool I wrote a few years ago called Molecules.

DM: Tell us about Molecules.

DANNY: It's a real-time molecular building tool. Replaces those ball-and-stick kits they flog to school kids and students.

DM: Any previous success stories?

DANNY: I think a lot of people would argue that Game.Dev itself has been an unbridled success. We went from having almost no visible local game development community or information to running workshops and inspiring an entire generation of developers, in less than 3 years. QCF is just starting out, but already our first project is doing very well and getting great feedback. I feel that's just the tip of the iceberg.

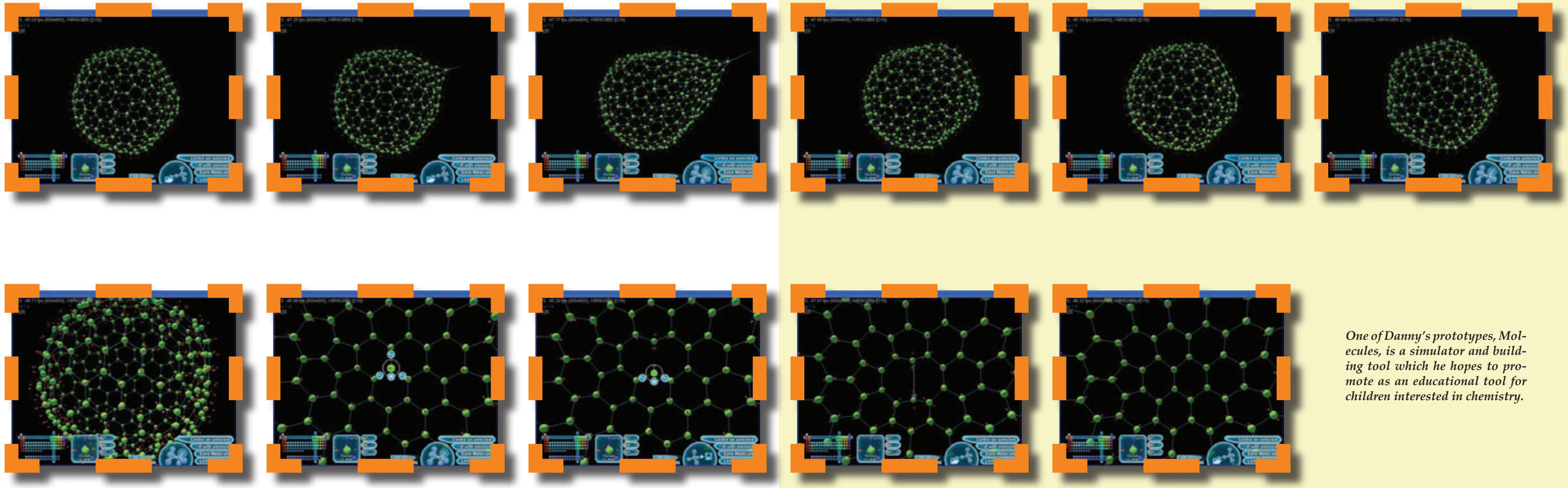
DM: What are your opinions on the South African game development environment?

DANNY: I think we've got a unique opportunity to build capacity as an industry of innovative and creative game designers, known for excellent, quirky, exciting games in a global games industry that's crying out for exactly that sort of authorship. Because we don't have the publisher-driven monolithic structures here that govern game development pretty much everywhere else, I feel that we have the chance to start from scratch and avoid all the pitfalls that take creativity out of the equation. We're also fortunate enough to be a country with lower-than-average living costs, so we've got the chance to sell games globally via the internet and make much better livings than indie developers in Los Angeles or New York. Sure, we've got infrastructure problems but those can be overcome.

The Eastern European developers are known for technical wizardry but often flawed gameplay, Japanese developers are known for cultural flashes like bullet-hell shooters and complex jRPGs. I would like to see South African game developers famous for being a breath of fresh air in the industry in the next 10 years. Our future is digital distribution and the unique talents our country produces. When most of our population is using computers, we'll see some huge changes to the role that games play in our society, both as tools and as means to earn.

DM: Do you have any advice for indie developers just getting started?

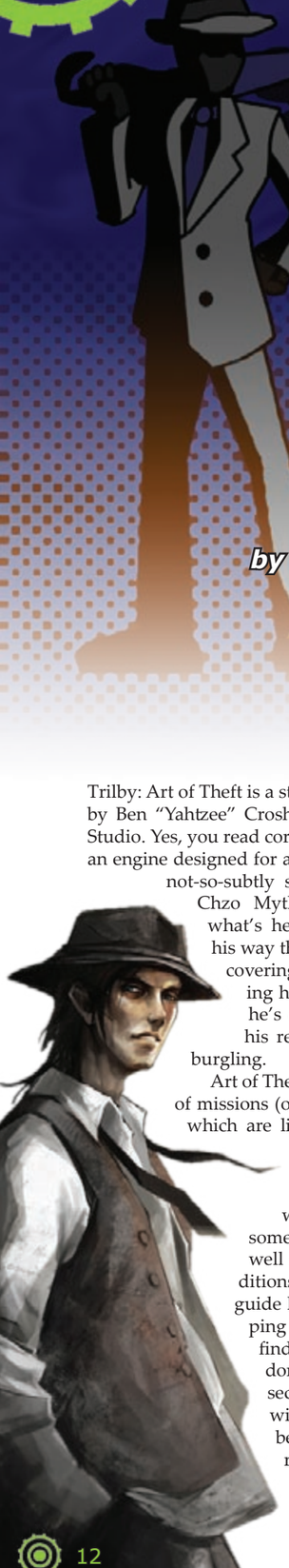
DANNY: An idea is pretty much worthless unless you turn it into a playable game. Start small. You're not going to make Quake 17 on your own. Work your way up from your core idea to the bigger ones. Always test your games, the smiles of your players will keep you going when things get tough. Work with others whenever you get the chance, but don't expect to have people beating a path to your door. Don't re-invent the wheel and don't accept "you can't do that" as an answer, there's always another way. If you keep trying, you'll get there… I feel like I've hardly started.



One of Danny's prototypes, Molecules, is a simulator and building tool which he hopes to promote as an educational tool for children interested in chemistry.

TRILBY the ART of THEFT

by Gareth "Gazza_N" Wilcock



Trilby: Art of Theft is a stealth-platform game created by Ben "Yahtzee" Croshaw using Adventure Game Studio. Yes, you read correctly - a platformer made in an engine designed for adventure games. As the title not-so-subtly suggests, Art of Theft stars Chzo Mythos protagonist Trilby. So what's he up to this time? Puzzling his way through haunted houses? Uncovering the secrets of reality-shifting hotels? Well, no. In this game he's doing exactly what got him his reputation to start with – cat burgling.

Art of Theft is comprised of a linear set of missions (or "heists", as they're called), which are linked together via a simple but entertaining storyline. Each heist follows more or less the same pattern. Trilby will first brief you, giving you some back story to the mission as well as outlining the victory conditions. Then it's then up to you to guide him through the level, snapping up whatever swag you can find while ensuring that you don't get spotted by guards, security cameras, laser tripwires or civilians. Should you be detected, the alarm will be raised. This doesn't cause immediate failure, but set the alarm off once too often and Trilby will decide to hotfoot it out of there before the cops show up. Complete the mission, and you're presented with a score screen that outlines your performance and assigns you a rating for the heist, ranging from a pathetic "C" to the elusive "Trilby" rating.

As you've probably gathered from the above description, the main component of Art of Theft is stealth. This is achieved mainly by the best friend of every burglar since burglary was invented - darkness. By sticking to dark or dimly-lit areas, Trilby is able to hide himself from view and avoid setting off the dreaded alarm. Different light levels offer him different degrees of invisibility. Completely dark areas allow him to move freely without being seen by anything but laser sensors. In dimly lit areas he is visible while in the open, but can hide by hugging himself to the wall behind him, an action that costs him the ability to move (this is also useful for dodging the aforementioned lasers). Slipping in and out of visibility without being detected requires careful timing of your movements and actions, but is made easier by the predictable patterns that guards, lasers and cameras follow. Bad timing can be fatal, though, since cameras and lasers will trigger the alarm almost immediately. With human guards, however, Trilby has a last-resort tool – a limited-use tazer built into his "grolly" (a grappling-hook/umbrella hybrid) that he can use to render them unconscious before they hit the panic button.

While dodging Security is the core mechanic of the game, Yahtzee has made an effort to spice the actual theft portion up a little by throwing some extra challenges into the mix. While lootable items are generally placed within relatively easy reach, more valuable items and mission objectives are often secured inside safes or locked rooms, which you must break into by means of simple reflex-based minigames. Written notes are often scattered across the levels, containing security codes or clues to unlocking hidden loot. There are also electrical panels in some levels that, when successfully tinkered with via a chance-based "cut the wire" minigame, will do anything from shutting down security systems to bathing the entire level in darkness.

Not that any of this is superfluous either – for each puzzle you solve and for each bit of loot you steal, you earn Reputation Points. These can be spent between missions on new abilities (such as "sidle", which allows you to move while wall-hugging), upgrades to existing abilities (such as improved lockpicking skills), or even such luxuries as "guard amnesia", which will gradually replenish your alarm count as the level progresses.

At this point, it would be pertinent to mention the influence of Reputation Points on Art of Theft's replay value. You are able to replay any of your completed heists at any point, but any abilities you have purchased in later missions will remain available to you during those replays. Even the earliest heists are designed especially with this in mind, with some areas made inaccessible until you purchase the necessary skills to reach them. In addition, the end-mission rating boost that these new areas provide comes with more tangible rewards than simple bragging rights. A progressively higher average heist rating will unlock special costumes that Trilby can change into, granting him special abilities to aid him in his work. All of this works together to make Art of Theft a fantastically rewarding game to play through multiple times.

I'm going to say it bluntly - Art of Theft is not an easy game. In itself this isn't a bad thing, but the difficulty is augmented somewhat by a few unfortunate quirks. Firstly, the controls tend to alternate between "over-sensitive" and "dead", which works against the whole split-second-timing part of the game. Nothing is more frustrating than whacking the wall-hug key to dodge incoming lasers, only to have Trilby blissfully ignore your command and set off three alarms in a row, or wall-hug and then suddenly pop out again because you held the key down for one nanosecond too long. Secondly, the enemies may follow a predictable pattern, but they have a tendency to go "out of phase". In these cases, at least one enemy in a given area is potentially able to see you, leaving you standing there stupidly for minutes on end waiting for a "vision gap" to sneak through.

That said, Art of Theft is not a bad game either. Despite occasional frustration due to the aforementioned niggles, the game has a lot to offer – it's unique, it's challenging, it's highly replayable, and above all, it's fun. It's definitely worth checking out just to see how far the AGS engine can be stretched with a little skill. Overall, if you're willing to work past the occasionally finicky controls, there's a lot to recommend here. Give it a shot.

THE FAMILY TREASURE

by Simon "Tr00jg" de la Rouviere

You can't help but smell the whiff of nostalgia that permeates the air as your eyes fall upon the gorgeous retro adventure game graphics of The Family Treasure.

This South-African created adventure does not tread any new ground in terms of plot and story, but it delivers in clean, fun gameplay.

You play an old bearded pirate named Bloodhook, and you are tasked to find your family's lost treasure. The rest of the characters are your average pirate crew.

The puzzles are quite easy, so the seasoned adventurer will probably whizz through this game. Even for non-experts, it's not too long. Some of the events are quite random, though, and at times the game doesn't quite make sense. Why a pirate would decide to give you a magnifying glass, no one knows.

The Family Treasure touches on the world of Monkey Island and references to Le Chuck, greatly enhancing the game with that special feel and humour.

As mentioned, the graphics are really well done for an indie project. The place where the game excels the most, though, is the music. The funky pirate tunes create the perfect atmosphere as you scavenge the small island.

If you have time (or a lunch break, even) and just want to rewind with some standard adventure gaming, then you won't go wrong with The Family Treasure.

Sumotori Dreams

by Simon "Tr00jg" de la Rouviere

What exactly is Sumotori Dreams? Well, it is Sumo-ish and not so much dreamy as it is a perfect example of "Drunken fighters, lol!"

A better explanation would be that Sumotori Dreams is fighting game, and a very unique one at that. Your goal is to simply push the opposite player so that he falls or steps out of the ring. Yes, that is all.

Well, okay, not really.

The average epic, grandiose fight takes a humongous 5 seconds. What happens after the 5 seconds is what makes Sumotori Dreams the gem it is.

After you've pushed your opponent over, both of you are probably lying on the floor. The AI takes over and its job is to simply stand up and take a bow. With strategic placing of objects and semi-drunken AI, hilarity ensues as the AI tries to stand up. The competing wrestlers stumble, break blocks (tsk, tsk) and generally just fall around like the pair of complete buffoons that they are. Once they are both standing, they take a gracious bow and the next match can start.

The controls aren't very intuitive, but it is done purposefully like that. The average person can't throw jaw-breaking uppercuts in real life, and it takes practice to throw a decent hit. Your keys are "push with one hand", "push with both hands", "lean forward" and "jump". They could've added a "kick with both feet", but that's just our opinion.

The graphics also aren't absolutely awe-inducing, but since when did we worry about that? The characters are fancifully pieced together by little blocks and the entire game fits into just a few kilobytes.

Sumotori Dreams is also a really interesting game from a design perspective, considering that the biggest part of the game takes place long after the player input has stopped. It is quite a neat idea that could perhaps gain some foothold in other potential titles.

As a game, Sumotori Dreams is a delight to watch. As a rule of thumb, fetch your roommate, housemate, grandmother or cat to play it with you. You'll be giggling like idiots for hours.

Game Graphics with Adobe Photoshop

This article refers to resources available at the "Contents" section of the Dev.Mag website (www.devmag.org.za). It is recommended that you visit the site and download these resources.

This month, Rishal "TheUntouchableOne" Hurbans shows us an example of creating a simple GUI graphics set with the help of Photoshop.

In this tutorial, you will learn how to create some decent backgrounds for your game's GUI. This would apply to things like the menu screens, loading screens and any other area in the GUI where a background would be required. The techniques used here can also be used to create in-game backgrounds and add extra effects to them.

The backgrounds created will be raster images so we will need to know the exact size of the image to prevent different stretching, which would cause the image to look distorted and unprofessional.

As usual we will create a blank Photoshop project. Choose a good usable size for the canvas. I used a 800x600 pixel canvas.

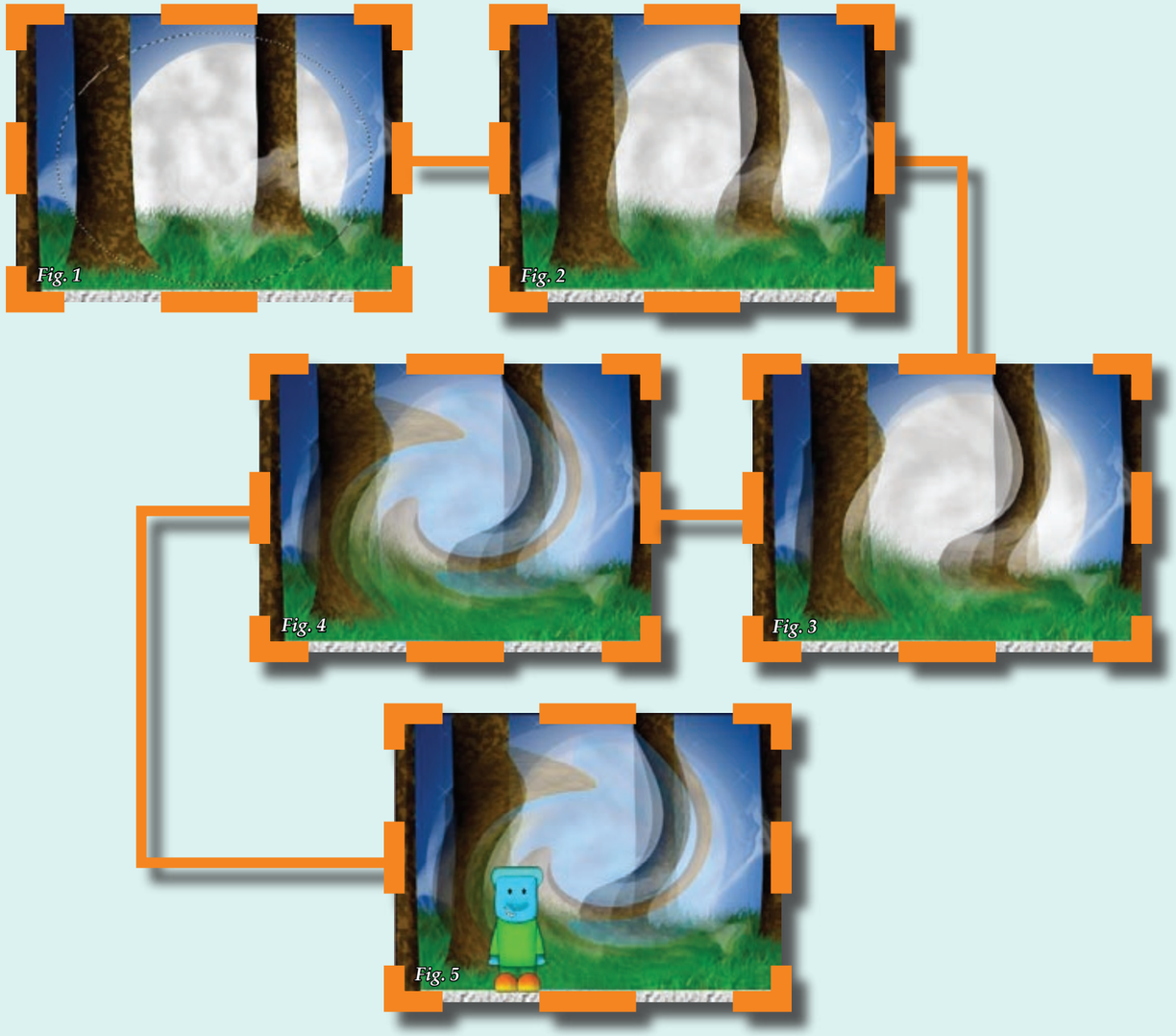
Remember, the colour scheme and theme of the backgrounds in your game should be in sync with the overall theme of your game. As this tutorial has developed, we've created the character, "Smurfy" and a "Moon-light forest" backdrop, so it will only be fit if Smurfy starred in a game that takes place in a forest. A title,"Smurfy's Forest Adventures" or something along those lines.

We will start with the menu screen for the game. The menu screen should be an attraction of your game. No-one wants to start the game looking at a boring or horribly done menu screen but on the other hand no-one wants to deal with an over done cumbersome menu screen. In this particular type of game, I would say the menu background should be somewhat humerous or at least, interesting.

At the stage of creating menu screens, loading screens etc, a good idea would be to use concept art that you have drawn physically or make use of the in-game art already created. The menu backgrounds can also have an abstract look. This can easily make the image look really amazing if done correctly and doesn't take too much effort. By giving the background an abstract look and giving your buttons a more definite and solid look, the menu should be really appealing to the player.

When thinking about the menu screen for "Smurfy's Forest Adventures", many ideas may come to mind. Since Smury is part alien, I thought a warp portal would be good. So a portal with Smurfy popping out in the deep dark forest would be ideal.

It's pretty obvious that we will make use of the previous created images, we will use the front view of the Smurfy vector sprite and the forest backdrop, so if you don't already have the resources, it would be a good idea to download them.



Once you have the blank project opened, open the Moon-light forest.psd photoshop project file. Select all the layers of the Moon-light forest.psd project, right-click, and duplicate layers to the blank project.

While all the layers of the Moon-light forest are selected, right-click and select Merge Layers. We don't need all the separate layers, just the forest in full.

Firstly we can create the warp portal in the center of the scene.

Now, while the forest layer is selected, choose the Elliptical Marquee tool in the tool bar and select an elliptical section of the image as shown in Figure 1. Once the desired area is selected, right-click and select Layer via Copy.

The new layer needs to be distorted to give it the "portal" look.

So, select the filter option in the task bar, select Distort>Twirl, set the angle to around 50 and apply it, now set the opacity of the layer to 50%. The image should look similar to Figure 2.

The image looks a bit blurry and distorted, which is what we want.

Duplicate the portal layer just created. Repeat the Filter>Distort>Twirl procedure but this time set the angle to something greater than 50 but less than 120. The opacity of this layer should also be 50%. The image should now look similar to Figure 3.

Duplicate the previously created layer and apply the Twirl Distort procedure once again, use an angle of around 190 and set this layer's opacity to 30%. We are going to give the last distort we created a blue tint. Select Image>Adjustments>Selective Colour. Choose White and set the Cyan and Magenta values up until you get a nice looking blue(the colour here can be any colour of your choice).The image should look something like Figure 4.

Thats the "portal" done for now. We need to put Smurfy into the scene now. Open the Photoshop project with Smurfy's front view and duplicate the layers to the current project that we are working on. I think the default size of the character sprite is just fine, but if you disagree, resize the vector layer using the Transform, Scale procedure.

I have decided to make the character's one arm look different to the other by simply, flipping it horizontally rotating it slightly(these options are available in the Edit>Transform Menu). After this has been done, right-click on the layer in the layers window and select, merge layers. This will make the vector sprite into a single raster layer (Figure 5) .


Fig. 6


Fig. 7


Fig. 8


Fig. 9

We are going to use the Transform, Warp procedure, so we need the sprite to be in raster image form. Select the Smurfy layer and select, Transform>Warp. Now move the points and "handles" in the warp grid to make it seem as if the character is being sucked into or thrown out of the portal as seen in Figure 6.

Now since all this is happening very fast, we can duplicate the warped character layer, set the opacity to about 50% and move it slightly to the left or right (Figure 7).

We can now use the smoke/fog technique as explained in tutorial four to give the image a few more effects. The effect can be applied to the surroundings around the character as well as around the portal. After you play around with the effect, your image could look similar to Figure 8.

Another effect that would be useful here is the sprayed stroke effect. Select Filter>Brush Strokes>Sprayed stroke. Choose suitable values for the Stroke length and spray radius (Figure 9).

Any other touches can be added to the image as you see fit. I have added a small black dot in the center of the portal to make the portal more definate. Using the blur and smudge tools will also add an extra effect of the portal. Experiment with the tools with different strengths and see what you can come up with (Figure 10).

Now remember, your menu background should be interesting but also not too out there in terms of dominance. You need the buttons and options to be clear and bold in the menu screen, though in the loading screens the background should be really interesting and attractive as a player does not want to wait for something to happen while looking at something boring. There should also be a space for the buttons where they won't be obstructing anything interesting on the background (Figure 11).

The backgrounds can be transformed into many different amazing styles, using the Filter>Artistic option (Figure 12).

Loading bars, buttons, checkboxes and the other GUI controls will be discussed in the next tutorial. By combining the background and the controls correctly, a striking GUI can be achieved. A challenge to you would be to use the animation techniques discussed in tutorial three to created an animated menu background as seen in commercial games.

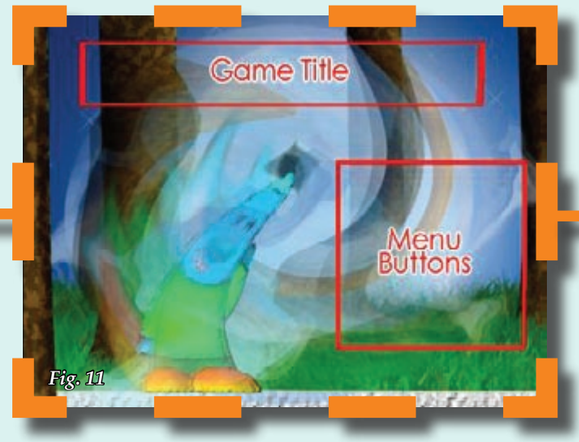


Fig. 11


Fig. 10

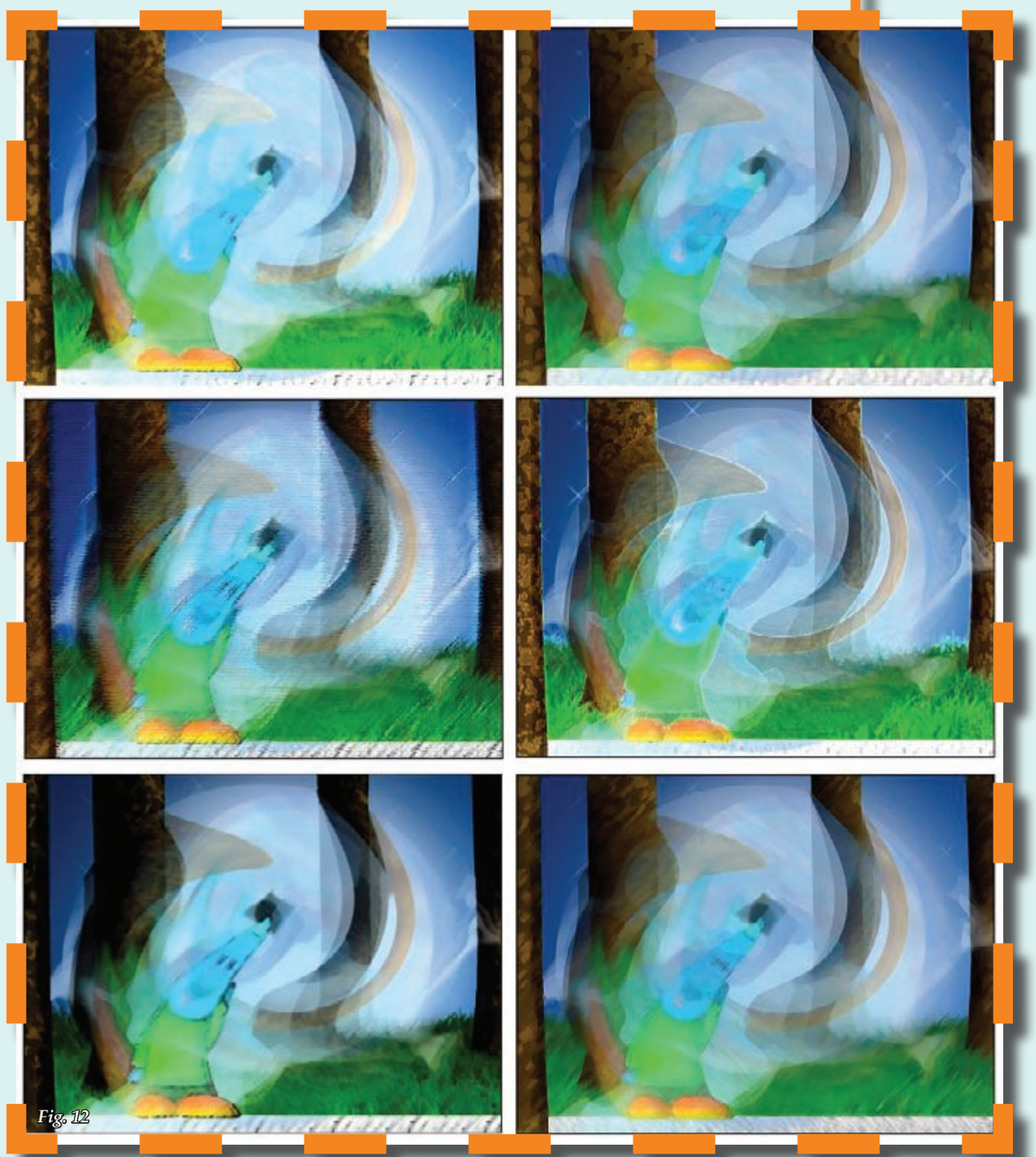


Fig. 12

BUILDING BLOCKS

Procedural level generation in gGuardian

Ever wanted to make entire worlds on the fly without having to manually stick in every blade of grass? This month, Gareth "Gazza_N" Wilcock offers readers a bit of insight into the process of procedural level generation.

We've covered procedural generation before in Dev.Mag (we had an excellent article on Perlin Noise two issues ago), but before we start it might be pertinent to refresh your memory. In game development terms, procedural generation is the creation of game assets on the fly by using code as opposed to standard editing tools. Rather than spending hours upon hours creating textures, levels, geometry, music, or whatever else your game requires, you generate it as the game is running according to rules that you specify.

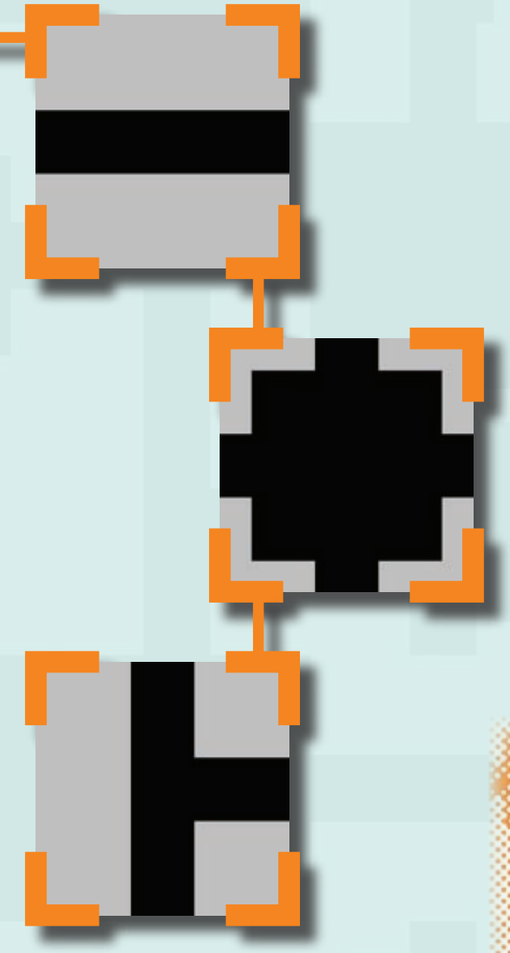
It sounds very complicated and, depending on what you want to generate, it can be. However, procedural generation can be a great asset to the aspiring game developer, and it needn't be that difficult. Since it's best to learn with a working example, I'm going to cover a simple method for procedural level generation that I developed for one of my own games.

A simple level generator

gGuardian is a game that I whipped up for one of the frequent competitions held by Game.Dev, the community that creates this magazine. The rules of the competition specified that we had a month to create any manner of game we chose – provided that the gameplay only lasted for ten minutes from start to finish. I decided to create a siege game where the player must defend a single key location from a multi-pronged assault by hostile aliens. Early on in development, I realised that unless it had a large variety of maps to play on, the game's replay value would be close to zero. I didn't have the time to build and test a whole lot of maps, so the answer was simple: I needed to procedurally generate them.

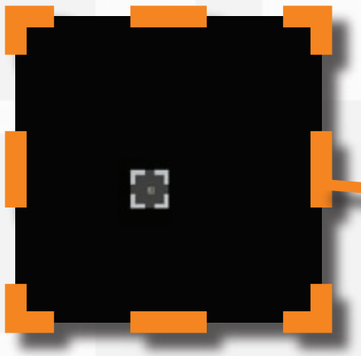
Fortunately, the levels that I had in mind were very simple. I needed a room that would contain the Cryobay, the object which the player was required to protect. I also wanted easily identifiable points from which the attacking hordes could spawn, which I decided would also take the form of special rooms. These two functional room types would be supplemented by other arbitrary rooms filled with decorative doodads, and the whole lot would be linked by a maze of corridors. All very straightforward in principle, but the question was how I could build something like that on the fly.

Eventually, after much experimentation, I came up with a rudimentary but very effective method. I split the level into large (512 x 512 pixel) tiles, and built room and corridor tiles that I could plug into this grid. This saved me several hassles. For one, it meant that I could always be sure that everything linked together properly (since generating and linking corridors on the fly can pose several problems). Secondly, it kept the actual level generation process simple and fast.

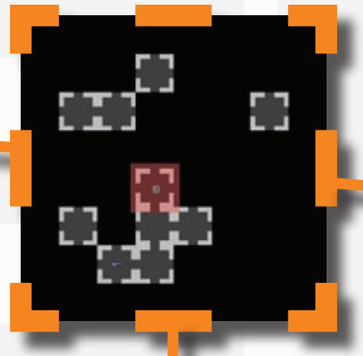


The actual level building takes place over several steps:

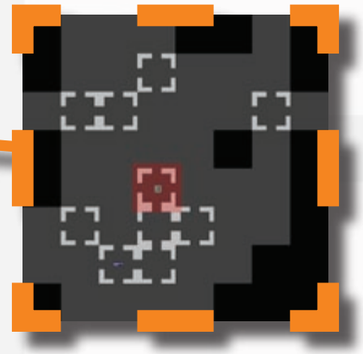
1) Place Cryobay room. This room is always staggered around the centre of the map and, as the name implies, contains the Cryobay, the object that the player must protect.



2) Place room tiles. These are instantiated randomly across the map. Random objects that serve as decorations are also instantiated within these rooms in a way that the algorithm sees fit.



3) Place "generic" path tiles. Each room fires off eight of these - two in each direction (up, down, left, right). If path tiles overlap other paths or rooms, they are destroyed. The idea is to form a "mishmash" of path tiles that fills the spaces between rooms and interlinks everything.

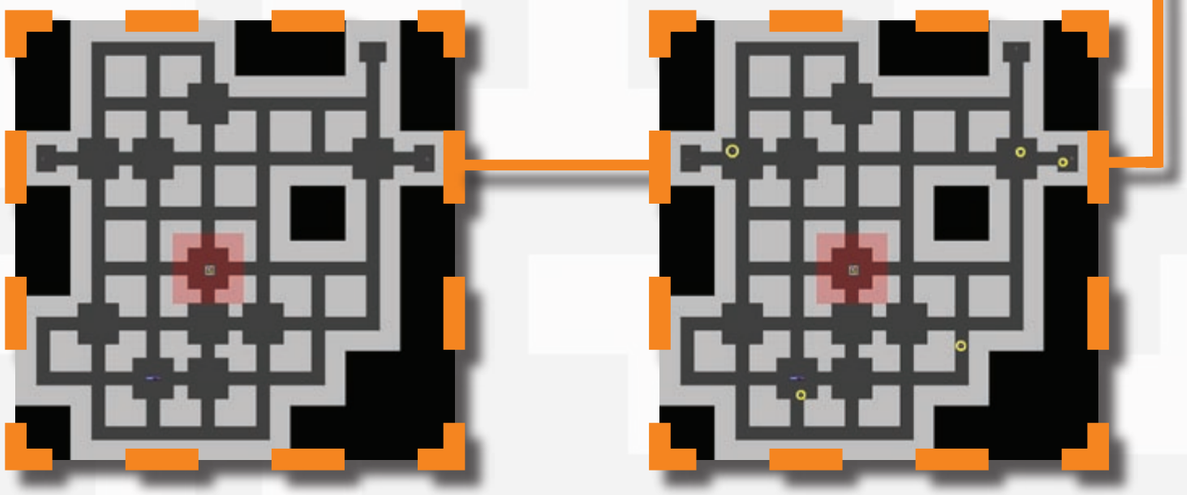


4) The "generic" path tiles scan the surrounding area to determine the relative positions of adjacent room and path tiles. They then use this information to decide what specific type of path they'll become ("L" path, T-junction, straight corridor,etc.), and assign the correct tile to themselves accordingly. As you can see in the screenshot, this forms a complex but logically laid out network of corridors.

If a generic path tile only has one neighbour, it becomes a "Dead End" tile. Dead End tiles are where I place the alien spawn points/portals. The number of portals created depends on the difficulty level. If a portal is close enough to the cryobay and there are sufficient others for the game to play properly, that portal is destroyed.

4) Once the level layout is created, I randomly scatter the weapon pods that the player must collect across the map, ensuring that they don't overlap walls or decorations.

And voila! One complete gGuardian level! Serve immediately.



Design Documents
(What you need to know)

by Sven "FuzzYspoON" Bergstrom

"Oh that's right, I remember now, I actually wanted a cheese gun for Potato Planet Avengers. The game was meant to be a top down shooter where you ran around collecting cheese and shooting spudlings. Now we have 12 weapon choices, 6 different view ports and no gameplay. Now we will never finish the project because the code is convoluted and doesn't compile any more. Let's make a new game."

Sound familiar? This is a horrible occurrence in the everyday world of game development. From the smallest of projects to the largest of code-monsters, there is nothing worse than yet another failed project. Most (not all) failed projects stem from one simple fact: there was no real design "system". This article aims to outline some of the ideas behind a design document, what it is, what it does, and how it can help within a project. Hopefully it can help you churn out a complete project at some stage in your pursuit of writing games.

The idea behind the design

The design document serves to explain the proposed ideas in a well-formed, neatly outlined manner. Above all, it is aimed at keeping the project a sane and manageable task, without losing sleep over adding ideas to a project without remorse.

The design document is not aimed at the end user, it's not aimed at your friends when you tell them what the game is about, it is a technical reference and design outline of what your goals were and should (within reasonable limits) remain.

The document (even if separated into pieces across multiple documents) should be classed as a whole and is considered by each part of the development process. The design document itself is to be a concise, relevant and complete reference in order to accomplish the design.

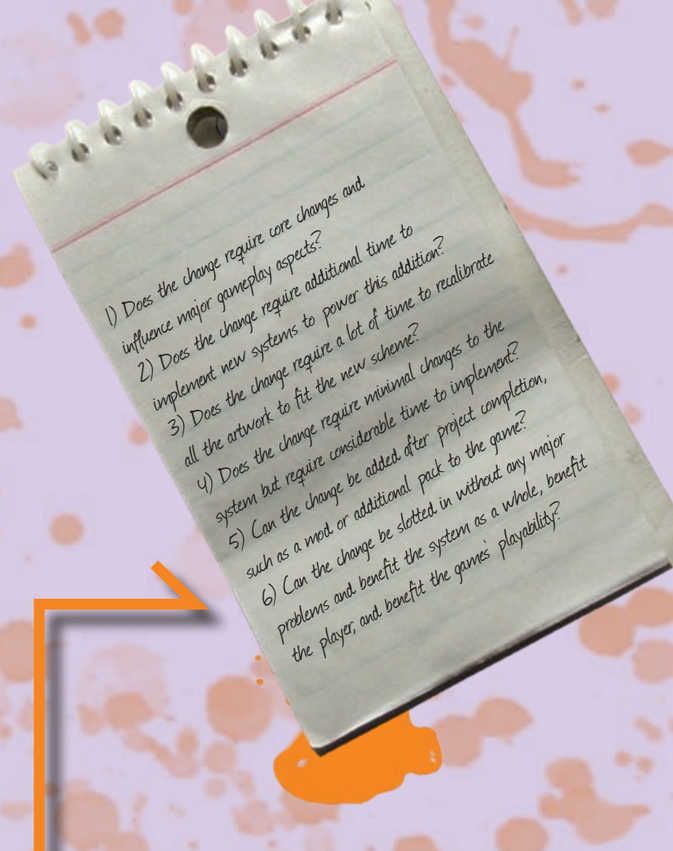
Without sounding too serious or convoluted, it tells everyone what is needed to be done to accomplish the game.

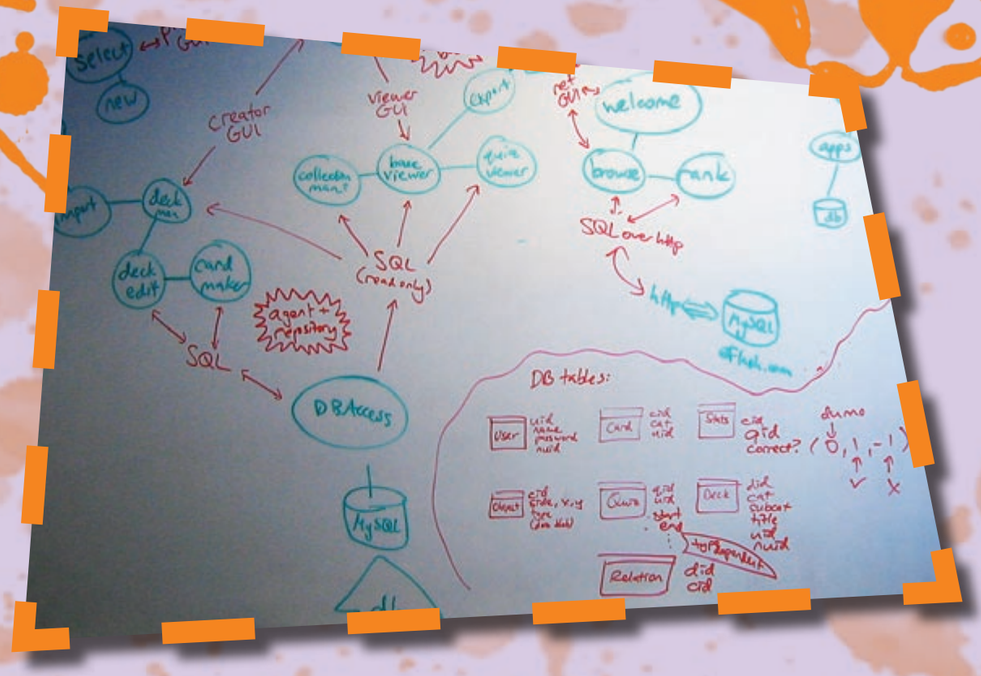
A design document is a very important part of any project, but it is more a guide than a rulebook. There must be someone sensible enough "in charge" of this document and there must be contribution from all teams and participants in a project.

The design document should be reviewed and either updated or reverted to at all major milestones in a project. Leaving no space for additions or subtractions of the design is foolish, but leaving no space for complex additions that can break a project and removing the need for expensive setbacks is what a design document can do for a project. The document helps to remember the idea behind the design, exactly what it is that was started and where it is going.

If you were getting on a train, and you asked the driver where it was going and his only response was, "Uh, I don't know," or a long complicated explanation about the difference between grilled pancakes and cotton jeans, there would be no motivation or even desire to get on that train. Without a destination it is near impossible to know where you are going. The thing with a train (and most games) is they have a starting position (the game idea) and millions of routes to choose from.

A design document is not meant to inhibit the development, or to hinder the changing of any original or additional ideas for a game. Evaluate each addition or change on the following set of criteria:



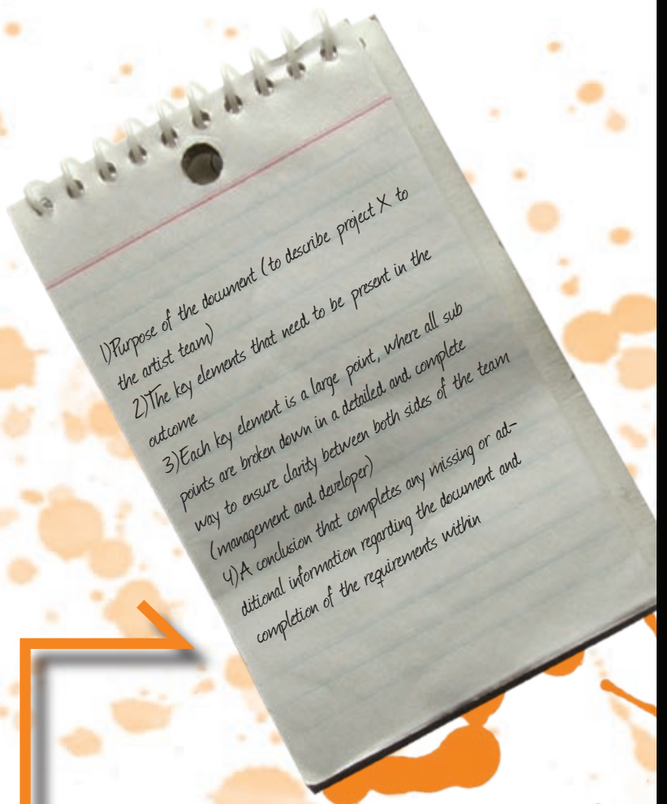


Chances are if the addition will make the game a lot better it is worth considering, but if it is going to hinder development in such a way that the project will be set back hugely, rather leave it out. Certain features in a game will make the game incredible: the development option is feasible, it will take some time (within reason) and will need the core to adapt to fit the new model, but these ideas are not what kill a project. If all participating members or affected members in a game design agree that the addition can be done within a time frame that suits the project manager, the addition is most likely to make the game better and it is worth considering. The important part is that there must be consistency when sticking to a document, and there must be wisdom when it comes to changes. Always check the document when developing parts of the game to see whether you are following what the game was designed for.

The document layout

There are millions of opinions, and always ones that differ. I think design documents change per team, per project, and per purpose, but I don't believe a project can exist without a guideline of what is to be accomplished. For example, a game like Monopoly won't need a huge design document for the programmers as it is common and well-known, but the game could be taking a Star Wars twist and include new instructions for the artists to abide by. Were it a 3D version and artists had a budget of what technology the game is able to use, it would be beneficial to state the key elements of what is allowed or not allowed to be created: for example, a poly budget that limits each scene to a number of polygons in-game.

A few major things a design should contain are a number of steps you may break down into any and all aspects of the project. Things you might want to keep regardless of the project is the following:



In closing

Opinions on formulated design documents are viable, but there is room to improve. If you have ever used a design document you will know there is consistency, flow, and usually success in completing tasks assigned to you. This is especially helpful if you are not working under your own design, have a couple of level-headed friends and pass your design along to them, asking them if it is worth the effort to dive that deep or whether you should leave out some options for the sake of actually getting a game complete.

Making complete games is a matter of wisdom, logic and perseverance. Having unrealistic goals or being too shallow in concept can easily be identified when a design document is in play.

TAKING THE HIT

Criticism is an important part of the game development process. Simon "Tr00jg" de la Rouviere gives a crash-course guide on how to take it, when to heed it and what can be learned from it.

So you have just created your masterpiece and now you submit it to the Internet to be praised by your loving fans. Instead you are met with a rabid herd of people claiming how you wasted their time with your retarded game.

Well, this article is here to help soften that blow.

Almost everyone's first game is bad. So, instead of going sulking, you should try to always see your first game as a learning experience and pave forward.

So, now that you have your first game out of the way, you will most likely start receiving more criticism for your upcoming games. I like to divide criticism into 4 categories.

The 2 extremes first:

1) "1337 crap"

This criticism is along the lines of "OMG! This suxxor. 0/100!". You can pretty much treat it the same as what it is like. Trash. Just ignore it.

2) "1337 awesome"

On the other extreme, you have people going, "OMG! This is awesome! 10000/10!". With these you won't know why it is awesome, so the best is just to reply with a simple "Thanks for playing. I am glad you enjoyed it."

The next two types of criticism are a bit rarer:

3) "Bad, but here's what's bad"

This criticism is bad, followed by a reason why your game is as bad as it is. This is probably the most important criticism you can receive and most developers don't want to even look at it.

When receiving criticism, it is important to try and be as objective as possible. If someone did not enjoy it, there must be a reason. This is why these criticisms are the best. Most of the times you discover things you never knew would be detrimental to your game.

4) "Good, but here's why it is good"

This criticism is the same as the above. It is really vital for expanding your game development. You will know what you did right so that you can re-implement it the next time around.

Responding to criticism:

It is often the case that when people are met with bad criticism, they feel insulted. It won't help at all to flame someone who went to the trouble of playing your game. That player won't come back to play your next game, and you will have learned nothing from potential helpful criticism.

However, taking heed to every criticism will most likely lead you to never finish your game at all. With criticisms like, "add a flamethrower", "add more maps", etc it is best to take it with a pinch of salt. In the end it is your game, after all. It is your creation and we all know art is subjective. One can't really prescribe what you should do, but it's generally a good idea to try remain objective about criticisms while taking your own idea of the game into account.

Giving criticism:

As mentioned above, your criticism will most likely fall into those categories. If you have the time and patience, it is best to try and give constructive criticism: add suggestions and give the places where you found bugs.

In the end, criticism isn't there to attack you, so take heed to it. Just don't take it all into account either, lest your game becomes something you never actually wanted it to be.

a matter of time

Think fast and get ready to roll. Rodain "Nandrew" Joubert has an in-depth look at the mechanics of time-based gameplay.

We've all experienced it: a warning klaxon goes off and a frantic radio message buzzes in your ears. You have to get your squad out of the secret weapons facility before the malfunctioning doomsday device goes off and blows everybody inside to smithereens.

At this point, a set of glaring red numbers appears on your heads-up display and inexorably begins counting down, ticking its way towards your impending destruction. You turn around to take the elevator, but an unexpected explosion wrecks the controls and bathes the corridor in flames. Panicked enemy operatives run about, firing wildly in all directions and adding to the chaos. The sirens get louder, the evacuation messages start booming. But the calamity outside barely presents a whisper in comparison to the all-too-clear sound of your ragged breathing and the pounding of your heart.

Nothing adds to the urgency and the immediacy of a game quite like a good old-fashioned time limit. One could be tearing through the Green Hill Zone as Sonic, racing against the clock in Time Crisis or even just dashing for that next checkpoint in that good old arcade racer. Whatever the situation, players invariably find themselves eyeing that steadily decreasing meter. If you as a game designer can capitalise on this concept properly, it's one of the most powerful ways to have the end user emotionally engage with your creation. Think "frantic". Think "intense". Think "awesome". These words have all been used to promote games, and they could apply to yours too if you work at it.

The importance of time

The Game.Dev community (www.gamedotdev.co.za) recently held a time-based competition that exploited this particular concept in gameplay. The premise was simple: build a game, any game, as long as it was over in less than ten minutes. However, as with all such competitions, the execution was a lot more involved than the premise, and it became clear to participants that constructing a game to capitalise on the potential of a time limit was more complicated than it looked.

As mentioned already, a time limit provides the developer with a valuable emotional hook with which to lure potential players and keep them going until the end. It's this sense of urgency which lends a sense of importance to what may be an otherwise passive experience.

In terms of functionality, time limits also serve an important role. Today's gamers are sitting in the lap of luxury: they're offered more high-quality titles than they can possibly find the time to play, and even the obsessed, 16-hour-a-day, twitch-reaction junkie has to be very discerning with regards to the games they choose to play. In an arena as saturated and competitive as indie game development, it's often unwise to create an epic that requires any great deal of time or effort on the player's part. By establishing, say, a ten-minute time limit, you're immediately offering the consumer two things:

(1) An assurance that they can complete this game within a very specific time interval. Even if they're

initially uncertain about picking up your creation, the fact that they only need to make a small time investment can encourage them to give it a go.

(2) A clearly defined and well-established goal from the start. Short-, medium- and long-term goals are an important consideration for any game developer, as they lend your project structure and give players a sense of purpose. With a timed game, the task of goal-setting becomes much easier: your long-term goal is to defuse the bomb in the next ten minutes. Your mid-term goal is to find a schematic of the building as quickly as possible. Your short-term goal is to clear the current room of enemies. Bam! You have your game structure.

Timing it right

Simply tacking a countdown timer onto your game isn't going to win hearts. It needs to make sense and, if possible, become an integral part of the gameplay itself.

For example, one of the top entries in the Game.Dev competition involved guiding a robot to collect material, build turrets and stop waves of aliens from reaching a spaceship's cryobay where the human passengers were hibernating. The key element was that this robot had limited fuel: after ten minutes, it would power down and any remaining aliens would overrun the cryobay. This was the basic time limit premise. It made sense, too: it was acceptable to players and fitted in well with the game world.

More importantly, however, the game touched on one of the great strengths of time-based games: scripted gameplay. While many entries simply attached a timer and allowed the player to engage in activities until their proverbial battery ran flat, the spaceship defender dealt with a clear progression in the game. The first few minutes were spent scrounging about for weapon salvage and useful equipment. After that, the player had to move on to establish turrets and prepare defenses. Then, after a pre-ordained interval, the alien portals opened and enemies began charging towards the cryobay. This was obviously the beginning of the combat phase. In the last few minutes (particularly on higher difficulty levels) the player was again required to pack up turrets and re-establish defenses in more heavily-hit areas to eliminate the last of the enemies.

The point? This wasn't simply ten minutes of the same repetitive action. In each gaming session, the developer used time as a weapon to create a buildup, a climax and a conclusion – in much the same way that an author would pen a story. The game kept moving forward and the timer forced the player to keep heading towards the next meaningful objective. It worked.

Aside from scripted gameplay, another way to make the timing more meaning is to provide regular feedback to the player and thus work on oiling those emotional hinges. Compare a simple ten-minute deathmatch to one which, after every two minutes, offers some sort of reminder that your hourglass is running towards empty. This can be done visually (your player is poisoned, and as he heads towards death your screen steadily flashes more green), aurally (consider the "10! ... 9! ... 8! ..." counter of Unreal Tournament) or through actual game events (the building you're in is collapsing, and the roof starts to crumble away).

In conclusion

A timed game is more complex than one may initially think, but the rewards for the player – and thus the developer – are potentially great. As always, there are other design aspects to consider when involving oneself in a timed venture, but this insight comes from the two most important developer weapons of all: making mistakes and learning from experience.

Hopefully, this guide will set you off in the right direction and help you with any initial time-based gameplay projects. From there, it's all about refining your craft, seeing what works and striving to make the sweat form on your audience's brow. And you have plenty of time to figure that out.

FROM THE COMMUNITY ...

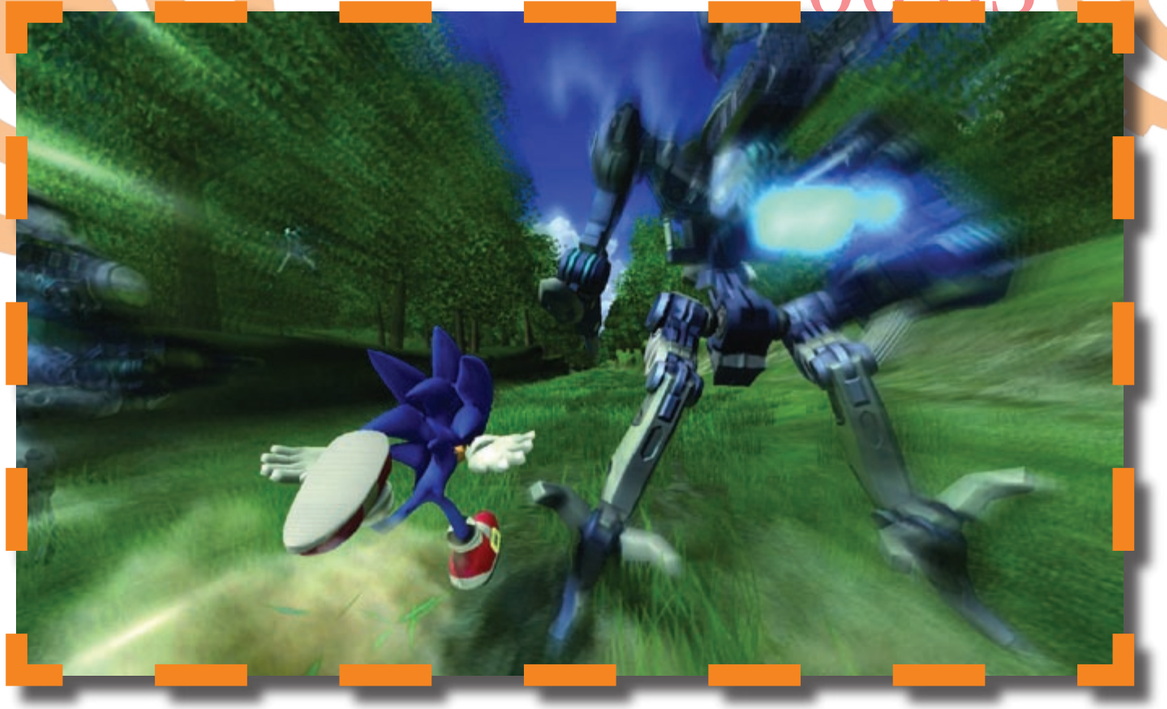
Q: "What did you learn from the time-based competition held by Game.Dev?"

CYBERNINJA: "I learned that there comes a point in development where one needs to focus on just one good idea/theme and see it through to the end."

THAUMATURGE: "I think that I learned that while one can come up with an interesting concept in a few days, it's rather little time in which to flesh out interesting gameplay."

FENGOL: "Simple ideas work. Your game doesn't need to be filled with different mechanics to be enjoyable."

GAZZA_N: "I learned a lot about pacing a game. Balancing the game so that the player had a fair chance, but was still required to plan and act quickly, was an enlightening exercise."



A trip down MEMORY LANE

A look at the history of Dev.Mag

In early 2006, Dev.Mag tentatively spread its wings and went forth into the world, delivering with it a message from a few game development hopefuls with fresh pens and starry eyes. Because nostalgia is always fun (and because it's also cool to give readers some perspective) Rodain "Nandrew" Joubert decides to grant some enlightenment and offer ...

By now, you've already looked at some of these 40-plus pages of neatly packaged game development wisdom from one of South Africa's most dedicated and enthusiastic communities. Hopefully, you appreciate the Features section, the lovingly-crafted reviews, the generous helping of tutorials, the copious design advice and this edition's special makeover.

It may be difficult to believe that a little over two years ago, the concept of Dev.Mag didn't even exist. It may be a further stretch of the imagination, however, to visualise the first edition of the magazine: barely ten small pages of hacked-together content provided by a few intrepid developers who were both excited and uncertain about the project they were embarking upon.

From there, the Dev.Mag pages – and fan base – have grown significantly. Thousands of readers, both locally and internationally, now enjoy a wide selection of game development articles every month and we hope to continue growing from here.

To put things into perspective, here's a few pages dedicated to having a look at where Dev.Mag has been so far.

EARLY DAYS

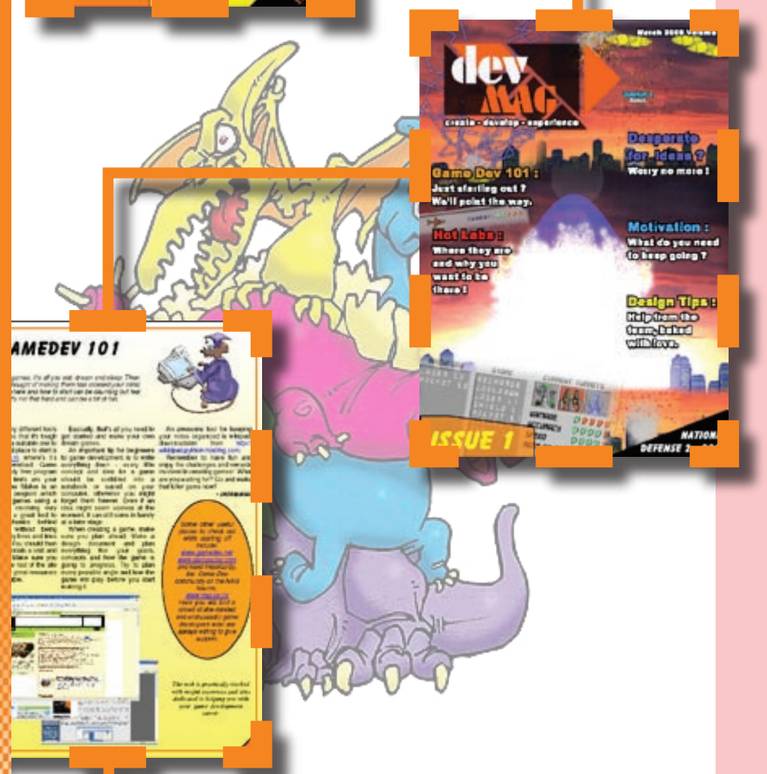
The first edition of Dev.Mag was rather creepy. It looked hideous, the content was much cruder than later offerings and it was, overall, a culmination of everybody's first attempt at the deal.

The initial Dev.Mag team consisted of about a dozen eager developers sitting about and scheming on a local forum, headed by one Stuart "GoNzO" Botma. The concept of the magazine was introduced in December 2005, and the first issue was released a little over a month later. It enjoyed a modest amount of success, spread by word-of-mouth and a raw bundle of enthusiasm. This was an exciting time for the community – the idea of a magazine was fresh and interesting. People were keen for more. Work on the second issue started soon after that.

One of the saving graces of the magazine early on was the entrance of our first designer, Brandon "Cyberninja" Rajkumar. Cyberninja was, as far as we could remember, a complete novice to the art of game development, and some of us were concerned that his rather sudden appearance combined with his lack of experience in the development field would result in an enthusiastic layout monkey who would design an edition or two before burning out and abandoning the project due to boredom.

We couldn't have been more wrong. Cyberninja soon bullied his way onto the staff through sheer force of talent – even though his work was patched together in his spare time between numerous other projects, his capabilities awed the team. The stuff he produced was, quite frankly, miles ahead of the work that we'd generated for Issue 1. He continued to serve on the magazine afterwards, got into game development itself and became a solid part of the community.

Dev.Mag's initial online presence was helped along by Google Pages, which provided a convenient springboard for our publishing ventures. Although it wasn't the most powerful hosting mechanism, it was free and convenient for our purposes. The site was humble and no-nonsense – it provided us with a platform from which we could distribute the mag and advertise our presence to those outside the community.



FOOT IN THE DOOR

During our earlier ventures with magazine promotion, we did the rounds on various forums including that of the Game Maker community. A memorable point in that endeavour was a comment from one of the members, quite possibly the most encouraging thing that I have ever heard in relation to the mag. Our publication had just hit the fifth issue and this individual mentioned that it was noteworthy because most online magazines petered out before they got this far. We'd survived the initial whittling phase, it seemed, and had moved on to establishing ourselves as a young and promising online release.

One of the most impressive things about Dev.Mag has been its ability to stand firm over time despite numerous upsets and the inevitable staff turnover. After the fifth issue, we moved from simply pushing out a monthly release to refining the work we sent out, doing our best to expand with new ideas, innovations and content. The magazine changed a great deal during this time, undergoing a metamorphosis which would ultimately prove to be beneficial. Our eyes were opened to the complexities of running a magazine, more focus was placed on recruiting writers for extended content and we started securing interviews from parties such as Introversion, Jacob Habgood and prominent local studio Luma.

Within this time frame fell rAge 2006, a South African gaming and technology expo. Interestingly enough, this wasn't just our first major marketing event: it was also the first time that the Dev.Mag team actually had an opportunity to meet face-to-face. One of the interesting things about an online publication is the fact that team members often correspond across the country – or even the globe – in the course of their work without ever confronting one another. It was a novel experience to shake hands with individuals who had hitherto been mere nicknames on a forum board.

The rAge edition was also a source of pride for us because it was the first issue that we could technically say weighed in at 40 pages – an achievement at the time, although our spreads back then were only about half the size of later designs. Content was the one of the aspects of our mag which swelled the most around this point, due in part to the incredible amount of people volunteering as new writers.

MAKING THE MOVES

By the time we'd been active for a year, Dev.Mag had undergone numerous redesigns and sections were either added or re-organised according to the magazine's needs. The most notable aspect of our development was the establishment of several highly popular article series, including those by William "Cairnswm" Cairns and Luke "Coolhand" Lamothe, two active game developers in the South African community. Though neither of these individuals are currently regular contributors to the magazine, they made an enormous difference in raw content and article style, raising the bar for Dev.Mag submissions in issues to come.

Over the next few issues, Dev.Mag began establishing relations with several game development groups and organisations, swapping advertising space and expanding the fan base considerably. It was during this period that monthly downloads from the site alone began to climb quite high – practically skyrocketing by the time rAge 2007 arrived. It was around this time that Dev.Mag had the privilege of reporting on the considerable advancement of local developers such as Luma and Retrotoast, the latter having just secured a game publishing deal after their success in a local competition.

During this time, the magazine was still constantly undergoing tweaks and reorganisation, something which we were able to do with little hassle due to our publishing medium. Experimentation with new ideas – such as the open Opinions section, the popular History pieces and the Blue Pill article series – was common during this time. Some concepts were retained, others were discarded and still others were changed about to become something else.

This era also introduced our resident Dev.Bot mascot, DB (pronounced "deebee"). Crafted lovingly by one Geoff "GeometriX" Burrows, DB has since established a strong presence on Dev.Mag's pages and serves as a valuable beacon for the magazine's identity. Further innovations are planned involving the Dev.Bot, so watch this space ...

HERE AND NOW

Despite the fact that the release of individual Dev.Mag issues has since slowed down (due in part to additional quality control measures and the sheer weight of content in each issue), the magazine has become more popular than ever and contributors regularly approach internationally-acclaimed indie and corporate game developers for interviews, reviews and general chit-chat.

Coverage of major projects such as Aquaria, Audiosurf and H-Craft Championships have become Dev.Mag's bread and butter. In addition to the core team of hobbyist developers, the magazine contributor line-up now sports several trained journalists, a helping of professional game developers, a handful of industry evangelists and one or two experts in related fields. All of these individuals have approached the magazine with unrivalled passion and enthusiasm, and to this day the magazine is still built on the entirely voluntary efforts of a dedicated and tightly-knit game development community.

With a new executive team taking the helm of the mag, it now only remains to be seen where Dev.Mag is taken next. Whatever decisions end up being made, one thing is certain: the magazine will continue growing and you, dear reader, shall reap the benefits. Thanks for coming along on this nostalgia trip with us, and we hope to see you again next month!