

ISSUE 7 2006



DEV.MAG

CREATE • DEVELOP • EXPERIENCE



rAge

SOUTH AFRICA'S FIRST GAME DEVELOPMENT MAGAZINE



DEV.MAG ISSUE 7 2006

REVIEWS: LEGEND OF SHADOW: GAMASUTRA
INTROVERSION, MARKETING 101: BRANDING

CONTENTS

REGULARS

03 - ED'S NOTE

04 - DIGITAL STOMPIES

FEATURE

06 - ROACH TOASTER POSTMORTEM

SPOTLIGHT

08 - INTROVERSION

REVIEW

12 - GAMASUTRA

13 - LEGEND OF SHADOW

DESIGN

15 - QUALITY TOUCH PART 3: ADDING THE QUALITY

17 - BLENDER TUTORIAL: MATERIAL AND TEXTURE

19 - PROJECT MANAGEMENT: PART 3

21 - FRAMEWORKS: THE GAME LOOP

22 - MAKING 2D ASSETS WITH 3D SOFTWARE PART3

MOBILE

25 - GAME DEVELOPMENT IN JAVA: PART 5

TECH

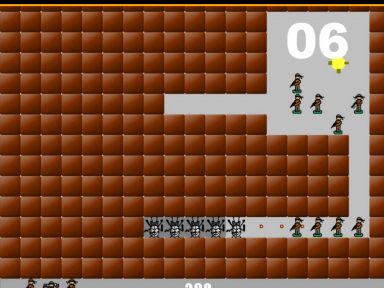
31 - DATA STRUCTURES: PART 2

TAILPIECE

33 - MARKETING 101: BRANDING

COMIC

36 - DIGITAL MAYHEM



ED'S NOTE

Yes, rAge is here. Or already past, if you weren't lucky enough to be at the expo when you got this edition of Dev.Mag. If you didn't go, I can only stress that you make every effort to be there next year, because it's a pretty big event for anyone to miss out on!

You may have noticed that the recent Dev.Mag release schedule has been altered. This was to accommodate for rAge, and after this issue you can expect us to return to our normal routine.

In this issue we have something very special lined up for you. We were lucky enough to get a hold of Introversion and interview them on their recent title, DEFCON. For those of you who don't know who they are you can start off by climbing out from under your rock. Introversion is the company responsible for the creation of Uplink and the award-winning Darwinia. You can find the interview by flipping just a few pages.

Keep an eye out for us if you're at rAge!

Editor

Stuart "GoNz0" Botma



THE TEAM

RANKING OFFICER

Stuart "GoNz0" Botma

SECOND IN COMMAND

Rodain "Nandrew" Joubert

DESIGN SQUAD

Brandon "CyberNinja" Rajkumar

Paul "Higushi" Myburgh

CEREBRAL SOLDIERS

Simon "Tr00jg" de la Rouviere

Ricky "Insomniac" Abell

William "clairnswm" Cairns

Bernard "BurnAbis" Boshoff

Danny "dislekcia" Day

Andre "Fengol" Odendaal

Yuri "knet" Oyoko

Heinrich "Himmler" Rall

Matt "Flint" Benic

Luke "Coolhand" Lamothe

Greg "Zphyr" Reveret

Geoff "GeometriX" Burrows

WEB WARRIOR

Claudio "Ch1ppit" de Sa

Robbie "Squid" Fraser

WEBSITE

www.devmag.org.za

To join, make suggestions or just tell us we're great, contact:
devmag@gmail.com

This magazine is a project of the NAG Game.Dev forum.
Visit us at www.nag.co.za

All images used are Copyright and belong to their respective owners.
If Chuck is reading this magazine:
All jokes within are for entertainment purposes and should not be taken seriously or acted upon ie: Please don't roundhouse kick us.

DIGITAL STOMPIES



Mobile Dragon

<http://www.hitech.herocraft.com/technology.htm>

As the mobile gaming world expands, so does the demand for mobile development environment. Mobile Dragon is a cross-platform 2D/3D game engine written in C++, designed for use with PDAs and Smartphones running any sort of mobile OS. It includes all the necessities for mobile game dev, supporting networking, 3D environments, numerous input systems and its own physics and math libraries. The 1.0 beta release is currently available, complete with documentation and several demo applications.

3pointD.com

Blog: 3pointD

<http://www.3pointd.com/>

"The Metaverse and 3D Web, as blogged by Mark Wallace and friends". Sounds simple enough, but has a surprising amount behind it, including some really handy and insightful looks at the world of computers and the Internet. Although not exclusive to the art of game development, this blog does provide some handy tidbits here and there, and merits a read for those interested in the greater IT world as well. If you don't fancy this stuff, give it a miss, but avid blog readers should enjoy this one.



Game Career Guide

<http://www.gamecareerguide.com/>

A new site, affiliated with Gamasutra, has just recently launched for people who are interested in learning about - and getting into - the gaming biz. It's already filled with features such as how to apply for your first game development job, as well as reviews of good game development books and a convenient "Getting Started" section for those who are new to the site. Round this off with an all-important newsletter for those who want to keep track of what's going on and presto, you've got a valuable online resource that's open to all.

Games - Serious Business!

<http://seriousgamesource.com/>

For those who are interested in the "serious" side of gaming and how videogames can be put to practical use, take a look at this site. Serious Games Source is a place dedicated to the coverage, discussion and promoting games which are used for training, medical, military, educational and governmental purposes. This includes news on Edinburgh University's development of an "Anti-bully" title designed to help children, features on the latest war simulations and reports on the Serious Game Summits. This site provides an interesting, and rarely explored, perspective on the world of video games.





Innovative mobile gaming

http://www.gamevil.com/eng_new/game_view.jsp?game_id=4&category=1

Nom 2 has received coverage from many gaming sites for its innovation, style and fun. Developed by Gamevil and priding itself on its "one button" gameplay, this 2005 title is a great example of how innovation is definitely flourishing within the mobile genre. For more info on this game, visit the Gamevil website, or check out the Gamasutra postmortem at http://www.gamasutra.com/features/20060908/shin_01.shtml.



Book: Artificial Intelligence for Games

<http://books.elsevier.com/us/mk/us/subindex.asp?isbn=0124977820&country=United+States>

AI in games is not the easiest development path to take, and with good reason. Few good books exist on the subject, which make gems in the series that much more worthwhile. Artificial Intelligence for Games has recently been made available, and reviews promise that it's worth any aspiring AI programmer's time to look through. The book goes through the AI process from beginning to end, citing examples from actual games and providing snippets of C++ source code and a helpful CD-ROM to assist readers.

XNA Game Studio Express beta release

<http://msdn.microsoft.com/directx/xna/>

Indie developers, rejoice! Microsoft has announced the release of a brand-new product that enables the average Joe Developer to create games for the Xbox 360. The XNA Game Studio Express, while still in beta, already promises free game development for the Windows XP and Vista environments. For an annual fee of \$99, these games can be shipped over to the 360 console as a perk of the XNA Creators Club, which will be implemented when the full version of the studio is released.



Autodesk community portal available

<http://area.autodesk.com/>

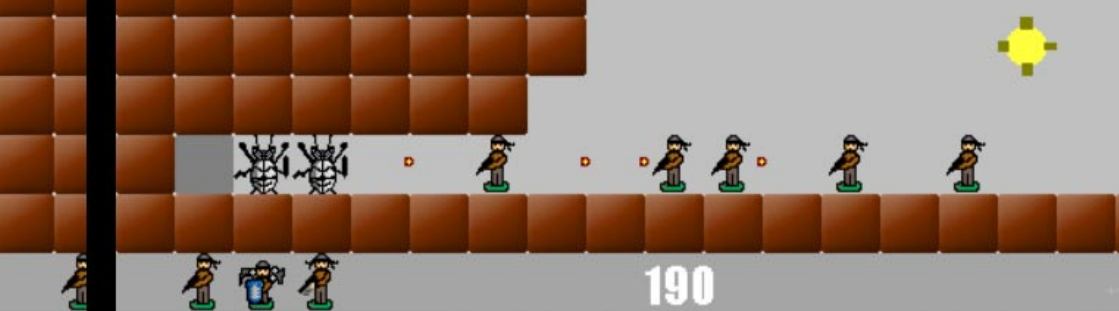
The creators of 3D Studio Max have now created a community portal known as "Area", designed to allow people to showcase creations made with the full range of Autodesk animating and rendering products. Not only does the site offer free membership, but has quite a hefty amount of perks associated with such membership as well - including blogs, catalogues, showcase areas and access to the local forums. The site also contains news and tutorials for enthusiasts, making it a valuable stop for anyone who works with Autodesk products or even 3D tools in general.

FPS "Quantum Leap" awards

http://www.gamasutra.com/php-bin/news_index.php?story=10720

A recent feature on Gamasutra requested readers to submit motivations for their all-time best FPS in terms of how much it added to and pushed forward the genre. Here's the resulting lineup of the finalists. Honorary mentions of typical favourites such as Doom and System Shock did the rounds, along with a few surprise entries here and there. It just goes to show how the bounds of the FPS genre can always be pushed back to bring about something new and refreshing - and how rewarding creative thinking can be!





Roach Toaster postmortem

It all started while watching the daily news on TV. What I wanted to recreate was a riot control squad. The basics behind the game detailed that you should first control a situation and then exterminate it. We all wish we could exterminate some rioters every now and then, but it seemed inappropriate. Due to some unknown reason, I decided to switch to a highly trained team of militia that will try to get rid of the neighbourhood's roaches.

The name evolved from the term "Roach Coach" taken from a Powerpuff Girls episode (yes, that name sticks in your head for a very long time) to "Roach Toaster", which aptly fits the game's premise.

I am very proud of Roach Toaster. It is my most popular game and has been received very well. It has appeared on gaming magazines and freeware games databases. Since the download counter was initialised on Roach Toaster in January 2006,

it has received 1000+ downloads. It is at the current version of 1.2. Work was started on version 1.3, but it was scrapped for the sequel.

Here is what I learned and I hope it will help you too:

Design everything from scratch



When I started development on Roach Toaster, it was a sort of hit and miss affair. I did not quite know how it would turn out. This resulted in design problems that popped up during all stages of development. The basic premise of the game turned into something I did not intend it to be. This luckily turned out to be a good thing in a way, but I felt

that it was not what I wanted it to be, which was control and then exterminate. I had to dig up totally new game mechanics to justify some design problems.

So, to avoid your game taking a turn for the worse, remember to design every aspect of the game and to play

every part of the game in your head. Develop with the end in mind. This might seem strange, but it is wise to design the way you are going to develop your game also. When you start on a game you do not expect your current development to hassle you later on, but believe me, it does! Instead of starting with parent objects, I made a separate object for every unit. This

meant that I had lots and lots more to do when I detected a general bug (no pun intended and for hereafter) in the units, and thus had to change "something" in every unit, which took quite some time.

So my advice is develop with the end in mind. You do not want to get stuck in a situation like this.

Resources (graphics and sound) do matter:

I have always had the motto that "gameplay is king". In every aspect I still believe it is true. What urges people to download your game? What little do they have to judge your game upon? Screenshots. This means that your game should also look pretty and have eye-candy.

When people commented on Roach Toaster I realised that if I had better graphics I might have gotten a few more downloads, and every download matters. If I had better graphics, the overall game would have been a much more pleasant experience.

I will give you a perfect example: When I first released Roach Toaster (version 1.0) to the Gamemaker Forums, it received almost no downloads. The main menu was horrid and it used resource pack sprites. I changed the menu, spruced up

the graphics in some parts, and now, on the 2nd time round, Roach Toaster nabbed about 250 downloads just from the Gamemaker Forums.

Sound is crucial too. If it is bad and cannot be muted, people will stop playing your game before they have even started it properly. I got this comment from another forum.

Develop for yourself:

One thing I noticed that some people apparently disliked Roach Toaster downright, while others praised it for its "excellent design" and "awesome fun". When asked why, it came to the fact that they simply do not like this genre, or these types of games.

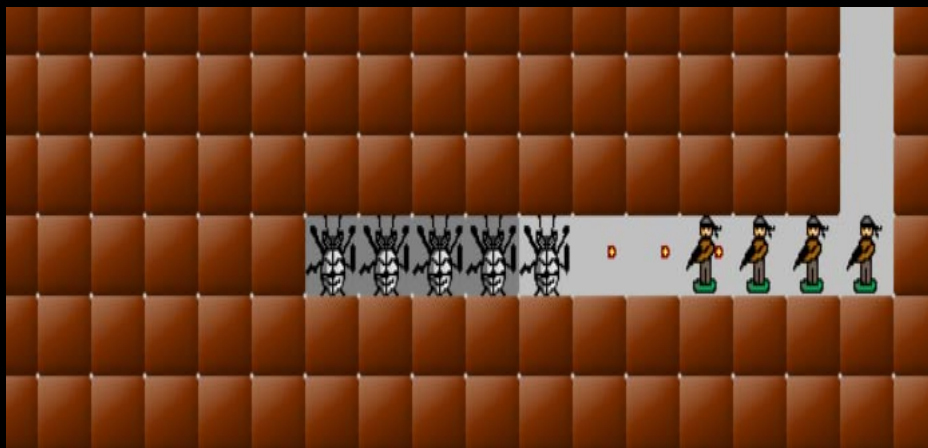
Since this a niche market, it did not get as many downloads as, say, an action title (although there are exceptions). Thought has been put into making Roach Toaster 2 more action-packed for those people who like action, but I decided against it.

I developed Roach Toaster because I liked the concept and thought it would be fun to play. If you develop games that are fun for yourself, regardless of how small the market is, there are bound to be others who will enjoy playing it just as much as you enjoyed making it!

TR00JG

Find Roach Toaster at Tr00Jg's website:

<http://www.shotbeakgames.za.net>



FEATURE

Introversion software

The last of the bedroom programmers

Recently, Dev.Mag was privileged enough to gain an audience with the people at Introversion - the self-confessed "bedroom programmers" responsible for indie gaming greats such as DEFCON and Darwinia. TROOJG takes the helm and has a chat with them.

What advice do you have for bedroom programmers all over the world?

Setting up your own company is extremely difficult and your driving impulse must be primarily for the love of gaming and creating games - although the money (once it comes in) does help. Above all, you need to have the courage of your convictions - we knew we had something worthwhile at Introversion, strong games and unique ideas; we just needed the grit and determination to see it through even when things got rough.

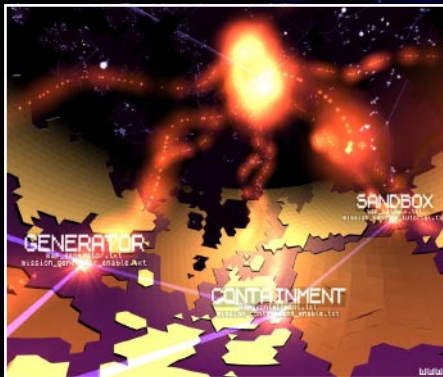
It's probably not what anyone wants to hear but being part of an independent developer involves a lot of hard graft and is not even half as glamorous as it sounds! Of course, there are enormous benefits to being independent - primarily you can do whatever you want! We own our own company, decide our work hours, and we don't have publishers breathing down our necks telling us what to do next. This allows us pretty much complete creative control but also means we have to remain focused and disciplined - harder than it sounds, especially if the cash flow is drying up. That said, we've had some great moments here at Introversion, and our success at the IGF awards this year made all of the blood, sweat and tears over Darwinia seem worth it.

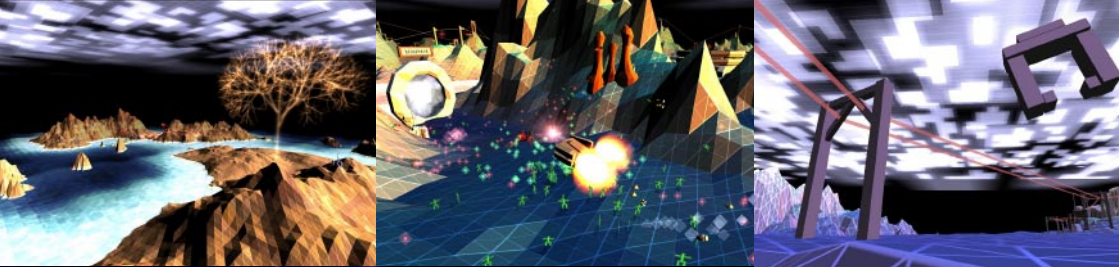
How do you see the indie industry at the moment?

Generally speaking, the biggest problem about being a small

indie developer at the present moment is that you simply don't have the time and resources to compete with other much larger developers in the making of certain types of games. Having such a small dev team means that we struggle to produce content, and a number of reviews said that Darwinia was too short, but it had taken us 3 years to produce a game with 10 levels! Content production is the major time drain on most games and we'd hoped to generate our levels automatically, but we ended up building each level by hand which took a very long time.

At Introversion, we have to make our small dev team an asset, which is why we concentrate on producing the kinds of games that are better suited [to us] - that is, games that are simpler and purer but more able to experiment freely with ideas and-





innovations. The second major issue for indies at the moment, or at least certainly when we started 5 years ago, is that publishers aren't really interested - that's the bottom line and it can be a real struggle to get yourselves noticed and taken seriously. When Darwinia was released, we were big enough to self-publish in the UK, but the US market is around ten times larger and we just didn't have the staff. It took a success story like the Darwinia launch on Steam for publishers to sit up and take notice and we were very happy to team up with Cinemaware Marquee to launch Darwinia in US retail in early July this year.

Where do you see it going in the future?

Here at Introversion we think that on-line distribution may well be the saviour of computer games. Modern game development costs millions of dollars that developers usually have to seek from a publisher. In doing so, the publisher takes a large financial risk and as such the royalty payments made back to the developers are not favourable. Once retail and the distributor have also taken their cut a game needs to be enormously successful before the developer actually sees some profit. This is destructive as it means that developers tend to shy away from making innovative and creative games and stick to what they are told to do by the publishers. On-line distribution may provide a route by which a developer can directly profit from each individual sale and cut out the middleman - thus requiring fewer sales to make a profit, thus enabling them to be more innovative and take more risks.

Certainly our deal with Valve on Steam last November has made our future prospects a lot brighter and allowed us to

continue making the kinds of off-the-wall games that we like to create. We first approached Valve when we originally launched Darwinia but it wasn't until we launched our second Darwinia demo that they got in touch with us. As far as game sales go - Steam was in a completely different league, giving us access to a much wider range of gamers. We sold more copies of Darwinia on Steam in three weeks than we'd managed to sell ourselves via UK retail and the Introversion store during the whole launch period!

What is your opinion on applications like Gamemaker, ie Rapid Game Development kits?

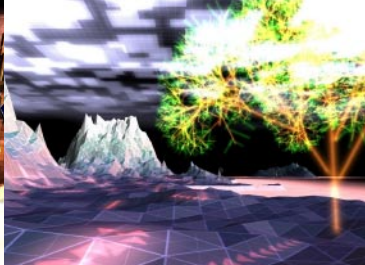
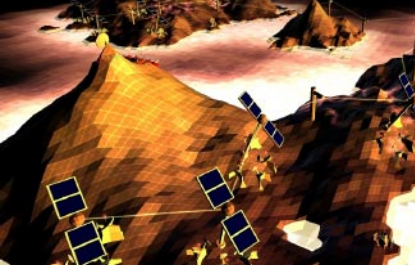
It's not something we would use, as we want the true freedom to create new ideas. The problem with game development apps, is that what comes out tends to be broadly similar to other games produced with the same app, even if the underlying idea is different. We've not had any experience of using these systems - we just develop our own, as it's easier and quicker for us in the long-run.

Any hints on your 4th game?

I'm afraid we can't divulge anything about this as present but rest assured, as with all our games we guarantee to make it exciting, unique and wholly different to anything else out there! Check out www.introversion.co.uk for all the upcoming news on Introversion projects.

How have your lives changed since winning so many awards at IGF?

Winning awards at IGF completely changed things for us - finally people are sitting up and taking notice of the games we



are trying to produce and that is both incredibly gratifying and rewarding for us - we have worked so hard to earn some merit in this industry and it can be a very tough path. In addition to this we have been able to gain much credible coverage and exposure, something that is really important in the run up to DEFCON.

We entered the IGF awards initially because we were encouraged by the fact that it was a games award ceremony for the independents which meant that we weren't trying to compete with hugely successful mainstream titles by big publishers; previous winners has included games like Terminus and Oasis so we thought we stood a pretty good chance becoming a finalist in one of the categories. We decided that if we could win even a small prize it would add a lot to our credibility in the industry and improve our profile. We're also in favour of any event that gives indie developers a voice - it can be difficult to make yourself heard over the noise of the major game developers.

What issues have you had (if any) with selling your games online? What advice do you have for indie developers who want to sell their games online?

Online price points tend to be driven cheaper than retail price points by the platform holders - e.g. \$19.99 for Darwinia on Steam and \$29.99 for Darwinia in retail. Retailers aren't too happy with that, so you need to work hard to keep retail happy. Whilst digital distribution is great for us, you can't ignore retail as a large number of games are still bought through that distribution network. We'd recommend that you find someone

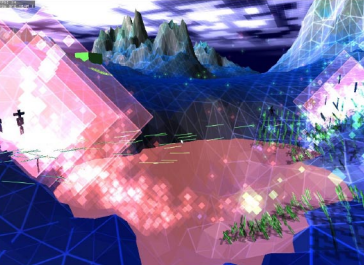
you can trust and then stick with them. What you don't want to do is to license the game to everyone, as you'll start a price war, and you'll just erode your earnings. Also any deal you get should be in advance of 50% return back to you...

Do you actively prototype ideas and see what is fun or is it a natural evolution?

Ideas just occur randomly really, although we do get a lot of inspiration from movies. We allow those ideas to mill around in our heads, letting them develop for a while before writing anything down. Months can often pass before we do actually get round to writing anything down, primarily because if you do this too quickly the idea can stabilize without having given it a chance to expand. With Uplink and Darwinia, the design process was on-going; we didn't have a big design plan from the start. (continued...)

"Setting up your own company is extremely difficult and your driving impulse must be primarily for the love of gaming and creating games"





How do you process input from Beta testing?

We bug test and play test the games. The first is obvious - the latter involves sitting some players down and watching them play the game. Usually at the beta testing stage the game play is locked down, and we're more concerned with the user interface, and making the game easy to get into.

We used that process to make a much better Darwintia demo second time around. The first Darwintia demo we put out didn't do the game much justice.

If you were to hire a new team member, what you look for in that person apart from the obvious (ie, being good at whatever he is doing)?

Talent aside, reliability is something we depend on when we get people in to do work for us. The flexibility of working at Introversion is viewed as a definite bonus, we don't set rigid work hours or inflexible project plans but at the same time we have to trust that people will do the work that we're paying them for and that they will get the work in on time. It sounds simplistic but trust is absolutely imperative to us because we're such a small tight-knit group and teamwork is essential.

Do you guys think there are lots of undiscovered indie talent, or would you say cream floats to the top regardless?

We'd say there probably is a lot of undiscovered or at least un-nurtured talent out there, but this is hopefully going to become

more and more a thing of the past. Traditionally the indies' main afflictions have been the lack of financial backing and the limited opportunities available to getting your games noticed. With the advent of digital distribution channels such as Steam, Xbox Live Arcade and the increasingly avid support of journalists and online bloggers the possibility for smaller developers to get their work out there is much greater.

Thanks Introversion for your time! This is General Tr00jg signing off... This article will explode in exactly 5 seconds...

TROOJG

Check out www.introversion.co.uk for all the upcoming news on Introversion projects.



SPOT LIGHT



Gamasutra

It's a sorry gamer who hasn't heard of Gamasutra. This website is one of the best resources out there concerning the gaming industry, containing news directed at industry professionals, casual gamers, market analysts and, of course, the aspiring game developer.

Gamasutra is run by the CMP game group, an establishment responsible for many gaming publications including Gamasutra and its related sites. Gamasutra itself is a professional and widely-recognized site which has already been in existence for several years, backed by a lot of experienced writers and professional IT journalists.

News is provided on a daily basis, and features concerning games, gaming habits and game development often crop up to interest and inform readers. Want some very real tips on what to do when designing games? Take a look at Gamasutra's excellent and ever-growing repository of game postmortems, where indie and AAA titles alike are dissected, examined and critiqued by the developers themselves, telling readers where they felt the game went wrong,

where it went right and how it went overall with regards to their initial expectations. The site also serves as a portal for far more resources. Content includes job directories, product guides, a place to upload your own resume, newsletters and an all-valuable RSS feed for those who need to keep in touch with current events. One of the biggest things about Gamasutra, how-



ever, is its extended family of websites and resources. Gamasutra has affiliations with places like the Game Career Guide, Serious Gaming Source, Game Developer Magazine, GDC Radio and many more establishments. Recently, podcasting has also reared its head with Gamasutra, consisting of recorded chats with game development gods and



other really, really smart people that you can listen to at your own convenience. Overall, you'll be very hard-pressed to find a more comprehensive source online for game development needs - and even if you do, keep this one in a special place anyway. Bookmark it, RSS it, send it to your grandmother and do anything else to make sure this doesn't disappear from your sight for too long.

Gamasutra is the holy scripture of gaming - a keeper for anyone who wants to take their game development seriously.

NANDREW

Last known update: Continuous
<http://www.gamasutra.com/>



Legend of Shadow

Looking for a fast, frantic and engaging game with beautiful graphics? Well then, let me introduce
Legend Of Shadow

The game starts off very quickly within the first few seconds a princess gets kidnapped, you are in control of a ninja and it seems it's up to you to rescue her.

The world has a scrolling view with an oriental theme and pumping music in the background. There are three actions you can perform: jump, throw a shuriken (ninja star) or swing your sword. The height you can jump is great and you usually find yourself able to gracefully jump from tree to tree. Shurikens will be your main source of attack and there are many different powerups which will give you new shurikens that have different effects. These range from one that will fire a barrage of three shurikens at once, to another type of shuriken which, after being shot, will fly back and forth until it takes out an enemy ninja.

All the usual features one expects from a polished game are there, such as configuring the controls or chang-

ing the resolution. Just about anything you want to tweak is available. Levels are structured in a fashion where you start off at one end of an area and need to make it to the other end all while different types of enemy ninjas will attack you.

Every now and again a special "mini-boss" appears which shoots deadly fire at you. Once you make it to the end of an area you have to fight a boss which will require a bit of puzzle solving to find their weak point. A nice touch is how each level has its own theme of scenery and different obstacles which provide variety.

The only way to describe the graphics in this game is: purely awesome! They are some of the best I've ever seen in a 2-D game. The animations are top class, from the motion of your character jumping, to

lightning going off in the background, to the way bad guys fly through the air after you have killed them - it's all breathtaking. The sound effects are great, realistic and really contribute to the immersion of being a ninja. Nothing bad can be said about the background music, it's fast and energetic which suits the gameplay.



REVIEW

One of the downsides in the beginning of the game is that it drops you in the deep end and enemy ninjas come at you from all directions while you still don't have any idea about the controls or where to go. You will definitely be seeing the "continue" screen quite a few times while you are getting the hang of things.

The whole downside of not knowing where to go initially can be pretty frustrating and to be honest I spent ages fighting ninjas in the same area waiting for something to happen until I eventually figured out that I was sup-

posed to be travelling to the left.

At least in a later level an up arrow gives you much-needed guidance. You start off with only three bars of health but thankfully there are powerups to boost your maximum health which are sneakily hidden along the way to make your quest easier.

There are a few bugs in the game, such as occasions when the frame rate slows down to an unplayable 1 fps and can force the user to restart the game.

Besides the initial steep learning curve

and minor glitches this game is a gem. It contains a well-rounded package of amazing graphics, sound and absorbing gameplay. I positively recommend that everyone give this game a try.

INSOMNIAC

Legend of Shadow was created by Darthlupi in 2004 and can be downloaded from:

<http://www.gamemakergames.com/?a=view.download&id=1382>



QUALITY TOUCH

PART 3:

ADDING THE QUALITY



Previously, this series of articles has touched on the various aspects a game must have to be considered complete, as well as the options that should be included in a game to improve the user experience. This article

Nearly all of the items listed below could be excluded from a game and still allow the game to be considered complete. A little bit of extra work spent implementing these suggestions will make the game seem so much better than other similar games. Some of these items need a small amount of effort to add into the game while others will need complete rewrites to sometimes large sections of the code. Each item will make a small improvement in the game, in some cases a single change will not even be noticed by the player, but taken as a whole these small changes will improve the overall impression the player has of the game.

Most people that download a game will run that game at least once. If they find that they do not know how to play the game in that first try they are very likely to uninstall the game. These players will often not first look at the help page or tutorial when they run the game. By **displaying the help screen automatically** when the player starts their first game the chances are that the player will then read the help and will therefore be more likely to understand how the game works immediately.

A large number of uninstalls are done due to player frustrations with how things happen within the game. Taking time to decrease these frustrations will result in more player

acceptance of the game. A key method to decrease frustration is to allow the player to **start playing at any level** they have already had access to, this is especially true when the levels do not have a direct impact on the game story line. When the game ends an option to **restart the current level** with a zero score and a full set of lives makes it possible for the player to experience an uninterrupted playing experience.

Displaying progress bars while loading levels makes it clear that the game is busy doing something and therefore keeps the player's attention on the game. Without a progress bar, the player may think the game is not doing anything while he/she is sitting waiting and may therefore terminate the program. Another way to keep the player's attention during **screen or level transitions** is to use some sort of effect between the screens such as fading one screen into the other.

The more **visual and audio feedback** that can be given for player actions the better as it makes it clear to the player that their actions are being responded to by the game. Perfect examples of this



Remember: your help screens actually have to be helpful ...

DESIGN

are the various effects that can be implemented within the in game menu. As the player moves the mouse over a menu option the image could fade slightly or even change colour to indicate that it is the currently selected option. In addition, a chime could be used to indicate that the mouse has moved over a menu item. Similar feedback could be used when the player actually clicks the menu option. This is an easy item to over-implement, a nice balance must be found between giving the player feedback and not irritating the player with massive animations each time they move the mouse.

A player expects the game to remember their current options between games. By **saving the various options** settings in a configuration file and reloading them when the game starts again the user will see the game behaving the same way during each session. This can then be extended to support multiple user profiles within the game. This way multiple users of the same home computer will be able to



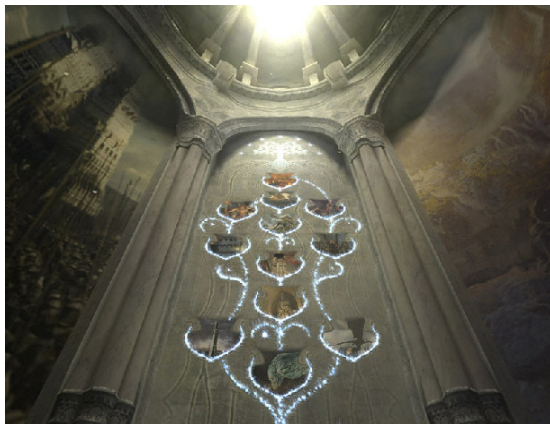
Lots of things have loading bars. Why not

set up the game in their own preferred way without affecting each other players' experience of the game. Many players would prefer for these options to not be stored in the registry, but rather in a game-specific configuration file. Games are often deleted from the windows explorer leaving any registry settings behind. By storing them in an ini file or other text configuration file, the file will be deleted along with the game directory.

Only a few suggestions have been made on ways to polish a game. Each of these on their own will make almost no significant difference to

the game itself. However, taken as a whole these small changes will make the game feel significantly more polished and will therefore improve the player's experience of the game. When you are trying to sell a game, the initial impression a player has of the game will have a direct impact on the sales figures of that game.

CAIRNSWM, FENGOL



In some game titles, even level selection



Blender Tutorial - Materials and Textures

For this tutorial, we will be continuing from the scene we created previously. If you would prefer to use a pre-created scene rather than the one you made in the last tutorial, the scene is available for download from the Dev.Mag website under the content section. (devmag.googlepages.com)

We'll be improving this scene by creating materials to give our objects some variety and add colour to our scene. This tutorial is far more technical than the ones that have come before, so if you're not yet very familiar with the controls and techniques we used it's best that you go back through the previous instalments again, or at least have them on hand.

First off – Material basics

Load up the scene you created earlier. You should have a sphere object, two light sources, and three planes that make up the walls and floor. First off, we'll be editing the sphere object, so select it with the right mouse button. Make certain that you are in Object mode. In the buttons window, select the shading tab. You'll remember that we used this tab to modify the properties of our light sources in the previous menu. However, when you have an object selected it will present you with different options with regards to object materials.

At the moment, our sphere has no material linked to it so the materials

panel is empty. Click the Add New button to create a new material for this object. The materials panels will fill with new options and two new panels will appear. For now, concentrate on the materials panel which is used to modify basic options with regards to your material.

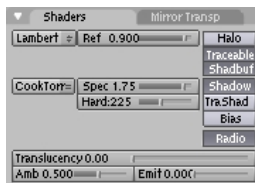


The 'material' panel. Change basic material options like colour and name.

The topmost box in this panel shows the name of your new material. You can leave it at the default value if you wish, but it is often a good idea to name your objects and materials. Below you'll see three rectangles, labelled 'Col', 'Spe', and 'Mir'. Clicking on the coloured rectangles with your mouse will allow you to change the colours using a colour chart or by typing in the red, green and blue values. Select a blue colour for the base 'Col' field and a slightly darker blue for the specular colour (The values I used were $R=0.25$, $G=0.25$, $B=1$ for the base colour, and $R=0$, $G=0$, $B=0.95$ for the specular colour). We won't use the mirror colour just yet, so you can ignore that for now.

Now we'll look at the shaders panel. Make sure the 'Shaders' tab above the

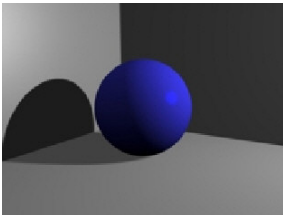
panel is selected. The drop-down boxes in this panel represent the shading and specular methods that will be used in rendering. The default values should suffice for most uses, so we'll leave them as is. The first slider marked 'Ref' represents how much light the material will reflect. Higher values make the surface brighter. Set this to 0.9. The second two sliders, 'Spec' and 'Hard', affect the shine that light's have on the material. The 'Spec' value determines how large the specular area is, and the 'Hard' value determines the smoothness of the shine. Set them to 1.75 and 225 respectively.



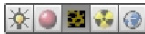
The 'shaders' panel. Change shading methods and modify specular and reflectivity.

If you render the scene now, you'll notice two specular regions on the sphere. This is because we have two light sources in our scene. However, this looks a bit strange because, if you remember, we only created the 'Hemi' lamp to add ambient light to the scene, so the two specular regions defy the feel that there is only one light source in the scene. To fix this, select the Hemi lamp, and under the Lamp panel,

click the No Specular button to disable the light from creating specular regions on the object. Rendering the scene now should present something like this:

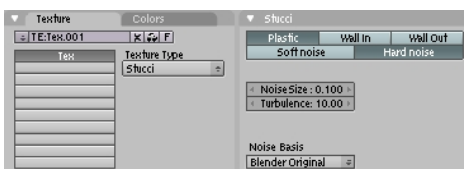


Now we'll give our floor and walls some texture. With the floor selected in Object mode, create a new material. We're going to create a cement floor look for this object, so the default grey colour should be fine. To give this object a bump map we'll need to create a texture. You can use any image file for a texture, but for the sake of this tutorial we can use Blender's procedurally generated textures. While in the Shading Tab, select the Texture Buttons or press F6 to bring up texture panels.



Click Add New to bind a new texture to the object. Select Stucci from the Texture Type drop-down box. A Stucci options panel will appear where you can fine tune your texture. Select Hard Noise from this panel, and change the Noise Size and Turbulence values to 0.1 and 10 respectively (see below).

Now we need to set the texture to be mapped as a normal map (i.e. bump map). To do this, go back to the Materials Buttons by clicking the red sphere icon next to the Texture Buttons icon, or press F5. The material texture panel on the right is the one we'll be using to change how the texture will



The 'new texture' panel. Create new textures and modifies their

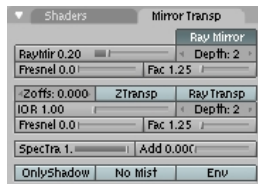
affect the material. Make sure your new texture is selected in the list, and then select the Map To tab. Deselect the 'Col' button and active the 'Nor' button to change the texture to map onto normals rather than affect the material's colour. Set the Nor slider to 0.3 to change the extent of the texture's affect. Then select the Map Input tab and change the SizeX and SizeY values to 10 each to make the texture tile (repeat) 10 times in the X and Y directions.

Now that we're done with the floor, we'll create another material for the walls.



The 'map to' and 'map input' panels. Change how textures are mapped onto materials.

Select one of the walls, create a new material, and set the colour values for base colour, specular colour and reflective colour to $R=0.8$, $G=0.6$, $B=0$; $R=0.75$, $G=0.4$, $B=0$ and $R=0.35$, $G=0.2$, $B=0$ respectively. In the shaders tab, set reflectivity to 1, and specularity to 0. Now we'll use the Mirror/Transparency tab to make the walls slightly reflective. Click the Ray Mirror button to enable mirror effect with raytracing. Set the RayMir slider to 0.2 to make the surface 20% reflective.

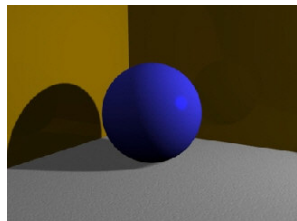


The Mirror/Transparency tab. Use to allow materials to be transparent or reflective, and change related values.

We'll be using the same material for both the side and rear walls, so it's best to name it. In the materials panel, the top drop down box should read something like *MA: Material.003*. Change this to any name you wish. Then select the other wall in the 3D view, and in the material panel use the drop down box to select this new material.

If you render the scene now, using the reflections you can just make out that there are no walls behind the camera. Simply duplicate the existing walls with SHIFT+D and drag and rotate them into place. Top view (7) will be best for this. You needn't worry about the roof, since it shouldn't be visible in any reflection. Your completed scene for this tutorial should look something like the following image and will be available for download from the Dev.Mag website's content section.

Now that the tedium of learning all the basics is out of the way, we can create



some more advanced things, so in the next issue we'll be having some fun. Using all the skills you've learnt so far we'll create a new scene from scratch. Until then, happy Blending.

CH1PPIT

Project Management: Part 3

This month, we will examine the content creation phase for your game, as well as the maintenance phase. Both of these are relatively straightforward aspects, and require little explanation, although there are certain elements of both that you should pay attention to.

Step 4 – Execute

It is critical that, before you begin this stage, your design document is as thorough and as detailed as possible. You should be using it extensively until the end of the development of your game, and should continue to use it as reference once your game is complete.

During this stage, you will be creating the content for your game. You will need to keep your design document as reference during this entire stage, as any drastic deviations from it will only extend the development cycle, and could create unforeseen problems.

As you create content, you will most likely find that certain aspects you have designed may need to change. Prototyping and testing are bound to reveal elements that may need tweaking, rethinking, or completely removing or adding. If you do choose to change aspects of your game at this stage, it is vital that you keep your design document updated – a log of version histories and changes will help keep things organized.

Once complete, your game will be ready for release. It is advisable to first release your game to a small, select group of people, as they generally could aid in testing and evaluation with more

constructive criticisms than a larger, typically anonymous, group would.

An essential element of any game is the manual. This could be online, within the game itself, a physical book or even a simple readme. Regardless of which type of manual you decide to use, you will find the best source of information for it in your design document. If your design document has been kept up to date and accurate, it should be able to detail all the aspects of your game you wish your player to know about.

Step 5 - Maintain

It is important to remember that although your game is finished, its development cycle is far from over. People, and especially seasoned gamers, expect the games they play to be as bug-free as possible. There are also expectations for certain types of games to be updated on

a regular basis. Most MMORPGs are updated monthly, weekly or even daily to keep their players happy and the game's intrigue fresh. You should apply these same elements to your game.

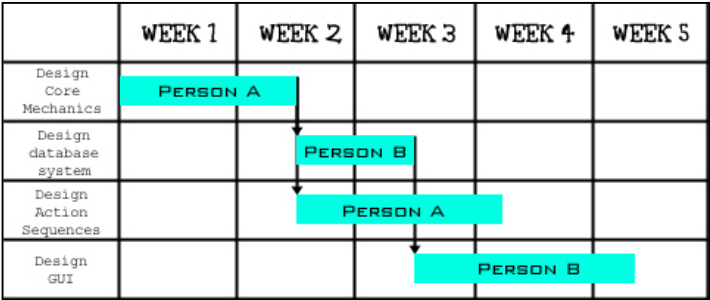
During this stage, you should be approaching each update with the same principles as your entire game. Each update should be initiated, planned, designed and executed. Larger updates (especially expansions) should even have their own maintenance stage.

Once again, you should be updating your design document as you update your game. By keeping a constantly up-to-date and well-organized document, you have an easily accessible summary of your game, without having to sift through mountains of code and images.



Most professional games are armed with a comprehensive design document by the time they hit their execution step. You should get into this habit for your products, too.

THE GANTT CHART



Tools of the Trade

The Gantt chart

Developed by Henry Gantt in 1910, the Gantt chart is a simple method of determining a realistic timeline for developing your game. Typically, it comprises of a timeline along the X-axis, and the procedures listed along the Y-axis. Each procedure is then given it's own timeline. Using a Gantt chart will help alleviate the difficulties in setting deadlines, and will also give you an estimated starting point for each activity.

See above for an example Gantt chart of a specific portion of the design phase.

As you can see, the database design can only take place once the core mechanics have been designed, as is also the case with the action sequences. Designing the GUI can only take place once the database design has completed, yet is not affected by when the action sequences have been designed. Also notice the names on the chart; this helps you to allocate resources and people to specific tasks.

Using this tool will allow you to lay out your goals in an easy-to-read format, and will give you some perspective as to how and when each goal will be achieved.

The Work Breakdown Structure

This diagram, although simple, will help to streamline the process of developing your game. It allows each task to be assigned a requirement (or multiple requirements). Using this diagram, you can plan the development of your game by breaking it down into sections, and breaking those sections down into further sub-sections, if required.

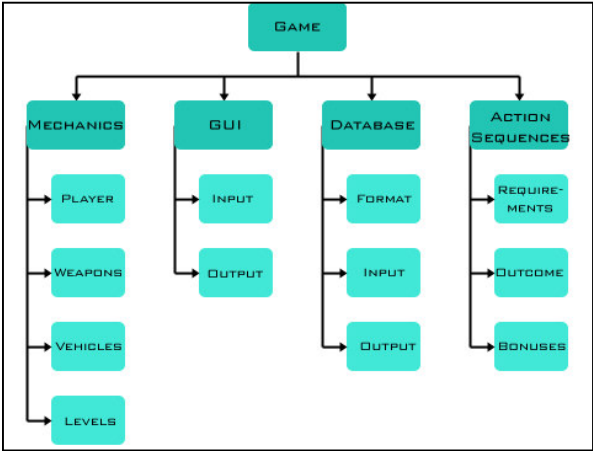
The use of this diagram creates an outline for the development of your game, and will be especially useful when creating a vast or complex game. It allows you to stay in control of development, by keeping all the main

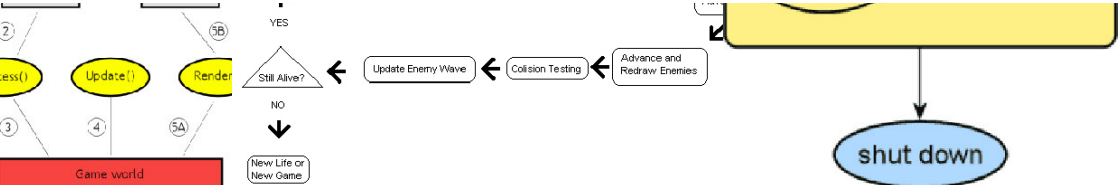
aspects of your game in a single, organized location.

With that comes an end to this brief discussion on Project Management. While there are many more aspects of PM that warrant attention, I trust I have given you at least an insight into the principles behind it. I hope you will use at least a few of the elements I have mentioned to help streamline the development of your games. In the next issue I will be reviewing a new PM application developed specifically for the development of video games. Until then – happy developing!

GEOMETRIX

THE WORK BREAKDOWN STRUCTURE





FRAMEWORKS - THE GAME LOOP

A game loop is a section of game code (or of code within a framework) that gets executed over and over during the running time of the game. Graphical based games have since the beginning used a game loop to control the inner processes of the game. Game loops can be implemented in many different ways, and with the rise of modern windows compilers these methods have become very, very easy to use.

A game loop is actually difficult to define as it is typically a very small snippet of code, but it plays a very important role in the program. Just like the human brain cannot function without a blood supply, a game will not deliver functionality without a Game Loop. Just as we consider the blood supply a minor function of the human body the importance of a game loop is often underestimated. At its core, the game loop controls when and in which order the various main functions of the game get called. As the game loop gets executed in every iteration through the code it can contain the functions that need to be called regularly in the life of a game.

The core functions of the game such as AI engine, Input management, Sprite movement and collision detection and the actual graphical display are called from within the game loop. Each pass through the game executes each of these functions from within the game loop – however, each function may not need to be called on each iteration. Games typically need to run at 30 frames per second (FPS) and therefore the Graphical Drawing function may only need to be called on every third or fourth iteration.

There are many different ways in which a game loop can be implemented. The easiest and most traditional way of implementing a game loop is to implement a loop that only exits when the game state allows the game to end. Within the loop the various game functions get called one after another. When the program execution point reaches the loop it will continuously repeat the series of commands one after the other. Each of these functions contains the game logic based on the various state requirements. This

implementation makes maximum use of the computer's processor and will typically use 100% of the available processing time.

Games can create an internal timer that begins execution when the game starts and processes the game loop whenever the timer executes. This method is very good for creating games that have a set FPS that remains constant. However, this method can cause very unexpected results when each frame cannot be completed before the next frame needs to begin, such as on low spec machines. Due to the amount of time the application may sit idle between timer executions, this method has very small impact on the processor of the computer.

Windows games are often created using a windows GDI form. The windows operating system includes a concept of idle time. The game loop can be implemented to execute whenever the operating system encounters a period of idle time. This implementation will give more time to other processes executing within the operating system and will therefore not use 100% of the processing power of the computer.

With the advent of more and more powerful computers, programs are moving toward a thread-based development design. The beauty of threads is that they can run independent of the main game loop, effectively outside the main game engine itself. While the game loop is executing the various input and display functions the modules in the threads such as AI can run uninterrupted. The game loop will then take input from the threads during each iteration and react on them as if they were external inputs.

A basic game loop is easy to implement, but as a game gets more and more complex the design of the game loop will also become more complex. Certain decisions need to be made on frequency with which functions get called, which functions get called based on the current game state and the amount of time that the game needs to sleep between display frames to allow other applications to run, while still displaying enough FPS to appear fluid.

State management in the game can often have a large impact on the chosen method of implementing the game loop. State management systems that implement unique objects for each state will manage the game loop very differently from state management systems that are built into the implementation logic of the game itself. State management is however a topic in itself and will be discussed in a future article.

When game developers move on to creating their own game engines within a traditional programming environment, the first part of the game that needs to be developed is the game loop, as it defines the implementation methods for the rest of the game engine. As the game loop is defined and extended the various functions needed in the game can be added. Typically the graphical drawing functionality is created first and then extended as needed within the game loop. In this article, a brief discussion of the various implementation methods has been made, but each of these methods must be extended for the game's needs.

CAIRNSWM



MAKING 2D ASSETS WITH 3D SOFTWARE

PART 3

This month, we will be using our sprite we created last month, add shadows to it, use it in the Game Maker environment and even learn how to create handy animated sprites. Open Game Maker, create a new object and add a new sprite. Load the sprite we created last month as the new sprite, and centre the sprite's origin. Now assign the sprite to the object, and let the games begin!

To make our object moveable in all directions, we will add some simple functions to it. In its create event, add a friction value to your liking. Now, for your turn left and right keyboard events, add a "set a value of a variable" action with the following values:

Variable : *direction*
 Value : *-3 (Positive for left, and negative for right)*
 Relative.

For your forward keyboard event add a "set the direction and speed of motion" action. Set the values to such:

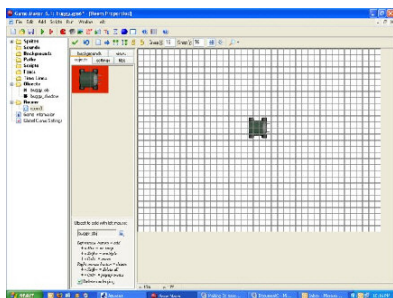
Direction : *direction*
 Speed : *5 (Play around with this value)*

Now duplicate the event for your reverse keyboard event and change the speed value to a negative number. In the step event add the following code:

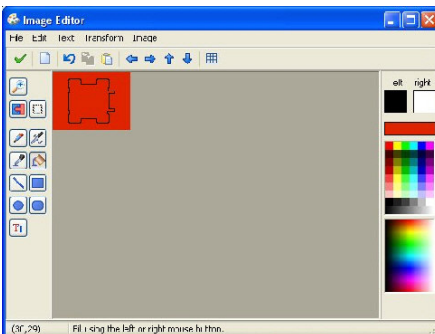
```
{
    image_angle = direction;
}
```

This will make the image rotate as you turn left or right. The orientation of your sprite is also important. My sprite orientation was out by 90° and I had to rotate it in game maker before it faced the correct direction.

Now that we have motion, we should create a room and place our object in it. Test it out.



Now to add the shadow. Duplicature the object sprite, and rename it to "objectname_shadow_spr". Open the sprite in game maker and go to the outline option. Make sure the selected color is black, and the click outline. It will ask you if you would like to delete the contents of the sprite, you answer yes.



You should have something similar to the picture above. Now flood fill the inside area of your sprite with black paint.

Create an object with the name "objectname_shadow_obj". Assign the new shadow sprite to this object. In its step event, add the following code.

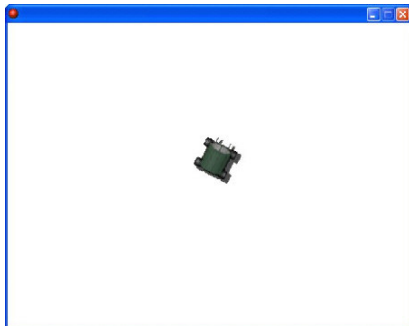
```
{
    x = (object.x + 10);
    // where above 'object' is your object name
    y = (object.y + 10);
    direction = object.direction;
    image_angle = direction;
}
```

The "+10" is to give a relative offset to your object so that it looks more like a shadow. Also in the create event add the following code:

```
{
    image_alpha = 0.7;
}
```

DESIGN

This will make the black sprite blend and look like a shadow. Now add the shadow object anywhere in the room, and test it. You should have something like this:

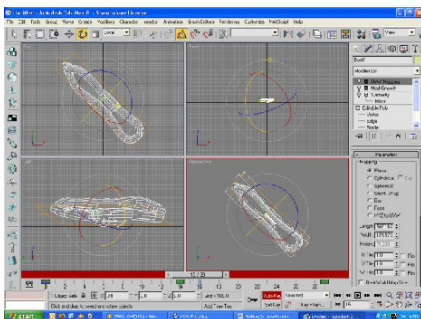


I'm sure this will have given you all lots of ideas on how you could make shadows for almost any object in your game. Now on to the next bit : making an animated sprite from scratch in Max and getting it into Game Maker.

ANIMATED SPRITES

For my game Mech's Destiny, I require a cursor for my menu. This is what I will be making in this tutorial. First, we have to make the model as always. I whipped up the model I wanted to use and applied my materials. This time around, you should not use a plane as a reference surface, as it would cause too much post work on the shadowing. Instead, a handier idea is to change the environment to a color not used in your model's texture map. I used a bright green, and made my cursor a metal/chrome color.

Next step is to animate your model in your 3D package, and test the animation until you are happy. This is how my scene ended up:

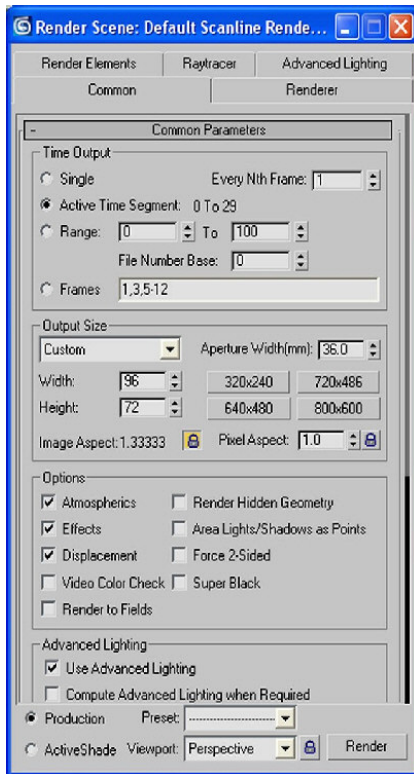


Now to set up Max to render the scene into loads of small images – this will be used to make our animation in Game

maker. My rendering setup looked like the image below. Note that the setup is set to active segment. My rendering resolution is also set to a higher number than I need, so that I can always reduce the quality if needed (you can never increase it if looks bad, after all!).

I then set my render output settings to save a file called cursor.jpg in a new empty folder. When I click "render", it actually creates 30 cursor.jpg files in that folder, named cursor1.jpg – cursor30.jpg.

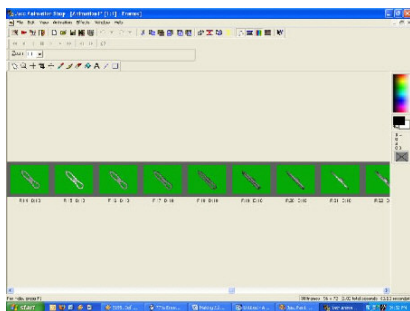
The rest of our work will be done in Paint Shop Pro's animation shop. I open up the Create Animation Wizard. The



first question we are asked is if we'd like to resize the images, or use the first image size. In most cases, using the same size as the first image works the best. The second option asks us what we want the default canvas color for the animation to be. I normally use the opaque option. The next option screen is applicable when you have differently-sized images and concerns itself with how or if they should be scaled to fit the animation size. We are not interested in this, as all our images are exactly the same size. The next option screen gives us a

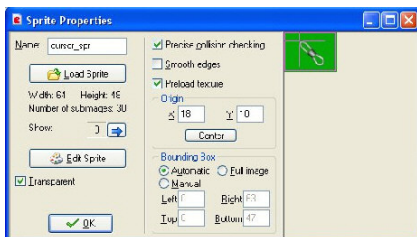
chance to set the animation speed of the animation. This is, in fact, not that important, as you can set it in Game Maker as well – but the “repeat the animation infinitely” option should be ticked.

In the next option window, we have to go and add the files we would like to include in the animation. Please take note that the order is determined by how you select the images in the window. You could always rearrange them afterwards, but try and avoid any extra work if you can. Click on finish and you should have something like this:



The animation shop has some great options you are welcome to play with, but for this, all I am going to do is resize the animation to 64x48 pixels, and resave it as a .gif file. The options I choose for the quality of the .gif file always set it to the highest quality. At the end you should have a small (64kb or similar) animated .gif file of your model.

Now let's get it into Game Maker and see how it looks. First, I create a sprite and load our .gif file. Make sure to set your x, y position to correspond to your cursor's point. This can be seen in the image below.



Now create an object and set the sprite to your newly created mouse cursor sprite. This is where I notice that my green opaque color isn't working correctly. The green color isn't uniformly the same. This causes game maker to not see it as transparent, which is a big problem. I have noticed that the best way to avoid these problems is to use “pure” colors. What

I mean by that is to use pure blue, red, green, black or white as your environment color in Max.

I quickly fix this problem by opening my saved Max file and just changing my environment color to pure white. I then re-render the images and remake my .gif file using the animator software. My result looks much better.

Now, in the step event of your cursor object add the “jump to the following position” action, and add the values for x and y as seen in the image below. Alternately, you could add a piece of code that looks like the code below into your step event.

```
{
    X =
    mouse_x;
    Y =
    mouse_y;
}
```



Lastly, change your global game setting, and deselect the “display the cursor” option. Add your object to a room, and see if it works.

Hope to see you all at rAge this year, and if you see me there, you are more than welcome to come and look at some of my work, the assets I have made and the methods I use.

HIMMLER

MOBILE GAME DEVELOPMENT IN JAVA



PART 5 : DIE EVIL BLOCKS!

In this edition of the tutorial, we'll jump right in, finish off the core gameplay and add a few little touches to our game.

Once again you need to exercise your artistic skills, this time to create an image to represent the blocks we'll be destroying. Make it a 25x10 pixel image and save it in your res directory as bar.png. As we need to use a number of blocks, we will put them into a new static variable blocks, which is a Vector.

Vector is a commonly used Java class that can store any number of objects (of different types if need be) that can then be accessed directly or by requesting an iterator with the elements method. It is sufficient for our simple needs, however it can be quite slow and in more complex situations you would be better off building a custom link list or similar structure.

It is part of the java.util package, so we add that to our imports at the top of the file. In the TutorialCanvas constructor, we load our new block image into a temporary Image object and use that to create enough sprites for four rows of blocks on the screen.

Also in the constructor, ballSprite's starting position is moved to the middle of the screen so it doesn't

start between all the blocks. Note that nextElement returns an Object, so we need to cast it before using it.

In our run method, we now need to check for collisions between the ball and blocks. Once again we iterate over all the elements in the vector using Sprite's handy collidesWith method to check for bounding box collisions. Once we have established that the ball has hit a block, we can do line intersections with the top and bottom of the block to determine whether we should bounce along the X or Y axis.

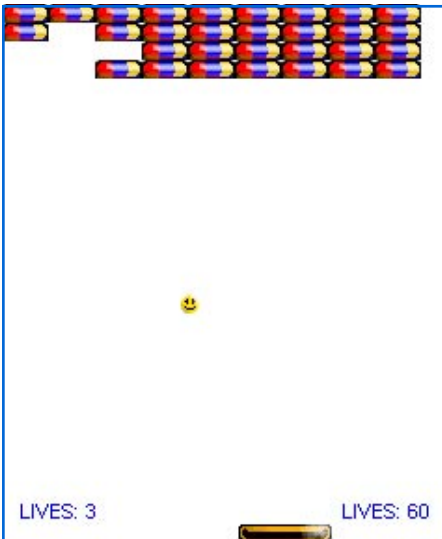
Unfortunately Java does not include any collision routines, so we have to implement one of our own. This has been added as a separate method, intersect, to keep things from getting too cluttered.

We also remove the block we collided with from blocks so it will no longer be rendered or checked against. We'll check the size of blocks to determine if the player has won the game and if we should paint a "well done" message.

Note that the lines adding the velocity to the speed has also been moved until after the collision tests. These changes are all in Listing 1 and when you compile and run them, you'll see we now have a (very simple) block breaker game!

Now we'll add simple sound, a score counter, and rearrange the interface a little to suit the now-busy screen. 'Proper' sound is one of the trickier things to implement successfully in mobile games and deserves an article of its own, but for our simple requirements we will stick to the basics and just use the phone's default alert sounds. The first thing we need is a way to access the midlet from the canvas, so we add a new static midlet variable to our canvas as well as a midlet parameter to its constructor. We simply store this object when the canvas is constructed. Bear in mind that alert sounds can take some time to play, so we don't want to use them in places that would interfere with gameplay (like every time a block is designed).

We will rather just use them when the player has lost all his lives, or has won the game. To do this, we check after updating our objects if either condition is true and call playSound on one of the AlertType objects available statically through the alert type class. For example, ERROR can be used for the death case, and CONFIRMATION when the player wins.



It's all coming together nicely!

The call to playSound is where our static midlet variable comes in handy,

we use Display.getDisplay(midlet) to get the display parameter we require.

Now we add our score. This is also very easy; we'll add a score static variable to the canvas, and add points to it every time the player destroys a block. Finally, we display the score in a similar fashion to the player lives. At the same time it makes sense to move the lives counter (and score) to a location on the screen that is not full of blocks. All of these steps, as well as the sound changes, are shown in Listing 2.

At this point we have a simple, but quite respectable, game. From this simple beginning it is a relatively small job to add multiple levels to the game that allow for different placement of blocks, or even different types of blocks that take more than one hit or release a power up. By now the game may have also have started showing some signs of slowdown as more sprites are shown on screen, and tweaking the speed values for the ball and bat, or even building in a proper timing loop- might be a good idea.

There are countless possibilities for improvement and optimization, and I will leave it in your, very capable, hands. Things are starting to get a bit messy because it's all squashed up in a single file, though, and it would be nice to work in a more interesting environment; so, next time, we'll move our code into a mobile project in the NetBeans IDE.

As mentioned in the first instalment of this series, NetBeans and its Mobility add-on are available for free from www.netbeans.org.

FLINT

Code Listings

Listing 1. TutorialCanvas after adding blocks.

```

import java.util.*;
...
//Canvas class
class TutorialCanvas extends GameCanvas implements Runnable
{
    ...
    static Vector blocks = new Vector();;

    //Constructor
    public TutorialCanvas()
    {
        ...
        Image blockImage = null;
        try
        {
            ballSprite = new Sprite(Image.createImage("/ball.png"));
            barSprite = new Sprite(Image.createImage("/bar.png"));
            blockImage = Image.createImage("/block.png");
        } catch(Exception e)
        {
            e.printStackTrace();
        }
        ballSprite.setPosition(getWidth()/2, getHeight()/2);
        barSprite.setPosition((getWidth()+barSprite.getWidth())/2,
                               getHeight()-barSprite.getHeight());

        int columns = getWidth()/blockImage.getWidth();
        Sprite blockSprite;
        for(int y = 0; y < 4; y++)
        {
            for(int x = 0; x < columns; x++)
            {
                blockSprite = new Sprite(blockImage);
                blockSprite.setPosition(x * blockImage.getWidth(),
                                         y * blockImage.getHeight());
                blocks.addElement(blockSprite);
            }
        }

        //run the game loop
        public void run()
        {
            while(!exit)
            {
                if(lives > 0 && blocks.size() > 0)
                {
                    //update our bar position
                    ...
                    //update our ball position
                    // REMOVED ballX+=ballVX and ballY+=ballVY
                    ...
                    //Check against blocks
                    Sprite blockSprite;
                    Enumeration blocksEnum = blocks.elements();
                    while(blocksEnum.hasMoreElements())
                    {
                        blockSprite = (Sprite) blocksEnum.nextElement();
                        blockSprite.nextFrame();
                        if(ballSprite.collidesWith(blockSprite, false))
                        {

```

```

        //hit the top or bottom
        int oldX = ballX - ballVX;
        int oldY = ballY - ballVY;
        int blockLeft = blockSprite.getX();
        int blockTop = blockSprite.getY();
        int blockRight = blockLeft +
            blockSprite.getWidth();
        int blockBot = blockTop +
            blockSprite.getHeight();
        if(intersect(oldX, oldY, ballX, ballY,
            blockLeft, blockTop, blockRight, blockTop)
            || intersect(oldX, oldY, ballX, ballY,
            blockLeft, blockBot, blockRight, blockBot))
        {
            ballVY = -ballVY;
        }
        else
        {
            ballVX = -ballVX;
        }
        blocks.removeElement(blockSprite);
        break;
    }
}
ballX += ballVX;
ballY += ballVY;
ballSprite.setPosition(ballX, ballY);
...
}

//paint the canvas
protected void doPaint(Graphics g)
{
    ...
    Enumeration blocksEnum = blocks.elements();
    while(blocksEnum.hasMoreElements())
    {
        ((Sprite) blocksEnum.nextElement()).paint(g);
    }
    ...
    if(lives == 0)
    {
        g.drawString("GAME OVER!", getWidth()/2, getHeight()/2,
            Graphics.HCENTER|Graphics.TOP);
    }
    else if(blocks.size() == 0)
    {
        g.drawString("WELL DONE!", getWidth()/2, getHeight()/2,
            Graphics.HCENTER|Graphics.TOP);
    }
}

//intersection test
boolean intersect(int l1p1x, int l1p1y, int l1p2x, int l1p2y,
    int l2p1x, int l2p1y, int l2p2x, int l2p2y)
{
    int v1x = l1p2x-l1p1x;
    int v1y = l1p2y-l1p1y;
    int v2x = l2p2x-l2p1x;
    int v2y = l2p2y-l2p1y;
    int ux = l1p1x-l2p1x;
    int uy = l1p1y-l2p1y;

```



```

        int d = vx*v2y - vy*v2x;
        if (d != 0)
        {
            int r = (uy*v2x - ux*v2y) / d;
            int s = (uy*v1x - ux*v1y) / d;
            if (r >= 0 && r <= 1 && s >= 0 && s <= 1)
            {
                return true;
            }
        }
        return false;
    }
}

```

Listing 2. Changes to TutorialCanvas and TutorialMidlet to add sounds and scores.

```

public class TutorialMIDlet extends MIDlet
{
    TutorialCanvas canvas = new TutorialCanvas(this);
    ...
}

class TutorialCanvas extends GameCanvas implements Runnable
{
    ...
    static MIDlet midlet;
    static int score = 0;

    //Constructor
    public TutorialCanvas(MIDlet midlet)
    {
        ...
        this.midlet = midlet;
        ...
    }

    //run the game loop
    public void run()
    {
        while(!exit)
        {
            if(lives > 0 && blocks.size() > 0)
            {
                ...
                if(lives == 0)
                {
                    AlertType.ERROR.playSound(
                        Display.getDisplay(midlet));
                }
                else if(blocks.size() == 0)
                {
                    AlertType.CONFIRMATION.playSound(
                        Display.getDisplay(midlet));
                }
            }
            ...
        }
    }

    //paint the canvas
    protected void doPaint(Graphics g)
    {
        ...
        g.drawString("LIVES: "+lives, 10, getHeight() - 10,
            Graphics.BOTTOM|Graphics.LEFT);
        g.drawString("LIVES: "+score, getWidth()-10, getHeight() - 10,

```

```
Graphics.BOTTOM|Graphics.RIGHT);  
    ...  
}  
    ...  
}
```

THE TECH WIZARD



Data Structures - Part 2 Linked Lists

In the previous article, we touched on the basic concepts and principles behind data structures. We talked about variables, which are the building blocks of data structures, arrays, which are used to organise lists of data, and structures, which are a way of encapsulating multiple kinds of data into a custom form that suits your needs.

Now that you have a better understanding about what exactly data structures are, we will move onto a much more useful (although complex) data structure known as a linked list.

Linked Lists

Linked lists are probably the most used data structures after variables, arrays, and structures. They are used in everyday game programming, and most experienced developers consider them essential for easy as well as efficient data storage and processing.

Given the somewhat complex nature of linked lists, it is vital that we call

upon the many topics covered previously in this series of articles, including variables, structures, and most importantly, pointers.

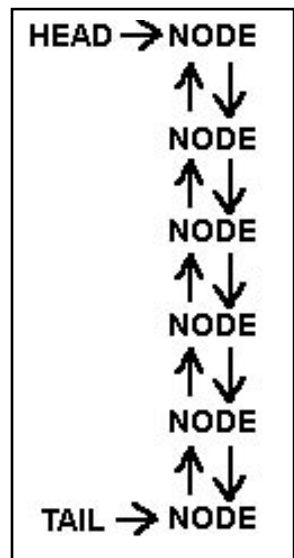
The basic principle behind a linked list is that each piece of data in the list belongs to a node, and each node in the list points to the neighbouring nodes in the list (ie. the nodes directly before and after it in the list).

The list itself is maintained by keeping track of the node that is currently first in the list, as well as the node that is currently last in the list. These two special nodes are normally referred to as the head and tail of the list.

In order to visualize this better, imagine a pearl necklace where each pearl is attached to the next one in a linear fashion. You can think of the pearl beside one of the clasps as the head of the list, and the pearl beside the corresponding clasp as the tail of the list.

Each pearl in the necklace is only directly aware of the pearls on either side of it, or in the case of the head and tail pearls, the fact that nothing exists beside it in one direction.

List 1



Storing Data

As mentioned, data that is to be placed into the list needs to belong to one of these nodes. Nodes are usually defined to contain a pointer to the data that is to be added to it, as well as pointers to the nodes that both proceed as well as follow this node. Linked lists such as these are generally referred to as doubly linked lists.

C-Based pseudo-code of the structures used to define a linked list:

```
//structure defining a single linked list node
struct
{
    ListNode *prev; //pointer to the node before
                    //this one
    ListNode *next; //pointer to the node after
                    //this one
    void *data; //pointer to the data at this
               //node
}ListNode;

//structure defining the linked list
struct
{
    ListNode *head; //pointer to the head node of
                   //the linked list
    ListNode *tail; //pointer to the tail node of
                   //the linked list
}LinkedList;
```

This structuring of linked lists allows for very easy traversal of all data in the list, similar to the way that arrays are processed. Iteration is started at the head node of the list, the data contained there is processed, then the iteration continues to next node and the process is repeated until the tail node of the list is reached.

C-Based pseudo-code of looping through a list:

```
//linked list containing my data
LinkedList myList;

void foo(void)
{
    ListNode *curNode; //node used to "step" through
                       //the list with

    curNode = myList.head;

    while(curNode)
    {
        //call a function to process the data at this
        //node
        ProcessData(curNode->data);

        //move onto the next node in the list
        curNode = curNode->next;
    }
}
```

Why use linked lists?

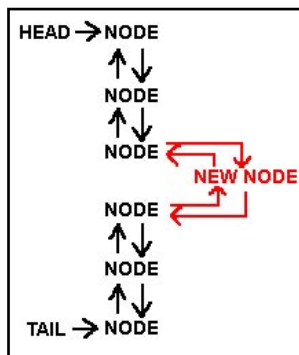
At this point you may be wondering

why you would ever want to use a linked list, as everything that has been mentioned so far about them seems to be handled perfectly by arrays. However, there is more functionality to linked lists than merely the ability to store and process data in a linear fashion. The main advantage that linked lists have over arrays is that arrays need to be statically defined as a fixed size, whereas linked lists can change their size dynamically as needed.

Once again, picture the pearl necklace. Imagine if you could add or remove individual pearls to or from any point on the necklace. If you could, you would see the length of the necklace grow or shrink as you did, which is exactly how adding or removing data to or from linked lists works.

This can be accomplished by merely changing the pointers of the affected nodes, telling them that they should no longer point to the nodes that they are currently pointing to, but to instead point to either the newly inserted node or the node on the other side of the node that has been removed.

List 2



C-Based pseudo-code for inserting or removing a node in a linked list:

```
//function to insert a node at the end of a linked list
void LinkedListInsertNode (LinkedList *list, ListNode *node)
{
    //tell the node that its previous node is the
    //currently set tail of the list
    node->prev = list->tail;

    //tell the node that it doesn't have a next node as it
    //is at the end of the list
    node->next = NULL;

    //tell the currently set tail of the list that its next
    //node is the new node
    list->tail->next = node;

    //set the list's tail to be the new node, as it is now
    //the last node in the list
    list->tail = node;
}

//function to remove a node from the list
void LinkedListRemoveNode (LinkedList *list, ListNode *node)
{
    //tell the node's current previous node that its next
    //node must point to the node's next node
    node->prev->next = node->next;

    //tell the node's current next node that its previous
    //node must point to the node's previous node
    node->next->prev = node->prev;
}
```

Conclusion:

Although they can seem quite daunting, linked lists are essential tools to game programmers. They allow for easy dynamic organization and modification of data, which is a core necessity in development of even moderately complex games.

The key to understanding how linked lists work (along with how they are best used) is to first not only understand how pointers work, but to become comfortable using them.

Once pointers become second nature to you, linked lists will follow suit and become a welcome part of your programming arsenal.

COOLHAND



New SUPER MARIO BROS.
 ニュースーパーマリオブラザーズ
 5月発売予定 4,800円(税込)

TAILPIECE

Marketing 101 - Branding: “What’s in a name?”

Since the dawn of modern industry, the practice of marketing has evolved into the money-making merchandising machine that all companies rely on today. They sell, promote, advertise and ship their product into the homes and into the hands of their potential customers.

These customers fall into targeted segments that are focused upon by the marketers. Their job is to make these products look and seem more appealing to their chosen target market than those products offered by the competition. Now, how does this all apply to gamedevelopment, you ask? Well, I recently listened to a very interesting podcast on [GDC Radio](http://www.gdcradio.net/2006/08gdc_radio_presents_developers.html) http://www.gdcradio.net/2006/08gdc_radio_presents_developers.html.

These lecturers compare the role of the marketing team to that of the development teams, where information can be misinterpreted, therefore wasting resources, not to mention the frustration of not being able to synergise these two key business units into one game production powerhouse. According to these lecturers, there is constantly conflict between marketers and developers. This proves an age-old saying that assumption is the Big Mamma of all

mishaps. First of all, a “game plan” is required to assist a marketing campaign. This plan starts off with an idea or concept of the game, which the developers have hatched together with the creative art and design teams. It is important to have all the necessary

consumer views your product in light of all that they have heard, experienced, and taken from the use of your product. Branding is thus the name of the game! (There’s a pun in there somewhere...)



Fantastic game brands that have been built up over the years include: Doom, Quake, Unreal, Ultima, Dungeons and Dragons, Warcraft, Halo, Pacman, Sonic the Hedgehog, Mario Bro’s, etc. Hmmm, you catch my drift? Now what makes these brand names so prominent and instantly recognisable? Well yes, they are all names of well-known and loved games. But look deeper into what messages they have conveyed over

checks in place to make sure that this grand plan can run to fruition from the word go. At the base of this plan is a message, a calling; something that differentiates this game from the rest. Something relevant and revealing. The most primary message is found in the game’s brand - its name, which, by definition, has many different meanings. However, I believe a brand is how the

the years and bear in mind that no brand is made over night.

These brands names will also carry a small “™” or ® which means that they are bona fide, certified, Registered Trademarks of that company’s achievement through years and years of hard work on developing the game as well as the brand. Every little bit of

information, both publicised and spoken, about these games has contributed to their unique brand development. Every teaser advert, demo package, add-on, sequel, web page, blogspot, forum, good and bad review, print and video advertisements all play a part in building the brand. In brand development there is NO such thing as bad publicity. When someone talks about your game, they are spreading the word and opening a doorway to further exploration and tweaking of your curiosity.

A great brand is an important - nay, an essential - aspect of your game. I hear you asking, "But surely the gameplay, storyline, and characters are the most important aspects of any game?". Well, yes they are. A good brand name and an effective marketing campaign will never salvage a weak or inadequate game. However, a thousand words, ideas and sensa-



tions can be experienced from hearing but one name: the brand. This is what marketers will use when designing campaigns and strategies to sell the games by creating deepened desires and wants within their target market (Xbox 360, anyone?).



Now one could go deeper into the whole psychology of branding, such as colour usage, shape and logo design, but I do not wish to bore you just yet with all that kibble. Just know that when you get people talking, you get them excited. Look at all the hype that the marketers of NC Interactive / Guild Wars have managed to create with their ever-expanding credo of player classes.

Also take note of what new engines, reworked network interface and killer marketing has done for the Warcraft brand. WOW! (Sorry, I really can't help myself) A brand is a game's identity, and conveys a message to the consumer from their first encounter with the game.

This identity needs to be maintained throughout the game in the form of gameplay, weapons, baddies, storyline, as well as any complimentary, visual, eye-catching "merch", such as packaging, limited edition box sets, action figures, in store displays, and booth babes. YAY!

However, it all has to start somewhere - and how do we turn an unknown name an international phenomenon? Let's put it this way: First, one needs to educate the target market and



establish brand awareness. This is achieved through exposing key decision makers to the game. These would include members of the press, retailers, celebrities, distributors and VIP gamers.

This awareness can be brought about through a supportive and informative advertising campaign, as well as direct response mediums such as discount coupons, trade specials, online bookings and registration, and, everyone's

TAILPIECE

favourite: competitions. Public Relations activities or stunts are also used to attract attention. Now that we have the consumers' attention we need to give them reason. By this I mean we have to give them some "bang for their buck". Why should they choose our new release over that of a rival game house?

Well, quite simply, we need to give them a "value proposition". Such a proposition will add quality to our offering and, as value is intangible, the proof is very much in the pudding. In other words, if our game makes use of the latest Pixel Shader 3.0 technology and it looks breathtaking on a HDTV, then the public has a right to know. This will then add to the positive way in which our target market perceives our game.

We would therefore have added an element of quality to our kick-ass final offering. This will, in turn, cause a favourable attitude towards our brand and start to solidify our brand reputation, thus growing and developing our brand or game into a household name. Ultimately, we'd like to establish a fan base from which the concept of brand loyalty is derived. If we can establish a loyal fan base, then we will have to spend much less on advertising and other marketing communication mediums.

The reason for this is the same as reason why two opposing fanboys will spend hours flaming each other on a forum or chat room. Preference.

So, to you developers out there, I can only but ask that you begin to familiarise yourselves with the inner workings of marketing in business and what the marketing team requires from you in order to generate the sales and make your games even more appealing by focussing on the subtleties that you

technical wizards often take for granted. Building a game is one thing. Keeping that game in demand for a sequel and beyond is another. I'm talking about building a brand, a legacy, a milestone in cyberspace.

BURNABIS



DIGITAL MAYHEM

YOU PLAY THEM



WE MAKE THEM



WANNA
KNOW
HOW?

GAME.DEV

FIND US AT **RAGE**

COMIC

The HOTTEST 1 Year IT Course in SA

Laptop Included

Microsoft
CERTIFIED
Application Developer

i.t. intellect
COMPUTER TRAINING SOLUTIONS

National Certificate 2007 *Information Technology*

Contents:

- | | |
|---------------------------------|---------------------------------|
| I X Desktop Specialist Training | I X SQL Database Administration |
| I X Business Skills | I X JAVA Development Toolkit |
| I X A+/N+ COMBO Pack | I X LINUX+ |
| I X .net Windows Development | I X IT Project Management |
| I X Network Engineering Pack | I X Website Development Toolkit |

**South Africa's Premier 1 Year
IT Certification For School Leavers...**

DBN - 031 277 2000
PMB - 033 386 6057

JHB - 0861 484 484
RB - 035 789 3115
CT - 021 421 8555

**FREE
INTERNATIONAL
EXAMS***

*conditions apply

Microsoft
CERTIFIED
Systems Administrator

www.itintellect.com

i.t. intellect
COMPUTER TRAINING SOLUTIONS

www.itintellect.com

Presents



HOT LABS

DIGITAL GAMING SOLUTIONS

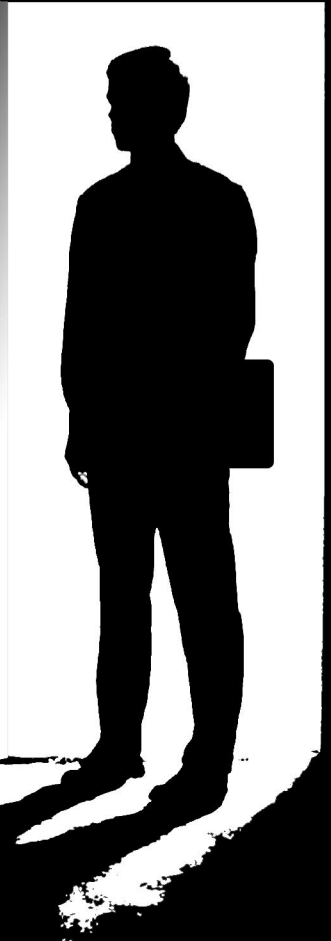
IT Intellect Durban are hosting Miktar and Danny Day for a weekend of Game Maker MADNESS...Come and learn from two of South Africa's leading authorities in local game development...

DATE: 10 - 12 November 2006

PLACE: IT Intellect Durban
Shop 405b, Level 3
Musgrave Centre
KwaZulu-Natal

For more info call: 031 277 2000

BOOKINGS ESSENTIAL!!!





O N L I N E

www.devmag.org.za