SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

JULY 2007

DEV.MAG

CREATE • DEVELOP • EXPERIENCE

REGULARS

SPOTLIGHT

OPINION

REVIEW

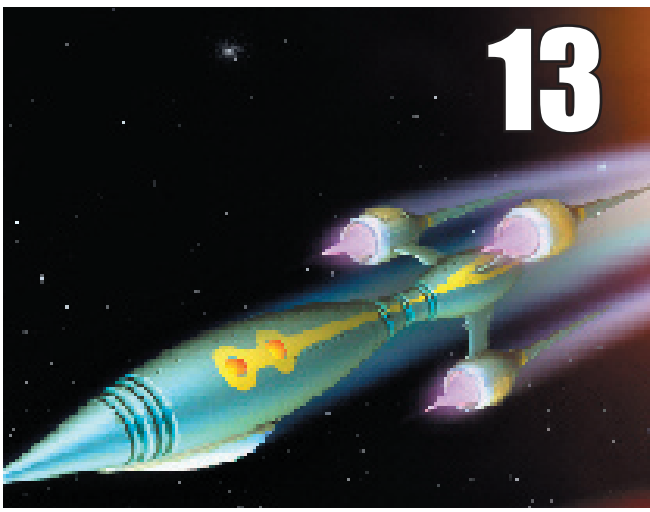
DESIGN

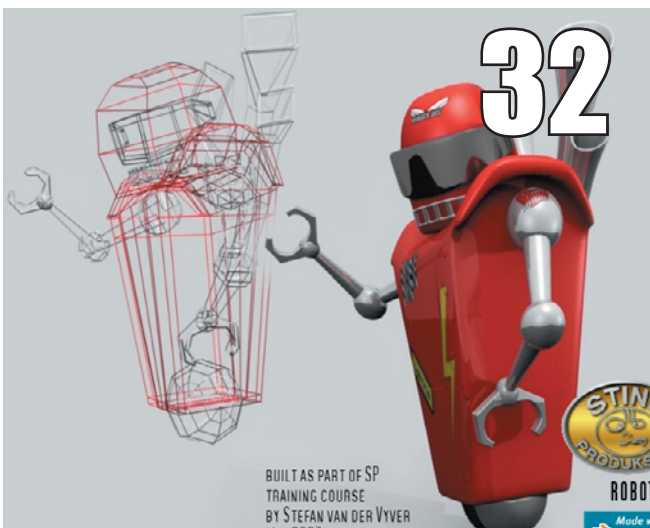
PROJECTS

TECH

HISTORY

TAIL PIECE





So, Comp 15 is over and the Game.Dev crew are moving over to a new forum. For local readers who dwell at school or on campus, I hope that the new term has treated you better than it has some of us! In fact, we're still not sure where that giant carnivorous hamster ran off to with our poor Dep Ed's arm, but it's a problem we're working on.

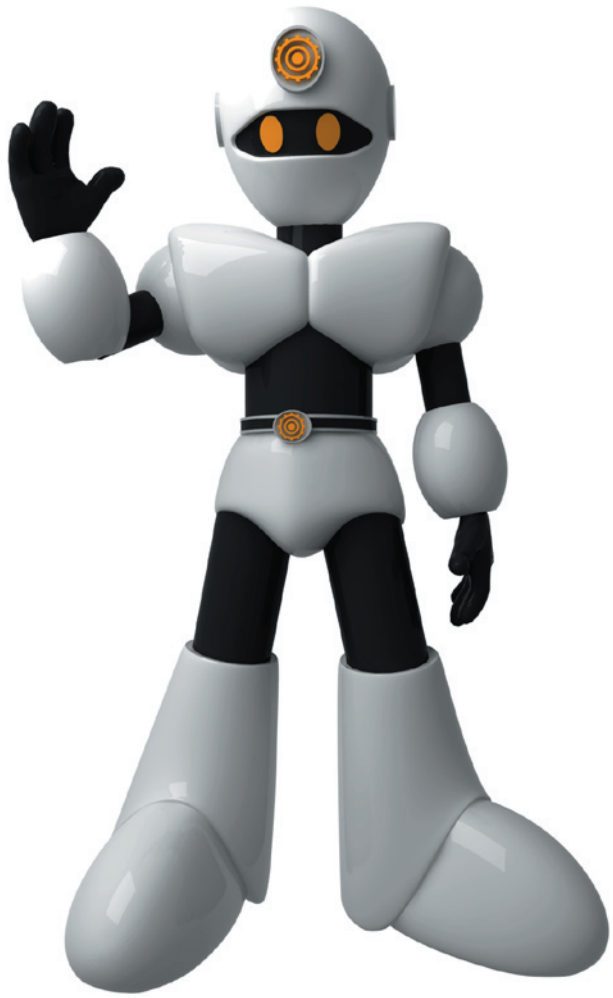
Putting the mag together was an awkward little deal this month, and we've skipped out on one or two standard sections in favour of other cool stuff. One of the gifts that this month provided us with was a nice little surprise shortly before deadline – one of our writers grabbed a fantastic 18-minute interview with Daniel Brewer, a local programmer who's been a great help at a lot of the Hotlabs down in good ol' Durban and happens to be lead coder on a Warhammer mod for Unreal Tournament called UT40K: The Chosen.

Thus, our feature this month is basically a 4-page Spotlight piece with Mr Brewer (officially the longest chat we've ever subjected anybody to, the poor guy), not only because he worked on this mod but because he's landed a job with Digital Extremes as a result. Some of you should have little alarm bells going off in your heads by now, since DE are the blokes responsible for the Unreal series in the first place. Being an Unreal fanboy myself (hey, we all have our weaknesses), it's really great to be able to have Daniel in the mag this month, and it was even more fun playtesting his mod (http://ut40k.planetunreal.gamespy.com/).

I was also flipping through the most recent installation of I-Imagine's history (oh, this may surprise some people, but I really do read the stuff that goes into this mag), and it's amazing how hectic the politics of mainstream game development can really be. More than ever, it's become important for eager new developers to get informed about the industry, even with the indie development scene, and anybody who has an interest in knowing what it takes to start a game studio should really check out the I-Imagine series, starting with Issue 10.

But hey, overall lesson of this editorial: Unreal Tournament is cool, and I'm off to play some after this edition of the mag has been released. Enjoy this month's offering!

Editor
Rodain " Nandrew " Joubert



DID YOU KNOW?
Sometimes, small technical issues can vastly change large aspects of a game. In the Dreamcast version of the original Unreal Tournament, most of the Assault maps were too large to fit into memory, so the game mode got replaced with a new Challenge mode instead.

This month's opinion columnists:
William "Cairnswm" Cairns
Ricky "Insomniac" Abell

DEVMAG
CREATE DEVELOP EXPERIENCE

EDITOR
Rodain " Nandrew " Joubert

DEPUTY EDITOR
Claudio " Chippit " de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNERS
Brandon " Cyberninja " Rajkumar
Geoff "GeometriX" Burrows

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

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

WEBSITE DESIGNER
Robbie " Squid " Fraser

WEBSITE
www.devmag.org.za

EMAIL
devmag@gmail.com

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

The rAge build-up begins

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

South Africa's annual gaming and technology expo, rAge, is just around the proverbial corner. In anticipation of this year's coming events, Game.Dev's official website has released an article detailing the shenanigans which the crew were involved with at last year's event, along with the promise that this year is set to be bigger and even better. Game.Dev also keeps an archive of game development articles that have been published in NAG magazine, so those who are interested in further reading (and some handy tutorials) can check it out.

Torque Game Builder update available

http://www.garagegames.com/products/torque/tgb

Torque Game Builder 1.5 is now available for fans of this game making tool. TGB is a powerful game creation utility from indie developers GarageGames, and the new version (free for those who have already paid for the initial product) sports several major improvements, including a much smaller file size for final executables, easier file access and platform support, several package updates and a new "Behaviors" system for game objects. For those who haven't tried TGB and would like to see what it's like, a free 30-day trial is available from the website.

New Pascal-centred online magazine release

http://www.delphigamer.com/

PascalGameDevelopment affiliate Delphi Gamer has just released its first edition of a quarterly magazine that deals with game development using the Pascal family of programming languages. Delphi Gamer is a showcase site for PGD's games and aims to promote the idea of game development with a showcase of articles, interviews and postmortems. The magazine is free for download from the site, and issue 1 features XNA Chrome, Pascal's contribution to game development on the Xbox 360.

AI Design for Turn-Based Strategy

http://www.gamasutra.com/view/feature/1535/designing_ai_algorithms_for_.php

An interesting challenge that most programmers are presented with in their games is the matter of AI. One of the most tasking genres for this is the Turn-Based Strategy game, where a computer's conventional advantages of speed and accuracy mean nothing against a skilled and carefully thought-out human player's plan. Gamasutra's recent feature on algorithm design looks at some of the challenges that face programmers in this arena, and provides a simple but useful technique to employ for a typical TBS game's AI.

TALKING TO DANIEL BREWER

Digital Extremes programmer and UT40K coder

Tell us about yourself

Well, it all started way back in the old days on the Commodore 64. I got into programming and really started getting into game dev when I was in varsity, where I decided to start working on a few projects, though they've kinda always been on a little bit of the 'back burner'. So while I'm at work, or during my spare time, I work on some games and mods – which is what has really been getting me out there.

Yeah, sure, we'll let him out in a few years. Until then, he's gonna be our code monkey. Muahahaha!

Awesome! You said you went to University. What did you study?

Ah... I went the University of Natal in Durban and studied Electronic Engineering. There was originally a choice between Computer Science and Engineering. In the end, I thought Engineering would give me a more well-rounded base of skills to work from, so it's what I decided to do. There was quite a lot of programming in the Engineering course – having to program electronics, physics and chemistry as well as all sorts of other subjects – but programming is something I've always done and it just grabbed me and just sorta came naturally ... so I managed to start there.

So what language did you start coding in?

Originally in LOGO for the Commodore, but Basic was the first language I really started on, and of course Turbo Pascal at school and from there moving through to C and C++, which is what I program in today.

A jack of all trades, it seems. What sort of traits separate you from the rest?

Well, I've got a strong background in C++ which was predominantly self taught and about 4 or 5 years ago I started looking at OpenGL, a 3D rendering library which is also cross-platform and can be used on Windows, Linux and Sili-

cone Graphics Workstation. It's fairly open and out there and very powerful. So I got into that and taught myself OpenGL from scratch and started entering a few demo competitions online. More recently I've started doing mods for the Unreal Tournament Engine and that works in Unreal Script, which is their own scripting language that the Unreal Engine uses. Quite similar to C++ and JAVA.

Sounds like a lot of time and effort. Where do you go if you want to DIY it?

There are some really good resources out there on the net like Gamasutra, which is a great site for articles on the industry as well as articles on how to do certain things. Also Gamedev.net which is a big international site that has a lot of information of a slightly more technical nature, as well as an open forum where you can post questions and share views. It's free and quite helpful. The place where I learned OpenGL is a place called Neon Helium which is done by a guy in the States who started writing some really good tutorials for OpenGL. They introduce you to the basics and progress upwards.

Hey that's pretty sweet, and you say this stuff is all free?

Yup ...

And would you definitely recommend a language such as C++?

C++ is pretty widespread for game development, mainly because it's very powerful and is cross-platform so you can code for Windows, Playstation, Xbox and so on.

Really? I see a lot of C# stuff coming about nowadays.

C# is a lot more recent and being punted by Microsoft. C# is also Microsoft Specific so it'll only work on Windows. With their new XNA setup and libraries they've got, you can write games for your Xbox 360. I'd say that C# is a good place to start learning because once you've been programming for a while and learned one language its very easy to learn another, especially if they are similar. And luckily, C# and C++ are quite similar. C# does handle a lot of things automatically where you have to do them manually in C++.

Well, enough shop talk! Let's get onto why you are here. Digital Extremes. Tell us more.

It's actually quite a long history. I've been applying for jobs in the games industry for 2 years now on the game dev sites Gamasutra and Gameindustry.biz, which have regular postings for jobs throughout the industry and the world. I got into the mod scene to do a mod for Unreal Tournament to try and get a bit of a portfolio going. A mod is really a good way to test your skills and shows a level of teamwork which is very important in this industry as well as being able to meet release dates and deadlines. This provides ample experience for you to base your portfolio on. So I then started looking for a mod to join and came across a Warhammer 40K mod for Unreal Tournament 2004. Being a huge fan of the Warhammer and Warhammer 40K universe because I play the tabletop war games. I decided to join the team and it just so happened when the team leader received my email they were desperately looking for a coder, as their programmer had left them. So they welcomed me aboard, I then had to learn Unreal Script and started going from there. That was 2 years ago and we have released quite a few versions of the mod, just keep adding more stuff and fixing bugs, improving the game play.

Where can we find this mod?

It's online at:
http://ut40k.planetunreal.gamespy.com
You can download it from there. You need Unreal Tournament 2004 with the latest patch and the bonus pack.

Fantastic! So you were searching, had the drive, had the ambition and it's worked out.

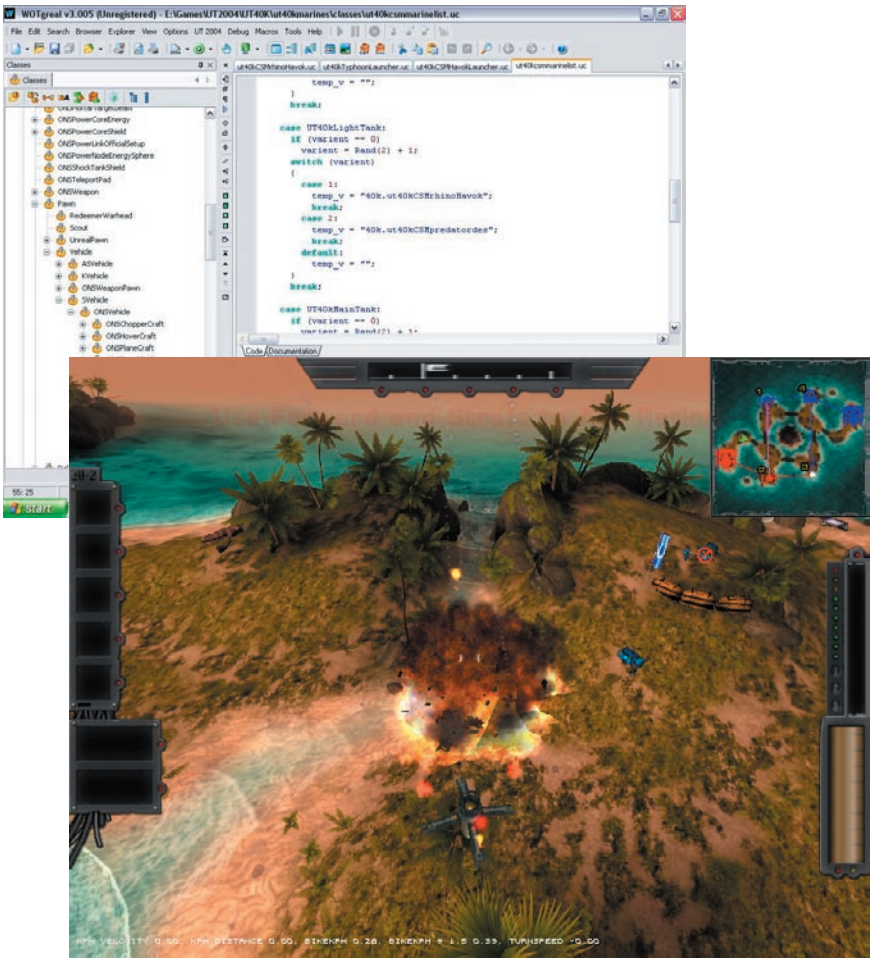
Yeah, well, I had enough of my current job and just sent out a whole flurry of job applications, and was even contemplating starting my own independent game development company. Then I got an email back from Digital Extremes. They said they liked my CV and my portfolio, which I'd put online, and the mod. Then they wanted to know if I wanted to do a programming test, so they sent me the test just to check firstly how good my C++ problem solving skills were, which they were happy with, then asked for a phone interview. I then had a very long phone interview with some of the big hotshots in the industry including the father of Unreal, their lead programmer, the producer, and some other people who grilled me for an hour. There was a lot of talk about the mod, which raised a lot of interest, as they probably get a kick out of seeing what people do with their game. Then they emailed me a little while back with a job offer and that was that.

Awesome stuff! So what do you know about the company itself?

From the few pictures I've seen on the website, it looks like a nice place to work in as there are about 60 developers and its quite big. I'm just really looking forward to getting there eventually once everything gets cleared with visa and customs and such.

How do you know about Dev.Mag?

It started on the NAG Game.Dev forum, where I frequent under the name SilentBob, and I found out about the Hotlabs initiative and decided to check it out. I then signed up to help grow the knowledge base by giving talks and lectures on game development related issues and some nifty coding examples and AI scripting. This was great as the Hotlabs became about pushing the envelope, getting the community together sharing the knowledge and sharing your time. As its no good sitting in your room coding the next best game engine that's going to beat Oblivion



if no one out there is going to see it. That's what Hotlabs are about, bringing together guys who are passionate about game development so that they can share their knowledge and share problems that they've had and just show off what they've been doing.

And having fun!

Definitely! I think some of the most rewarding programming that can be done is in game development. I mean, it's one thing to just program an accounting package. When you're programming a little fighter zapping across the screen blowing up stuff it just doesn't compare, and even in the Hotlabs here when you see people starting off with Gamemaker and getting their little flying sprite moving across the screen, it's a time where you can see that they're really getting into it.

What does this all mean to you? The success, this opportunity that's just come to you now... What is your plan for the next 3-5 years?

I'm extremely excited to begin with going off and working at Digital Extremes, as I've been a fan of the Unreal games for a long time now. Well, I've just really gotta see where it goes when I get out there. There's a game I'll be working on which hasn't been announced and will be released next year or the year after for the PC, XboX360 and Playstation 3. I'll really have to see where the ride goes, as it's been a lot of hard work to get this far.

Sounds cool! Any advice for our readers?

You gotta just keep plugging away, keep at it and keep the momentum going. Get your stuff out there, share it with other people, put it out on the web and keep going and eventually someone will answer. Focus on teamwork. These days, companies do value that sort of skill as it takes many, many people to make a game and you can't just do it on your own. And remember, have fun doing it!

MUSHIMUSHI

The ONLY Pascal Game Development Magazine

Issue 1

Chrome & XBOx 360

Airblast Postmortem

And MUCH More...

Out 1st August 2007

Available from www.delphigamer.com

"Business models: thinking outside the proverbial box"

Game development can be a lot of fun and a passionate hobby for many of us, but I'm sure most have also thought about how some money can be made from this awesome hobby, whether you thought about making a bit of extra cash to help fund that next graphics card or would like to make a living from it. The problem is that while we all have bucketfuls of great ideas for our games, in my opinion not enough time is spent on thinking of innovative ways for our games to earn some cold, hard cash. Most of us just think of the standard business model where you make a game and then sell it for some arbitrary amount to as many people as possible. While this may work for the big players in the industry such as Nintendo, EA etc. things are way too crowded in the industry for your game to have a good chance of being noticed.

I had been trying to think of how games could bring in money in a new way and then inspiration struck from a question in a recent Computer Science exam. I won't go into the gory details of the question, but basically we had to code an applet that was a game advertising for a food shop. If the user won the game they got a voucher, otherwise they got told that they won nothing. This simple little example struck the idea of how powerful a form of advertising games can be.

Heaps of companies are always competitively trying to market and advertise their products. The old and traditional ways for companies to reach mass markets such as adverts on television just don't work any more, as not only are people watching less TV due to other distractions such as the Internet, but people have also become desensitized to most adverts and hit the mute button as soon as some come on.

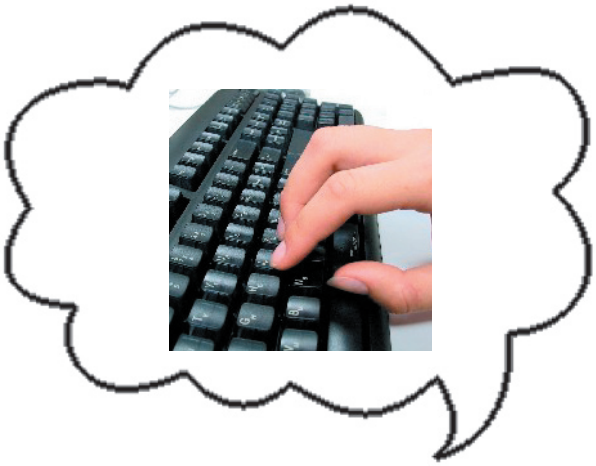
That's why an avenue such as games which advertise a product can be a very powerful and attractive way for companies to reach audiences. Think about how much exposure a company can get if, say, a recent popular game such as Peggle - which a lot of people got hooked on, including me - had advertised a product. With all the talking on forums and other places around the world Peggle received I'm sure any companies sales would dramatically increase by a huge margin from the exposure.

As long as the advertising is kept subtle and ties into the gameplay rather than the gameplay tying into forcing a product down the players throat then surely everyone can win. For a great example at how a game and brand can be fused successfully just look at South African company Luma and their game Mini #37(check Dev.Mag issue 12 for more on Luma and the game). The game manages to have a fully featured, fun racing game while incorporating the sheer coolness of the Mini brand.

Trying to pitch an idea to a company to convince them to just hand over money for a game may sound a tad crazy, but with the proper amount of research into a market and a solid way to incorporate the brand into the gameplay I can't see any reason why a company won't at least consider it and the viral nature of exposure games can offer.

INSOMNIAC

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



"Hard coding is bad! (?)"

Hard coding is bad? Says who?! I suppose it depends on what you consider hard coding and where in the project life cycle you are. But let's take an example:

```
Print("Hello World");
```

Okay, so where is the hard coding? Do you consider the string the hard coding? Well then what about this:

```
SetWindowTitle("Text Editor");
```

Is that bad? What if your application is called "Text Editor"?

Hard coding is considered a bad development practice as it restricts the ability of the software to be modified to suit new requirements. For example if the string "Text Editor" is used 33 times through the application code it makes it very difficult to modify the name of the application to something like "Textpad". So the idea is to ensure that values are not hard coded into the system, but rather stored in such a way that they can be easily modified.

Sometimes hard coding is not bad. Sometimes hard coding is the clever way of doing things. In South Africa, for example, almost all applications are done in English. Why go to the effort of allowing alternate language options for button captions, etc. - especially when you are prototyping a new idea.

When prototyping, the whole idea is to do it AS QUICKLY AS POSSIBLE. Take the shortcuts, etc. But when implementing the "bad" hard coding, keep in mind that you may want to fix it later if the prototype leads to a large project. Make use of constants, set a value in one place and reuse the variable storing the value everywhere else.

```
AppTitle = "Text Editor";

SetWindowTitle(AppTitle);
```

Another example of this "bad" hard coding is database connection strings. When developing a prototype, the connection string should be hard coded. As soon as the prototype is ready to hand over to users to test or use, the connection string should be moved into a configuration file.

Let's look at our first example again:

```
Print("Hello World");
```

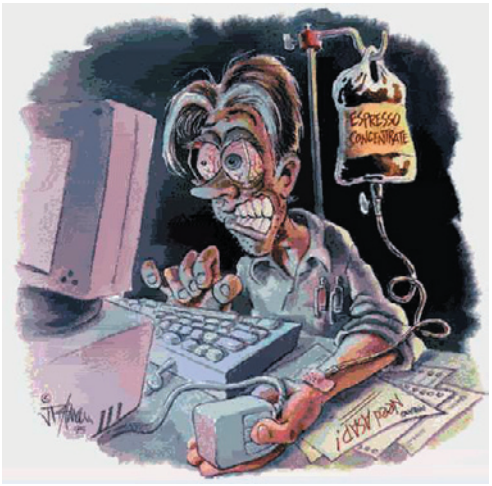
What if I told you that Print is also hard coded? In fact every little bit of code we write is actually hard coding logic into our programs. So in fact, our job as developers is to hard code logic - so is this good or bad? If all hard coding is bad, then we need to develop self modifying code all the time. In fact the whole purpose of unit testing is to ensure our hard coded logic is correct.

Don't just accept that hard coding is bad. Give it some thought first. Hard coding is good, in the right place. It saves time, saves effort and often saves a project!

CAIRNSWM

If you fancy saying something about game development and have enough faith in your writing ability to do so, feel free to submit content to our monthly opinions section.

Send your work, 400-500 words in length, with the subject title "Opinion Submission" to devmag@gmail.com, and you may just wind up in our pages. Please note that we reserve the right to decide what material is suitable to publish in the magazine.



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

TACTICS ARENA ONLINE

Developer:

Digital Seed Entertainment

Year of creation:

2003

Website:

www.tacticsarena.com

Genre

Turn-based tactics

The advent of online gaming, as well as its inevitable rise in popularity, has seen numerous old formulae rehashed to work in a dedicated multiplayer environment. While Tactics Arena Online cannot claim original gameplay, it has nonetheless succeeded in bringing an addictive turn-based combat system to any online user.

Boasting tactical gameplay similar to established games such as *Fire Emblem*, Tactics Arena pits two player's wits against each other in turn- based combat. With a relatively simple user-interface, the game is easy to pick up, but the hidden complexities of the game system may take a little while to decipher. However, these very same complexities grant the player a lot of tactical opportunities to utilize in play and forms a major part of the game's appeal.

The greatest attribute of Tactics Arena Online is that it can run straight in your web browser without needing any additional downloads or software besides Macromedia Flash. Within moments of creating a game account you can test your battle-worthiness against other players on one of the game's online servers.

Offering players customizable starting unit lineups and arrangements, no two battles will ever be alike. However, a large portion of available units and customization options, including two entire races, are not available to free accounts. A small monthly fee of $5 is required to obtain a gold account which allows players access to extra units and game features.

Overall, Tactics Arena is a great way to bend your mind and test your wit against other players, and, for fans of tactical gameplay, this is a gem. Single-player experiences against AI can never truly compare to battles against human players.

CHIPPIT

TAKING THE BLUE PILL FOR...

Microsoft XNA Framework

What is XNA? XNA is a set of libraries created by Microsoft for its C# programming language. Apparently it stands for "Xna's Not Acronymed". A bit paradoxical...

XNA is decent set of libraries for the aspiring game developer. Although XNA is not recommended for total beginners, it is definitely a great way to get yourself accustomed to a standard programming environment.

Where do I get it?

http://msdn2.microsoft.com/en-us/xna/default.aspx.

Just follow the links.

You need the XNA Game Studio Express, Visual C# express and the .NET framework. Installing XNA is easy, but it requires some extra steps.

You must first install Visual C# Express. This is the basis for XNA, since XNA is just a set of libraries. You must also attain the .NET framework, which comes with Windows XP service pack 2 and Vista by default. And then last, but not least, you should install XNA Game Studio Express. This sets up your libraries and creates easy templates to get coding as soon as possible.

Tutorials:

http://www.xnatutorial.com
http://www.xnaresources.com

A Spacewar game example is included with XNA. Unless you understand the basic coding structure, this example is actually very complex to understand for any beginner.

XNA-Tutorial is a great site for getting info and examples although the blogger hasn't posted in a month.

Comments:

As mentioned, XNA is a very decent offering from Microsoft for the intermediate indie developer. Game Maker will still be your best if you are a total beginner.

XNA has some cons. It has poor sharing capabilities for the Windows platform. If you want to hand out your game to your family they will need the XNA framework and .NET framework on their PC!

The other thing is the poor audio support. It can currently only support .wav and similar formats. This means, while your code is very small, a minute of music is in the range of 20mb. We can only hope that OGG Vorbis support could be added in the future.

Fortunately, XNA's ability for any developer to develop games for the Xbox 360 makes up for any of its cons. To create a game for a console so easily is just pure magic!

TR00JG

Beginner's Guide to Making Games Part 2 - Collisions

(Feeling lost? Check Issue 14 for the first part of this series.)

Welcome to the second part in the Beginner's Guide to Making Games series. Each month a single important concept required for making games is discussed in detail. This month we are taking a look at collisions. Collisions are core to most games because you need to have working collisions to blow up enemy space ships or to land missiles accurately on top of enemy cities.

The main goal of the Beginners Guide series is to try and ensure a detailed understanding of the various concepts so that they can be applied to other new and exciting games. While most traditional tutorials will show what code is needed, these guides will ensure that you walk away actually understanding each of these concepts rather than just having done an example.

This article is aimed at someone who has just started learning to make games. While it is expected that the reader of the article has completed the first Game Maker tutorial and can thus create sprites and objects it is quite possible to follow the article without having done so. The article is structured to introduce a new programmer to the concept but yet give some value to the intermediate level programmer as well.

The vast majority of games require collision scripting. Some games such as word puzzle games get away without having collisions but these are the exception rather than the rule. To keep things simple we will only look at collisions in 2D space.

Today's article will contain the following sections:

1. A brief Game maker tutorial showing the basics of getting two objects to collide.
2. This is followed by a discussion, in detail, of the various Game Maker collision actions available to you
3. Game Maker Language (GML) is a powerful way for the developer to get closer to the action, and contains numerous functions for collisions

Game Maker Tutorial

This tutorial is going to create a room that contains a ball and a wall. The ball will move in a random direction until it hits the wall.

This tutorial is not as such a game, but is designed to show various methods of checking for collisions. A few small changes could make this into a fun game. You might create a bat that gets moved by the mouse, and give the player a score each time they hit the ball, or even create a two player tennis game. Or how about adding blocks that blow up when hit by the ball, or perhaps make the player click on different blocks to make them change shape and then bounce the ball differently around the screen, like a reverse pinball game.

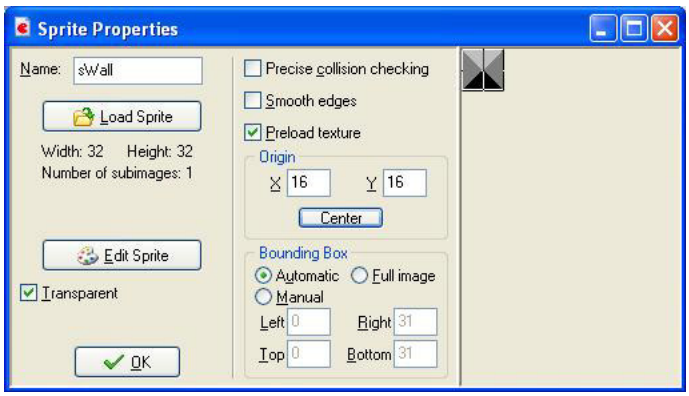
To demonstrate how collisions work we need to create two objects that can collide with one another, as well as the events to define what happens when they collide.

Step 1 - Create sprites

In Game Maker, add a new sprite. Select an image for it, such as one of the balls in the default graphics.

Set the origin of a sprite to the centre of the sprite to assist with collision detection. To do so press the Centre button on the Sprite screen. As the ball sprites that come standard with Game Maker do not completely fill the image it is a good idea to use precise collision checking for them.

Transparent sprites usually look a lot better than a solid sprite unless for some reason the sprite is a single block of colour. The colour used to make the sprite transparent is the colour in the top right hand corner of the sprite.



Now create a second sprite to represent the walls we are going to need.

Note that the wall sprite does fill up the complete area of the image. Therefore exact collision checking wont get you any change in behaviour. To improve the performance of the game, switch off the precise collision checking.

Step 2 - Making the wall object

Using the wall sprite, create a wall object. The difference this object will have from the other objects we have made is that it needs to be solid. Game Maker has two main categories of objects - solid and non solid. Non solid objects are typically the player object and other creatures in the game. This allows these objects to float over one another while using a main category to check that they don't go through solid walls etc.

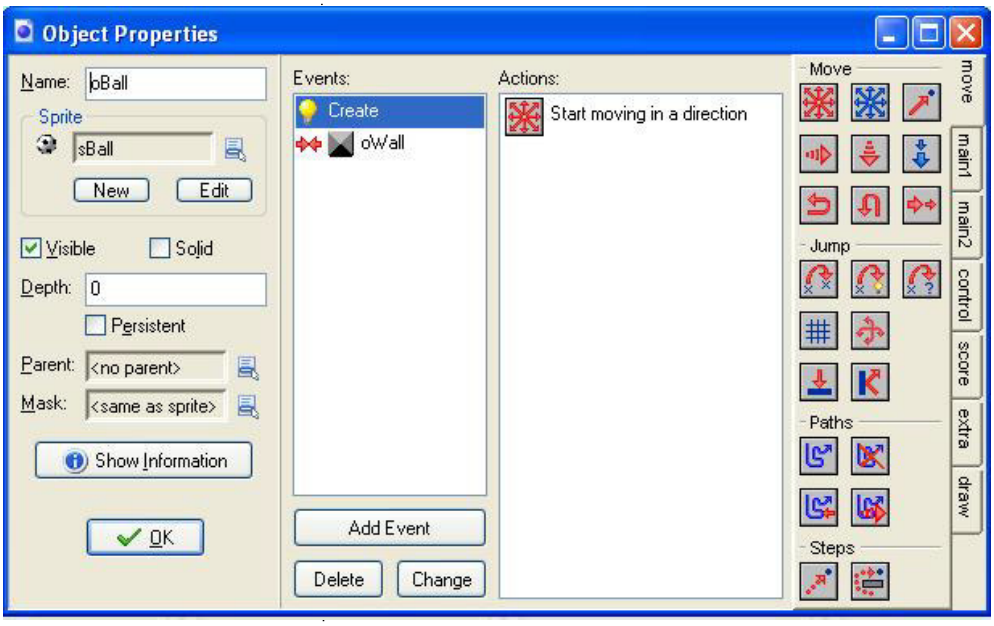
Often the solid objects in the game don't have events added to them. The idea of solid objects is that they sit in the room and act as obstacles of one sort or another for the player object.

Step 3 - Making the ball

Make the ball sprite an object.

As we want a moving ball we add a Create event with a "Start moving in a Direction" action. Select all the directions and give it a speed. When the game starts the ball will move in a random direction.

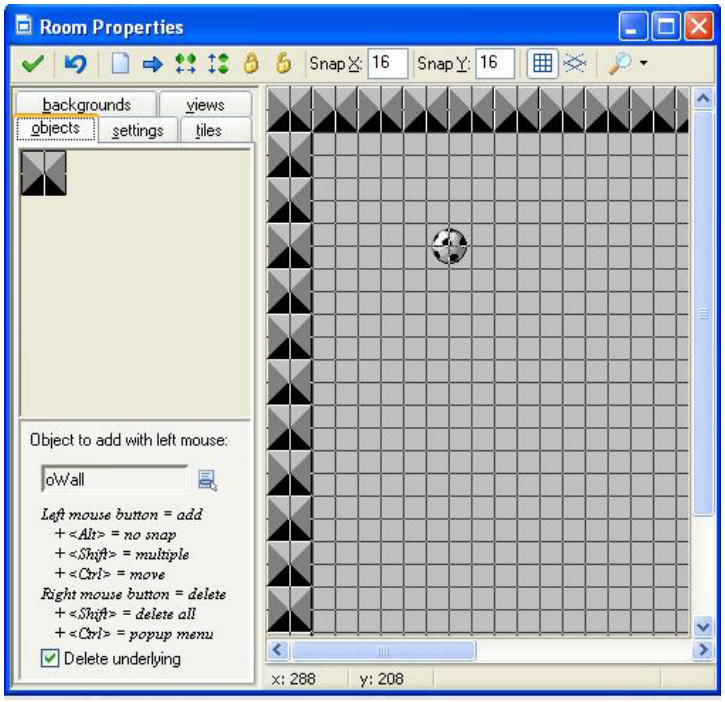
Now for the collision code. Add a new event, select Collision and select the wall object. Add a "Start moving in a Direction" action with the center button selected and a speed of 0. This basically tells the ball to stop moving as soon as it collides with a wall object.



Step 4 - Creating the room

Rooms are where everything happens in a Game Maker game. Without a room objects can do nothing. Using the Wall object build a frame around the room. The idea here is to ensure that the ball cannot leave the screen.

Now add a ball object in the middle of the room.



Step 5 - It works

Run your game by pressing the green “Run Game” button. The beauty of Game Maker is how your game runs every time without fail (I think I’ve said that before!).

Watch the ball go flying off in a direction until it hits the wall, and then stops. Using the standard, most basic, Game Maker events and actions we've got collision checking working.

Colliding Objects and Actions

Game Maker is cool because it's easy, and actions are one of the most important things to making Game Maker such as easy tool to use. To get the most out of the Game Maker action, you need to understand what they do, and what alternate options are available for each thing you want to do.

Often in Game Maker there are alternate options to achieve the same thing. However by understanding what the various options do, they can be better used to achieve the effects that your game requires.

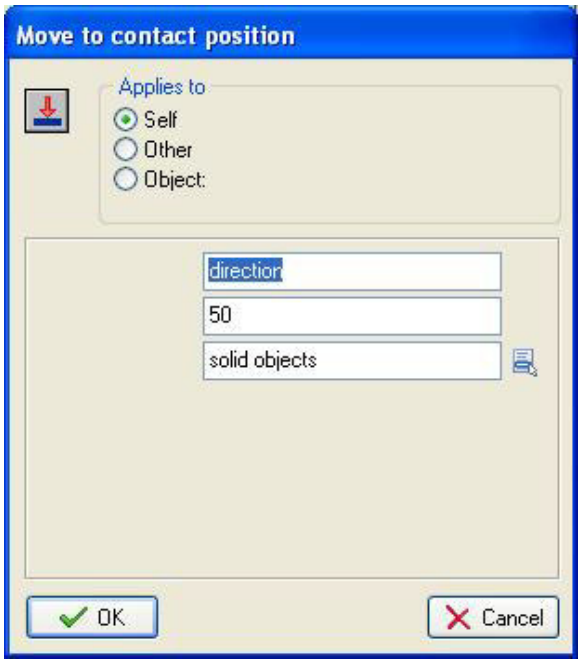
Stop on collision

There are different ways of making objects stop moving when they collide with each other. In the tutorial we have just done, you will notice that when the ball stops moving it is actually not touching the wall. This is because the ball will move 5 pixels on the screen each step it takes. If the wall is ever less that 5 pixels away the ball will stop at its previous position, often leaving a few pixels open between itself and the wall.

Move to Contact Position

Game Maker gives us a nice easy action to solve this problem. Move to Contact Position allows you to instruct the object to move from where it currently is to the nearest contact point in a direction. A nice feature is you can define the maximum distance it should move to collide (think of magnets - they attract up to a certain distance).

Earlier I mentioned that the wall should be solid. In the Move to Contact Position action you can select to move to the nearest solid or non solid object. If we had more than one ball in the room, setting the action to only solid objects will ensure that balls can't collide with one another.

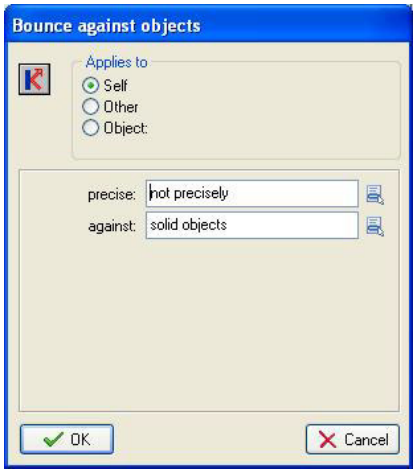


Change Movement on Collision

Many games that use balls make these bounce around the screen. When they hit a wall, or the player's paddle they bounce neatly off the obstruction and head off into a new direction.

Bounce against objects

The easiest way to make an object bounce is to use the “Bounce against objects” action. In the real world an object bounces off an obstruction at the same angle as it hit the obstruction - in other words if you drop something straight down, it will bounce straight up. The bounce against objects implements the same behaviour.



You will notice that you can select a bounce action to be precise or not precise. Consider very carefully before choosing precise as it means the game needs to do many more calculations to do it exactly, and in most cases wont have a material difference on the way the game works.

Change Movement

Quite often there are simpler rules for making objects change movement in the game. For example if you add a bat to the game, and the player needs to move the bat left and right to bounce the ball back into the screen, the ball will only ever need to change its vertical movement when it hits the bat. As we saw last month we can just use the “reverse vertical direction” action to implement this. This will again save a lot of time in the game as its much quicker for the game to execute this command than work out the details for a true bounce action.

Options

There are almost always different ways to make an object behave the way your want it to. Always try the various options to find a way to make the object behave the way you want it to, and choose the option that most suits the current requirement.

Moving objects with GML

Game Maker actions are typically shortcuts to certain Game Maker Language constructs. While that sounds complicated, it basically means that actions are the easier way of doing things that can be done in Game Maker Language, but they only implement a subset of the possible GML functions.

When using the collision event, checking is based on the object currently executing (i.e. the object executing the script), and other objects as defined by the developer. The various collision checking methods compare the collision parameters with a certain type of object. The type of object can be based on an object type or a specific object.

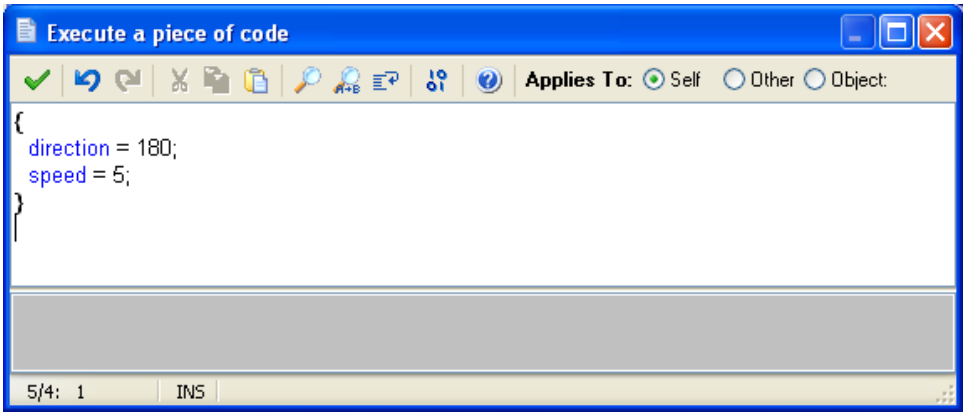
The collision event will only execute if a collision actually happens. The various GML collision methods are typically used to test for a collision to decide if movements are possible or to check that other objects are where they are expected to be. In addition the collision methods will allow for checks such as determining which object is below the mouse.

Calling a script

To call a script for an object, add an event to the object and add the “Execute a piece of code” action to the action list. In the popup text editor add the commands you want to be executed.

Scripts can contain any of the Game Maker Language functions as well as various control structures such as for loops and if statements. GML allows statements to be grouped together through the use of braces to identify start and end sections { }.

Remember also that it is possible to access properties of other objects. So it's quite possible to do things like moving toward other objects or using the position of other objects to check for collisions.

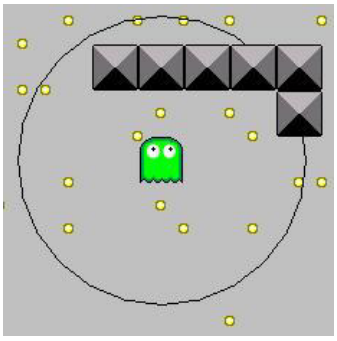


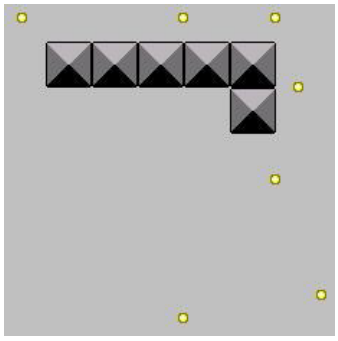
This example code below will find all instances of an object type within a 100 pixel radius of the base object.

This can be used for something like blowing up all boxes around a grenade exploding, or for causing damage to all enemies within the explosive radius of a fireball.

Identifying objects

Everything in Game Maker is identified by its id. This id can be the object or an instance of the object, as well as things like sprites, paths and even scripts and rooms. EVERYTHING in Game Maker is identified by an id.





Each of the collision methods returns an id of an object that has a valid collision or a -1 if there is no collision. The id returned can then be used to check if the object is of a specific type using the object_index property.

Checking for Collisions

Here is a piece of example code that checks for objects within a 100 pixel radius of the current object. Once the object id has been retrieved using the collision method, the object can be interacted with.

```
while collision_circle(x,y,100, ObjectType,false,false) >= 0
{
  d := collision_circle(x,y,100,ObjectType,false,false);
  .. Do something with object D ..
}
```

Collision Methods

Game Maker supplies 5 collision checking methods. Here is a small section copied from the Game Maker help files:

> All these have three arguments in common: The argument `obj` can be an object, the keyword all, or the id of an instance. The argument `prec` indicates whether the check should be precise or only based on the bounding box of the instance. Precise checking is only done when the sprite for the instance has the precise collision checking set. The argument `notme` can be set to true to indicate that the calling instance should not be checked.

collision_point(x,y,obj,prec,notme)

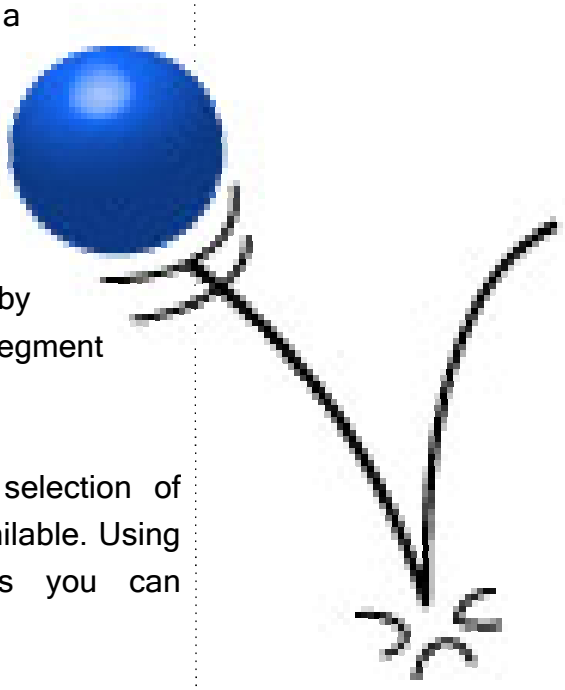
This function tests whether at point (x,y) there is a collision with entities of object obj.

collision_rectangle(x1,y1,x2,y2,obj,prec,notme)
This function tests whether there is a collision between the (filled) rectangle with the indicated opposite corners and entities of object obj. For example, you can use this to test whether an area is free of obstacles.

collision_circle(xc,yc,radius,obj,prec,notme)
This function tests whether there is a collision between the (filled) circle centered at position (xc,yc) with the given radius and entities of object obj. For example, you can use this to test whether there is an object close to a particular location.

collision_ellipse(x1,y1,x2,y2,obj,prec,notme)
This function tests whether there is a collision between the (filled) ellipse with the indicated opposite corners and entities of object obj.

collision_line(x1,y1,x2,y2,obj,prec,notme)
This function tests whether there is a collision between the line segment from (x1,y1) to (x2,y2) and entities of object obj. This is a powerful function. You can, for example, use it to test whether an instance can see another instance by checking whether the line segment between them intersects a wall.



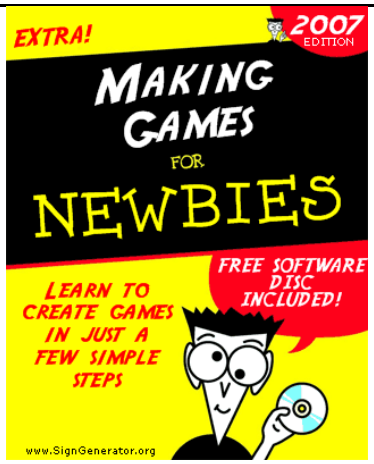
As can be seen a reasonable selection of collision checking methods are available. Using these various collision methods you can implement various different effects.

The power of scripts

While it is certainly possible to create a fun playable game without using scripts, the person using scripts will probably be able to create a more immersive game as they will be able to include proper AI and more believable behaviours. Taking the time to learn to use GML will certainly improve your games.

Using the various collision methods allows the game developer to have a lot more control over what happens in their game. Checks for collisions with objects, and the resultant actions forced on the colliding objects offers many opportunities for special effects and effective game controls.

CAIRNSWM



Beginner's Guide to Making Games - Part 1

EXTRA!

Last month we had a look at different ways of moving an object around the screen in game maker. This month's Extra will look at doing the same sort of thing in other programming languages. The benefit of using Game Maker is of course that you don't actually need to understand all the intricacies of the process.

When moving 2D objects on the screen there are two basic methods of doing so:

1 Frame Base Movement (as used in Game maker)
2 Time Based Movement

Frame Based Movement

Frame based movement is based around a fixed distance an object can move on the screen each frame.

Typical code for a frame based movement loop is

```
{
  Object.X += 2; // Move the object 2 pixels right
  Object.Y += 2; // Move the object two pixels down
}
```

Frame based movement can only be used when the number of frames being refreshed on the player's screen each second is constant. (Game Maker by default refreshes the screen 30 times per second). Because the game is therefore being throttled in its refresh rate, it often uses less of the user's processor than a similar Time Base Movement system.

Time Based Movement

Time based movement moves the object a certain distance over a set amount of time. This is very useful when the actual screen refresh rate is variable.

An example of the code used to implement Time Based movement would be something like this:

```
{
  GameTime = now - lastTime; // Get the amount of
time since the last movement
  GameDelta = GameTime / 1000; // Typically time is
based on milliseconds.
  Object.X += (20 / GameDelta); // Move the object
20 pixels right each second
  Object.Y += (20 / GameDelta); // Move the object
20 pixels down each second
}
```

Time based movement is very important in games where the power of the computer running the game may vary greatly. An old Pentium 1 may only run at 20 frames per second, while a new dual-core with the latest video card may run at over 1000 frames per second. By implementing Time Based Movement, both players will have pretty much the same gaming experience.

Appearances

One of the great things about developing games for humans is that there is a limit in what they can perceive. Time based movement has a vastly superior smoothness of movement across the screen - just consider an object moving across the screen in 30 frames or 1000 frames, each change in position is obviously smaller for the 1000 FPS movement. However the human eye cannot distinguish any difference once the refresh rate exceeds about 30 frames, and often a whole lot less. This can be shown by the fact that 35mm movie cameras use a standard exposure rate of 24 frames per second

Your Choice

Neither method of moving objects is better than the other, each has its place. Whichever method you choose to implement in your game must support the user base you have defined as well as the type of graphic application you are developing.

If you are developing an application that shows the progress of items on a conveyor belt, it is doubtful that the user will be too happy if you use the full power of their CPU. On the other hand, if you have a fast paced action game with a large number of missiles and terrific explosions on the screen, it is just as doubtful that a player will thank you for not using the full power of the CPU.

CAIRNSWM

Windows Vista

Open Source Win64

GameBoy Advance

DirectX 10

.NET 2.0

XBox 360 via XNA

Linux

Mac OS X

Nintendo DS

OpenGL 2.1

SDL

Object Pascal has a lot to offer...

www.Pascal Game Development.com

Chrome

Free Pascal

Turbo Delphi

Delphi for Win32

Lazarus

ROACH TOASTER 2: PICKING OUT THE BUGS PART 6

(Wondering what we're talking about over here? check out Dev.Mag Issue 10 for the beginning of this Project series!)

Roach Toaster 1 has been successfully remade in XNA! By the time you read this, I might have already been placed in the Top 20 - or not. Who knows? I seriously do not know what to expect. *Holds thumbs*

Anyway, this project series is all about Roach Toaster 2. Here is the current development status.

As you might know, I could not release the beta of Roach Toaster 2 some while back, because I did not have a decent internet connection. Well, that is about to change, and again, by the time you read this, I will have my internet!

Roach Toaster 2 beta can commence, but during my hiatus from its development, I took a back seat and looked in depth at the gameplay. As I have not played "Big City" since its inception, I stumbled upon some things I want to change.

These changes stem from the following design issues:

1) A player hates it when he is punished for things out of his control.

Seeing "Big City" breed, I felt a loss of control. I could not really efficiently curb the spread, and it made me feel incompetent.

2) The importance of letting it flow.

After playing addictive games like Peggle, I realized how important it is to let the game flow. When you finish a level in Peggle, it immediately takes you into the next level, without an option to quit. Although a tad diabolical, I enjoyed it and it kept me enthralled. This is what Roach Toaster 1 did right. The game kept flowing into the next level.

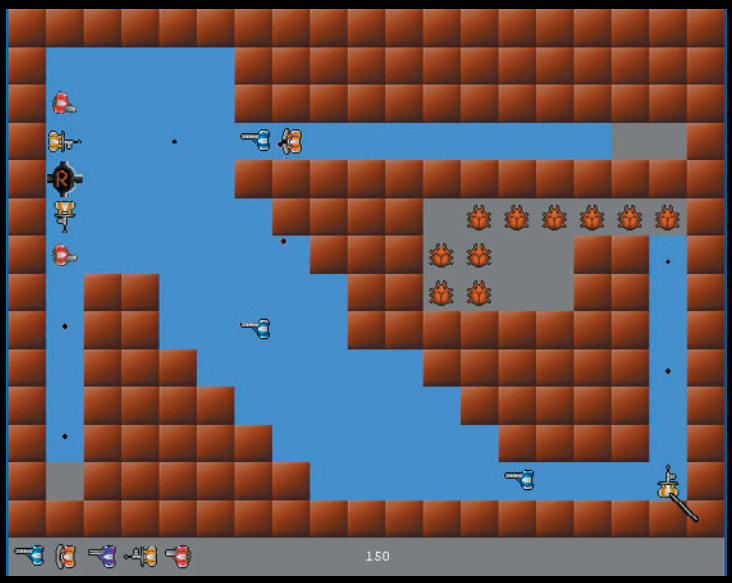
So, taking the above design issues into account, I am going to develop two different prototypes for "Big City". The first is the current non-linear approach it has now, i.e. after each level you are taken to "Big City" to choose your next level, and then a more linear approach. The testers will have to choose between them. I am for the latter, though. We will see how it turns out.

Unfortunately, I can only begin development on it again after July 31st. I have to work on my matric Computer Studies project. I am lucky though. XNA code is very similar to Java, which means, I can easily port it Roach Toaster: XNA.

Having Roach Toaster 1 in Java is also a very good thing. This means that I can put a web version online, which in turn means that I can draw many more visitors to my site, and I can use it to test the finer gameplay in Roach Toaster 2. So all in all, this brief hiatus from development has proved to be very beneficial!

For more info on Roach Toaster 2: Big City, head on to http://www.shotbeakgames.za.net or the new site http://www.roachtoaster.com

TROOJG



CODING ETIQUETTE: DEFENSIVE PROGRAMMING

It is a fact of programming life that programmers will make mistakes in their code. These mistakes or errors are not usually down to "bad programming" or any lack of talent, but merely due to the sheer scope that most programs are comprised of. When any one programmer is churning out hundreds upon hundreds of lines of code a day, it's only a matter of time before some of those lines will contain errors.

The kinds of errors that I am referring to here are not, however, the simple ones (i.e. syntax issues) that are easily picked up by compiling the code or doing a quick execution of the newly written logic. Unfortunately, the errors that I am talking about are the ones that sleep in your code and only rear their ugly heads when very specific or unlikely situations arrive. These are the ones that seemingly only show up during the final phases of testing, or perhaps they appear to come and go at random, making it difficult to actually track down the cause of them.

Debugging these kinds of errors or crashes in code is probably the most frustrating as well as one of the most time consuming aspect of any development project, not to mention the fact that bug fixing is ultimately the single most likely reason for a project missing its targeted deadline. Therefore, it is essential that programmers take the time to properly check for any and all potential errors in their code. Ultimately, this is achieved quite easily by merely writing simple verification checks at any point in their code where resultant data has the potential to be incorrect. These "checks" are usually referred to as assertions.

A few common places where assertion checks should be employed are:

After memory allocations
There may be no free memory available or the size requested may be either too large or potentially a corrupt number.

On data returned from a function call
It may not be the result that you are expecting or the function may have had some form of internal error.

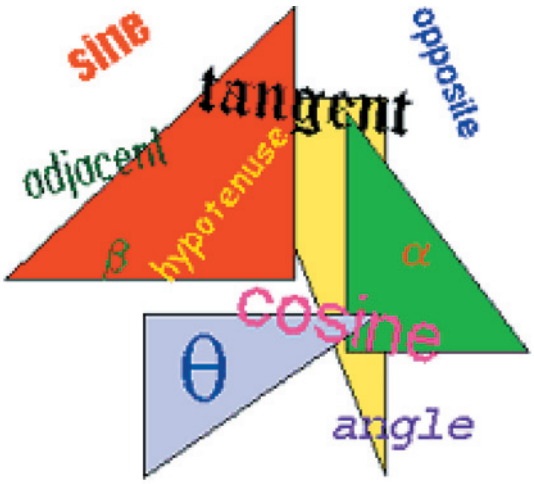
On data assigned from a mathematical operation
The operation may have had incorrect or corrupt inputs, or the result may not be within the valid ranges for whatever you may need to do with it next.

As with most aspects of proper coding etiquette, many programmers consider these checks to be too time consuming for them to do, which is once again merely an excuse to justify their coding laziness. A few minutes spent here and there performing very simple checks will, in the long run, save immeasurable time when tracking down what are quite often nasty errors that could easily be caught with simple assertions.

COOLHAND

Simple C Code Example of Error Checking

```
char *DynamicStringCopy(char *pcSourceString)
{
        char    *pcDestString;
        int     iStringLength;

        //determine the size of the source string, and allocate some memory to
        //store a copy of it (add '1' to in order to hold the NULL terminator)
        iStringLength = strlen(pcSourceString);
        pcDestString = (char *)malloc((iStringLength + 1) * sizeof(char));
        if(!pcDestString)
        {
                //there was an error allocating the memory... either we are out of
                //it (unlikely...) or 'iStringLength' is somehow a very large
                //number...
                printf("Unable to allocate destination memory for string copy.");
                return NULL;
        }

        //now that we have a chunk of memory, copy the string into it
        strcpy( pcDestString, pcSourceString);

        //return our new string
        return  pcDestString;
}

void main(void)
{
        char    *pcMyArbString;

        //store a constant string into memory for no reason whatsoever...
        pcMyArbString = DynamicStringCopy("I am a useless string!");
        if(!pcMyArbString)
        {
                //output an error message for our invalid string copy call
                printf("Unable to store the arbitrary string into memory");
        }
        else
        {
                //do arbitrary stuff with the arbitrary string...
        }
}
```



GAME CODING WITH TRIGONOMETRY PART 2

Last issue we covered the core principles of trigonometry and learned how to use these to calculate unknown sides in a triangle. Well, I hope you're rested and ready Commander, because we'll be jumping straight back in with another essential technique - calculating unknown angles.

Angling 101

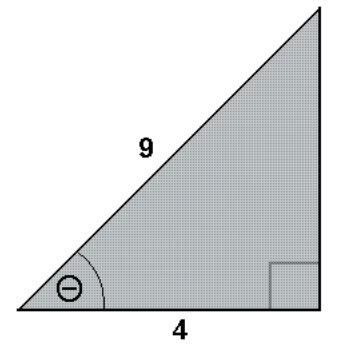


Figure 1: Mystery Angle θ

As always, let's start with a triangle. This one has two known sides (the hypotenuse and one other), but the angle, theta (Θ), is unknown. Bummer. Let's fix that.

Okay, let's use our super trig techniques from last time. Last time we learned to solve sides by substituting into the magic formula that follows.

$$Ratio(angle) = \frac{\text{Unknown Side}}{\text{Known Side}}$$

"But hold on," you might think, "we're not solving for an unknown side this time!" You're absolutely correct. When solving for angles we need to perform a little tweak to the formula, but the core principle remains the same. Watch and be amazed:

$$Ratio(\text{Unknown Angle}) = \frac{\text{Known Side 1}}{\text{Known Side 2}}$$

Still looks familiar, right? Let's see how it works. Since we're solving for Θ, we'll name our sides relative to that angle in the usual way. This leaves us with h equal to 9 units, and a equal to 3 units. Now we need to set our ratio. As you can see in our adjusted formula, we'll be using our known sides for this. However, if you study the ratios from last month, you'll realize that there are two possibilities for the ratio depending on which side of the division line we choose to put our numbers. For instance, if we choose:

$$Ratio(\Theta) = \frac{a}{h}$$

we would use Cos(Θ) as our ratio. However, if we choose:

$$Ratio(\Theta) = \frac{h}{a}$$

our ratio would be Sec(Θ). The choice is really up to you, but since Cos is generally the easiest and most frequently used ratio, that's the one I'll be working with. While we're at it, I'll plug the numbers in as well.

$$\cos(\Theta) = \frac{3}{9}$$

Now we need to solve for Θ. That means that we'll have to cancel Cos out to get Θ on its own. For this, we'll use the arccos(Θ) trig function.

Huh?

Inverted (arc) functions are essentially the antimatter of the trigonometry world. By applying the inverted version of a function (arcsin, arccos, arctan, etc), we can cancel it out to obtain an angle on its own. However, in accordance with the algebraic method, this means that we also have to apply it to the numbers on the right-

hand side of the equation. As a result, our example will play out a little something like this:

$$\arccos(\cos(\Theta)) = \arccos\left(\frac{3}{9}\right)$$

$$\Theta = \arccos\left(\frac{3}{9}\right)$$

$$\Theta = \arccos(0.333)$$

$$\Theta = 70.529$$

Note: In the Windows Calculator, using inverted functions is as easy as selecting the "Inv" checkbox in the upper left of the screen before clicking a function button.

There's our answer! Angle Θ is roughly 70.5 degrees.

By now you should have a fairly good idea of how to calculate both unknown sides and unknown angles in a triangle. Soon you'll be ready to learn how to apply all of this to your games, but first...

Degrees and Radians

Before we begin with game applications, one last tidbit of essential information that will make or break your ability to use trig in your games - are you aware that there are two units of measure for angles? Most people are taught to measure angles in degrees, and for this reason I've been using degrees as the angular unit in all of the calculations so far. Now, while there's technically nothing wrong with degrees as a unit of measure, radians are considered to be more mathematically correct. For this reason they are the primary unit of angular measurement in science and engineering. Luckily, most decent programming languages have trig functions built into them by default. Unluckily, they mostly operate using angles measured in radians. Because of this, you'll need to be able to convert between the two. Radians are not difficult to understand. Degrees are based on one full rotation being measurable in 360 units. Radians are based on one full rotation being measurable in 2π (π = +-3.142; 2π = +-6.283) units. Therefore, conversion is just a matter of looking at the angles proportionally. Don't panic! It's as easy as using these two formulae:

$$\text{Angle in Radians} = \pi * \left(\frac{\text{Angle in Degrees}}{180}\right)$$

$$\text{Angle in Degrees} = 180 * \left(\frac{\text{Angle in Radians}}{\pi}\right)$$

GAME MAKER TIP:

Game Maker saves you from all this mathematical nastiness by compressing all of it into two simple functions: degtorad(angle) and radtodeg(angle). Just be sure that you plug the right measurement into the right formula!

Let the games begin!

At long last! We're going to learn how to use trig in our games! We're going to start with one of the most essential trig applications - accurate angular movement.

Imagine that we have a player character that we want to move at a rate of 2 units per frame. We go ahead and program the game to move the character at that rate when the movement keys are pressed: move +/- 2 pixels vertically when the up/down keys are pressed, and +/- 2 pixels horizontally when the left/right keys are pressed. When we press vertical and horizontal keys together, the player simultaneously moves 2 pixels vertically and 2 pixels horizontally. Diagonal movement is sorted, right? Well, you would think so, but you'd be wrong. If you implemented your movement that way (and many people do), you'd probably notice that something was a little off... To elaborate, let's take a look at your player movement in terms of a triangular arrangement.

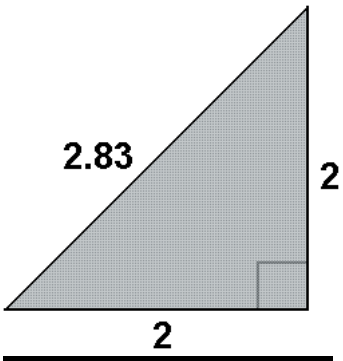


Figure 2: Pythagoras > j00

See the hypotenuse? That represents your diagonal movement rate. When we apply the theorem of Pythagoras, we get the amount that you see in the diagram. No, neither the numbers nor your eyes deceive you - moving diagonally with this method, the character is travelling nearly 50% faster than it would normally! Definitely not too good for our gameplay. And what if we didn't just want to travel in cardinal directions? What if we wanted to move at any angle? We'd have to find a way to map hundreds of key combinations with different horizontal and vertical movement rates, which is hardly practical.

What can save us in our time of need? Trig to the rescue! Firstly, we'll need to reconsider how our movement system works, since we know that this one is fundamentally broken. Well, we know that our speed is a constant 2 pixels/frame. We also know what our eight angles of movement would be (0, 45, 90, 135, 180, etc. degrees). This situation can be represented by this diagram:

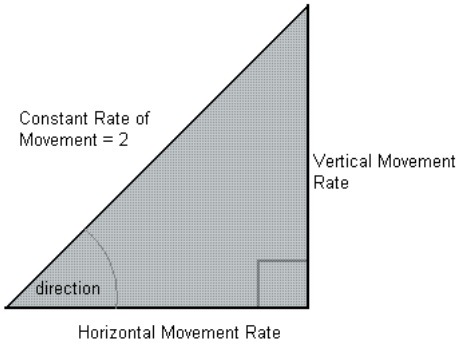


Figure 3: Reworked movement system

Taking the above arrangement into account and applying our trig theorems, we can calculate accurate horizontal and vertical displacements for that angle. You'll notice that when converting trig statements into algorithms we forego all the algebraic number juggling and only use the final arrangement of numbers and variables for our calculations. As a result, our algorithm will look a little something like this:

Horizontal Displacement
= speed * cos (direction of movement)

Vertical Displacement
= speed * sin (direction of movement)

All you need to do is alter the direction (in radians, remember!) and speed of movement, and these two algorithms will happily give you the correct movement rate/displacement for each of the two axes. As you can see, this means that you can move an object accurately in any direction at any speed. This is especially useful for elements such as bullets or other projectiles which you want to fly in all directions!

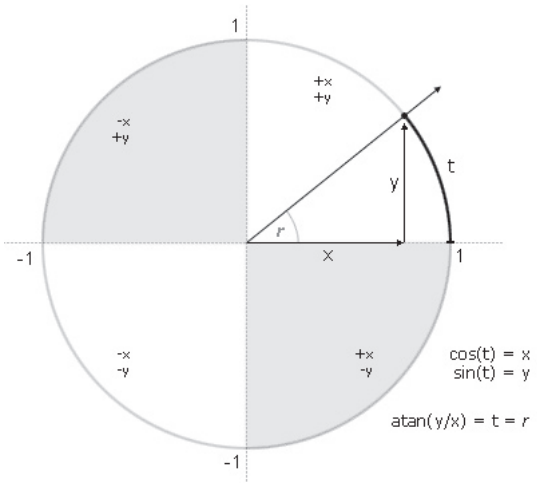
GAME MAKER TIP:

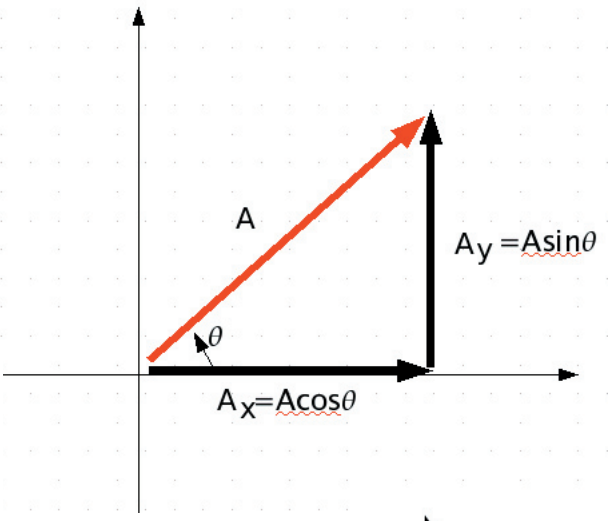
Although the above is very easily implemented via GML scripts, Game Maker has built-in speed and direction variables in every object you create. All you need to do is assign number to these, and GM automatically does all the calculations and movement for you! Hax!

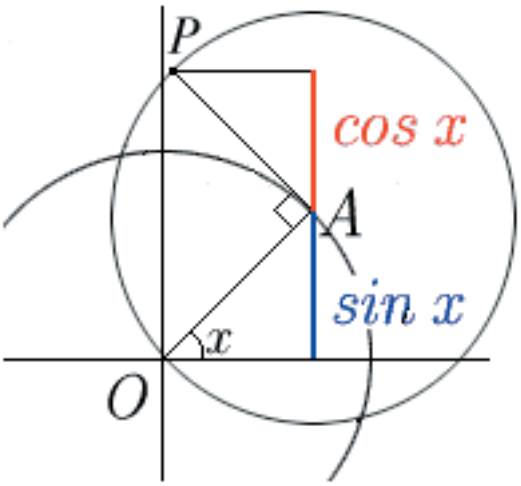
That's all folks!

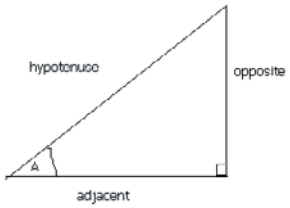
Yes, we're done - for this month, at least. Hopefully the little taste of applied trig has made you hungry for more, and the fun doesn't stop here! I still have a few trig tricks to show you: tricks such as relative angle calculation, circular/elliptical movement and positioning and wave/helix movement. Until next issue, happy coding!

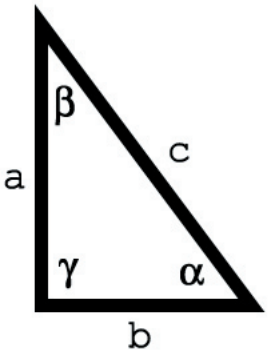
GAZZA_N











THE HISTORY OF I-IMAGINE

Part 6: Xbox, Lies, and Videogames

(Who? What? Where? If you're lost concerning I-Imagine, check out Dev.Mag Issue 10 for the start of this article series!)

After returning from E3 and before we started to get geared up for ECTS, there were a couple of issues that had to be taken care of. The first was to send our current demo of Chase to Infogrames for evaluation, which we did as soon as we all got back to work after our short lived holidays. The second was to decide whether or not we wanted to develop a game that Ubisoft had approached us about developing while we were at E3*.

Ubisoft had the Batman license and were developing a driving game around it, where various Batman villains would trigger crimes in a city and the player (as Batman) would have to rush off to apprehend the villains before they could escape. The game was in an incomplete stage as the previous developers of the project had either gone bankrupt, or Ubisoft had just decided to take the development of the game out of their hands. However, they were not looking to spend a lot of money on having the title completed as they had already given a lot to the initial developers of the game. Unfortunately, at the end of the negotiations, the highest amount that they were willing to pay us was about three times lower than the amount that we were willing to do the work for, and we reluctantly had to tell them that we would not be able to do it.

While these talks with Ubisoft were going on, Infogrames was busy evaluating the copy of Chase that we had sent them. Pretty much around the time that we had decided not to do the Batman game for Ubisoft, we received a response from Infogrames, but it was the news that we didn't want to hear. As far as I can remember, they said that while they found our game to be fun and graphically impressive, they weren't able to reconcile themselves with the overall idea that the game put the player into the role of a Hollywood stunt driver. I believe that the exact words that they used were: "The stunt driving angle is disposable.". Oh, the irony...

Regardless of our post-E3 struggles, September was quickly approaching us and we knew that we had to have a killer product to bring to ECTS with us. We had built up some momentum at E3 and a lot of publishers were keen to see what we new stuff we would have to show to them when next we met, and if we didn't impress them at our next round of meetings we would probably never do so. Luckily for us, the demo of our game that we sent with Dan to London at the start of September was easily our best effort to date. We had six fast-paced missions set in our Asian themed city that were not only fun to play, but our game engine was very good looking for the time with lots of nice effects and a busy, bustling city full of vehicles and pedestrians.

Dan had set up meetings with all of the usual suspects and I will be honest when I say that I do not remember the outcomes of any of those meetings except for the one with Infogrames. The reason why I do not remember what happened with the other meetings is most likely because just like at E3, nothing came from them. Another reason is probably down to the fact that something quite interesting happened in one of the meetings that Dan had set up with someone who we hadn't really met with before.

I say "hadn't really met with" because at the previous E3, we had one of the main guys

** I had mentioned in the last article that nothing particularly interesting had happened to us with publishers at E3, however I was reminded afterwards that our dealings with Ubisoft were during E3 2000, and not ECTS 2000 as I had originally thought.*

from Microsoft Games Publishing come by our booth to check out what we were doing. I can't remember the exact reason why he was there, but I think he ended up coming via roundabout means as one of my friends who was working at Nintendo brought his old producer (who was now at Microsoft) by our booth, and then he came back a second time with this other guy. Anyways, they had shown some interest obviously for the one guy to come a second time, but all that I can remember from the main guy's visit were the critical things that he had to say about our game at the time*.

Now that I-Imagine's "history" with Microsoft has been established, you can see that we didn't have much hope in what our meeting with them at ECTS would hold for us. From what I remember, Dan was expecting to meet up with some of the European Microsoft guys however, so we felt as though we weren't just wasting our time trying to impress someone who already didn't like what we had to offer. Now, I probably don't have the following story 100% correct, but I think that I have the gist of it. Basically, when Dan arrived at his meeting with Microsoft, instead of the people whom he thought that he was going to meet, there in front of him sat Kevin Bacchus and Seamus Blackley, the creators of the Xbox game console.

He talked to them about Chase and then when he tried to install it on the PC that they had there, it wouldn't run. Dan was quite puzzled as to why it wouldn't work and naturally, Kevin and Seamus were more likely thinking along the lines of, "Loser!". However, Dan did some tinkering on the PC and found out that it didn't even have DirectX installed on it. Yes, on a PC used for demoing games in the Microsoft meeting room at a major game trade show. Luckily, Dan had a copy of DX7 with with him on another CD, he installed it on the PC, and Chase ran perfectly on it afterwards. Needless the say, the guys were quite impressed with the whole turn of events and they were equally impressed with the demo of the game. Unfortunately, Kevin an Seamus were not the correct people to speak to about publishing our game, but they were interested in helping us to find a publisher and in providing us with Xbox development kits if it was something that we were interested in, and so they told Dan that they would be in touch with us in the next month or so.

The meeting with Infogrames was just as exciting for us. After Chase had failed their evaluation, we had pretty much lost all hope with them, but Dan received an email just prior to ECTS from the guy at Infogrames who had been our main contact there, telling him that he had a hot property that he wanted to discuss with him at the show. So Dan grudgingly set up a meeting with him and was quite surprised when he heard what Inforgrames had to say. It turned out that they were looking at porting Driver 2 from PSX to PC, and they thought that we would be perfect for the job due to the technology that we had already developed. Dan was naturally very intrigued at the idea of this, so when he was asked for permission for Infogrames to send a copy of our new demo to their internal studio Reflections (the creators and developers of the Driver series) in order for them to approve us as potential developers to work on their franchise, he didn't have a moment of hesitation. Perhaps he should have...

So Dan returned from ECTS with a lot of excitement and enthusiasm, and it was only a few days before we started to receive contact from Kevin Bacchus regarding out involvement in the Xbox program. It turns out that he and Seamus had been thinking of starting a development

** One thing in particular that stands out and that we laughed at quite heavily afterwards was that he wasn't happy with the fact that the particles from the booster engine on our desert buggy vehicle would penetrate the other meshes in the world. I mean, what game developer in the world in the year 2000 wasted CPU time on making sure that all particles behaved completely realistically in their environments?*

program for Xbox that would assist independent developers who didn't already have publishing deals to be able to develop for Xbox. We would still have to pay for development kits, but we would be treated as fully fledged developers, even though we didn't have a publisher behind us. Microsoft would also help us to set up meetings with the correct people at publishers and push our game to them for publishing on Xbox.

This program of theirs was to be named the Xbox Incubator Program and we were being invited to be the first members of it. Needless to say, we were very excited at this prospect, as in our hearts we always wanted to develop for a console audience, but were unable to at this time3. Part of me actually has always wondered if we just happened to meet Kevin and Seamus at the correct time, right when they were thinking of launching the Xbox Incubator Program, or whether only after meeting with Dan did they think up the idea of the XIP as a way to help out independent developers like us. Either way, as history went on to record, we were quick to take them up on their offer and officially became not only (to the best of our knowledge) the first console game developer on the African continent, but also the first Xbox Incubator Program members in the entire world.

All of this excitement was offset however by some concerns that Dave, Matt, and myself had over the creative side of I-Imagine after Hoang's departure. We were happy to go on with our roles as programmers, but we felt that design and art direction were basically being left behind. To respond to this, I tried to persuade Dan to contract a friend of mine (Marco Falsitta) who had worked at DigiPen as well as Nintendo to come to I-Imagine for a year in order to train up our new art team and instill his years of experience into what was without a doubt a talented group of individuals, but who were also without question lacking the experience necessary to tackle the scope of developing a video game for the world market on their own. Unfortunately, Dan declined the offer to bring Marco on board, but he did decide to bring a world renown game designer in from America for a week in order to help give the entire team some design direction, as well as to train up the new game designer we had hired.

This American's name was Noah Falstein (http://en.wikipedia.org/wiki/Noah_Falstein), a freelance designer and owner of a company called The Inspiracy (http://www.theinspiracy.com/). He had previously worked for Lucas Arts amongst other companies on some very high profile games and had twenty-plus years of experience as a video game designer. You can actually find quite a few articles that he has written about game design online, along with a series that he did a few years ago for Game Developer magazine (later re-published on Gamasutra). Noah was a great influence on us and taught us a lot about successful game design. His main game design philosophy was that you must always reward players and never punish them, as well as give players as much choice as possible at all times. These ideals were taken heavily to heart in what would eventually become the final gameplay dynamics of Chase: Hollywood Stunt Driver.

While Noah was visiting us in sunny South Africa, a very interesting thing occurred. I was doing my daily gaming news reading on the Internet and came across quite a surprising announcement on the old Games Radar website.

Infogrames had just released a press release detailing the new game that their internal studio Reflections were working on. It was called Stuntman and it revolved around the world of a Hollywood stunt driver doing scenes for various movies. I don't think that I have to tell you not only how shocked, but also how angry we were when we read that news. It essentially boiled down to the fact that for the past five months, we had been in correspondence with Infogrames and they knew everything about our game but we knew nothing about theirs.

Nobody outside of Infogrames and Reflections will ever know the entire story. Personally, I doubt that they stole our idea even though I, along with everyone else at I-Imagine did at the time. More than likely, they were either already in development of their idea when they first heard of Chase, or in the least they already had the idea of doing a game like that and the news that there was going to be competition to their idea spurred them on into making Stuntman quicker than they had originally intended on doing. This meant that either their left hand didn't know what their right hand was doing, or that they blatantly tried to dissuade us from developing a competing product by talking to us about Chase and by having us send it to them for publishing evaluation, even though they had no intention of ever doing so.

To make matters worse, throughout all of this we were still in contact with Infogrames about doing their port of Driver 2 to PC. Although we were upset and angry with the situation that we had gotten ourselves into with them, we were still determined to exploit this opportunity if at all possible. At first they seemed generally serious about us doing the work for them, but slowly as time went by, it became more and more apparent that they were using this situation as just another stall tactic for us so that their stunt driving game would come out before ours would. For instance, they wanted us to create a demo of some of the missions in Driver 2 using our engine and when they asked us about this, they actually inquired as to how that would effect our development of Chase. We just told them that we would hire more people!

However before we got around to doing the actual demo for them, they came back to us with some feedback from inside of their company saying that it wasn't going to be very profitable for them to release Driver 2 on PC. Apparently, the PC version of the original Driver didn't sell very well, and they saw this translating to Driver 2 also. So they came back to us saying that we would have to do the job for what would basically amount to no upfront money and only royalty revenue. We were still very intrigued by this possibility as we desperately wanted to have our name associated with this well known franchise. However, after much consideration, we decided that we wouldn't be able to do a project like this for free as we would more than likely lose too much money on it, so for the good of the company we decided that we would have to turn this opportunity down. All in all, these drawn out negotiations with Inforgrames lasted from September until the end of December while they tried to stall us in our development of Chase.

By the time the year 2000 drew to a close, many interesting things had happened to us as a company. All of it ended up being an invaluable learning process for us, and helped us to grow in our knowledge of the video game industry. As well, we also grew physically in terms of a company as during the year, as many South African's were hired to both expand our team as well as fill in the gaps left by the departure of Felix, Kenny, and Hoang.

Our first local hiring, Kevin, happened fairly early in the year and he filled the critical role of being our primary texture artist. Shortly after hiring Kevin, Brett was hired as a modeler. By the middle of the year, we had also hired Kirk who was also modeler, and Henk who became our primary world builder. We also hired Martin as our game designer to help fill the design void left first by Kenny, then by Hoang. The end of they year saw us add two more to the art team with Dave joining as a conceptual and texture artist, and Jeanine coming on as a conceptual artist. We also made our first addition to the programming team with Derek who was brought in to work on our tools. By the end of 2000, the entire I-Imagine team consisted of 12 people, 9 of whom were South African's. This meant that our transition from being a company in South African staffed by almost entirely overseas individuals to being one that employed primarily South African's took just over a year to complete. These would be the people who would help lead I-Imagine in the next year and a half towards our goal of completing Chase and having it published for Xbox internationally.

COOLHAND

THE CHASE TEAM:
Top Row: (Dave, Matt, Luke, Henk, Kevin, Derek)
Bottom Row: (Kirk, Martin, Dave, Dan, Brett, Jeanine)

STINK PRODUKSIES

All about small fish in a very big pond...

Stefan van der Vyver (known to most of us as ?rman) is an external writer for Dev. Mag who happens to be a bigwig in a local multimedia production company known as STINK Produksies. Just about every month, we get stuff from him aside from his regular Blender articles which showcase what the STINK guys have been most recently getting busy with. This month, we decided to ask him more about his involvement with it all, along with some info about the company itself. He swung us this ...

At the risk of sounding boring, the following information shows my progress from employee to business owner. Read on if you are at the point where you might want some real world information...

The business of earning a living, as a biblical principle, is a challenge.

I would venture to say that, for most of us, that challenge is represented by working for a living, and following our passion after hours. For me, it has always been a challenge to be free of that setup.

I had a plan, which was this:

1. Work a day job to pay the rent.
2. Work at night to do work for clients.
3. Minimise the risk of earning my own salary by building up a broad client base.
4. Spot the right time to make the transition.

It took me five years of part time 3D animation and music production to reach the point where I could throw my weight into STINK Produksies full-time. Boy, what a change! Gone was the constant stress-related heartburn, to be replaced by lots of open working hours for me to pursue the work that I really want to do.

For me the break came in October 1996, when a company for which I produced a range of children's CDs sold a large number of these to the big music retailers. That gave me the financial push to make the transition from employee to business owner. I am immensely glad that I entered into that agreement, and I am extremely grateful to them.

I decided long before going full-time that animation was my business, but that I do not possess all the skills necessary for completing big productions. My work experience, professional training and days of playing in a band have taught me that one must manage skills to reach a goal. That means that that I must be open to the knowledge that I need to buy skills from people around me, and be willing to really pay for them.

One thing that I see time and again, is that people are too selfish, or too scared to work in a team. We have all these 'independents' producing various products all around us. We probably all know a video guy, photographer, web designer or 3D animator who is trying to make it on their own. Yet, how many finally make it?

By building up a broad client base, I have built up many contacts. This puts me in a position where I can work on productions that reach far beyond my own capabilities. It has also sped up the process of production - when my part of the process is finished, I pass the work on to the next business, who take the product to the next level. Through all this I have worked through the fear of letting go of my work, allowing others to influence it positively, enhancing its capabilities.

One very important factor of running my business, is that I made a conscious decision to finalize productions. I am sure that everyone has, at some point in time, attempted something, only to lose focus and not complete what was started. For obvious reasons, this principle

can not be disregarded by any business. I decided to call it the "Wooden block principle":

Imagine that there is a conference for knowledgeable folk. Consider that the need may arise for a small wooden block.

Now, being a bunch of knowledgeable folk, there might arise some serious discussion regarding the exact nature of such a wooden block. There might be discussions regarding its size, shape, weight, etc. Being extremely knowledgeable, everyone wants to air their opinion regarding this wooden block. Consequently, there is a lot of discussion around the subject, with immensely wise statements being made.

Now imagine that the caretaker of the building where the conference is being held walks past. In ignorance, he hears that there is a requirement for a wooden block. Good ole' soul that he is, he walks out to his work room, cuts a piece of wood off the end of a beam, and presents it to the person who originally inquired regarding a wooden block.

Astonished, the knowledgeable folk are suddenly left behind, because the caretaker has produced what they could only talk about. HE finished a task. Sure, he could be asked to refine it, but he was there first. For me, that forms the basis of my business. I want to finish the projects that I start, enabling me to comfortably move on to the next project. This is no easy task, and you might want to consider this very thoughtfully - business owners are usually quite sharp, and have probably met many personality types. Your presentation and production history will speak for itself. Finished products will land you more work than many great ideas. We can all generate ideas, but few can follow through and complete them.

Thus it seems that, from the STINK Produksies side, there are two main points up for consideration: Don't be scared to work as a team. It will enhance the final product. Finish what you start. This becomes your track record.

Chatting to one of my business friends, she mentioned that many times we lose faith in our efforts, or the quality of our work. We feel that we can do it better, or that someone else is doing it better. This is the point where many people will give up. Should you want to make a mark for yourself, follow through on what you started, thereby earning the respect of people who have done so themselves, and who are now running successful businesses.

The computer game industry lends itself brilliantly to this principle. How many unfinished games are there in your closet, and how many of your games are being played across the world?

Starting out your business you might find yourself just a small fish in a big pond, but you'll be a fish. And you might become part of a bigger fish if you can find the right team.

?rman

Stink Produksies is a multimedia production company which has been operating for four years as a close corporation. Productions include corporate logos done in association with video production companies, television ads, television and radio voice overs, corporate and private original musical compositions, and children's audio CDs available in the major CD stores nationwide. Stink Produksies works mainly in association with other production companies, as it fulfills a niche market requirement for original composed musical scores and 3D design.

Check it out at www.stinkmedia.co.za