SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

MARCH 2007

DEV.MAG

CREATE • DEVELOP • EXPERIENCE

MINI#37

REGULARS

FEATURE

SPOTLIGHT

OPINION

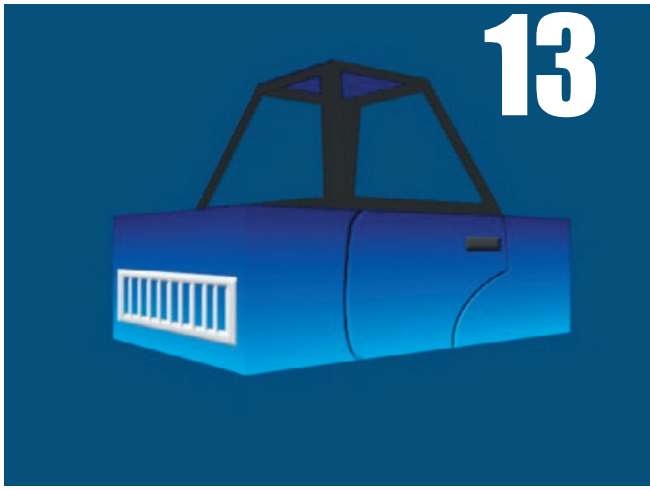
DESIGN

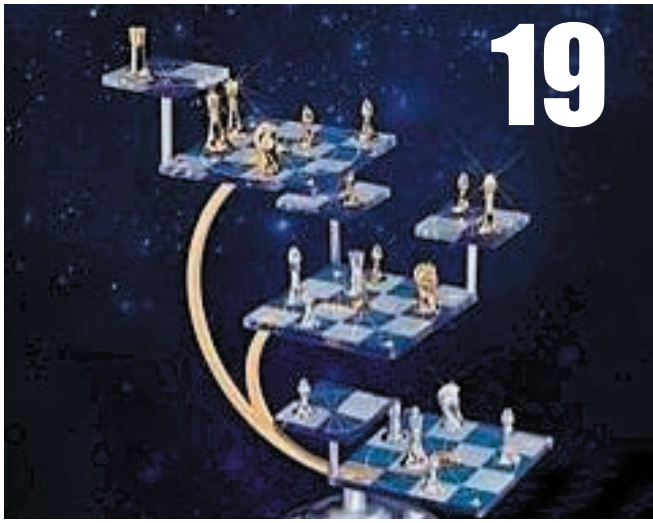
PROJECTS

TECH

HISTORY

TAIL PIECE





This month has been quite a neato one indeed, Yes, it still seems that our release date has been firmly planted at being towards the middle / end of every month (and we may just give up and keep it that way, if we feel lazy), but that doesn't mean for a moment that exciting developments aren't being made before our very eyes and keyboards.

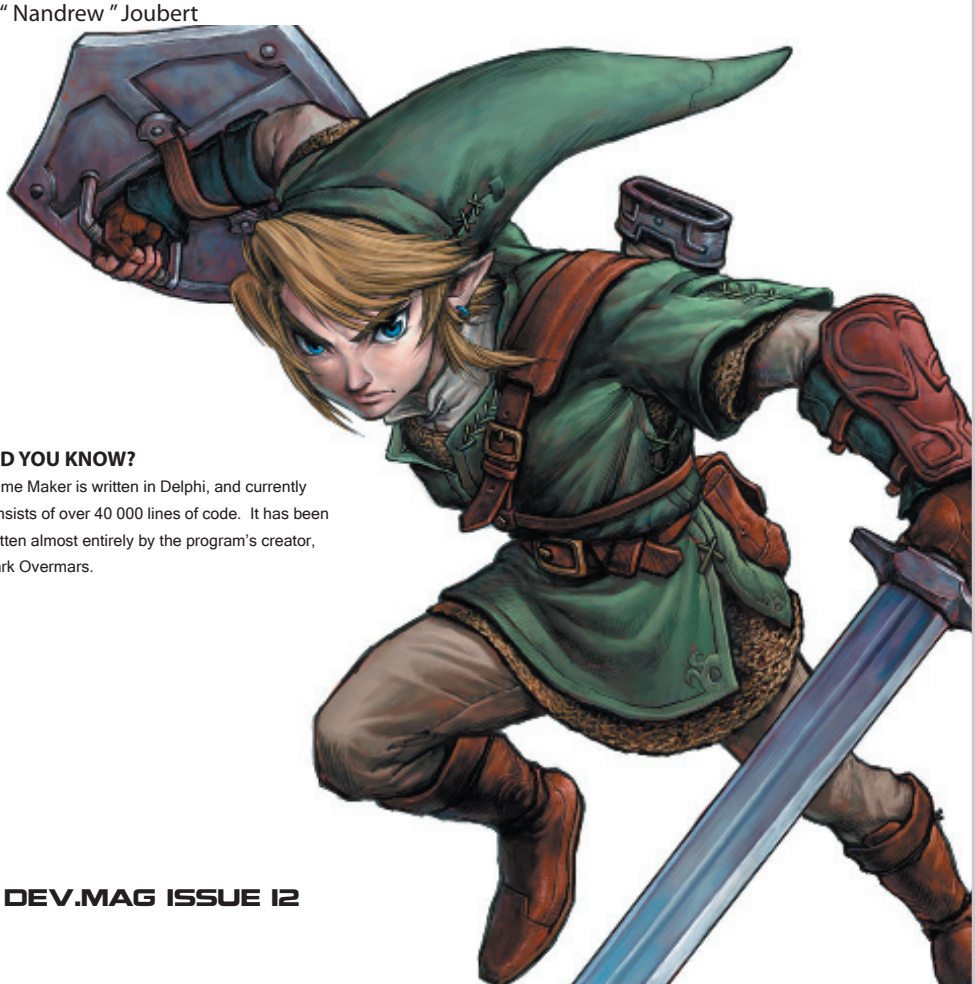
First off, the game development Hotlabs really seem to be kicking off due to some favourable press coverage and excellent marketing, which is great news for us and even better news for enthusiastic game developers out there who want to network with like-minded people. As usual, we have a Hotlab report hiding about in our pages (hint: look at this month's tailpiece) so that interested parties can see what it's all about and maybe get in on what they've been missing all this time! We might have slotted the labs into our feature column (as we thoroughly enjoy doing), but this month we're looking at local animation-studio-turned-game-developers Luma and chatting to them about their top notch, freely downloadable racing game, Mini #37. These guys are top-notch and have the resources to boot, not to mention several experienced I-Imagine developers working on their side!

Secondly, there's been quite a bit of interesting goings-on this month regarding content of the past, present and future. For the past, Dev.Mag is (unofficially) making its first payments to three of its writers -- Tr00jg, Cairnswm and Chippit -- to the amount of R 300 each for a top secret internal article writing competition we held an issue or so back. Hopefully we'll hold more of these competitions in the future, open to all regular Dev.Mag columnists, so any pros out there who know a thing or two about a particular game development field or aspect -- well, take the hint and scribble some stuff for us every month.

Of course, if you're still an eager little writer but don't really think you know enough about game development, then you'll be happy to know that as far as "present" is concerned, we've opened up our new Opinions section for the magazine, where people rant, rave or ponder about stuff in the game development world. The main thing about opinions is that they're the one part of the mag that's open to not only staff, but guest writers as well -- take a look at Garson's opinion column for more details on potentially submitting your own masterpiece!

For future, there's two things I'll mention -- first off, we've reached a very happy little understanding with pascalgamedevelopment.com, whereby we'll be exchanging ideas, advertising and lots of happy good times, starting with a full-blown interview / feature on them in next month's mag. My other important (and exciting) announcement is ... oh bugger. I've run out of space for my editorial. Maybe next month.

Editor
Rodain " Nandrew " Joubert



DID YOU KNOW?
Game Maker is written in Delphi, and currently consists of over 40 000 lines of code. It has been written almost entirely by the program's creator, Mark Overmars.

DEVMAG
CREATE DEVELOP EXPERIENCE

EDITOR
Rodain " Nandrew " Joubert

DEPUTY EDITOR
Claudio " Chippit " de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNER
Brandon " Cyberninja " Rajkumar

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

CARTOONIST
Paul " Higushi " Myburgh

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

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: **forums.tidemedia.co.za**.

OGDC website interviews

http://www.ogdc2007.com/index.html

Some of you may have heard of the Online Game Development Conference, but if the cost involved in attending the conference itself is a little too steep for you, then there's always the handy resources available on the conference's website to see you through. Of particular interest is the list of interviews that dots their news posts -- always great for people who like to hear from the big dogs in the industry. Interviewees come from the list of speakers attending the conference, and cover a wide variety of topics. Worth a look.

MUD co-creator gives keynote address

http://www.gamasutra.com/php-bin/news_index.php?story=13550

Self-confessed "virtual world dinosaur" Dr. Richard Bartle recently gave his view on game development at the Indie MMO Game Developer's Conference in Minneapolis. Bartle is credited with the co-creation of MUD, perhaps the single most popular massively multiplayer online game to hit the Internet before the advent of more modern MMOs such as Everquest and World of Warcraft. In his keynote address, Bartle stressed the importance of imagination over orthodoxy, believing that indie MMO developers still need to put more emphasis on innovative ideas. Read more in the Gamasutra report.

Gaming on Linux? Huh?

http://www.gamedev.net/community/forums/topic.asp?topic_id=443228

That's right. The Cedega tool (reported by gamedev.net to have just come out in version 6) promises to allow games originally intended for the PC to run on Linux "out of the box". The list of currently supported titles includes some pretty impressive names, including Elder Scrolls: Oblivion and Battlefield 2142, which brings up the question: will Linux soon become a more appropriate system for indie game development than it is now? Will Python game developers soon be receiving more company in this typically game-devoid zone?

Spacewar starter kit for XNA

http://msdn.microsoft.com/msdnmag/issues/07/05/XNA/default.aspx?loc=en

Still not sure about breaking into the XNA system? Maybe this handy little resource from Microsoft will help you decide. It's an article which outlines the introduction to XNA, a walkthrough of the basic interface and code structure as well as a full code resource for the sample spacewar starter game. The article is highly comprehensive and should be a good start for anybody who has a bit of programming experience already and is eager to start learning about XNA. The basic XNA system (for developing Windows, but not XBox, titles) is freely available for download at msdn2.microsoft.com/xna.

luma
create. lead. inspire.

LET THERE BE LUMA!

So this month, Dev.Mag is touting Luma. Why? Well, because they're cool, they're South African, and they make games. Enough said yet? Well, maybe not, otherwise we wouldn't have needed to run a full feature on it. Luma has already been kind enough to co-operate with the Game. Dev organisation in numerous ventures, and when we approached them to fill a few of our pages in Dev.Mag, they were happy to oblige. Here, straight from Luma's mouth, is an idea of what the company is all about. Turn a few pages and you'll also find our interview with Dale Best, one of the founders of Luma. You don't want to miss out on any of this ...

"Luma started out in 2001 as a digital studio specialising in 2D and 3D animation. Way back then, (which is a lifetime in this field), we found ourselves in a relatively new industry (with few competitors) and the cost of technology a little less affordable than it is now. It was a very nice place to be, if you had your businessman's cap on. The company was started and funded by creative people, I believe that is what kept us evolving over the next six and a half years into the different animal that is Luma today.

However, we now find ourselves in a much more competitive environment, where the handful of studios that were around at that time now number a few dozen; from larger studios like ourselves right down to one man shows with little or no overheads. The advantage of being a larger outfit has luckily allowed us to experiment with new ideas and ventures. That combined with the fact that we do not have any outside investment, means that we can allocate resources to projects that might not look great on paper, but instead rely on gut feelings.

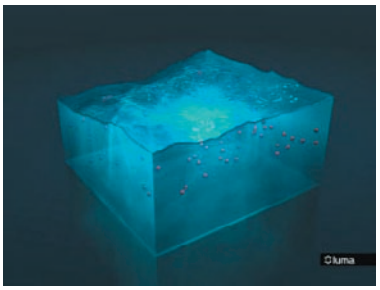
Creating content and developing technical ability became an underlying intent for us in 2005, as our role as a post production company seemed limited in terms of the future of the type of business we were in, not to mention the type of work that we ultimately wanted to be doing. Basically, we believe that by creating our own content and reinventing ourselves constantly, we will maintain an edge in terms of who we are and what we do. Alternative sets and types of media has always had a strong focus in terms of our development, and there aren't any better ways to create entertaining ways of interacting in the digital realm than by developing games.

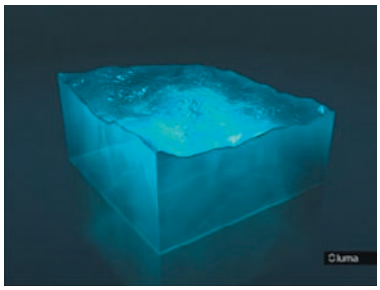
So, drawing from our existing client base, we built a prototype game concept using as little resources as possible. We used this game as a tool to show our clients a more exciting, further reaching, and much more immersive way of communicating and interacting with people than the brain dead spoon fed 30 seconds that irritate us while we try to watch Prison Break.

That said, our intent wasn't to simply rehash the superficial existing model of marketing after-thoughts slapped in the "cool" section of some bland website in the form of flash and shockwave "games". Our goal was to make real games that people actually wanted to play, and wouldn't mind actually spending money on. Only this time round, the consumer isn't paying to drive a (insert brand name here) licensed product, but the (insert brand name company here) is paying for the consumer to be able to play the game."

To see what our local neighbourhood Luma people are getting up to, visit http://www.luma.co.za/

For those eager to learn more about their main gaming project, Mini#37, additional info and downloads can be found at http://www.mini37.co.za/





SPEAKING WITH DALE BEST

Co-founder of animation and design studio Luma

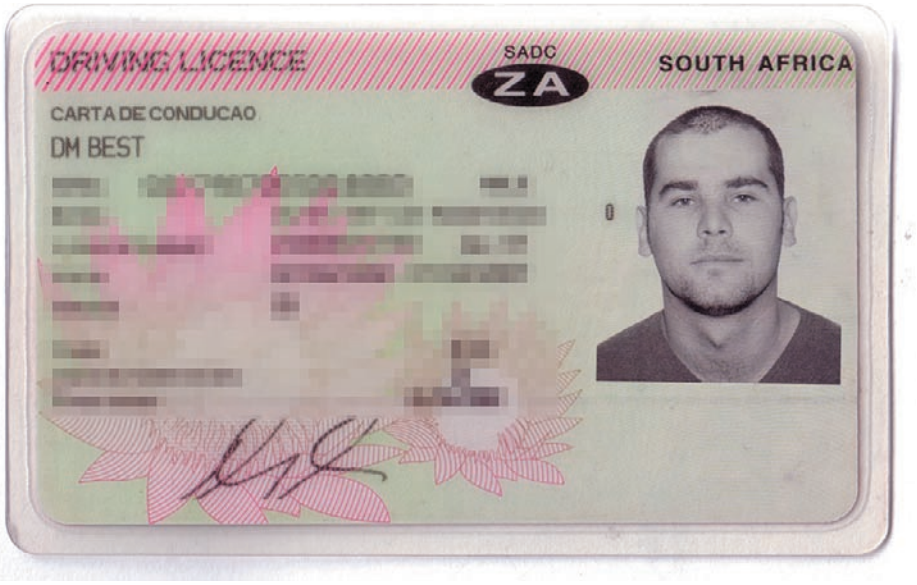
What made you guys decide to work on games?

Quite simply, it has been something that we have always wanted to do. There have been a lot of comparisons made in the media regarding games and movies, and being in the post production and animation industry we identified some of those synergies ourselves, and figured: “Hey, were almost halfway there, why not try it?”.

What game-related projects are Luma currently working on?

We are currently developing an arcade racer called MINI#37 (http://www.mini37.co.za). You can download the trailer here: (http://www.mini.co.za/MINI37/images/MINI37.zip).

MINI#37 is an episodic racing game and every couple of months or so, new sections of the game will be released. The first public release (Episode One) will be available in May of this year. It will include tracks set in Newtown (which is a part of downtown Johannesburg) as well as Cape Town
(tracks include Camps Bay, the V&A Waterfront, and the dock area). Each city/location will consist of three tracks, and new cars will also be available as downloadable updates or as part of the next release. All of the MINI vehicles will eventually be available, including vintage MINIs and some hotted up MINIs



Dale really likes racing games. That, or he simply didn’t have any other pictures of himself to send to us. We’re going with the former.

like the GP and JCW. There will be two game modes: Arcade Mode will feature single races on any of the unlocked tracks, and Championship Mode will require players to compete through a series of tracks in Grand Prix style racing. Both game modes will be playable as single player or multiplayer experiences.

We are developing this project for MINI South Africa, and if there are enough registered users and enough interest in the project, it will continue to grow. By registering, you will have access to freely download the game, get updates, eventually be able to post your best times and suchlike.

What sort of tools do you use?

Art tools:
-Photoshop, 3D Studio MAX 7, DelEd, After Effects, Lightwave, Stitcher.
Programming tools:
-Visual Studio C++ 2005 Express Edition, Torsion, Tortoise SVN.
In terms of game engine, we are using a customised version of TGE 1.5 (http://www.garagegames.com).

The soundtracks and sound design are being done by Tripwire (Lead singer of the band Pestroy (http://www.pestroy.co.za)), and Paul

Norwood from 16Stitch (http://www.16stitch.co.za). The CG intro trailer for the game was done in house at Luma by a team of 3 artists/animators.

The core team of guys working on this project at the moment consists of 3 artists and 2 programmers. The artists are: Dale Best (one of the founders of Luma), Dave Baxter (former I-Imagine Lead Artist) and Chris Cunnington, while the programmers are: Luke Lamothe (former Technical Director of I-Imagine) and Herman Tulleken.

What sort of obstacles have you encountered in developing your games?

The first obstacle was obviously experience. Unfortunately, South Africa has a very short history of game development on a professional level. However, we were fortunate enough to have two former I-Imagine guys join our team. The second obstacle for our current project was budget, and as a result, we had to find an engine that fitted into it. As well, as far as the bigger picture went, we wanted to be able to build tools and technology that were reusable, without having to worry about big licensing contracts and on-going usage fees, which is applicable to the more expensive and current bigger budget engines.

The first three months of the project were dedicated to not only having an Alpha version of the game done and ready, but also trying to get an art asset pipeline working, not to mention fixing bugs and adding more functionality to Torque. Also, designing our game around the current limitations of Torque (which was really designed to for use with indoor or terrain based First Person Shooters) has proven not to be so much a difficulty, but more of a challenge in order to create a game that is not obviously "lacking" in any areas. In other words, designing it so that it is not compromised as such, but just approached from a different way.

What would be considered Luma's biggest game development-related achievement to date?

I think the biggest achievement was our first deadline earlier this year in which we delivered a fully functional multiplayer Alpha version of MINI#37 to complement the launch of the new MINI. It is a muliplayer rig (up to 4 seats), which will go around the country during the months of March and April. Literally hundreds of people will have played the game almost continuously over that period of time. What I think is especially interesting about this is that the game has become a social event, as opposed to a insular experience between one person and their PC. The company responsible for running the events that are traveling the country have since told us that the game has been a real ice breaker, and that complete strangers would sit next to each other, and as a result of playing the game, would start talking and so on.

Do you believe that there's adequate room for "big-scale" game development in South Africa?

I guess it depends on what your definition of "big-scale" is! If you mean 30-man teams with 5 million dollar budgets, then I would have to say no (unless the day comes when Ubi-Soft or EA decides to open up a studio here...). But I think that the success shown by I-Imagine in the past of releasing games on 3 separate gaming systems (Xbox, PS2, PSP) has shown that there is both the talent and dedication in South Africa to achieve relatively large-scale game development dreams. The sticking point is that the bigger your development is, the more wide-spread your target audience needs to be, and going down the route of world-wide publishing/distribution means not only having to worry about making a great game, but also means having to worry about the business side of things quite considerably as well.

What can we immediately look forward to from Luma?

We will launch Episode 1 of MINI#37 in May this year on the MINI#37 website. It will be yours to download and distribute freely.

Any major plans for future titles?

There are a couple of new and interesting projects on the horizon. Our aim is to continue to develop games, and it doesn't matter what platform or how elaborate they will be. As we develop and grow, the core thread of making playable and enjoyable games will be our primary focus. The bigger picture which stretches into the next five years will also include our own titles. How these will be funded, or how they will be published/distributed will no doubt be dictated by the times, and will most probably not conform to the mainstream model (especially if history finds itself repeating).

What's your advice for indie developers?

The best advice that I can give is also the simplest (not to mention the most cliched): "Keep on following your dreams." One of Luma's dreams was to be able to get into a situation which allowed us to develop games in a financially viable way. We have now reached that and intend to continue down this path into the future.

NANDREW

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

Favourite colour?

Depends on which day of the week it is. today it's ... BLUE.

Do you drive a mini?

Yes, superchargers are cool.

What's your favourite game genre?

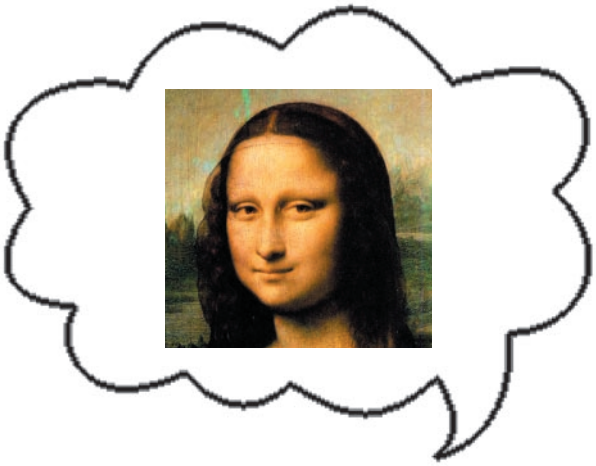
Simple old school arcade games like Space Invaders (on my old Atari), Puzzle games like Tetris (on my Gameboy) and Lumines (on my PSP), Racing games (on just about anything).

Who's the best gamer at the Luma office?

On crazy Japanese side scrolling shooters - definitely Dave (is that politically incorrect?), and I guess he's better than anyone at everything else too.

Is he a gracious winner?

You hear "TAKE IT! TAKE IT!", as he crushes you, controller in hand. Not sure, is that gracious?



"Are games art?"

Personally, I don't really know who cares anyway. But as it seems to be a pretty popular question at the moment, I thought I'd also give my opinion.

I suppose the real answer comes down to how you define art. The word "Art" to me, in its current use, is effectively an abbreviation for the term "Fine Art" - you know, those paintings on the wall or the novel you read last week. Fine Art is basically an expression of something the artist feels, putting it into a medium that others can experience. The art item itself tells you something that the artist was thinking about.

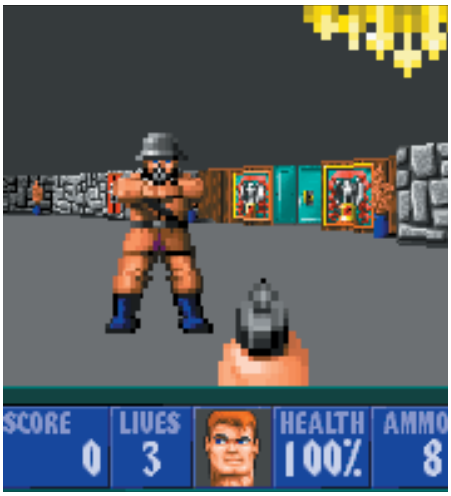
To me, using this Fine Art definition you cannot call games art. Fine Art tells you what the artist was thinking, meaning you don't influence it, whereas games are interactive and the choices and actions made influence the final outcome of the game. As you are changing the result, the game itself no longer conveys what the 'artist' wanted to convey.

While a game certainly consists of Artistic elements (those sprites, background and music scores) the composite result is not Art.

Is even an art studio truly Art? Doesn't an art studio have exactly the same components as a game? It is designed, built from a collection of art assets and allows the person experiencing it to choose how they will experience it.

So to me, as a game designer and developer, I am not an artist, I am closer to being an engineer. Just as a building or a motor vehicle is built, so I build games. I do not paint, compose or author a game, but rather I develop a game. After all, isn't that why we call ourselves GAME.DEV? Somehow, GAME.ART just doesn't seem to be the same thing.

CAIRNSWM



"TOUCHING HEARTS, CHANGING MINDS"

During development, I sometimes get lost in a tangent of distractions, the biggest being, surfing the web. While on an Internet surfing expedition, I stumbled upon a couple who happily got married… This might seem insignificant, but this couple met through World of Warcraft.

It might not seem odd that something like this could happen. An MMOG (Massively Multiplayer Online Game, for the gaming-impaired) is after all a huge mass of communication and like-minded people would be found questing through its world. To the average Joe, this is just a fairy-tale story, where two strangers ended up living happily ever after, but to us, as developers, it is something much more rewarding!

Without World of Warcraft, these two people would not have met and would most likely have gone their separate ways, meeting someone else, hopefully marrying and have an average of 1.2 kids.

As a developer, this just makes my hobby and life so much better. Most games these days have a social aspect to them. I "create" something and through it, people meet and their lives change for the better (or for worse). A seemingly innocent concept of just a "game" is turned into something greater, almost epic: a ripple in the primordial pond of life.

Please tell me that isn't something wonderful. To have that feeling that your creation is changing people's lives.

This, of course, also does not only happen with World of Warcraft. Other creations, like instant messengers, have brought people together who would not have met each other any other way.

I started thinking if any of my games (which are all single-player) could have changed people's lives. Perhaps someone became enthralled in a bout of Roach Toaster, and then perhaps decided to leave 5 minutes later and continue with his life… What if he did not play Roach Toaster? He could possibly have been in a car accident and with him killed the next "Google"-guys or Albert Einstein!

I know I am exaggerating, but even on a smaller scale, it is just as satisfying. Just think that someone played and hopefully enjoyed your game in simplistic bliss.

There is no sure way I could know… All I know is that somewhere, out there, I have made a difference to the well-being of some individual, just like me and you - and this through the best entertainment of them all: games. That thought is definitely something that makes game development oh-so-worth-it in the end, that proverbial cherry on the cake.

Don't you as a developer feel the same? If you haven't delved into game development, now is the time.

TROOJG

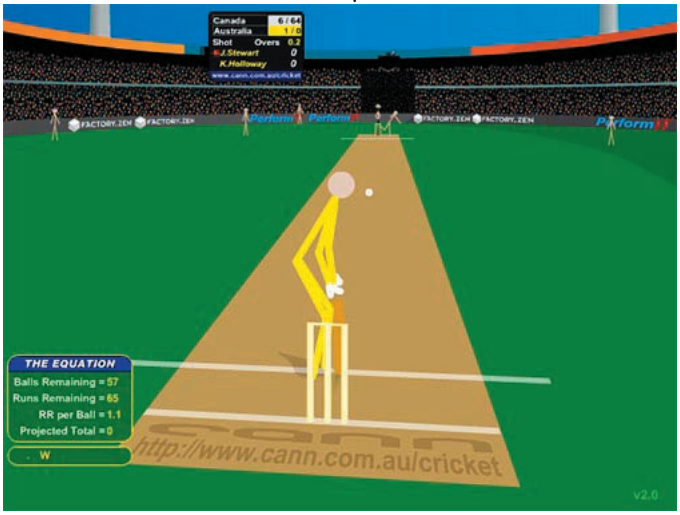
"FOLLOWING THEM INTO THE FIRE?"

Here I am, 1am in the morning, bored out of my mind, thinking what I can actually write about for a magazine tilted towards game development. I'm thinking to myself "Game development?" Yeah as weird as it sounds I probably have no interest in game development, or any forms of programming whatsoever. It might be misleading that I take Computer Sciences well into my second year of study, but I have as much faith in passing it, not due to lack of raw ability, but interest, as I have in the world ending tomorrow due to a killer bunny. Why am I writing this then? Support -- it's really that simple.

If you've been watching the television or reading the newspaper you'd notice that the cricket world cup is currently going on, a few supporters flying over from South Africa to support their home team. It's not so much a case of patriotism, but just willingness to support. You'd also have noticed the big problems they are facing over in the Caribbean as no local support is pitching up. Why is this you might ask? High asking ticket prices. Thing is, Dev. Mag is completely free, it's just 6mb to download no matter where in the world you are located. It really becomes a matter of "why not support it"? There just isn't any valid reason.

I've looked through most of the issues, reading a bit here and there, whatever caught my eye. What really amazes me is the amount of effort these guys put into it, not just the magazine itself but the bigger Game.Dev too - competitions, promotional stands etc. I don't really get why these guys have so much to give for game development; to throw around ideas and implement them into something simple. Maybe that's the whole thing, being able to implement an idea into something small, an idea no high-profile game development studio would pick up on. Maybe it's the thrill of success in seeing the number of downloads rising on something you designed. Whatever it is, someone will hit a gold-mine and be picked up by some studio or another, and just that is enough to support them for, for the possibility of playing something completely new in 5 years time. Of course, in the meantime we can enjoy a bunch of little games created by fellow gamers.

It's just a matter of support and this magazine, this -free- magazine, can really become something important in the game development sector, promoting it and stretching out to new people, with ideas and talent. Just spread the word and support. I'm giving it my support, not because I am in any way interested in game development, but because I love games.



GARSON

Ed's note:

Garson is a member of the NAG forums, which itself serves as a host to the Game.Dev forums. We didn't pay him to say any of this stuff about us. Well, not much at any rate.

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

BLENDER INTERMEDIATE TUTORIAL: Simple Modelling and UV-Texturing

Now that we've gone through most of the Blender basics with you, Dev.Mag will be featuring a more advanced set of graphics tutorials from Stefan "?rman" van der Vyver, a local trainer for 3D applications who runs an animation club in Cape Town. If you are not experienced with Blender and would like to know how to start creating your own 3D models, have a look at our issue archive on the Dev.Mag website (www.devmag.org.za) for the earlier tutorial series, then return here to pick up where the last set signs off.

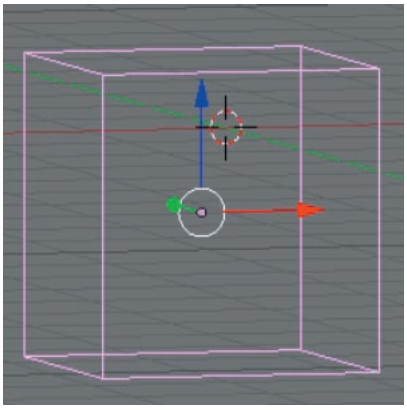
This tutorial makes use of Blender version 2.43. It is significant for the reason that the buttons and functions may be located in other places with earlier versions of Blender.

This tutorial will attempt to show you how to use UV-mapping to apply a texture to a basic mesh. If you don't understand the terms used, please read on. Better to read on and learn, than get scared and turn away!

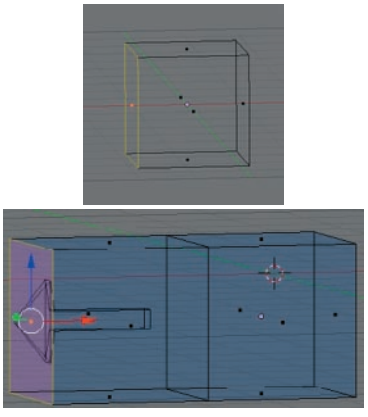
One Blender 3D convention is that one refers to keys as AKEY, meaning that YOU NEED TO PRESS THE "A" key on your keyboard. As another example of notation, an indication to press Ctrl-AKEY means that you have to hold "Ctrl" and press the AKEY at the same time. RMB is a short term used for right mouse button.

TO BEGIN:

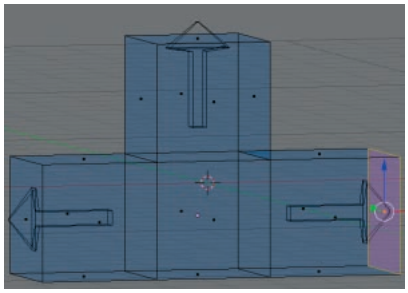
Start up Blender, select the default cube RMB and go into edit mode TABKEY.



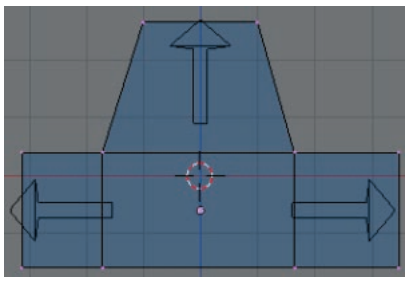
What we want to achieve, is to pull out the back, front, and roof of the cube. Select one of the side faces RMB and extrude with EKEY.



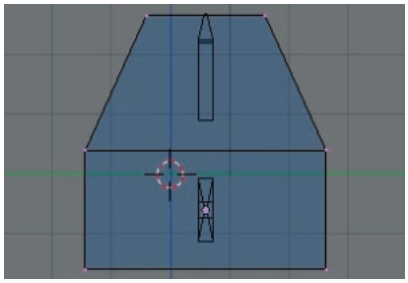
Carry on extruding the opposite face as well as the top face. Eventually, you should see something like the figure below.



Looking at your model from the side, move the vertices so that it resembles the following image.



Then look at your model from the front, and work on it until it resembles something like this:



You can exit edit mode now with TAB-KEY, and we now have an extremely basic shape that can pass for a car body. Great. Now comes the texturing part.

What on earth is UV-texturing?

UV-texturing allows you to make pictures outside the 3D program, and then stick those pictures onto a 3D model.

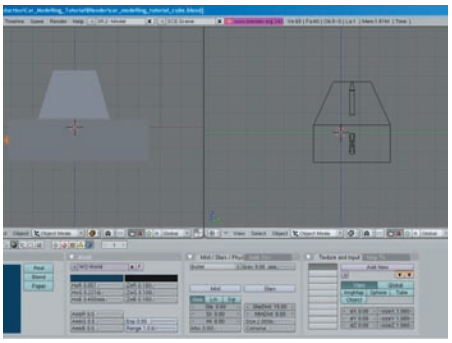
Initially, this is a new thinking concept. Give your brain time to adjust to the idea! I'll attempt to explain how to do it with one sentence:

You take parts of the model and place it on a picture, so the parts of the picture know where they should go and stick to the model. Two questions commonly arise from this:

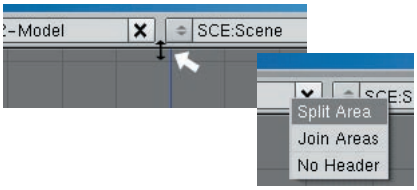
1) How on earth do you "take parts of the model"?
Well, there are two screen modes that allow you to do that, so don't worry.

2) How do the parts of the picture know where to go and stick to the model?
This part happens automatically. Don't worry.

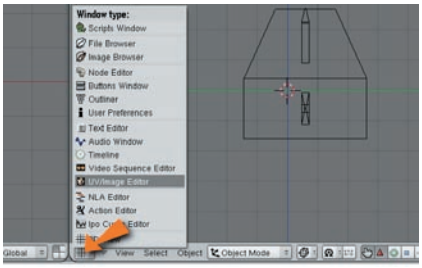
First step: Set up your Blender screen. Divide your main 3D screen in two, with a "Buttons"menu at the bottom.



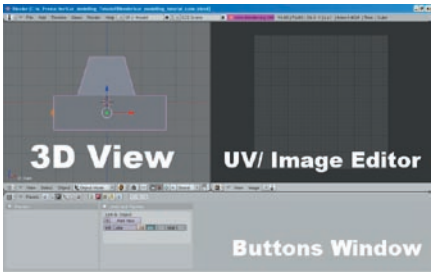
Don't know how to split your screen? Don't worry. Blender makes it simple, which is one of the nice features associated with the program. Hold your mouse cursor over the border line under the "File" menu. A double arrow appears. RMB and select " Split Screen" from the menu. Now simply position the split where you want it. You can use this function to split the screen vertically as well as horizontally.



Now, change the right hand side to a UV/ Image Editor screen.



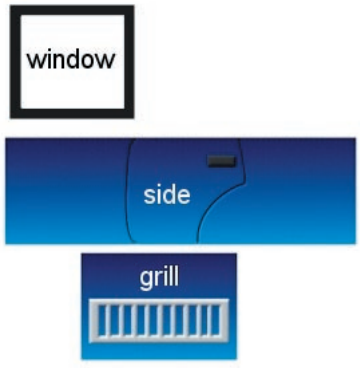
Your screen setup must look like this:



Now we will begin working on the texture. An overview of the process would be:

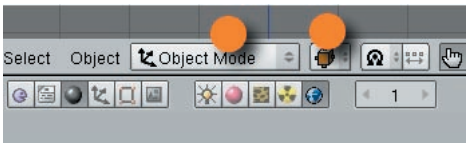
Select a part of the model. This part will automatically appear in the UV/Image Editor Window. Load the picture that you want to put onto the model, into the UV/Image Editor window. Move the vertices around so that the correct part of the picture is displayed on the model. Load the picture into the material for the model.

The picture that you will be putting onto the model should look something like this:

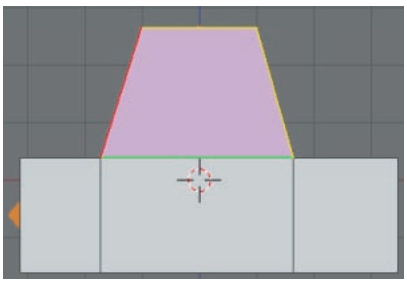


We will first "map" (being the correct term for this process) the car windows, then the front and back grills, the sides, and then one or two other faces.

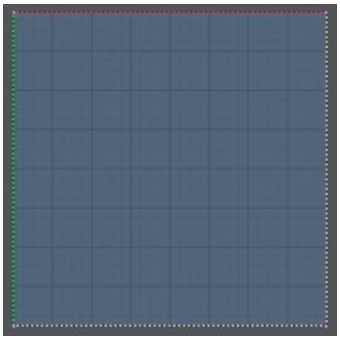
Ensure that your left hand window is in "Object mode". Also check that the button next to it on the right is set to "Solid".



The next step is CRUCIAL! Keep your mouse cursor over the left window, ensure that you have the model selected, then press FKEY. This places that window into "Face select mode".



RMB click on one of the windows of the car. It should look like the above figure. You should also see the following in the UV/ Image Editor:



The face that you selected is now displayed as a flat square in the UV/Image Editor. At this time, the square fills the whole UV/ Image Editor. We will soon change this.

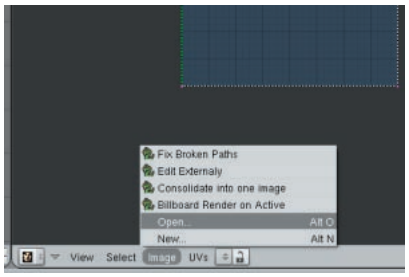
NOTE: UVW coordinates is a name chosen to describe the coordinates of your textures. It is the same as XYZ, just different letters. Since we are working in three dimensions, your textures have to know their position is three dimensions as well, hence UVW coordinates.

We need to load the picture so that you can position the UV coordinates (I"m referring to the face that you selected).

Buttons to be used for moving things are:

RMB	(select)
SHIFT-RMB	(add to selection)
RKEY	(Rotate)
GKEY	(Grab/ Move)
SKEY	(Scale)
BKEY	(box select)
AKEY	(select/ unselect all)

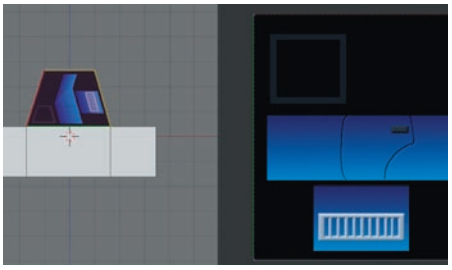
To load the texture, make sure that you know where the texture is located on your hard drive. Click on the "Image" button as shown here. Find the picture, and load it.



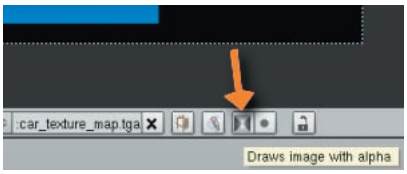
Two things happen:

1.The UV/ Image Editor loads your car "map"

2.The selected face on the model displays the entire "map"

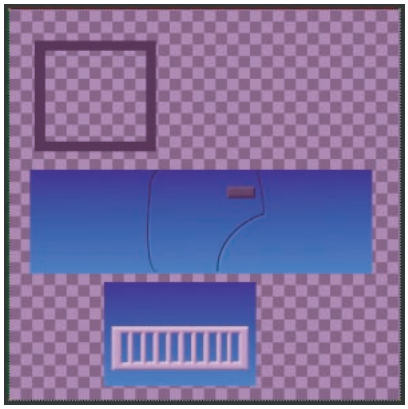


The car map is in Targa format, with an alpha channel. This means that there is transparency information built into the pic. Click on this button to see the transparency:

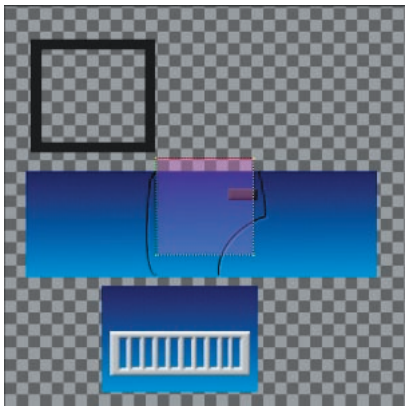


With the transparency switched on, the UV/Image Editor looks like the picture to the right:

You can still see the four vertices of the selected face in the corners of the UV/Image Editor. Select all of these vertices, pressing AKEY until the area turns pink:

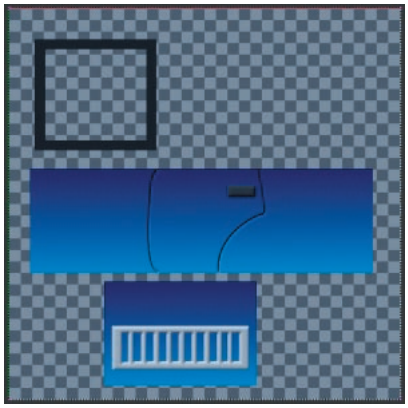


SKEY allows you to scale the selection. LMB finishes the scaling procedure. You will see that the face on the model displays only the area where the face is located in the UV/ Image Editor.



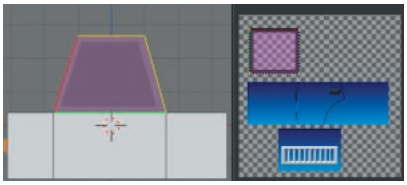
Getting back to my one line explanation, see if you understand it better already:

"You take parts of the model and place it on a picture so the parts of the picture know where they should go and stick to the model."



Now, GKEY allows you to grab and move the whole selection. Move it to the window frame, checking the effect on your model all the time.

Here you can see how I have positioned the vertices in the UV/Image Editor so that the window frame fits nicely onto the model. (uvmap9.jpg)

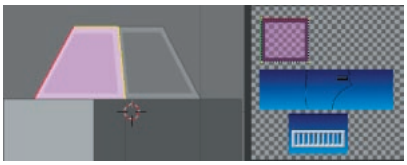


Your turn

Rotate the model so that you can select the other windows. Map each of them in turn.

When you select a face, and the face is displayed in the UV/Image Editor with no texture, just click on this button and select the image map:

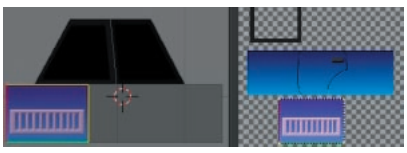
When you are finished, you should see them all mapped something like this:



I have changed the color on the windows so that you can see the effect better.

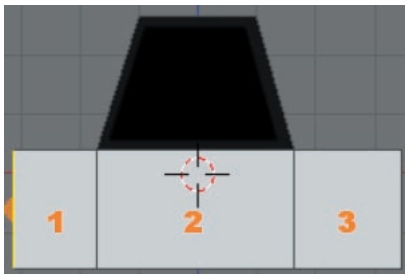
An important thing to note is that we have only started with the uv-mapping. You will not be able to render the image yet. If you render the image now you will only see a gray model!

When you are finished mapping the windows, you can do the front and back of the car. Here I selected the front, and I have mapped the grill in the image to that face.

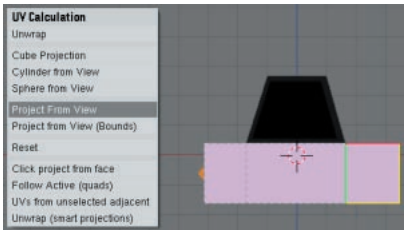


Challenge yourself and map the other end of the car in the same way!

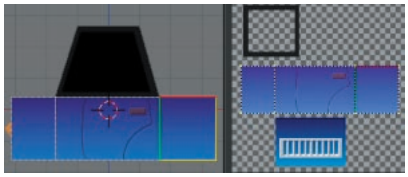
Now for the sides. There is a way to map all three faces in one go.



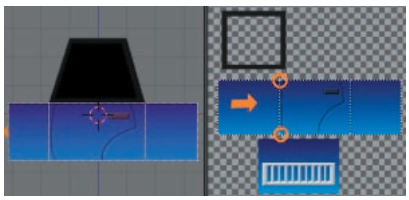
Make sure that you are looking at the car from the side. Select the three side faces (one side of the car model).



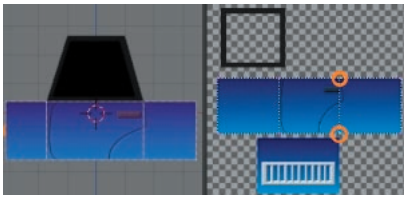
Keeping the mouse cursor over the 3D window, press UKEY. Select "Project from view". This transfers the exact layout of the faces as they are in the 3D viewport, to the UV/ Image Editor. Using the scaling (SKEY) and grab (GKEY) keys, position the faces onto the sideview on the texture map.



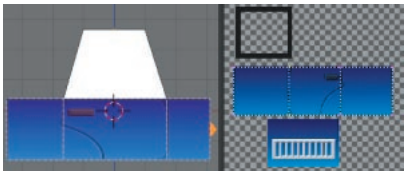
Can you see that the door on the map doesn't fit under the window? In my opinion the door should start and end below the window supports. To do this, use AKEY to deselect all vertices in the UV/Image Editor window. Now SHIFT-RMB the two vertices as shown in the image, and move them to the right. Can you see that the texture map is adjusting itself on the model? Now choose the two vertices on the side of the door handle, and move them so that the door handle is mapped closer the the end of the window frame. (uvmap15.jpg)



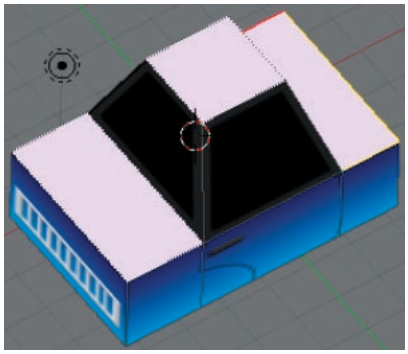
Mapping the other side of the car model involves a trick. When mapping the other side, you will notice that the texture goes on the wrong way round. Firstly map the texture as you have already done. Then, use the menu button: 'UVs' --> Mirror --> X Axis
This will flip the uv's horizontally.



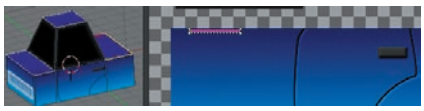
Note that the actual map does not flip! You will see the effect of the menu command on your model, where the texture will have flipped horizontally.



What remains to be mapped now are some faces on the top and bottom of the model:

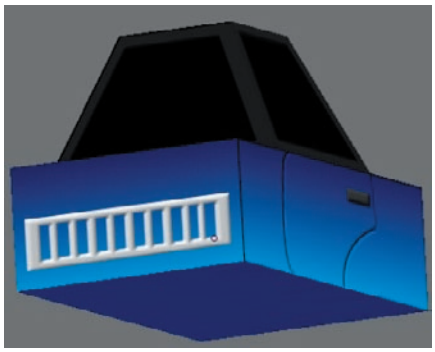


My approach here is to simply map those faces to dark blue areas on the map. The reason for that is that I feel that I can reuse parts of the texture without having to add unnecessary detail to the texture map.



You can map the bottom of the car in the same manner.

Finally, your mapped car should resemble something like this:



DON'T TRY TO RENDER THE IMAGE YET! We haven't applied the image map to the material on the model, so you'll only see a grey block.

Go to the material button:

Ensure that you have a material assigned to your model. If not, add a new material.

I'd like to explain in words what I'm going to do next. You have so far managed to map faces from your model onto a texture/ image map. This is sort of like a street address for those faces. They will always go and place themselves on the image at that address. This was the first part of uv texture mapping.

The second part involves loading the image map into the material that is applied to the model. The process involves a couple of settings. Simply loading the image as a texture will give the wrong result. The default setting for loaded textures is 'Orco', or rather "Original coordinates". We've done UV texture mapping, which gives the texture a very specific destination.

I don't want to go into lighting techniques, so I will activate a button that ignores lighting and simply displays the texture without shadows or highlights. This will be the quickest and best way for you to check your uv mapping.

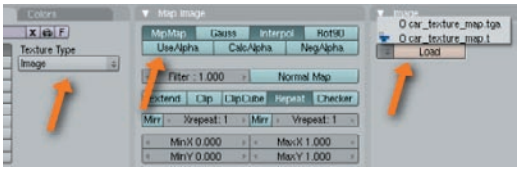
I struggled for three years to understand this process (using internet tutorials), because the two parts of the process did not connect in my mind.

So, let's launch into the final part: Assigning the image map as a texture in the material.

Number 1 shows that a new material has been assigned to the model. We will click on the leopard skin button (Texture) at number 2 to load the image map. Number 3 shows that the material currently uses "Orco" as the co-ordinate system for the material. Number 4 is close to the "Shadeless"button, which will tell the material to ignore lights, shadows, etc.

Click on the texture button. Your screen will look like this:

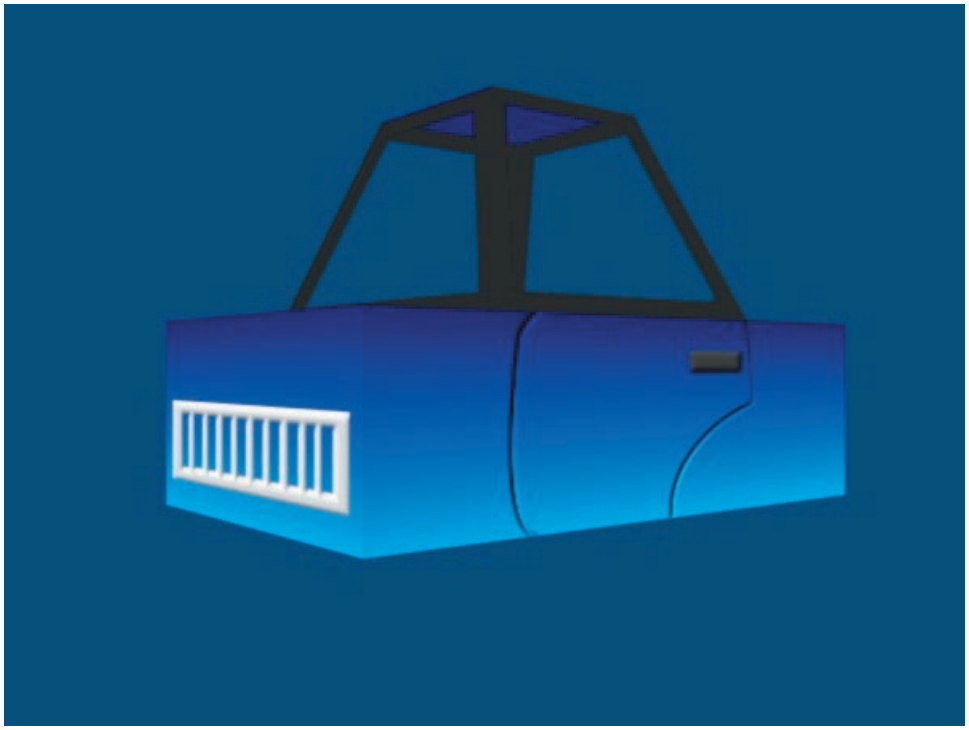
Click on the "Add New" button. Then choose an "Image" texture. Click on the triangles next to the "Load" button. You should be able to choose your car_texture_map from the list that pops up. Now activate the "UseAlpha" button. This will tell the texture to activate the alpha channel embedded in it. Huh? Alpha Channel? If it's unfamiliar to you, think of it as switching on the transparency that's already a part of the targa image.



Below are the final material settings for the car material.

Make sure that you have a camera set up. Press "F12" and render.

Below is what my uv-mapped car body looks like once rendered.



As I mentioned towards the start of this tutorial, this process involves new concepts. Therefor, assume that it may actually take you some time to get behind the concept of UV mapping.

?rman

Notes for material settings:

Ztransp is ON
Shadeless is ON
Alpha is at zero
In the "Map To" tab on the right "Col"and "Alpha" is set to ON

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

HTML FOR NOOBS - PART 2: Links, Pictures and Tables

Let's face it. A page without pictures is boring. Now we can't have boring sites can we? Luckily inserting images is very easy in HTML. Simply insert a <IMG></IMG> tag where you want the image to be. Inside the <IMG> tag you need to provide the source parameter. This tells the browser where to find the image it needs to display. For example you would put <IMG SRC="http://www.mydomain.com/header.jpg"></IMG>. Whenever specifying a file location you can use the root directory. If you use the root directory, your browser will look for the file in the same folder that your HTML file is saved. If your file is in the same directory as your source file, just put <IMG SRC="/header.jpg"></IMG>. This will work for any file. Animated .gif files cam also be used in a standard image tag.

If you have lots to say or want some kind of structure in a website, you probably will need to spread it out over several pages. To enable easy movement between these pages, we will make links. A link can be created with the <A></A> tag. In between your tags you can put your text for the link. Should this text be between any other formatting tags,the text will be formatted in the same way. You also need to specify the destination URL of the link in the form of the HREF parameter.

For example:

```
<A HREF="/badgers.html">Click for my badger page!</A>
```

Notice how the root directory can be used for specifying other HTML pages. Links and images can also be combined into link images! Simply put an image tag where your link text would usually be.

For example:

```
<A HREF="/badgers.html"><IMG SRC="/bad.jpg"></IMG></A>
```

One of the more complex ideas to grasp in HTML is that of tables. Tables allow you to align things easily and give your site structure. First start with a <TABLE></TABLE> tag. Tables need height width parameters to work effectively. When specifying dimensions you can use pixels as a unit or percentage. If you say WIDTH="40" then your table shall be forty pixels wide. Should you say WIDTH="100%" then your table will span the entire screen. This is useful because it allows a page to scale to any resolution. To be able to see where exactly your table's borders are you might want to enable CELLPADDING, CELLSPACING or BORDER. For the moment lets set them all to five pixels.

The table tag should look like this (spanning multiple lines here due to page constraints):

```
<TABLE WIDTH="100%" HEIGHT="70%" CELLPADDING="5" CELLSPACING="5" BORDER="5">
```

Now we need to add some rows and columns into the table. Add a few <TR></TR> tags. Inside each tag you need to put some <TD></TD> tags to start columns. You should now be able to see your basic table. Now experiment with throwing some text and images into the cells. The same height and width parameters can be used on rows and columns to manipulate cells into the desired shape. To have one cell taking the place of several. The COLSPAN and ROWSPAN parameters can be put into the <TD> tag.

That about clears matters up for the moment. Make sure you don't miss the final piece of this beginner's kickstart series!

SQUID

Item 1	Item 2	Item 3	Item 4
	Item 5	Item 6	Item 7

```
<TABLE WIDTH="50%" HEIGHT="70%" CELLPADDING="5" CELLSPACING="5" BORDER="5">

    <TR>
        <TD ROWSPAN=2>Item 1</TD>
            <TD>Item 2</TD> <TD>Item 3</TD> <TD>Item 4</TD>
    </TR>
    <TR>
        <TD>Item 5</TD> <TD>Item 6</TD> <TD>Item 7</TD>
    </TR>
</TABLE>
```

Divide and Conquer with Encapsulation

Everyone understands that a large problem is easier to solve if you break it down into smaller and smaller pieces. The smaller pieces can then be broken down into even smaller pieces and finally as each tiny little piece is solved, they can all be put together to get the big problem solved. Encapsulation is a mechanism in computer programming to divide and conquer the large problems by making them into smaller problems.

Solutions to problems typically have two aspects; firstly the solution itself and secondly an explanation of how to use the solution. A solution by itself is pretty meaningless if no one else can understand how to use the solution to resolve the greater problem. Often the same explanation on how to resolve a problem can be used for various different solutions such as requiring transport to move a computer across town (the explanation) and fetching a truck, car or bicycle (the solution) to do the actual moving.

Within a computer game (our large problem) there are many different aspects (smaller problems) that can be separated from the main program, and these in turn can be broken down into various little pieces (tiny problems) that can be implemented one at a time. A game framework for example will contain the main game loop, a sprite engine, a sound system and possibly a high score system. Each of these can be implemented (the solution) individually and combined to make the game work. The sprite engine in turn may consist of various different types of sprites such as a player sprite, a monster and a dot that can be eaten.

Encapsulation allows us to work on each aspect of the game independently and to get it working in isolation from the rest of the code using good abstraction. These independent parts of the game can then be combined to create the bigger solution and even used later in other applications. The key aspect of encapsulation lies in defining to other objects what methods and properties may be used in the object by other classes through the definition of a clear and understandable interface (the explanation of the problem solution). Now the real purpose of encapsulation is to ensure that consumers of the object do not try and use portions of the object that are likely to change in the future, which means that the changes can be implemented very easily, with no impact to the users of the objects.

An Object Orientated Programming approach

supports Encapsulation very well, as an object or class by its very nature defines an interface (the explanation) to the consumers of the class while ensuring that the core functionality is hidden away (the solution). The level of visibility of the methods and properties of the classes are defined in the interface through the use of private, public and protected definitions which define what consumers may see. A class inheriting from the base class may have very different visibility requirements from the parent class as a game library has when making use of the class itself. A key aspect to consider when defining encapsulation is that the various objects and classes should be broken down in such a way that they are distinct in their functionality, and that there is as little overlap in functionality as possible.

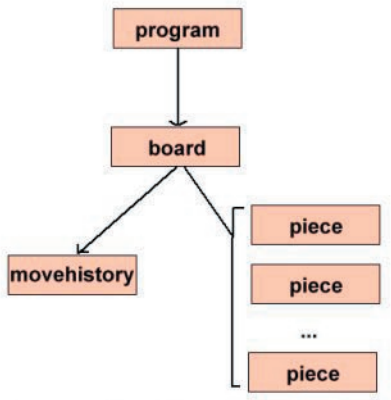
Encapsulation can be utilised in many ways to improve the quality, maintainability and development time of a game. By breaking down the game into its various parts and very carefully defining the interfaces that each part will make available to the game itself a clear technical design for the game can be generated. Each class once developed will immediately be available for use by other classes as they already know how to interact with the new class. In many cases classes can be defined as empty containers but accessed by the game itself before the implementation is complete. Consider for a moment a 'Score' label on the screen. When creating a Score class there would typically be an AddScore, DrawScore and ResetScore methods. The player character can call the AddScore method when killing a baddie, the screen draw functions can call DrawScore, and the New Game method can call ResetScore before anyone has defined how the various methods do what they are supposed to (even before the game contains a font class to actually draw the score to the screen).

Another aspect where Encapsulation is often used is to separate the presentation from the content of an application, in game development terms, to separate the game engine from the display engine. So a chess game could contain an encapsulation of the chess game engine with a clearly defined interface. Various game developers could then use the chess class and build their own presentation layer to turn it into a web based game, a single player board game or a multiplayer networked shooter style game.

Encapsulation is most important when code is being developed that is going to be made available for other people to use, such as the example of the chess game engine mentioned above. An engine like this must have very clearly defined interfaces to allow the programmer to use the engine easily while not really understanding how it all works. Of course it goes without saying that the interface needs to be documented in some way that makes it easy to understand and use.

Small solutions are often the best way to solve a big problem. Game development is a big problem that needs a lot of small solutions. Breaking down the problem like this makes it a lot easier to find the smaller solutions while not impacting the rest of the game. Taking the time to define the smaller problems before trying to tackle the big solution will often mean a better final solution in less time than would otherwise be required. Using Encapsulation to wrap the smaller solutions the bigger solution can be put together a whole lot more easily.

CAIRNSWM



a program can implement a fully functional chessboard just by creating and using a board object.

ROACH TOASTER 2: PICKING OUT THE BUGS PART 3

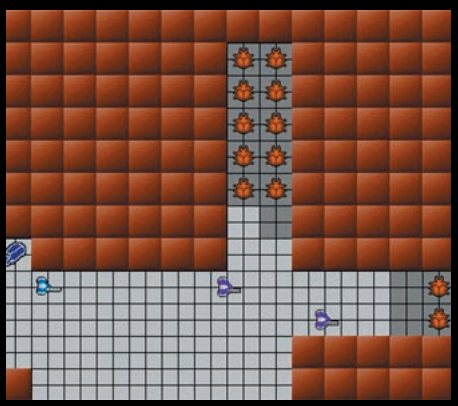
I sort of believed other developers when they said the last 10% is the longest (or something like that), but it seemed kinda ridiculous all the same.

This is exactly where I am with Roach Toaster 2. I am stuck on the last 10%. All the nitty gritty things have to be done, like polishing, adding eye-candy, balancing and last but not least, testing it for the gazillionth time.

The bane of a developer is the fact that by the time you finish your game, it will not be "fun" to you anymore. All the testing has made me numb to the fun factor of my own game.

There is some security in that fact too. If you manage to develop a game that is fun to play while you develop all the way through, then you know you have a hit!

I still have a lot to do in Roach Toaster 2, the biggest being the testing phase. At this late stage of development, after some closed beta testing, I changed a few key mechanics to the game (without too much hassle, luckily). The biggest being the way your placing radius increases. Instead of growing per turn, your own units determine the placing radius.



I also changed the way enhancements work. It now beautifully ties in with your global budget. Instead of getting random enhancements, you will now have to buy them.

A work in progress of the new enhancement system can be seen below.

Roach Toaster 2 has also hit the phase where I should start doing some pre-release marketing. I am stuck without a proper net connection, so that is pretty much something I will leave for later.

I have thought of interesting new ideas to market Roach Toaster 2. I have thought of various areas of the net that could do wonders, like YouTube. Did you know that a music video by the name of "Ok Go" received over one million hits? The immense pulling power of something like YouTube could very easily be harnessed.

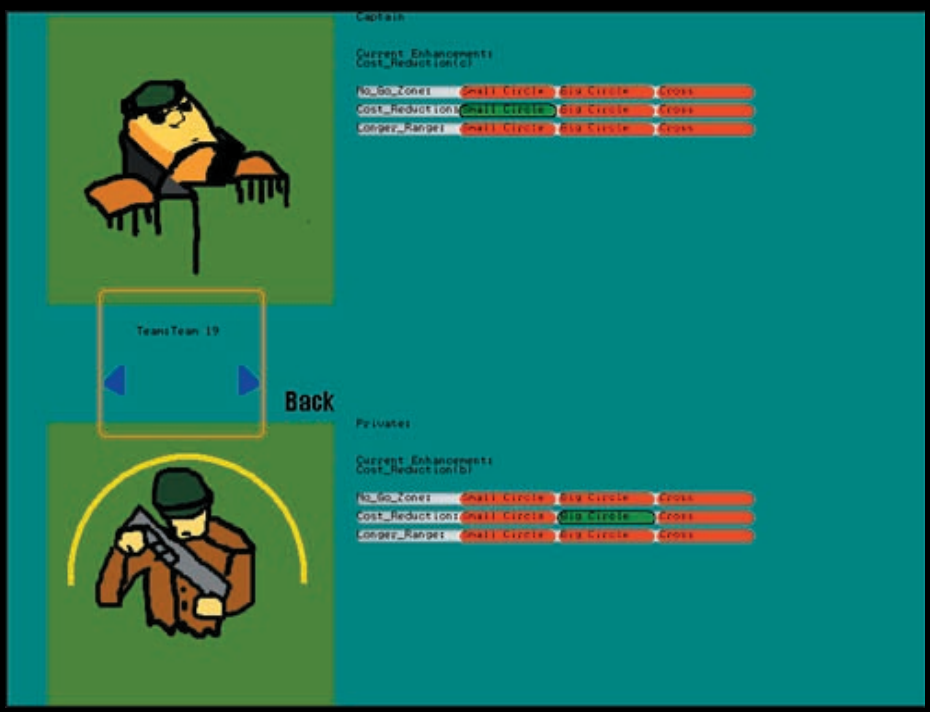
I will definitely give feedback once I am done with that.

Other aspects I can include to further Roach Toaster 2's gameplay is online features. While these features aren't a necessity - you can play off-line if you want - they add to the whole package.

These features include having global stats and an online community where you can upload and download user-created levels.

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

TROOJG



CODING ETIQUETTE: NAMING CONVENTIONS

While commenting your code may be the best passive way for you to keep it manageable and maintainable, the use of a standardised naming convention system can help you to do much the same in an active manner. In other words, by merely naming variables, structures, functions and the like in a consistant and descriptive way, you can perform a lot of the same functionality as code commenting automatically. This isn't to say that just by naming things in your code smartly you can ignore the need to add comments. However, can can reduce the need for certain comments or the length of them in your code as long as the areas in which they would exist are populated by descriptive coding practices.

In terms of specifcs on existing naming conventions, the most well known one is Hungarian Notation. The basic format of Hungarian Notation is this: variable naming is made up of two specific parts, with one part being the actual variable name and the second part being a prefix for that name. It is designed in a way so that the variable name part should be descriptive towards what the variable itself represents. It will begin with a capital letter, and if more than one word is necessary for the description, each word should only be differentiated by the capital letter that it begins with. This method of word concactenation is known as CamelCase. As for the prefix of the variable, it is made up of lowercase letters which are used as a mnemonic (a memory aid) for the type of data that the variable is. Therefore, different letters are used to represent intergers (i), floats (f), booleans (b), etc.

Simple, syntax-correct statement in the C language:

```
a = b + (b * c);
```

Same statement, but with simple Hungarian Notation:

```
fTotalCost = fPrice + (fPrice * fVat);
```

As the simple example above shows, there is clear difference in what code written in the Hungarian Notation style looks like compared to the bare minimum that a language like C requires in order to be syntax-correct. Hopefully you will also be able to easily notice the benefits of writing code in this method, as not only does each variable in the statement read easily in terms of what it represents (TotalCost, Price, Vat), but you can also immediately tell what type of data each variable stores (floating point values). The principle of using CamelCase style naming can also easily be applied to all other sections of code including functions, structures, etc.

At the end of the day, the most important thing is for you to choose a naming convention system that is best suited for yourself. It must be something that you feel comfortable with practising, while at the same time remaining descriptive enough for not only yourself, but for other people who may be working with your code now or in the future. You may wish to implement Hungarian Notation (or another style) in its strictest sense, or you may with to take ideas from various styles to create your own. When all is said and done (and coded...), the exact style (as long as there is a style!) doesn't matter so much as the discipline that you are requiring yourself to follow does.

However, whatever methodology you do decide on implementing should be done so in a clear way and maintain some form of consistency for as long as you use it. This doesn't mean that you can't ever evolve your practices, as they should in fact change from time to time in order to be in line with whatever you feel comfortable doing, not to mention your own personal growth in programming knowledge and abilities. What it does mean though is that you shouldn't be chopping and changing your conventions at any given moment while you are programming. Ideally, any changes to your system should be thought out to a degree so that they are actually providing an improvement to your code, and not just happening because you may be too lazy to adhere to your own self-imposed guidelines!

If you are interested in more information on Hungarian Notation, or Naming Conventions in general, you should visit Wikipedia (www.wikipedia.org) and search for both "Hungarian Notation" as well as "Identifier Naming Conventions", as both of these entries go into much more detail on their respective topics than this articles does.

COOLHAND

THE HISTORY OF I-IMAGINE

Part 3: Killing Time

I must be honest that it was quite hard for me to focus on my work back in Vancouver once I had returned from my trip halfway around the world. The school year was nearly over and I had decided to resign my position at DigiPen at the end of May, which was only two and half months away at the time. However once the routine of things had started up again, both time and work went by quickly enough. I soon told my co-workers about my situation and what I was planning on doing. Needless to say they were quite surprised, although their shock soon turned into humour as they couldn't resist joking around with me about what I was getting myself into... lions in the streets and all! Their ignorance-induced mirth was short lived though once I revealed to them that I had actually spent time there and had experienced South Africa for myself, and that they were only wasting their time with petty scare tactics!

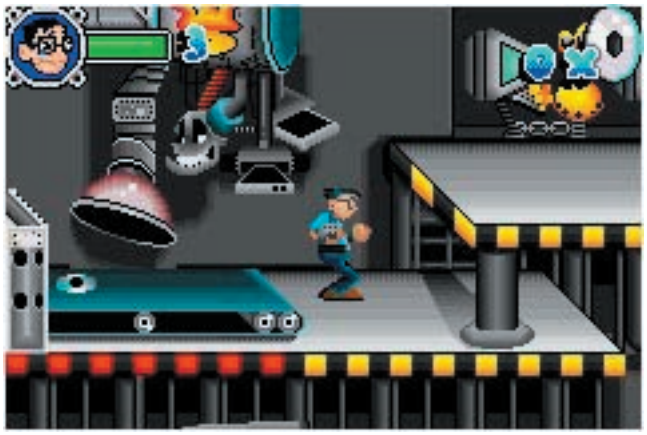
About a month after returning to Vancouver, the Easter Weekend holidays were upon me, and I had plans to go back home for the long weekend. The timing surrounding all of this eventfulness in my life couldn't have been any more perfect as all of my siblings (three brothers, two sisters) and their families had decided to travel back home for Easter as well, as it was also our Grandmother's 90th birthday party that weekend. I had previously only told one of my sisters about the whole "South Africa thing", (she also lived in Vancouver and I wanted at least one family member to know where I was in case there actually were lions in the streets!) and I decided to use this family reunion of sorts to tell the rest of my family about the future that I had decided to pursue for myself. As expected, all of my family members were quite taken aback by the news (especially my mother!), but everyone (except my mother...) congratulated me and wished me the best of luck in the choice that I had made.

Next up in the preparation schedule was the E3 trade show that was taking place in Los Angeles during the middle of May. Dan had decided that it would be the perfect place not only for all of the team members to get together in one place for the first time, but also for us to begin approaching publishers to let them know that we were on the block. At the time, E3 was getting a reputation for being a place where any game developer could rock up at a meeting with a publisher, pull a rolled-up game design "document" out of his back pocket, and get 2 million dollars to make a game. Sadly, that wasn't the case. While we didn't have any specific design documents per say, we did have a couple of ideas that we lobbed at publishers (if I remember correctly they were both 3rd person action games... one set in the Wild West and the other in Ancient Greece...). Unfortunately (but not unexpected), publishers were not ready to bite on what we had to say, but for the most part they were interested to see what we would have to show in the future, and asked us to keep them up to date with what we were working on. So while we didn't walk away from E3 with a signed game contract (not that we actually thought we would pull something like that off!), our meetings were not without merit as we did make some very good contacts at various publishers, although one person in particular would end up causing us quite a bit of grief at a later time...

Publishers aside, the E3 get together was a great experience for us. The core group of the six I-Imagine founding members were all able to meet each other and get a feel for the people with whom we would be working alongside for next two years. Since the initial meetings that took place in South Africa back in March, a couple of the original guys had changed their minds on moving to South Africa (Jim and David from Paradigm, who were classmates of Dan and I at DigiPen), and so there were two new guys for the rest of us to meet. Dan had met and worked with Matt and Dave (another Dave... not David from Paradigm!) while he was on contract in Portland, and he was able to convince them to come and join us at I-Imagine. So, along with meeting them for the first time during E3, I also met Felix's brother Kenny who was to be our designer. All six of us hit it off quite well, even though Dave kept looking over his shoulder -- he didn't want to be seen with us too much as he was still under contract at Looking Glass and was feeling a bit paranoid!

Once the fun of E3 was over, I headed back to DigiPen to finish up my remaining two weeks there. When they had ended, I returned home to North Bay to wait for my South African work permit to be granted so that I could finally make my way to Johannesburg. Well, as anybody who has dealt with any kind of governmental service in South African knows, it was not something that happened quickly. When all was said and done, I spent just short of four months loafing around, leeching off of family members in North Bay, Connecticut, Toronto, and even back in Vancouver while I waited for the work permit to finally be granted to me on September 24, 1999. The very next day, September 25, 1999, I boarded a plane in Toronto that was bound for New York. Once there, I was to swap airports and catch my flight aboard South African Airways direct to Johannesburg where my future awaited me.

COOLHAND



HOTLAB MARCH REPORT

So many people!

After the unfortunate cancellation of the hotlab in February, it seemed everyone was amped and ready to go in March. Thanks to MushiMushi's marketing and Nandrew's recent article about the hotlabs in the March issue of SA Computer Magazine, many, many new and eager game dev'ers were attracted to the KZN Hotlabs. I can safely say this was one of the largest crowds we've ever had for a hotlab, with well over 25 people attending. Luckily we had a new, bigger lab to fit everyone in. One unfortunate drawback of the new lab is that there are no computers and everyone needs their own laptop or PC, but fortunately almost all the eager game dev'ers brought their own, so things worked out.

It was such a great sight to see so many new, keen people ready to make games. After everyone installed GameMaker and a brief introduction about the hotlabs was given from MushiMushi and SilentBob, things kicked off. Due to the fact that virtually everyone was new to game developing and the GameMaker framework, the hotlab focussed on the basics and how to make a simple platform game. Insomniac and Skinklizzard headed up the platform tutorial and in no time people with hardly any prior game development experience were making games. Due to some people progressing faster than others, the class was divided into 2 groups so the people struggling could get more attention while others could go ahead and experiment and make their own additions to their games. Some people got so into it they even spent the short break improving their games. At the end of the hotlab everyone had a basic platform game running and their homework was to improve and add to the game for the next hotlab. If only homework was always that fun!

A great day was had by all and it's great to see even with so many people the hotlab's trademark relaxed and friendly enviroment still held. The month of March was a definite milestone for introducing so many people to the awesome art of making games and is a great omen for things to come in future hotlabs. Make sure that if you are in the KZN area, you come along to the next hotlab. Good times and powerful knowledge are guaranteed!

INSOMNIAC

Hotlabs run on a monthly basis at IT INTELLECT in Durban (031 277 2000) and all participants are contacted via e-mail. To be added to the mailing list, please send an e-mail describing your area of interest to bernard.boshoff@gmail.com