



SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

SEPTEMBER 2007

DEV.MAG

CREATE • DEVELOP • EXPERIENCE

rAge





REGULARS

Ed's note	03
From the net ...	04

FEATURE

Chatting with Travis Bulford	05
<i>An interview with the creator of South Africa's first game development studio</i>	

OPINIONS

Why doesn't life come with background music?	08
<i>Q-man takes a look at how games put the grooves in our moves</i>	
For the love of games!	09
<i>TheUntouchableOne touches on the number one reason to develop games</i>	

REVIEWS

Knytt	10
<i>A game which charms with simple graphics and a simple premise</i>	
Great Games Experiment	11
<i>A networking portal with the avid developer in mind</i>	
Gamedotdev.co.za	12
<i>If you don't know about our alma mater yet, then it's time to listen up!</i>	

TUTORIALS

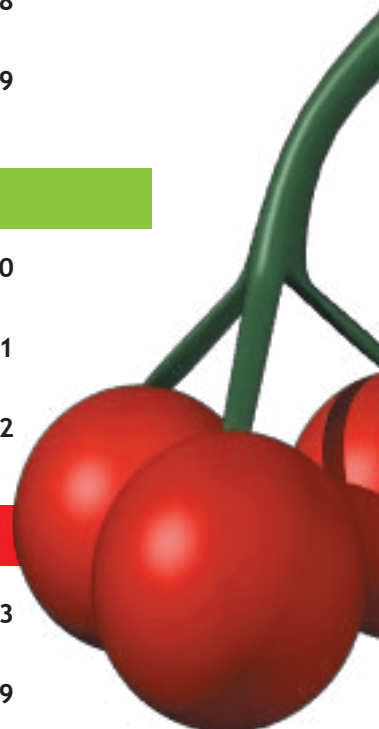
Blender — intermediate series	13
<i>Extracting 2D animations from rotating 3D objects</i>	
Beginner's guide to making games	19
<i>The fourth tutorial in the series, dealing with particle effects in Game Maker</i>	
Game graphics with photoshop	22
<i>A new series, the first part dealing with an overview of the basic Photoshop tools</i>	
Going slow-mo	25
<i>A quick look at implementing in-game speed adjustment</i>	

DESIGN

Project — Line Wars	29
<i>The highs, lows and middle ground of creating a game with ... lines</i>	
Game.Dev's edutainment competition	32
<i>The story of what happened when Mindset decided to give away R10 000</i>	
The history of I-Imagine	34
<i>The eighth part of our long-running series on these local game developers</i>	

TAILPIECE

We are legion	37
<i>If you're in South Africa and do game development ... know that you're not alone</i>	



DEAR READER ...

Yee-haw! Oh man! Sweetness! Dev.Mag Issue 17 is here and it absolutely rocks!

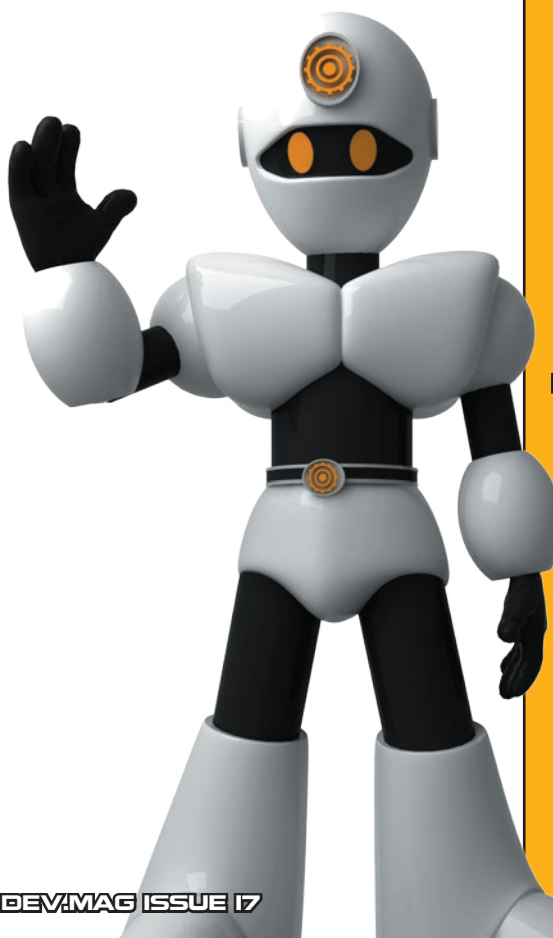
Yeah, sure, I know: that's rich coming from me. The more cynical readers will probably be rolling their eyes about in their sockets right now (mind they don't fall out) and the more outspoken ones shall be calling verbal justice down upon my arrogant and assuming manner.

But in all honesty, I'm really happy right now, and you should be too. Not only is this a super-special-uber rAge expo edition (an event which most South African readers will be familiar with), but we're sporting a full redesign. And more than 40 pages of game development writing. Does it get any better than this? One of our biggest jumps to date, Issue 17 is the result of an entire month of very hard work done by a lot of committed people.

So, gushing aside, what's new? Well, this editorial is being typed up on the eve of this month's big expo, and from what I can see already there's a whole lotta stuff that's going to be happening in the game development arena this year. It's really inspiring to see so many organisations banding together for a few days and working towards a common goal. Luma's racing title, Mini #37, has been rigged up in and around the Game.Dev stand, and a few of my colleagues have already had a chance to try it out with some super-rad driving seats thrown in for good measure. I missed out because I was naturally assigned some grunt work that kept me occupied for most of the time – but I'm already plotting my revenge, so all is good.

If you've missed out on rAge this year, don't despair. We're doing a full report-back in the next Dev.Mag (which is likely to be tweaked even further), and I suggest that if you haven't had the rAge experience at all yet, you really should try make it sometime in the near future. It's a truly fantastic expo where some truly fantastic dreams are realised. Ciao for now!

RODAIN "NANDREW" JOUBERT
EDITOR



DEV.MAG ISSUE 17



This was the first ever proposed cover for Dev.Mag. It was crude. It was ugly. It didn't even have the same name. Fortunately, some heroic designers stepped in to salvage everything, meaning that this little Frankenstein project never ended up seeing the light of day.

EDITOR

Rodain "Nandrew" Joubert

DEPUTY EDITOR

Claudio "Chippit" de Sa

SUB EDITOR

Tarryn "Azimuth" van der Byl

DESIGNER

Brandon "Cyber ninja" Rajkumar

MARKETING

Bernard "Mushi Mushi" Boshoff

Andre "Fengol" Odendaal

WRITERS

Simon "Tr00jg" de la Rouviere

Ricky "Insomniac" Abell

William "Cairnswm" Cairns

Bernard "Mushi Mushi" Boshoff

Danny "Dislekcia" Day

Andre "Fengol" Odendaal

Luke "Coolhand" Lamothe

Rishal "UntouchableOne" Hurbans

WEBSITE ADMIN

Robbie "Squid" Fraser

WEBSITE

www.devmag.org.za

EMAIL

devmag@gmail.com

THIS MONTH'S OPINION COLUMNISTS:

Quinton "Q-man" Bronkhorst

Rishal "UntouchableOne" Hurbans

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

All images used in the mag are copyright and belong to their respective owners.

RAAAAAAAAAAAGE!

CASUAL GAMING: AN OPINION

http://www.gamasutra.com/php-bin/news_index.php?story=15570

There's been something of a fixation on casual gaming news and articles lately. No surprises there, what with all the statistics and big companies and other fancy things involved. Gamasutra has a brilliant opinion column on the issue, and money-making matters for both casual and general indie developers are taken into consideration. The piece covers aspects such as in-game advertising, online portals and price adjustment, all reflecting on how marketing has been affected by new Internet technologies and progressions. A worthwhile read.



IGF OPEN TO STUDENTS

http://www.igf.com/rules_student.html

Arguably the indie world's most prolific event, gaining mass approval at the annual Independent Games Festival (IGF) is something which developers in the arena aspire to. Success is rewarded with fame, fortune and the guarantee of even more success. Now, students are also allowed to try their hand at getting onto the red carpet themselves, having been given their own section in the IGF awards. Of course, if you aren't interested in trying out yourself, check out the showcase section for some of the previous IGF winners - the likes of And Yet It Moves and Toblo are great examples of what even the humble learner can achieve with a suitable amount of passion and creativity.



GUERRILLA MARKETING

http://www.gamasutra.com/php-bin/news_index.php?story=15402

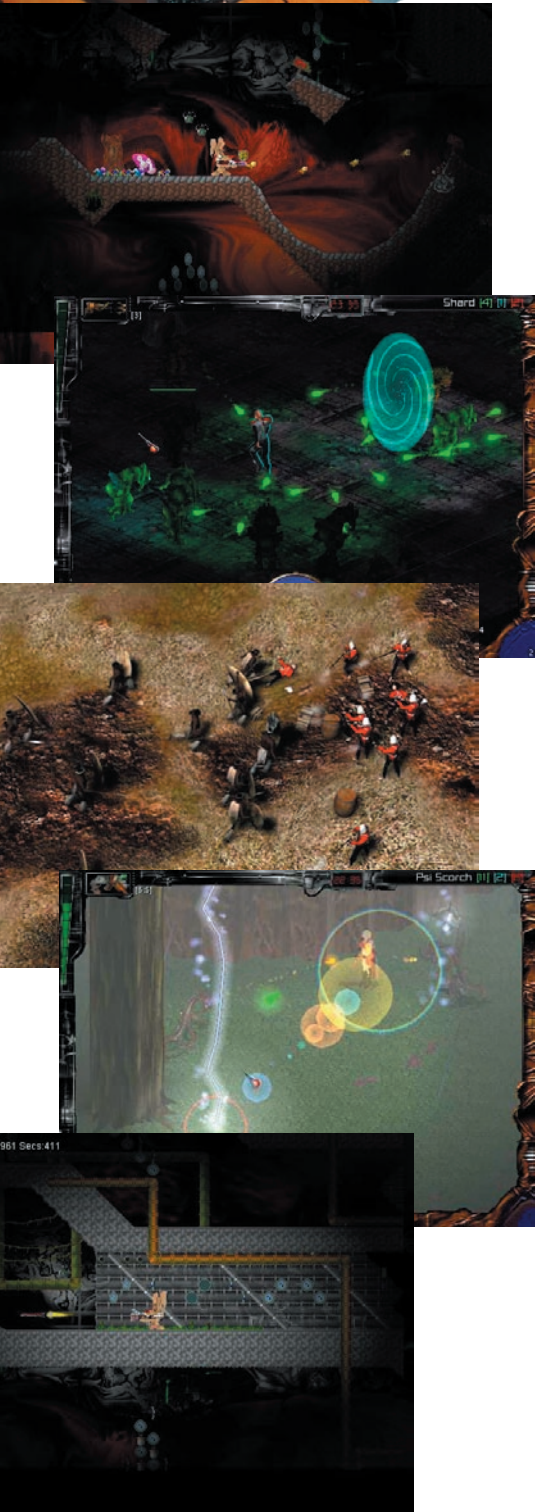
If you want to figure out how to make your games known, but don't know where to start, then listen up! We have a smashing little gem from the big WWW sitting in our laps, brought to us by the Austin GDC and reported by Gamasutra. One of the GDC speakers, Jay Moore, starts off with the most valuable advice of all: "the worst games are done strictly for the money... If you don't have the fire in your gut that says 'if I don't make this game there's nothing on the planet I want to do more,' you're not going to have the quality of content to differentiate yourself as an indie." From there, it gets better. The article explores a talk dealing with being a market-savvy and active gamer, outlining the importance of keeping up the work even after the game has been finished.



PIONEERS OF GAME DEVELOPMENT

A CHAT WITH TRAVIS BULFORD

by Simon "Tr00jg" de la Rouviere



It's not easy being the first. Travis Bulford was the managing director of Celestial, one of the first game development studios in South Africa. The company (later known as Twilyt) stormed onto the scene with South Africa's first reputable AAA title, *Toxic Bunny*, in 1996, and several other projects (including a CRPG called *The Tainted*) were worked on before closing Twilyt closed its doors in 2001. Even though the company itself is no longer around, it was a valuable spark in the fire of South African game development, and proved that a local industry was indeed possible to establish. So, what's Mr Bulford up to now? One of our writers investigates.

Firstly, tell us more about yourself. Who are you?

Well I'm 34 now. I am into games, of course. Reading, kung-fu, diving. Various sciences also interest me. Roleplaying and people, of course. Also play board games. Have two kids and am married now. But my wife likes mostly the same stuff, so that's very fortunate. As for the "who I am": well, hell, I'm still working that out. I think I'm academic at heart - I love knowledge and sharing of that. I want to learn and teach in everything I do.

In terms of developing games, do you think it's better to have a varied background, in the sense that you are then open to new ideas? That is, a gamer will make games, while a non-gamer might make interactive entertainment?

Hmmm ... better and worse are so difficult to judge. I expect that people do better at the things they like in all walks of life. Games are

topical, so games are not just games for the sake of it. They require creative input - well, one would like to think so - and I believe that for the creative person in a games team, variety opens them to new ideas. But a team is made from more than just the story teller. A dev team needs to be more structured and be able to apply logic to that creative information. So better or worse, I don't think it applies if you love games, you can work on them, just find the right role for yourself.

Now, you were in Celestial/Twilyt, South Africa's first real development studio. What was it like? Who were the members?

Original members were Caleb Salisbury, Nick McKenzie, Brian Johnson, and me too, of course. It was fantastic to work on games. I worked more hours then I have since or will again. But it was so much fun I didn't care! We were actually second when it comes to publishing any games here - a Cape Town company beat us by a few months with a title called *Cyril the Cyberpunk*.

But I think we can agree that you guys were the first studio with decent success, that came in the form of Toxic Bunny right? Tell us briefly what games you made.

Toxic Bunny was a successful game financially speaking. But I am not sure I would make the statement you did. Other games? Well, we published *The Tainted* three years later. *The Tainted* failed financially, yet we learnt so much more than we did with *Toxic Bunny*. It's ironic but kind of obvious that one learns more from mistakes than any other way. Anyway, we failed to make *The Tainted* what





it could have been. It had a lot of great points but its failings meant most people never saw those points. After that we started working on Zulu War. We never got that title completed as we ran out of steam and money.

Have you ever considered getting back into the SA game dev scene?

Every day, haha. If the right opportunity arrives, I will come back into game dev.

We've heard that you were busy with a Toxic Bunny port of Java. Tell us more.

I am porting TB to Java to sharpen my long-dormant game dev skills. That's one of the reasons.

And the other reasons for porting Toxic Bunny?

It's a really good game and we want people to play it again. I am also sick and tired of seeing crappy java games. The language can do more, and I will champion that. I'm always tempted to do things that people tell me can't be done. A friend of mine has also been nagging me since we did Toxic Bunny — in fact, from before we finished it — to do a Linux one and this kind of does that at the same time.

What do you think of Game.Dev's initiatives?

Game.Dev started just when we were putting down our tools and closing up shop, so until very recently I've not really looked too much at it, but I must admit it, it's damn nice to see so many people working towards game dev here in South Africa. I can't help but wonder if it was at this level when we stopped if it might not have been just what we needed to carry on. It's terrible working on an island, especially if you're academic by nature.

So what was the most important lesson you took away from working on that island?

The most important game development lesson was humility. It's ridiculous to compete with each other (I mean in a nasty fashion) when the rest of the world is better than you, more experienced, better financed and working together! Personally, I learned not to take myself so seriously.

Any final points you'd like to raise?

I would like to add that for us to succeed as an industry it takes a lot of people succeeding, and not just one or two. Music makes less money a year than games software. We have a great music industry. We as South Africa should have a fantastic games one too. 🎮

QUICK QUOTES

What's your favourite game?

Well, right at this point it's WoW. But to say that of all time would not be accurate. My favourite game of all time is a toss-up between Star Control 2 and Ultima 7.

Alliance or Horde?

Haha! My main is Alliance lvl 70 Pally. On the Horde side there are no high level characters. I like the lore aspect, so I want to play both.

Pizza or Steak?

Pizza... and coffee, of course.

Colgate or Aquafresh?

Colgate I guess.

Xbox360, Wii or PS3?

I'd pick PS3, but I will admit the Wii is quite tempting. Microsoft ruins my PC enough thanks! I don't want them near my consoles either. Next thing you know, they'll want to run my fridge too.

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.PascalGameDevelopment.com

Chrome

Free Pascal

Delphi For Win32

Turbo Delphi

Lazarus

“Why doesn’t life come with background music?”

by Quinton “Q-man” Bronkhorst

I was walking through varsity just the other day, casually listening to my iPod (as most frivolous students do these days), and as I walked past a group of fellow students, a very familiar tune started playing in my ears. Why, it was the Balamb Garden theme tune from Final Fantasy 8! How delightfully apt; for here I was, walking around a very large learning institution, and for a few minutes, I found myself within the game itself, experiencing, in reality, a fantastical world.

You see, real life has many good things to offer (feta, ham, naked people), but the one thing it just doesn’t have is spontaneous and booming music playing in the background as we go about our everyday lives (and no, listening to your iPod or having music play on the radio doesn’t count). Just think back on

a few key moments of your life, and picture those same moments with an appropriate soundtrack playing as the events unfold - how much easier would it have been to act in the right way?

Take for instance that romantic moment with that special someone - except you’re completely oblivious to the signals being given by them. Add, say, Marvin Gay or Barry White to the mix, or some harmonic orchestra playing soothingly in the background, and all of a sudden, there’s this romantic mood that only a fool wouldn’t recognize.

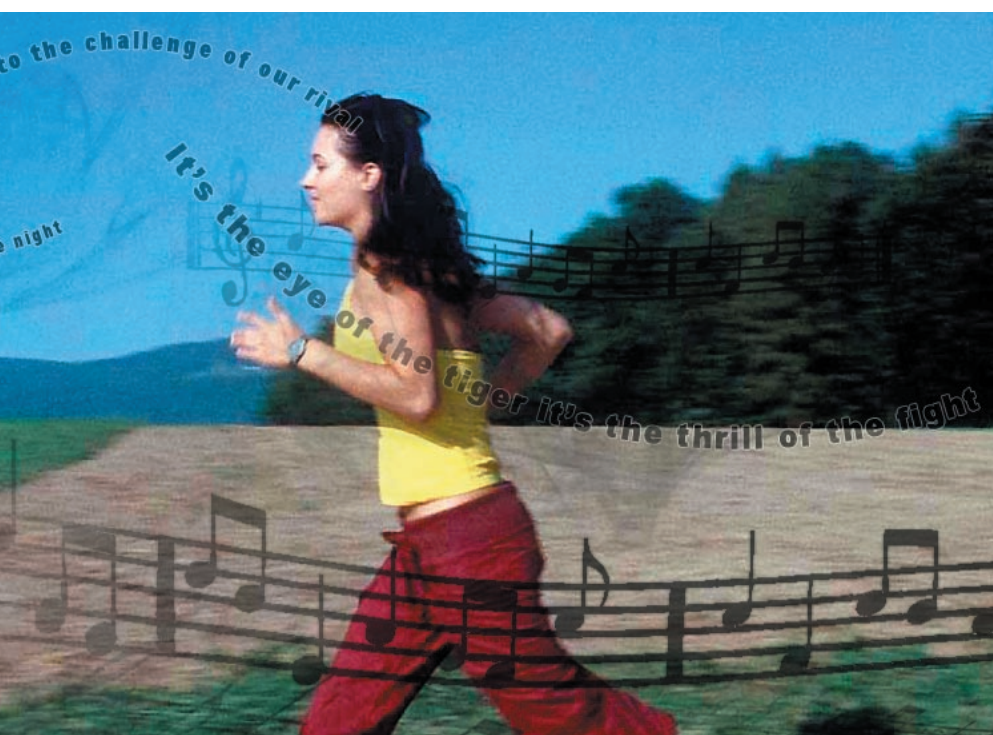
Or how about that fat kid that wants to meet you after school behind the prefabs at 3 o’clock? You’re sure to get your ass kicked two ways to Sunday...unless you have some

up-beat, high tempo action music playing in the background. Your heart starts racing, and you feel the vibe sending you into a sporadic bout of flying fists and kicks.

With background music, we’d know when we’re in danger and we’d know when we’re safe - everyone remembers how it felt to hear that comforting music and seeing a typewriter in Resident Evil. We’d even know when freaky things are going to start happening around us (trust me, you hear sirens and industrial-style music, you’re going to want to find a torch and a radio pretty quickly, because the fog is going to start rolling in real soon).

It’s undeniable that life with background music guiding us would have been pretty awesome, and if you really think about it, it’s just one of those things we take for granted in the games we play. Knowing when to act, what to expect, how to react, which people are good, which people are evil - everything a game can tell you through music, escapes us here in the ‘real world’.

Quite a convoluted path to walk to make a simple point, but the message should be pretty clear. Music adds so much to the experience of something, and plays such a pivotal role in setting up mood and atmosphere. And even though, for us poor sods here in the real world, such wonders of background music can’t truly be appreciated; when we get home and put that game into our system, we get to experience and enjoy such benefits reality keeps from us. 🎧



“For the love of games!”

by Rishal “TheUntouchableOne” Hurbans

Many of you have started developing games, whether it be with Game Maker or another programming language of your choice, while others have just recently discovered the wonderful world of creating games and are filled with excitement and enthusiasm. And what inspired you to pick up this fun - and productive - hobby? The answer to that question should be “For the love of games”. If you answered otherwise, there’s something seriously wrong.

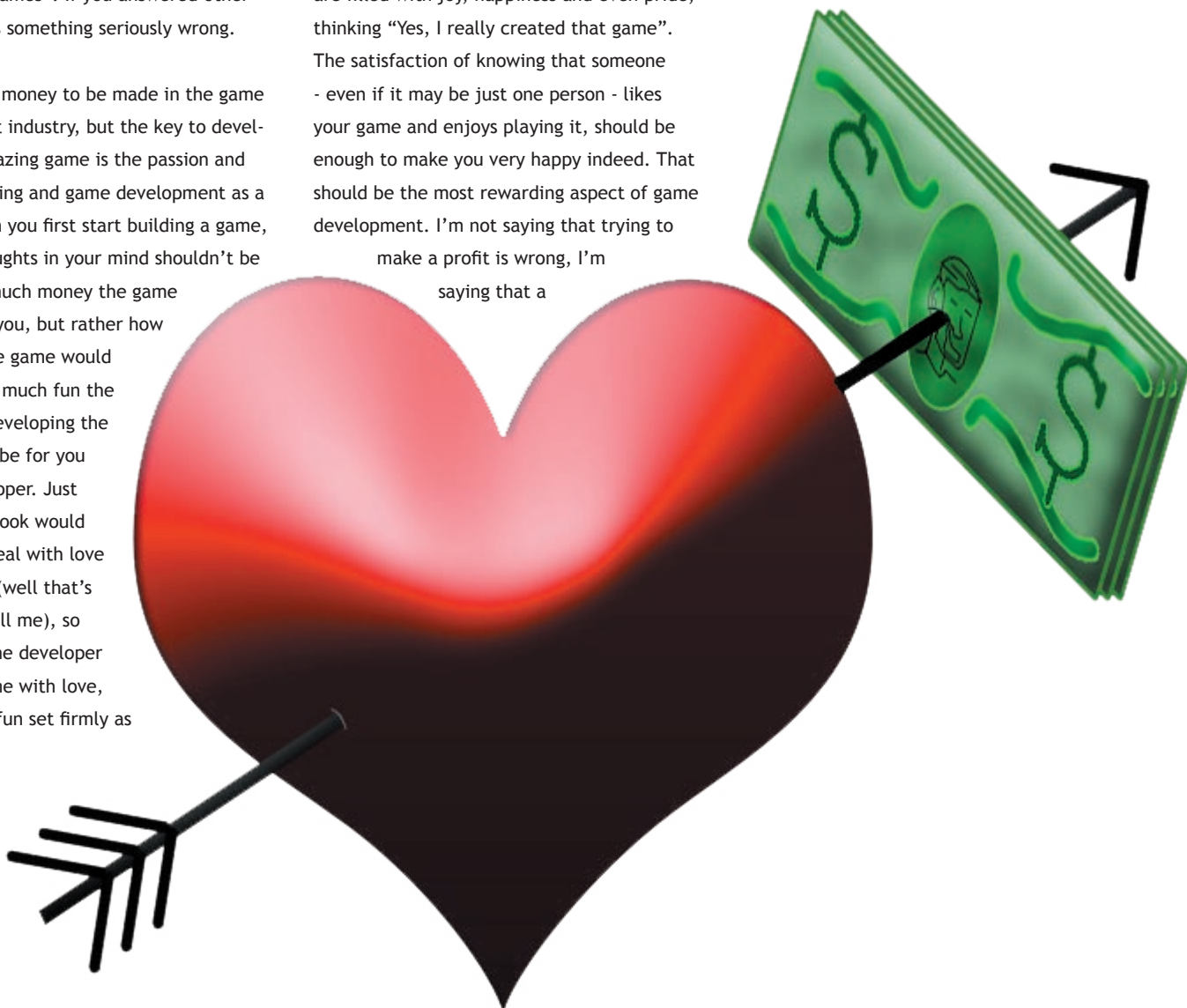
Yes, there is money to be made in the game development industry, but the key to developing an amazing game is the passion and love for gaming and game development as a whole. When you first start building a game, the first thoughts in your mind shouldn’t be about how much money the game could make you, but rather how much fun the game would be, and how much fun the process of developing the game would be for you as the developer. Just like a good cook would prepare a meal with love and passion (well that’s what they tell me), so should a game developer create a game with love, passion and fun set firmly as

a foundation. For hobbyists like ourselves, the driving force behind our work should be: having fun while molding a masterpiece that people out there can enjoy.

Developing a game is a very rewarding experience, and I don’t mean financially rewarding! Rewarding in a sense that when you sit back and see what you have done, you are filled with joy, happiness and even pride, thinking “Yes, I really created that game”. The satisfaction of knowing that someone - even if it may be just one person - likes your game and enjoys playing it, should be enough to make you very happy indeed. That should be the most rewarding aspect of game development. I’m not saying that trying to make a profit is wrong, I’m saying that a

game should be built on prospects of how fantastic the game could turn out and not how rich it can make you. The financial side should always come second if you have your mind set on creating an enjoyable game that will truly stand out.

So, for the love of games, develop for the sake of fun! 🎯



KNYTT

<http://nifflas.ni2.se/>

by Simon "Tr00jg" de la Rouviere



On a strange planet somewhere in the existence of our imagination live small acrobatic rat-like creatures called Knytt. In this epic adventure, an alien captures a Knytt. In an attempt to abduct it, the alien's UFO crash-lands on an even stranger planet.

This is how Knytt starts.

This charming exploration platformer created by Nicklas "Nifflas" Yurgen is a prime example of "less-is-more".

The goal of the game is to scavenge the strange planet in search of the alien's broken UFO parts. In Knytt, all you do is run, jump and scale the side of walls. This might seem boring to some, but it is exactly that, that makes Knytt such a fresh experience. There is no "attack", there are no "upgrades", there's just you and your two little legs.

The biggest strength of Knytt is its simplistic, beautiful landscapes - from desert wastelands and dark swamps to green shrubberies. As you wonder in eerie loneliness in this strange world, you can't help but gawk at this wonderfully crafted world. It is much like Shadow of Colossus in the way that, as you progress through the whole world, you can't help but wonder more about this world and its inhabitants (or the lack thereof).

The "inhabitants" are scattered around the world. They include humanoid creatures staring into space, swimming, fishing and just "being". Then there are the more animal/monster-like creatures that are the most enthralling; most of the time when you encounter them, you can't help but stare at whatever they are doing. These include a stout giraffe-like creature munching leaves and blue centaur-type guys shaking fists at each other across a waterfall.

Knytt is an extremely relaxing experience that is doubled by the soothing and engrossing music, but isn't all just running around an empty world - there are plenty of puzzles and secrets too.

There are things in Knytt all developers can learn from. In the end simplicity reigns king. Sure, having +3 sword of uber strength in World of Who-cares is awesome, but just exploring a world can be just as fun! 🎯



GREAT GAMES EXPERIMENT

www.greatgamesexperiment.com

by Claudio "Chippit" de Sa



Many of you have started developing games, whether it be with Game Maker or another programming language of your choice, while others have just recently discovered the wonderful world of creating games and are filled with excitement and enthusiasm. And what inspired you to pick up this fun - and productive - hobby? The answer to that question should be "For the love of games". If you answered otherwise, there's something seriously wrong.

As a developer, it's always a good idea to keep on the lookout for resources that can help you. Whether these resources are knowledgeable contacts or informative tutorial sites, nobody can deny that a little help here and there comes greatly appreciated. The Great Games Experiment is a website that may help on the marketing side of things. GGE is something of a social gaming website, where gamers can congregate and find others with similar taste. Registered users build profiles about themselves and their gaming

preferences. Users earn experience points by being active on the site, as well as additional popularity points based on numerous behind-the-scenes factors. Deserving users can be awarded kudos by fellows in recognition of achievement or other feats. Gamers can also form groups so that those of similar taste or other affiliation may assemble and discuss topics of interest.

Games are submitted to the site by users or developers alike. Submissions are categorised by tags that are assigned to them upon submission, and these allow the site to find games that are similar to each other, as well as group games by defining features. Each game has a highly customizable home page that can feature a wide array of information about it, including defining features, screenshots and gameplay videos. A similar rating system to that used on registered users is also employed on game pages, as well as an additional 5-point score calculated by averaging user ratings. Users are also free to leave comments about the game or even

submit entire reviews should they so desire. All these features are geared towards gamers, but the site has an equally large appeal for developers. Developers can use the site like normal users, in addition to using it to build portfolios of their own games or others that they assisted in creating. It serves as an excellent tool to promote upcoming or existing games, especially since it already has a large user base despite being a comparatively new website. Users on the site are also encouraged to provide feedback on any submissions that they play, and all developers know that feedback from the community is one of the most important resources that one can have. This is another great factor that lends to the appeal of GGE for developers. Because anyone is free to upload games to The Great Games Experiment, it is a truly brilliant tool for promotion and will result in greater exposure than can usually be achieved this easily. For this, it comes highly recommended. 🍵

Game.Dev

Last Online: 8/7/07

Game.Dev is a South African game development community dedicated to growing a world-class game development industry in our country.

We hold competitions every two months. Give lectures, hold workshops, run development lans and other regular events. Each year we run a huge conference as part of the rAge Expo.

Feel free to check out our members, their games and our other efforts. To join, simply be either from South Africa or a South African living abroad who is interested in game development. Everyone's welcome :)

Report

Headlines and Upcoming Events

Dev.Mag Issue 15!
The Dev.Mag guys give us yet another issue to enjoy.

Competition 16
1st September 2007 - 28th September 2007

rAge Expo
28-30 September 2007
We'll be there in force!

Group Member

visit the **Group Forums**

Ratings

★★★★★

Overall Rating: 4.7

11 ratings

Views: 200

Your rating: 5.0

game.dev

It's what you wish you were doing.

GAME.DEV

www.gamedotdev.co.za

by Claudio "Chippit" de Sa


At Dev.Mag we constantly feel a strong urge to publicise our roots. We simply wouldn't be around without our parent association, Game.Dev, a local South African initiative whose goal is to grow the local game development scene, and, like any good son, we feel that it is good to talk about our parents every now and again.

While Game.Dev was conceived as a subsection of a larger gaming forum, it has grown and now hosts its own website, acting as a front for all Game.Dev's activities. News of major Game.Dev events is regularly posted here, along with information on local development related events. It also links to Game.Dev's showcase on Great Games Experiment, where a large portion of members' games are available.

The greatest treasure of the Game.Dev site lies in the knowledge archived in the

article section. Game development wisdom is hoarded here, usually a scant few months after the same content is published in the local gaming magazine, New Age Gaming. These articles are written by Game.Dev's very own virtuoso, Danny Day, better known as dislekcia, and cover game development topics most pertinent to indies. However, the hints, methodologies and general wisdom contained in these articles are invaluable to any and all game developers.

Navigating to the links section of the site will yield a variety of local development related websites. The most important of which being the Game.Dev forum, still graciously hosted as part of New Age Gaming's forum. This is where Game.Dev was effectively conceived and the most relevant activity still occurs. Here, games are showcased and advice is dispensed, but the most notable activity comes from the regular competitions that are hosted, some of which include cash prizes. In fact, the first thing you'll notice when you visit the greater Game.Dev site is the list of the winners of the previous competition, with prizes totalling R10 000.

Game.Dev's home site is a trove of knowledge and a great place to stop for some insightful development wisdom. 



Game Maker in more detail

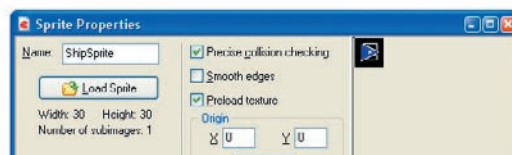
Wednesday, 16 May 2007

Last month we introduced Game Maker 6.1 as a tool for aspiring game developers. This month we expand on the basic introduction to GM by taking a look at some of the systems that give GM its power. It's important to note that you shouldn't be running GM in simple mode, even when you're just starting out. Advanced mode isn't as complex as it sounds and some of the settings and features we're about to cover are only available in advanced mode. To switch modes simply check or uncheck the Advanced Mode option in the File menu.

First up: Sprites

As mentioned last month, sprites are 2D images (as opposed to 3D models) that make up the graphics in many games. Exactly what you use to create your sprites doesn't really matter, you could render them from a 3D package, scan them in, draw them in a painting program or even grab them off Google image search. All that matters is that they look the way you want them to and they're in a format that GM can understand. This includes most popular image formats, so finding a program that can't output a format that GM supports counts as quite an achievement.

Once you've created a new sprite (either by clicking on the "New Sprite" button that looks like a red pac man, or by right clicking on the Sprites folder and adding a new sprite from the menu there) and loaded it from the original graphics file, you're presented with something rather similar to this:



Competition 15 Results

Saturday, 25 August 2007

Many thanks to [Mindset](#) for sponsoring Competition 15, the R10 000 prize pool got a lot of people excited and produced some great games and a lot of potential. Perhaps most exciting of all is the interest in taking quite a few of these game ideas further :)

And now, the official results:

1st Place: R5000

[Cartesian Chaos](#) by Evil_Toaster

2nd Place: R2500

[Rockets!](#) by Gazza_N

3rd Place: R1500

[Math Attack](#) by Squid

Best New Entrant: R1000

[Typing Tower](#) by ShadowMaster

There's a [full writeup](#), including reviews of the rest of the entries, on the Game.Dev [forums](#) as well as coverage on Fengol's [blog](#). Congratulations once again to all the winners and well done to everyone that dared to explore guerrilla learning and enter a game. Those of you who come to [rAge](#) at the end of September will be able to catch the Competition 15 prize giving on stage at 2pm on Saturday the 29th.

[Back](#)



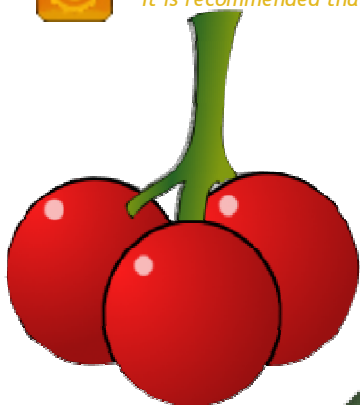
BLENDER TUTORIAL

Lucky Fruit Bonus Tutorial

By Stefan “?rman” van der Vyver



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



The adjacent images are of the Pacman fruit variety. I created the images both in 2D and 3D. Initially I thought of doing a tutorial showing you how to draw the fruit in 2D and convert them to 3D, but building them in 3D would take the same amount of time, with infinitely better results.

Seeing as this is a 3D tutorial, let us travel the road of the 3D fruit.

The idea is to create a bonus object in 3D, animate it to spin around once, and export it as an animation. This concept is especially well suited to bonuses and power-ups, in my humble opinion. If we do the animation correctly, the object will be able to spin infinitely and seamlessly.

Work flow:

- Decide on the object
- Decide on a basic build structure
- Which primitives should we use?
- Model
- Texture
- Animate the object (What is the context, how long should it take for a 360 degree spin?)

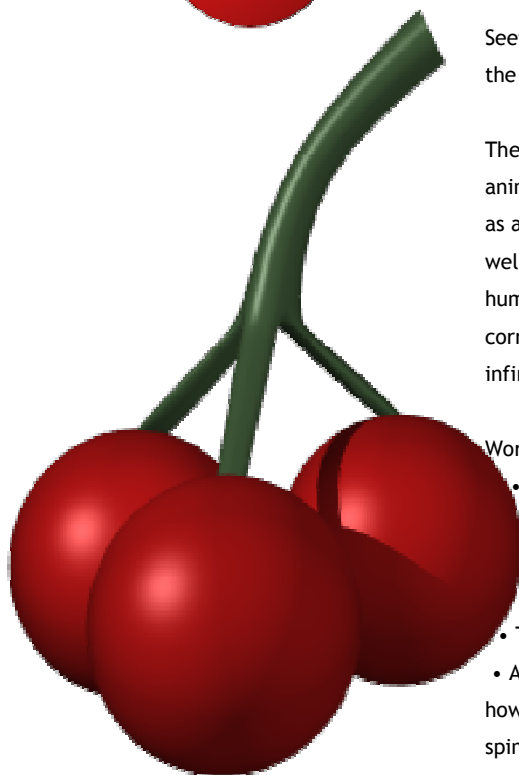
Set the render parameters

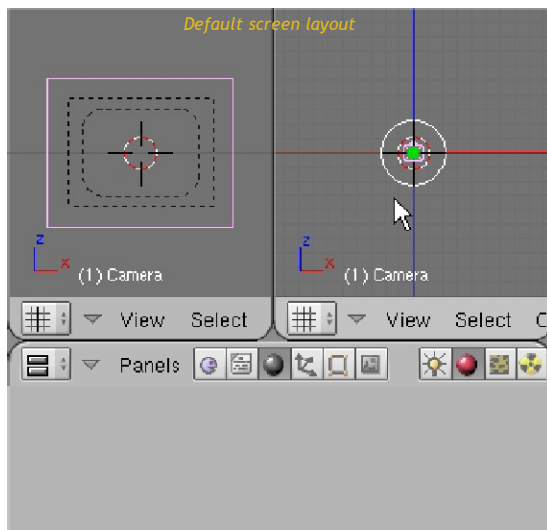
It would like to build the 3D cherries with some nice shadows and highlights on them.

To build the cherries, we will build a sphere and duplicate it three times. Then we'll build a stem that splits three ways. For the stem we will start with a simple plane in top view, and extrude it to the right dimensions.

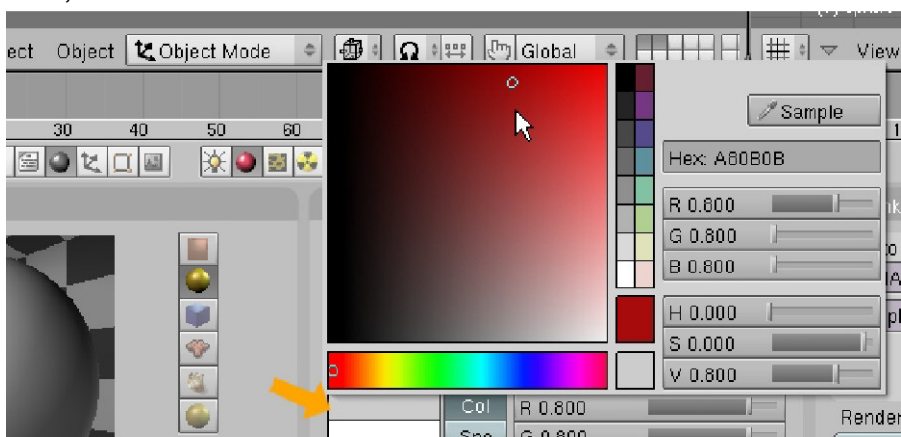
For this tutorial, I'll accept that you have the default Blender screen in front of you. If not, create 2 viewports along the top of your screen, and 1 viewport below them. Keep your mouse cursor over the top right viewport, and press NUMKEY7

We'll start in a top viewport by adding a sphere. Why the top viewport? Well, in Blender, the z-axis runs up and down. When you create an object in Blender, it is built with the z-axis pointing in the direction of the view that you built it from initially. If you start building an object from any other view than top view your object will not have the same axis orientation as the Blender world axis. Later on, that could account for extremely funny behavior when you animate.





Colour for cherries



SPACEBARKEY brings up the Add menu Add a mesh UVsphere with attributes set as above. When you add the sphere, the program takes you into Edit Mode. On screen you will see the sphere as a series of yellow dots that are connected with black lines. It would be a bad idea to move the sphere at this point. There is data related to the object, such as the center point, location and rotation that will not move with the vertices. Pressing TABKEY will take you out of Edit Mode and into Object Mode. It is in this mode that we will be duplicating the sphere to form two more cherries.

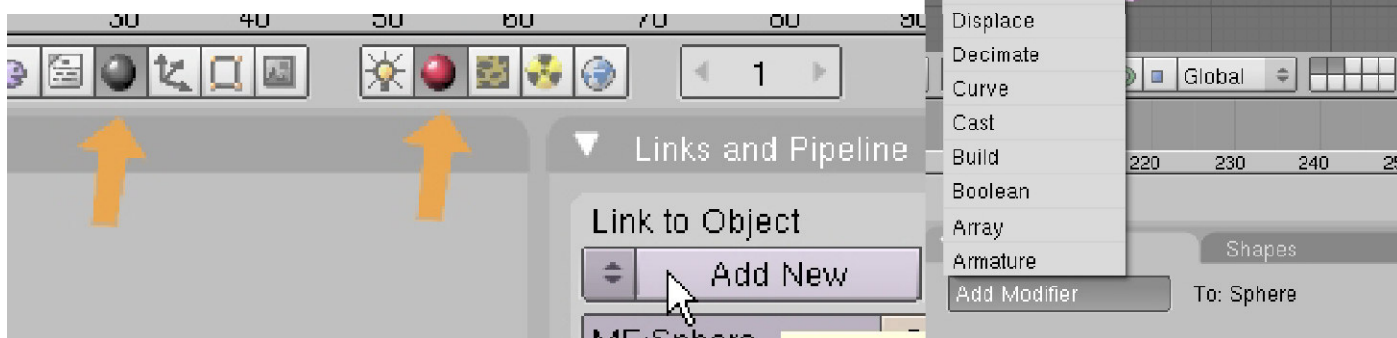
Before we create the other cherries, we might as well create the red material, and set the sphere to be smooth. Go to the materials button, and add a new material. Click on the gray color swatch, and select a nice deep red cherry color. Next step: Go to the edit buttons and click Set Smooth. The object appears to be more smooth, but we need still more smoothness. To achieve this, we add a Subsurf modifier. This subdivides

the surface of the object, giving us a far smoother appearance, without actually altering the mesh.

Make sure that you have the editing button selected. Still looking at the object from in the top view, make sure that you have the object selected (RMB) (RightMouseButton), and duplicate the object with SHIFTKEY+DKEY. After adding the "Subsurf" modifier, the object appears elongated. This is due to the way that the mesh is constructed, and is influenced by the way that the "Subsurf" modifier subdivides the mesh. My suggestion is to use the bounding box properties to get the ball into shape again. What is the bounding box? It is the smallest rectangular shape into which the whole of your object will fit. In this case, because my sphere is taller than what it is wide, the bounding box will have a higher Z value. Press NKEY to see the object properties. Look at the DimZ value. You can clearly see that the Z-axis has a different value than the X- and Y-axis. SHIFTKEY+LMB (left mouse button) on the DimZ value. Type in the same value as the DimX and DimY values. Your sphere will now be round again. You have made the sides of the bounding equal, making it square, in turn forcing the sphere to become round again.

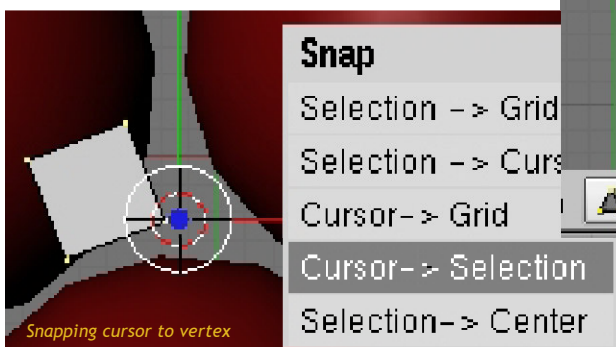
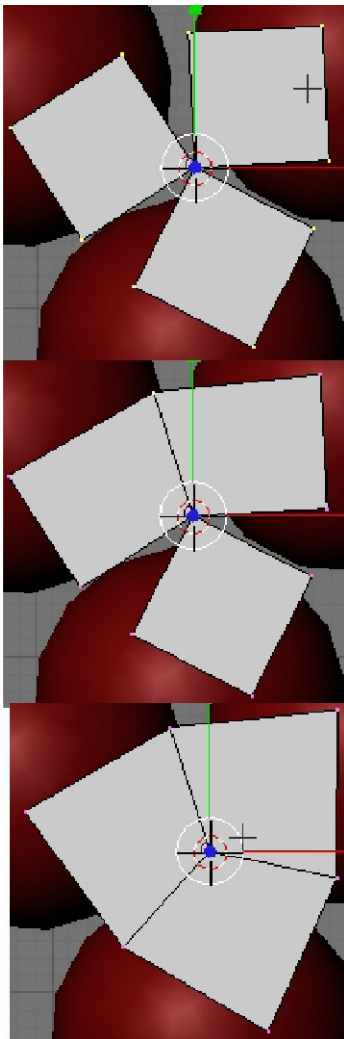
We have now changed another value - the ScaleZ-value! Many animators will simply leave this as it is. However, once again, this might cause problems later on. The best option is to apply the scale values to the object, allowing it to be addressed by Blender as a uniform scale of 1. CTRLKEY+AKEY brings up the dialogue to apply scale and rotation. After you have done this, you will see that the sphere is now perfectly round, with a uniform scale value of 1.000 for ScaleX, ScaleY and ScaleZ.

Time to duplicate the sphere. We have applied a material, set the smoothing, applied a subdivision modifier and made the object round with a uniform scale value. Make sure that you are in the top view, and have the sphere selected. SHIFTKEY+DKEY creates a duplicate of the object. Create two duplicates in the top view, and position them as you see fit. Go to front view (NUMKEY 1) and move the spheres so that they are slightly offset in the Z-axis.



Now for the stem

When modeling in 3D, we always try to keep our polygons four-sided. They don't have to be square, but we do want them to always have four sides. This leaves us with a bit of a dilemma, as we need a stem that splits three ways. To do this, we will start with a plane object (which is four-sided), then move the 3D cursor to one of the corners, and rotate the plane around the 3D cursor. Using some Blender shortcuts, we will join and delete some vertices. Below you can see a quick flow diagram. At the end, we have three four-sided polygons joined together.



The 3D cursor causes havoc in Blender if you don't understand how to use it. My suggestion is that you think of it as a place holder. When you add a new object in Blender, it will be added at the 3D cursor's position. The interesting thing is that a simple left-click of the mouse puts the 3D cursor where you just clicked. Many times, as a new Blender user, this will happen by chance, and you may encounter some unexpected behaviour because you, unknowingly, repositioned the 3D cursor. Another feature of the 3D cursor, is that you can use it to reposition objects, or have objects rotate around the 3D cursor.

Position the 3D cursor at the point where you want the stem of the cherry to split up into the three smaller stems. Then go to top view (NUMKEY7) and add a plane object (SPACEBARKEY). Scale (SKEY) and rotate (RKEY) the plane so that it fits the position of your cherries.

You should be in edit mode now. Make sure that you are in vertex mode (CTRLKEY_TABKEY). Select the vertex (RMB) as shown in the diagram, and then press SHIFTKEY+SKEY. Choose Cursor to selection. You have now positioned the 3D cursor on that vertex.

The next step is crucial. Click the Pivot point icon as indicated. Select 3D cursor. Any rotation you do now will happen with the 3D

cursor as the center point for the rotation. Select all the vertices of the plane, and hit SHIFTKEY+DKEY. This will duplicate the

Select Mode

Vertices

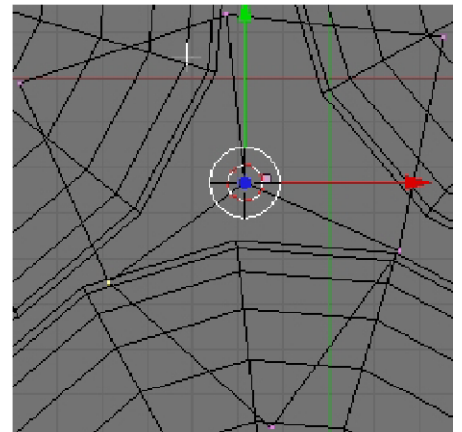
Edges

Faces

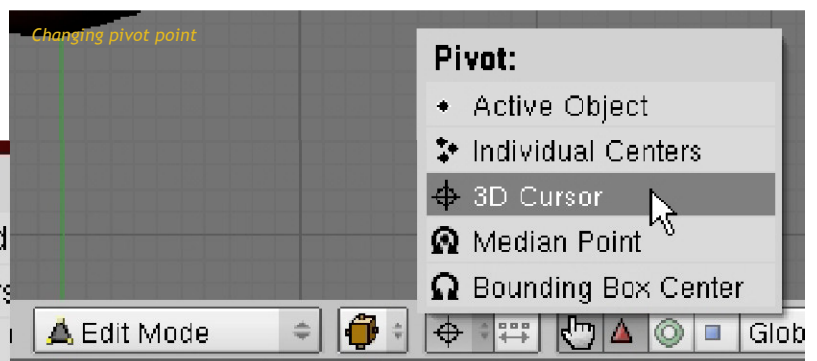
object. To rotate the selection hit RKEY. Hold down CTRLKEY to restrain the rotation to 5 degree increments. You can read the rotation degrees at the bottom of the viewport. Rotate the selection through 120 degrees. Duplicate and rotate the plane a second time.

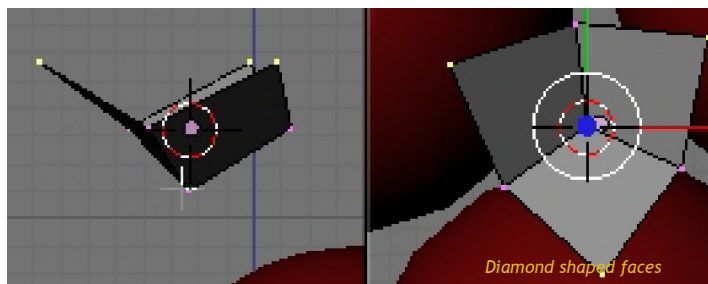
Now we need to merge or join certain vertices so that we are left with only three four-sided polygons. Change your shading mode to wire frame, by pressing ZKEY. RMB select the vertices as shown in the image, and hit ALTKEY+MKEY. Select "At center". The vertices will merge and you will receive a message indicating that 1 vertex has been removed.

Keep merging the outside vertices until you have the following result:



Now, what you cannot see is that the inside vertices are still overlapping. To merge those, deselect everything (AKEY). Box select (BKEY and LMB drag) the center vertices and press ALTKEY+MKEY to merge those vertices.





and position the extruded area so that it ends inside each of the cherries. I scaled the ends (SKEY) so that they would be thinner on the ends. Now go

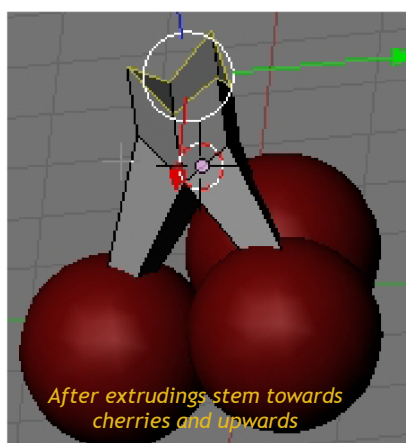
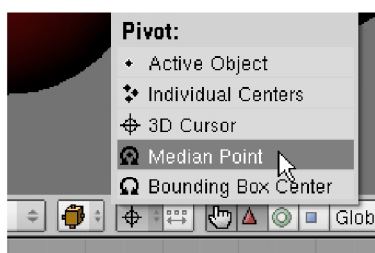
When you select all the remaining vertices, you should see the following information displayed in the title bar of Blender:

- Ve: 7 (Vertices)
- Ed: 9 (Edges)
- Fa: 3 (Faces)

If your amounts are higher, i.e. if you have more vertices, you have duplicated something in the incorrect way. Your best option will be to go into object mode (TABKEY), delete the stem object and start again with a plane.

I selected the three vertices on the outside edges of the polygons and moved them up slightly. Then I selected the middle vertex and moved it down. That left me with four diamond shaped objects, with the diamonds sort of facing in the direction of the cherries.

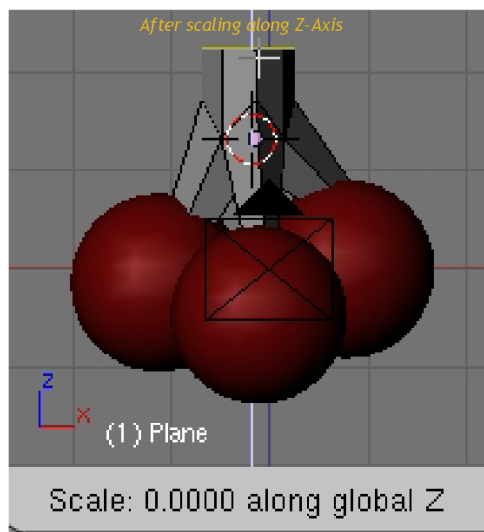
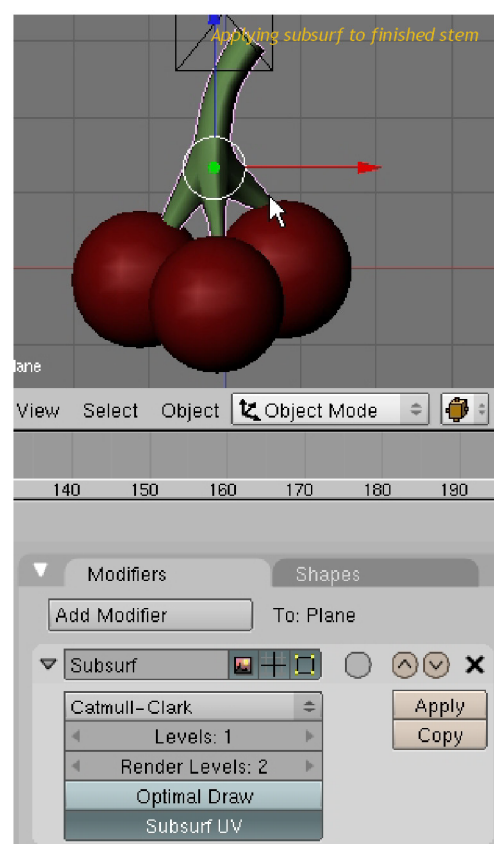
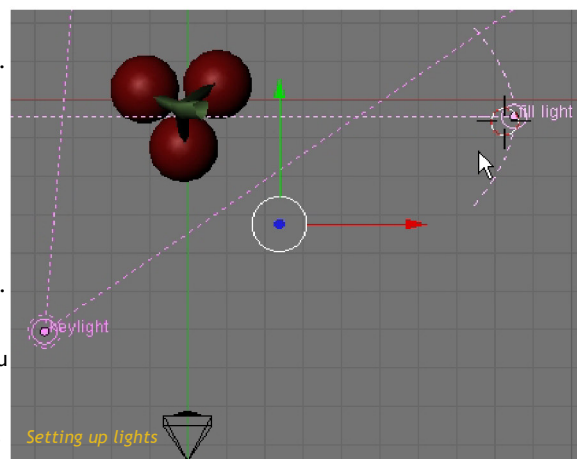
Next step is to go into face mode (CTRLKEY+TABKEY) and extrude each face down to a cherry sphere. It might be a good idea to change your pivot point before doing anything else. Now you can extrude the faces, one by one, using the EKEY. Extrude



into edge mode (CTRLKEY+TABKEY) and select the open edges at the top of the stem. Press EKEY, then ZKEY, to extrude the edges in the Z-axis. To make the end look neat, you can hit SKEY, then ZKEY, then hold in CTRLKEY while you drag the mouse. This will scale the edges in the Z-axis. Keep an eye on the bottom of the view port until it reaches zero. You can now extrude the edges until you are happy with the shape of your stem, scaling and moving vertices until you are happy with the result.

I've added a basic green material to the stem. I would suggest that you add a Subdivision modifier to the stem, and activate the "Set Smooth" button under the editing options. This will make the stem look nice. If you have black areas that look funny, your normals are probably inverted. Select all the vertices of the stem, and hit CTRLKEY+NKEY to make the normals face in the same direction.

Make sure that you have a camera in your scene or add one (SPACEBARKEY---->Add--->Camera). Change one of your view ports to a camera view (NUMKEY0). Add one or two lights and press render. I added a spotlight with shadows as the key light, and a hemi light as the fill light.

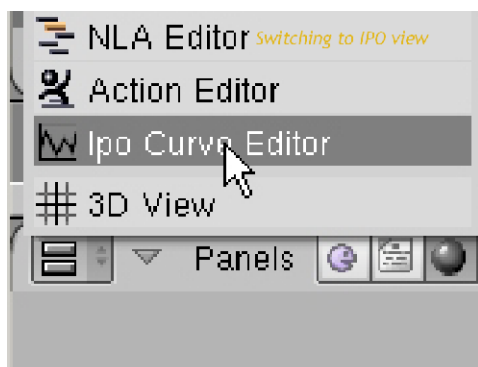
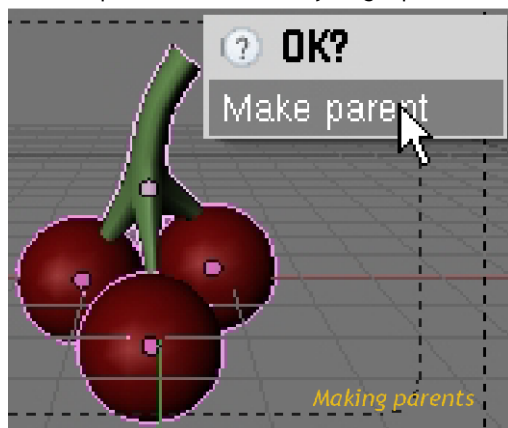


Animating the object

A good principle in animation is to use proxies, or helper objects, to animate objects where possible. It is always best to leave your object in as unaffected state as possible, in order for you to make changes to the object at a later stage, without affecting the whole chain.

In this case we will add an Empty object. We will parent the cherries to the stem, and the stem to the Empty, and then animate the Empty to spin around.

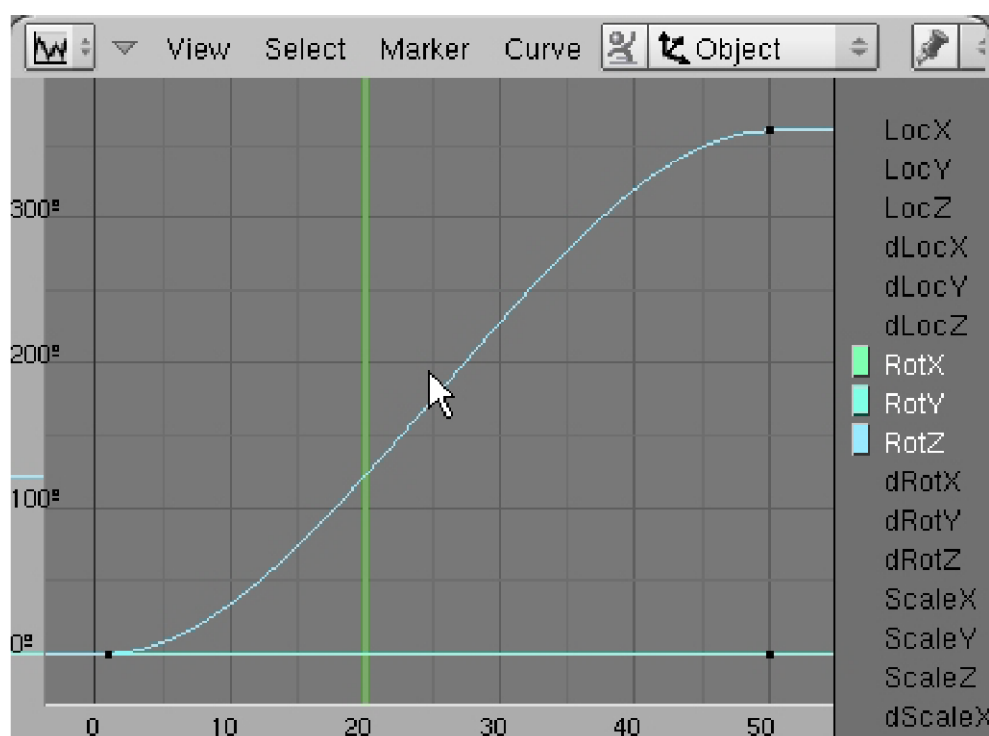
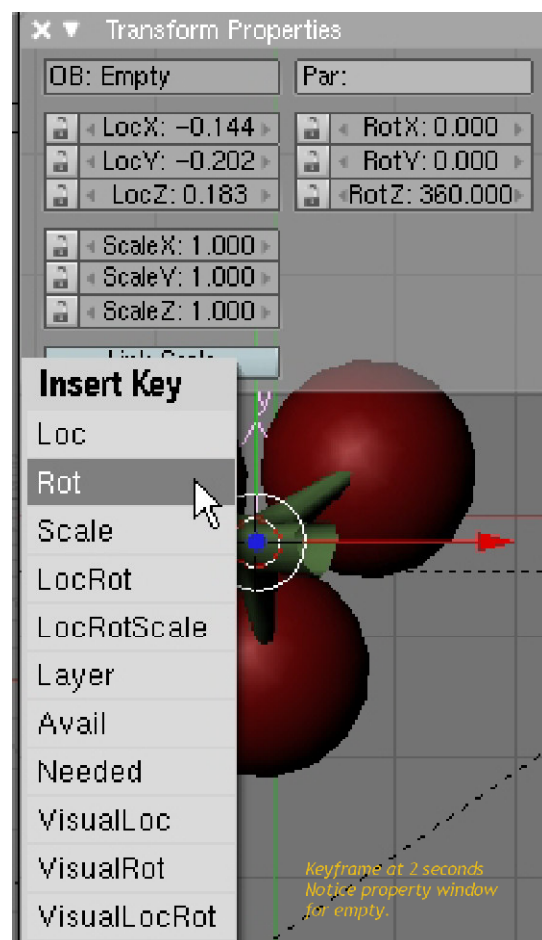
Parent the cherries to the stem. Do this by selecting (SHIFTKEY+RMB) all the objects, one at a time, and selecting the stem last. The stem should be a lighter pink than the cherries. Now hit CTRLKEY+PKEY. Now go to top view, and still with the cherries and stem selected, hit SHIFTKEY+SKEY. Choose Cursor to Selection. This will position the cursor in the middle of the imaginary bounding box enclosing all the objects. Simply put, it centers the 3D cursor on the selection. Now add an Empty (SPACEBARKEY ----> Add ----> Empty). Select the stem, and parent it to the Empty. Now we can animate the Empty. Test the Empty by hitting RKEY and moving the mouse. The cherries and stem should rotate along with the Empty. If not, retrace your steps and make sure everything is parented

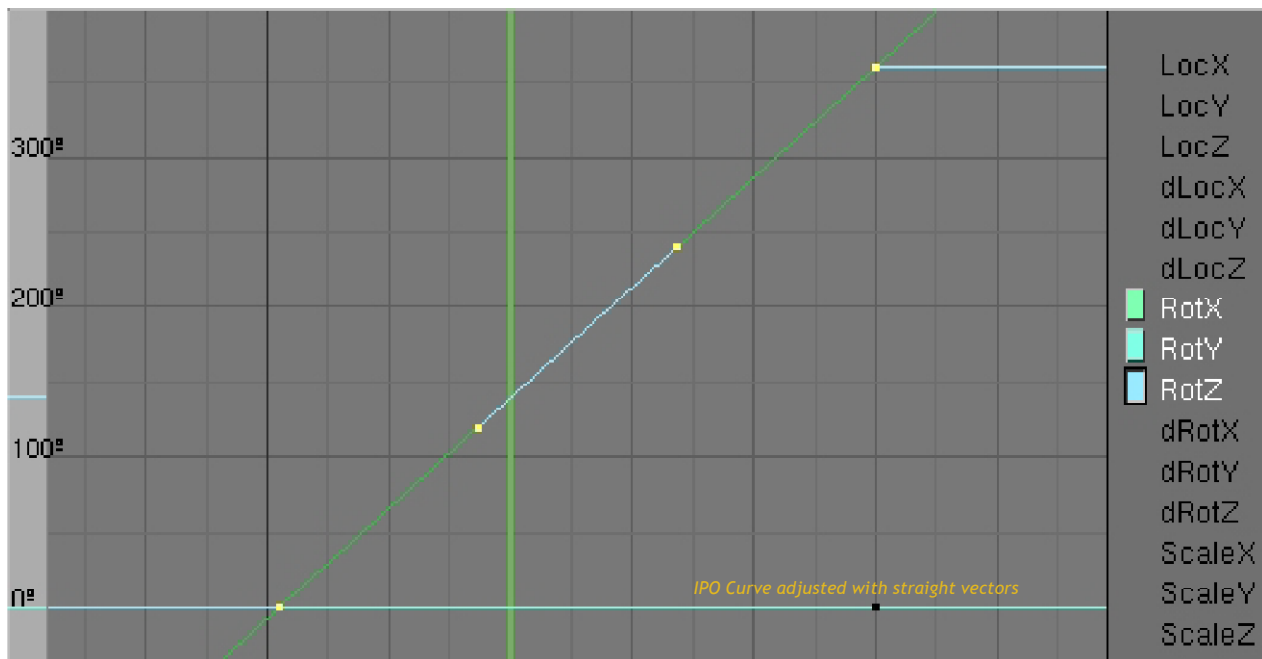


correctly. Select the Empty, and press the NKEY. The properties of the Empty will come up on screen. You should be able to see that the Z-rotation is currently zero. A full rotation will be 360 degrees, and we will use this screen (NKEY) to change the degrees for the animation. Change your lower viewport to an IPO Curve editor window.

In the top viewport, ensure that your Z-rotation (by checking the properties window NKEY) is zero. Ensure that you are on frame zero of your animation. Keeping your mouse cursor in the top viewport, press IKEY. Select Rot. This will insert a keyframe to fix the rotation at this point. You will see that the key is inserted when you look at the IPO Curve editor. Let's say that we want the cherries to rotate one full rotation in 2 seconds. That will mean that we need to do the animation over 50 frames (25 frames per second). (Press (ARROWUPKEY) to advance to frame 50. ARROWKEYUP advances 10 frames, and ARROWKEYRIGHT/LEFT advances or rewinds 1 frame at a time.)

Go to the rotation properties, and type 360 next in the RotZ space. Hit IKEY and insert a Rotation keyframe at this point. Your IPO curve should look like the image below. If your lines are running out of the screen, either zoom in, or click on "View" and select "View All".





You can now play back your animation by using the arrow keys to go to frame 1, and pressing ALTKEY+AKEY (with your mouse cursor over one of the top viewports). Currently the default settings will probably be set to 250 frames for rendering, so the animation will play, pause for a while, then play again. We'll sort that out in a moment. What is important to note, is that the animation starts slow, then speeds up, then slows down again. This is not what we want for the animation. We would like a constant speed throughout.

Select the speed curve with RMB. Hitting TABKEY will take you into edit mode. Hit AKEY until both curve points are selected in yellow. Press VKEY. This will change the character of the curve handles so that the interpolation is of the handles are vector type. This will change the line so that it runs straight between the two points. What this line now shows is a constant movement over time. This is what we want. If we want a continuous rotating object, it stands to reason that we can now use frames 1 to 49,

and simply loop the animation, or, described in other terms, the animation can keep repeating itself from frame 1 to 49. We leave out frame 50, since it is the same as frame 1.

Press TABKEY with your mouse cursor over the IPO window. That will take you out of edit mode. Change the IPO window back to a Buttons window, and select the Render Options.

In the Scene/ Render dialogue you need to set the following options:

- Output directory (where do you want to store the images? Click on the icon next to "/tmp/")
- Oversampling (smooth edges)
- Render frames (the first and last frames)
- Image size (consider how big the image will be in your game)
- Filetype (Targa and PNG supports transparency)
- Alpha channel (RGBA means "Red, Green, Blue and Alpha")

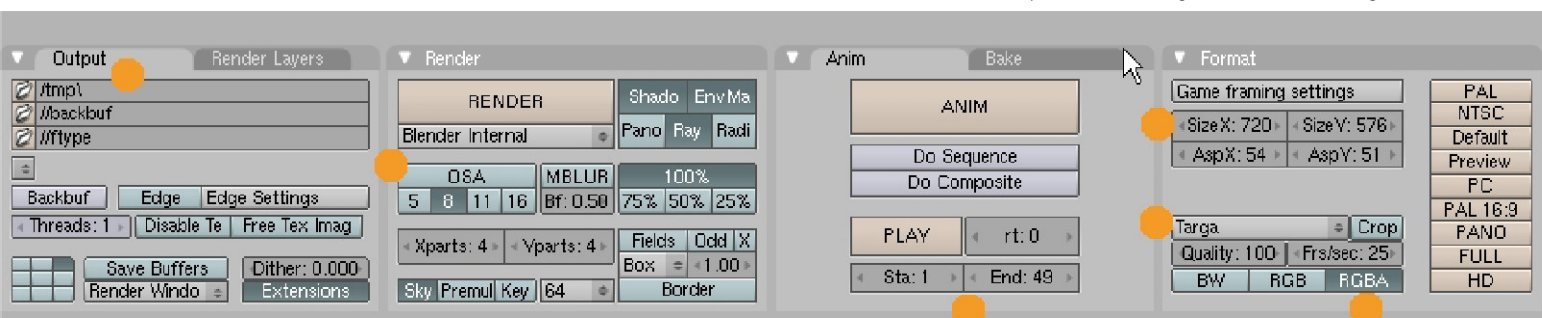
Make sure that you have a camera and lights in your scene. Without these, the animation will not render.

Click the "ANIM" button. Blender will now render a sequence of images to the directory/ location as specified. I use GIMP to composite my images as an animated .gif. There are various programs available on the Internet to do so. Most probably your game software will be able to use the sequence of images to create an animation.

Should you need more animated spinning objects in your game, you can now always replace the cherries with that object, parent the object to the Empty, and render the animation. You now have a type of template for such an animation.

Happy Blending ☺

Your final render settings should look something like this



PARTICLES

Beginner's Guide to Making Games

By William "Cairnswm" Cairns



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

This is the fourth article in the Beginner's Guide to Making Games. So far we've looked at making objects move and checking if they collide with each other, as well as different options available for setting the background of the game. This month we'll extend the fancy look of our games with particle effects.

The main goal of the Beginner's 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 logic is needed, these guides will ensure that you walk away actually understanding each of these concepts.

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 therefore 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 still be of some value to the intermediate level programmer as well.

This month's article will contain the following sections:

- A Game Maker tutorial on creating particle effects
- Other Options available with particles through Game Maker Actions
- More effects on particles through GML

Particles are only available in the registered version of Game Maker.

Particles can be used to make a game look that little bit more polished and professional - in other words, they add that bit of spice that makes a game look more than just a cluster of sprites.

Step 1 - Create a Sprite and an Object

Create a new Game Maker game containing a sprite and an object. I've used the dynamite sprite from the Maze example that comes with Game Maker.

If you need help creating a sprite and an object, have a look at previous articles in this series.

Step 2 - Create a Room

Create a room and place the dynamite in the room.

If we run the game now we have a room in which nothing happens. But now the fun stuff starts. We are going to blow up that dynamite!

Step 3 - Prepare for Particles

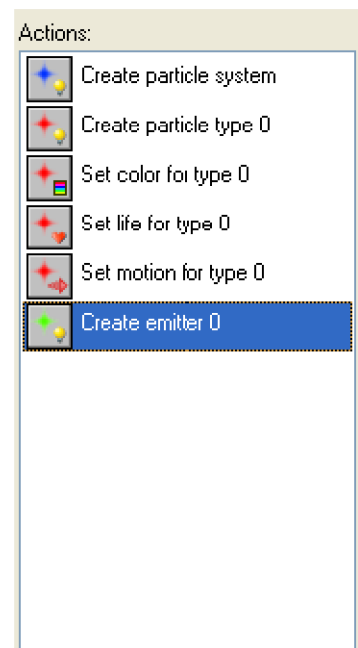
Go back to the dynamite object, and add a Create event.

Particles do not exist in Game Maker on their own, but rather exist within the Particle System object. So the first thing to do is to create a Particle System to hold all the particles we are going to create.

Just as Particles cannot exist without a Particle System, so particles cannot be created without a Particle Emitter. Particle Emitters control the way particles are created, so as to create different effects.

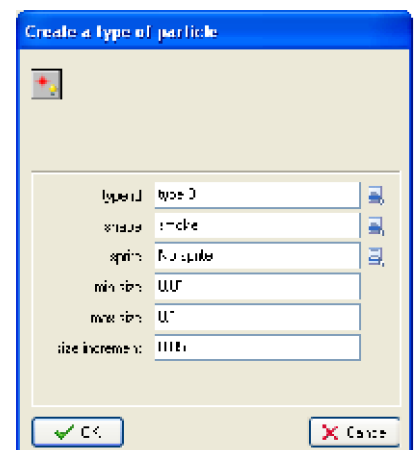


While Emitters control the way particles are created, the particle type defines the way in which particles behave.



Within the Create event, we need to create a Particle System, define the behaviour of a particle type, and declare a particle emitter to create the particles.

The Default value for the Particle System Depth can be used.



Use a smoke shape for the particle as this creates a good explosion effect. Play a bit with the sizes and size increment values to adjust the effect to your liking. For the Particle Colours, use a red and a yellow as these are mixed to give various shades of orange.

For the Particle Life set a value of min:10 to max:20.

Define the motion of the particles - for a platformer a min direction of 30 and a max direction of 150 makes the explosion appear to go upwards, while for a top-down shooter you probably want a min of 0 and a max of 360 to allow the particles to go flying off in any direction.

And lastly, create an emitter to create the particles when we need them. I quite like the diamond emitter shape, but for this example as we want all the particles to be created on the sprite it doesn't really matter what shape to use. Set the minX to x-10, the maxX to x+10 and the same for the y values.

At this point we have everything ready to create particles. The only thing left to do is to actually emit the particles from the emitter.

Step 4 - Make Particles

On the dynamite object, create a Mouse-Left click release event, and in the event add the "Burst a number of particles from an emitter action". Ensure you use the same emitter as defined in the create event and the same type of particles defined before. For a neat small explosion use a total of about 20 particles.

And there it is, a neat little explosion system for Game Maker. The same emitter and particle type can be used when two trucks collide, or a missile hits a plane, or Doctor Death's super explosive nuclear device lands on his enemy's city.

Actions - Other Particle Effects

Each of the particle actions takes a number of parameters and for each parameter there are various options for creating a different result within your game.

Particle System

The biggest extra effect that can be created by the particle system is the depth that the particle system is created at. This results in particles being either in front of or behind the other objects in the game. So for example you could explode a tank, letting the explosion be above the other ground level units, while being below the level a plane is flying at.

Also note that there is an action to clear all the particles within a particle system. This can be very useful when levelling up a player or something where the level resets itself.

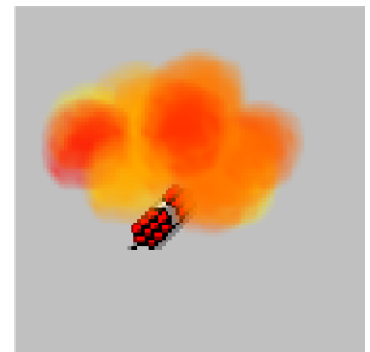
Particle Types

Game maker has a number of predefined shapes to use for your particle types. It is also possible to add your own sprites to use as particles. Sometimes it is worth playing with the various shapes before deciding on the shape to use for your particles. For example I find the smoke shape better for explosions than the explosion shape.

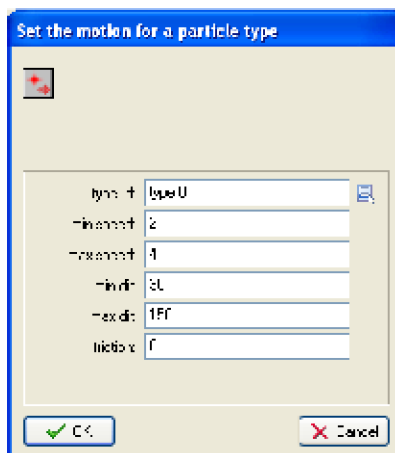
Also spend some time playing with the size of the particles. You might prefer other than the default sizes for your effects. In the tutorial above, for example, I use very small particles for the explosion but you might prefer larger particles.

Particle shapes

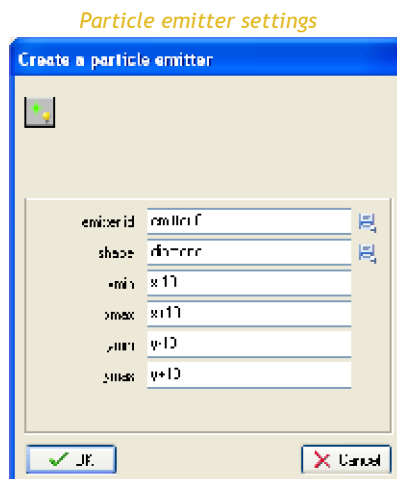
pixel
disk
square
line
star
circle
ring
sphere
flare
spark
explosion
cloud
smoke
snow



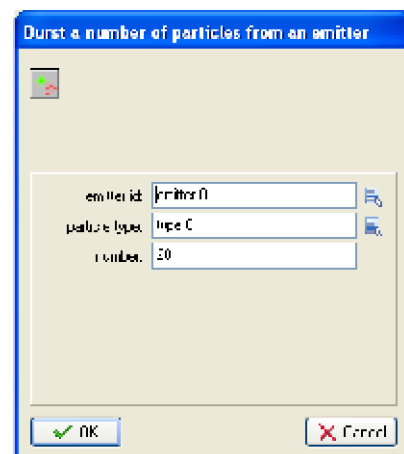
Explosive result



Particle Motion Settings



Particle emitter settings



Burst settings

Particle life and Colour

The colour facilities in the Game Maker particle system are actually really nice. You don't need to define the change of colour the particles go through, instead let Game Maker transition between the two colours and give you a wide range of effects for free.

Like Particle Type, the Particle Life Span needs a lot of trial and error to find the right values. The randomness built into finding the particle life is effective in that it results in different particles living for a different time and effectively fading the particles over time on the screen.

Particles in GML

Particles can be created and managed using GML in the same way that particles can be created and managed using actions. Just as with actions, particles in GML require a Particle System, a Particle Type and a Particle Emitter. These can all be created using GML.

Setting up for Particles

As with using actions, it is necessary to set up everything first. I prefer putting all this in a controller object (i.e. an object that

cannot be seen on the screen), and these commands in the Create event:

```
global.ps = part_system_create();
part_system_depth(global.ps, -10);
global.pt = part_type_create();
part_type_shape(global.pt, pt_shape_smoke);
part_type_size(global.pt, 0.01, 0.05, 0.01, 0);
part_type_orientation(global.pt, 20, 45, 0, 0, 0);
part_type_life(global.pt, 10, 20);
part_type_color2(global.pt, c_gray, c_silver);
global.pe = part_emitter_create(global.ps);
```

This will set up the particle system, type and emitter for a smoke trail system.

Making particles

In a little motor car's step event, we can move the emitter to where we want it and emit a particle to look like smoke behind the car.

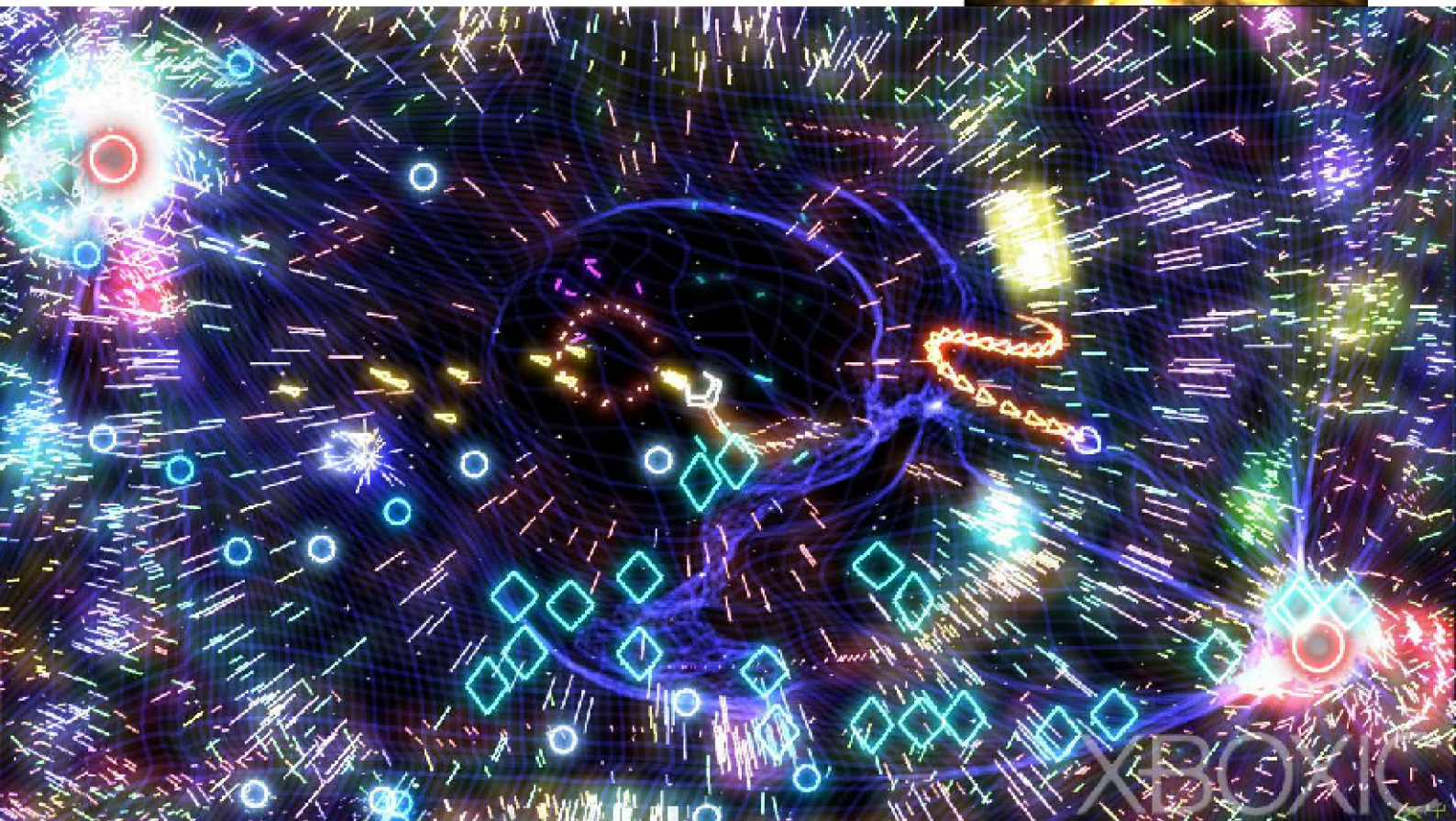
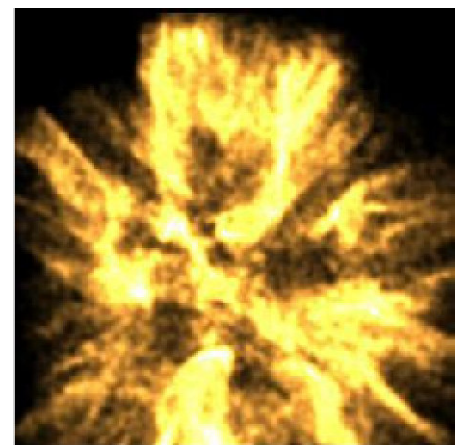


```
part_emitter_region(global.ps, global.pe,
, x+6, x+10, y+4, y+7, ps_shape_rectangle, ps_distr_linear);
part_emitter_burst(global.ps, global.pe,
global.pt, 1);
```

Now with each step we get a new particle trailing away behind the little car.

Other

There are lots of other things we can do with particles - think of cool things like fireworks, etc. Many of these can be done through actions, but as is often the case you can have more impact by using GML to control your particles. 🌀



GAME GRAPHICS WITH ADOBE PHOTOSHOP

GAME GRAPHICS DESIGN

Part 1: The right tools for the job

By Rishal "TheUntouchableOne" Hurbans

So, you're developing games with Game Maker or you want to start developing games and are on your way to making the next best game ever. To do that, you can't be using the default images that come with Game Maker or the images that you can muster up with MS Paint. You need some nice polished images to make your games really attractive and fun to play. By creating your own images, you have absolute creative control - you can design exactly what you want in the game, and exactly the way you want it to look. It also adds that professional feel to the game. You could have a great game but if the appearance is poor then the overall feel of the game could pay the price. You will learn how to create useful sprites, backgrounds and all other images related to developing a game in this series of tutorials.

To make use of this tutorial, you need the Adobe Photoshop software suite. You can download a free trial version from www.adobe.com/products/photoshop. I am currently using Photoshop CS2, so if you have a really old version of Photoshop you might need to upgrade, but the basic tools are present in most versions.

Let's get started. In this tutorial you will be introduced to the most basic tools used in Adobe Photoshop. These tools will be the

basis from which you create your game artwork. More complex tools and methods will be discussed in future tutorials.

One of the main concepts that you need to understand in Photoshop is "layers". Layers are exactly that - layers of images. If a top layer should overlap a bottom layer then the bottom layer might not be visible or could be partially visible, but we will learn later how to use layers to our advantage by manipulating them to suit our needs. The different layers currently available in the project will appear in the layers window. If this is not present in the workspace, simply press F7.

Note: Some tools can only be used on the selected layer/layers. Select the desired layer in the layers window. Select multiple layers by holding the Ctrl key, and choose the desired layers by clicking on them. Tools that are not essential for images for game development have been omitted. Shortcut keys appear in brackets next to the tools name.



Colour Box: This shows your currently selected Background

and Foreground colours. You may select any desired colours and store it in this box. You

can quickly swap around the Background and Foreground colours with the X key.

Move Tool (V): This tool is used to move the different layers in the project. Its purpose is to move any layer or selected piece of the image as long as the layer is not locked. A layer is locked when a small "padlock" icon appears next to the layers title; the background layer is usually locked. You wouldn't need to move it but if the need arises, you can simply right click and select "Layer from background".



Marquee Tools (M): These tools are used to make selections in an image. The different tool shapes allow you full control over the

parts of the image that needs to be selected. The selected area stencil can be moved without modifying the layer at all - just hold mouse click and drag, although if you would like to cut a piece of a image to create something else, you may do so by selecting the Move Tool while the desired area is selected, and drag the piece of image. (Note: you cannot cut if more than one layer is selected but if the need arises, you may merge the layers by selecting all desired layers>right-click>merge layers or pressing Ctrl+E).



Magic Wand Tool (W): This tool is another select tool; it can be used to select different areas on the image based on the flow of colour in that area of the image (also known as the colour range). You may select/deselect the Contiguous option to determine whether or not the colour selection is localised. You can also adjust the colour tolerance - you may only want to select a very particular shade of green, for example, or all shades of green indiscriminately. Everything but the currently selected area can also be selected by: right-click>select inverse.

Lasso Tools (L): The Lasso tools

are another set of selecting tools, but they give you more control over the detail of the



selection. There are three types of Lasso tools, namely the basic Lasso tool, the polygon lasso tool and the magnetic lasso tool. The lasso tool is a freehand tool used to select any desired area on the currently selected layer - this is probably the best tool when using a stylus. The polygon lasso tool draws a straight line between each click. The magnetic lasso tool is attracted to the nearest solid contour line, and will move along the line as the cursor is moved. You can fine-tune and add to your current selection by holding down the Shift key when drawing a new selection, or subtract by holding down Alt.



Crop Tool (C): This tool is similar to the marquee tool and is used to change the angle of a selected area of the image and

disposes of the rest of the image. The crop tool does not apply to a specific layer; it applies to all the layers and will change all layers respectively.

Brush Tool (B): The brush tool is used to paint on the canvas; it will paint the selected colour in the colour box onto the selected



layer. The size of the brush, texture of the brush, hardness of the brush, the opacity of the brush (transparency) can be adjusted in the options tool bar located on the top panel of the screen, as well as hotkeys. Quickly

change the size of your brush, for example, with the [and] keys, or the opacity by tapping a number between 0 - 9. The flow can also be adjusted. This is the smoothness of the brush as you paint over the canvas. There are also various modes related to the brush that can be used. Experiment with them to see the difference in their use.



Pencil Tool (B): This tool is the same as the brush tool. The flow of the pencil cannot be adjusted though. It is usually used for more detailed manipulation.

Hint: By holding the Shift key and drawing with the brush or pencil tool, you are able to draw straight lines.

History Brush Tool (Y): This tool

can be applied to one layer technically but it is seen on all the layers before the current layer. This tool “paints” desired areas on the image to the state of when the image was first open for editing or back to a predefined point. You can define this point by opening the History window, and checking the History Brush box to the left of the desired step.



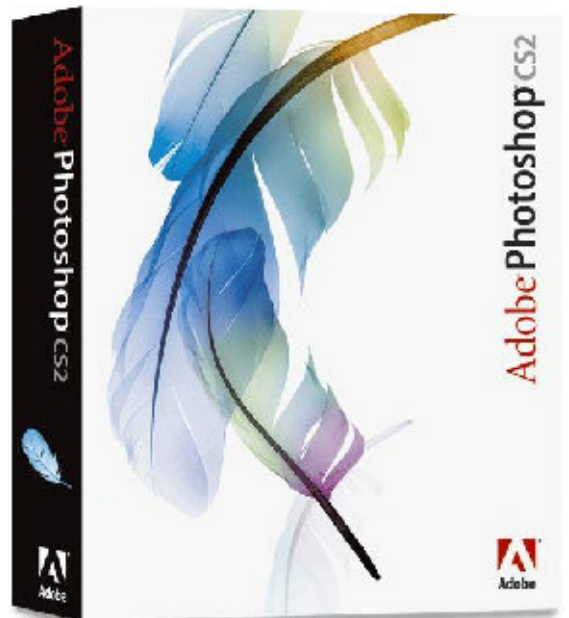
Art History Brush (Y): This tool has various “artistic” styles that can be applied to the tool when using it. This tool “distorts” the current selected layer using the selected style.

Gradient Tool (G): The Gradient

tool applies a gradient to the selected layer. The colours and styles can be selected and customised to achieve the effect that is needed. Spectrums of colours are available to make your images bright and attractive.

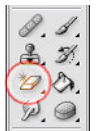


Paint Bucket Tool (G): This tool colours a selected colour range in the layer, it will use the colour currently in the colour box to replace the colour that it is applied to.



Eraser Tool (E): Yes, you guessed it - this tool erases parts of a selected layer.

Magic Erase Tool (E): This tool erases a section of the image where the colour is closely matched; it is basically a magic wand tool that erases.



Blur Tool (R): This tool will blur a desired area on a selected layer, it is commonly used to smooth-out rough edges on an image, to make the image look cleaner cut and polished in some cases.

Sharpen Tool (R): This tool does the opposite of the blur tool; it “sharpens” the selected layer. This reduces the colours in the image to simpler lines with basic colours if used excessively on the same area of a layer.



Smudge Tool (R): Everyone knows what smudging is; in Photoshop it is not much different. If you observe closely while smudging in Photoshop, you will see that is the blending of colours in the space where you have applied the smudge tool. It works well to blend colours that are slightly mismatched.



Dodge Tool (O): This tool increases the contrast of a desired area in a layer (i.e. It lightens the image). It is not

effective on extreme colours such as black and white.

Burn Tool (O): The burn tool decreases the contrast of a desired area in a layer, the area is darkened and if used excessively in one area, it will result in absolute black hence absolute black will not be affected by the burn tool.



Type Tools (T): These tools allow you to type text and add it to your image. Text can be written horizontally or vertically (if the vertical type tool is selected). The text forms a new layer in the project. The text colour will be the current colour in the colour box.

Mask Type Tools (T): These tools also allow you to type text and add it to your image but it only displays the text's mask, this means that the "outline" of the text will be selected and it can then be edited and manipulated. It does not form a new layer, if necessary, a new layer can be created manually by pressing Ctrl+J.



Shape Tools (U): The shape tools allow you to draw many different types of shapes and create them

as a new layer. These could be particularly useful for developing simple but polished sprites for a game.

Pen Tool (P): This tool is used to create paths in an image. The paths can be used in many different ways. It can be used for selection purposes, drawing shapes etc. There are various sub-tools linked to this. You can experiment with them to get a better understanding to this set of tools.



Eyedropper Tool (I): This tool allows you to select a colour from an image; this colour will now replace the foreground colour in the colour box. The colour can now be used in editing the image.



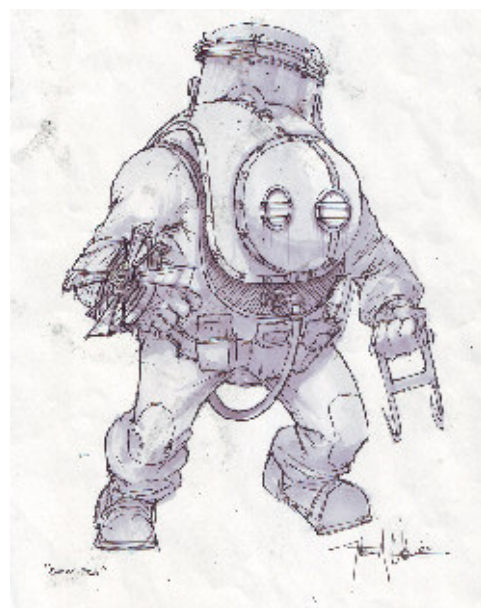
Hand Tool (H): This tool is used to navigate around the entire image, just click and drag. It is particularly useful when the zoom tool is used (see below). You can quickly jump to this tool by holding down the Spacebar.



Zoom Tool (Z): This is pretty obvious, this tool zooms the

image to your requirements, this tool is important if detail is required. This is a very useful tool indeed.

There you go, that's it, you now know the basic tools used in Photoshop for creating game graphics and their fundamental uses. I suggest that you play around with each and every tool mentioned. Once you get the feel of each tool and understand their functions, you will be ready to tackle the second part to this series. Good luck. 🍀



GOING SLOW-MO

How to adjust speeds in your game without grinding to a halt

by Rodain "Nandrew" Joubert

There's something in the gaming world that seems to be best known as the "matrix effect". People aren't talking about anything technical or mathematical here, they really are referring to that pop-culture, slow-motion, bullet-dodging, my-name-is-Neo phenomenon that seems to be all the rage recently.

An idea that a lot of developers are swooning over at the moment is to have slow-motion effects in their games. Whether it's some world-changing Max Payne tribute or just a means to emphasise a radical explosion, slow-mo is something that is attempted often, yet rarely executed flawlessly.

This tutorial is hardly the de-facto standard in successful slow-mo implementation, but it can steer a few wannabe programmers away from some common mistakes. Those who read on will also be offered a fairly reliable method of securing snail-paced gameplay.

The following code doesn't aim itself at any particular language or tool. Instead, it's hoped that the glorious secrets you uncover while reading here can be applied universally, so you should be able to fire up anything from C++ to Game Maker after this and gain the greatest benefit.

Mistake the first: frame-skipping

Frameskip. This is usually the first stop that an enthusiastic programmer will make in their quest for slow-motion success. With pun fully intended: skip it. It may be quick, it may be cheap, but it's also nasty and really looks awful.

The premise behind frameskipping is simple: quite a few games (usually the 2D ones) run at a respectable 30 frames per second. This

is a regular rate for a regular speed and some pretty regular gameplay. To slow a game down, some developers are tempted to grab the frame-controlling code and mess with it, which is generally a bad idea. In some instances, this may be a system timer, forcing a wait of several milliseconds after a single iteration of the main game loop code has been completed. In other cases, it could be a "wait for refresh" method that depends on specific hardware events (though this is rather archaic and inefficient in itself). If your development tool is particularly friendly, you may even find a handy function such as `framerate()` that happens to trigger all the dirty maths for you. Some programs (such as Game Maker) simplify this even further by allowing the room speed to be controlled by merely changing a variable.

Interfering with these sections of code will have several consequences. First off, your game will become slower. This is what the programmer is looking for, and has succeeded in doing. However, the side-effect of this is that your game will get a lot choppier as well. Halving the game speed in a program that runs at 30 fps will set it to 15 fps. This

is VERY noticeable for the player. Moreover, to gain any truly appreciable slow motion it's likely that you'll need to slow the speed down to between 20% and 10% of its normal rate. This means as little as three isolated frames of action every second. Yikes!

Some programmers at this stage decide to get clever and up the game's normal framerate to a ridiculous number like 300, so that the decrease in framerate isn't as easily noticed by the player. This is also a bad idea - your machine effectively needs to become ten times as powerful to run all the game calculations, which it would only usually have to do 30 times a second.

Mistake the second: manual labour

Programming well isn't just about putting down the code to make things run as quickly as possible. Sometimes, it's important to



have physical brevity in your code file as well, both for readability and debugging capability.

Unfortunately, while most people know about optimisation and getting their programs to run faster and more efficiently, few acknowledge the importance of the latter point and suffer massively as a result.

A trap which a surprising amount of people fall into is poor planning. They decide to implement slow motion capability after having programmed a fair amount of the core game already, or have implemented it rather patchily from the start, and realise only then that they've effectively "programmed themselves into a corner". In practice, this is the problem of people implementing quick fixes: they want to slow down the hero and the monster that they've created so far, so they decide to enter the code of the hero object and the monster object in turn, and put in a line or two (usually an IF statement) to manually set the speed. Simple, right? Well, only for the time being.

Such people usually get the idea of having a global variable to determine speed. Something like `global.goslow` to satisfy the aforementioned IF conditions. This is on the right track, but simply isn't good enough. In a game where hundreds of objects may eventually exist, developers have to repeat those "simple" lines they've hard-coded into the hero and monster, as well as all the associated code that eventually crops up for every monster, weapon, bug or blood particle that the programmer adds to the game. There's enormous room for error here, and if the programmer ever decides that they want to change the slow speed from 30% to 25%, they either have to hunt through all of the code and change everything that's been done already, or simply be content to remain with 30%. Not good.

Mistake the third: universality

Regardless of a programmer's methods, they sometimes make the mistake of adopting the slow property to every entity in the game.

You may not think this is a bad thing to start with, but it soon becomes problematic when you realise that the entire game, rather than just the game world, has become sluggish.

For example, animations on a character's HUD may be slowed down undesirably. Tool-tips which appear after holding the mouse cursor over an object for one second may now only appear after ten seconds. A popup menu which usually expands in moments suddenly takes ages to show up.

To avoid this, each object needs a class variable indicating whether or not it can be slowed down (defaulting to true or false, depending on personal preference), rather than blanketing the entire game with "slowcode".

A much better idea

An effective method to slow down your game is contained in a simple global variable: for purposes of this article, we'll call it `global.gamespeed`. The idea is that this



variable is by default set to a value of 1 (full, or normal, speed). When needed, it can be set to a fractional value such as 0.1 (for one-tenth normal speed) or even a greater value such as 2 (which would in fact double the game's speed). It needs to be built into your game from the start - if introduced later, it could end up forcing you to perform repetitive chores or even prompt a restructuring of your program. Either way, that would be far more work for pretty much the same result.

Once you've created your global variable and had it set to 1, you need to examine both your development tool and your programming know-how. If the situation permits it, you should create two basic template objects: one called **NoSlow** and the other called **YesSlow** (or, preferably, some names which are a little more inspired). All other objects in your game should then be set to inherit the traits of one of these two templates. The idea is to then place code in **YesSlow** which sets spatial and quantitative object values to be affected by `global.gamespeed` (something to be repeated every game step, like `objectSpeed = objectBaseSpeed * global.gamespeed`). **NoSlow** has no such code included.

If this is not possible, or simply too confusing, then an equally viable (though not quite as classy) method is to make sure that whenever you'd usually type an object's spatial



movement rate (for example, `speed = 5`) you instead type in code that factors in the global variable (`speed = 5 * global.gamespeed`). Make sure that you put this in a section of code that updates with every frame, or at least updates whenever a change in game speed is registered.

This is a reasonably effective technique, but be warned: holes in your logic can still appear if you're not careful. Although most people will know to change the movement speed of an object, there are other factors which are often left unconsidered:

-*interval-based events* (such as how many game steps must pass before a gun fires)

-*<value> per <time> events* (such as how much air you lose for every step you're underwater)

-*rate of velocity change* (how quickly you can skid to a halt)

Moreover, while this technique is good for slow motion, additional care must be taken if you're planning on speeding the game up beyond normal. One of these problems, for example, is collision detection - if a character is moving at ten times its normal speed, it can run right through a narrow wall without ever registering a collision, because the wall was thin enough to "skip over" entirely. Another issue crops up when a gun usually fires a bullet every single frame, and is suddenly required to shoot twice as fast. Either the programmer forgets to cater to this at all, or has to adjust the code to have multiple bullets fired in that single frame - and not occupy the same space, either.

There may very well be more efficient ways to handle speed changes in games, and you're welcome to try your own techniques. What's been offered here, however, is a reasonably solid technique, along with a few pointers for avoiding bad habits. If you're a newbie to slow motion, or maybe are just looking for something better than the old framechange habit, then maybe it's time to dodge that bullet and create a masterpiece that even snails would outrun. 🎯





www.ultimategfx.co.za

The ultimate graphic design community.
Take your game design to the next level.

Line Wars

A game project analysis

by Rodain "Nandrew" Joubert



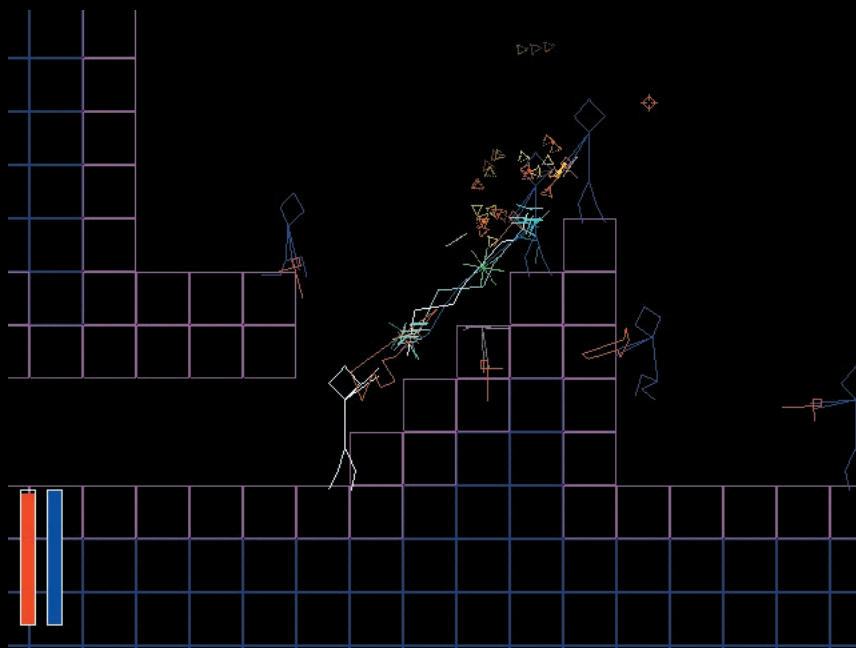
This article refers to resources available at the "Contents" section of the Dev.Mag website (www.devmag.org.za). It is recommended that you visit the site and download these resources.

I'm a programmer. Or at least, I do some dabbling and study it a little. I'm also a game designer. I believe that the projects I crank out have a reasonably good premise, work well and often bring something reasonably new to the table. One thing I am not, however, is an artist. Back in the day, I did everything in my power to make every one of my games a little visual masterpiece, which provided some rather gruesome results - great for a horror game, but not so wonderful when I was trying to craft Pony Island 5.

That has recently changed, and one of my favourite game development projects is something I worked on a bit last year. It was called Line Wars. The idea was simple: I forced myself to use lines. For everything. All artwork, all sprite animations, all particles were reflections of this geometric marvel. Did it look great? Well, that depends on your definition of great. I'm not an artist, no matter what sort of tricks I try, and I'm lazy to boot. But did it look better than anything else I'd done before? Well, sure!

Line wars was a simple platform shooter, complete with that all-too-awesome "bullet-time" effect and enemies which could be blown back (and apart) by the various weapons. The plotline was terrible - in fact, almost pointedly so - but it was great to jump, double-jump, go slow-mo, and let forth a devil-may-care salvo of liney death at stacks of hapless enemies.

In these pages, I'll present a postmortem of the game in the traditional sense. In other words, it's a nice little piece with some nice little headings, waxing lyrical about the pros



and cons of the game's development - what made it strong, and where it got pulled down. All in my own humble little opinion.

At this point, I suggest that you follow the instructions at the top of the page and head on over to the Dev.Mag site to grab the game. It's not a complete project, but there is one mission available consisting of six levels to play through, and having a go at it will help you understand what's going on in the rest of this article. That, and it's not half bad to spend some of your hard-earned time playing it.

What went right

The graphics

The main idea of this game was to make something reasonably pretty without having

to insert even a modicum of artistic talent. Great success, here. The entire game ended up consisting of lines with the exception of text, though even then I tried to use a simple-looking font. The art resources for a complex forest scene were as simple as a few dozen-lined creations in the closest drawing program handy, and I could place props and doodads liberally without the risk of everything looking overly ugly - if the player could accept that the game was based on lines, then the rest was in the clear.

In some instances, using lines actually made things better. Stylistically, particles such as explosion effects and bolts from a lightning gun suited the line-based environment really well, and for some reason a whole lot of other elements just seemed to click as well. I could make my backgrounds more varied and interesting because it was so easy to generate new material. However, the true

shining point came with elements such as enemies and even the player character itself. Instead of drawing frames to represent each of the individual motions (crouching, jumping, running, etc.), I decided to patch together a rudimentary vector editing program and constructed character “models” using points and line indexes. Once I’d created one frame, I’d save it to a file and simply adjust the co-ordinates of the points already in place to strike a different pose, then save that to another file.

The true value of this approach came when I threw the models into the game. Instead of simply flipping between frames of animation, such as when a character was spinning through the air, I implemented a crude method of tweening to make the points in the model move gradually from their original positions to those of the destination frame. Tweening is a method most commonly in the Flash application, and is basically the concept of inserting multiple frames of animation between two preconfigured ones.

The result was cool. I produced seven frames in all for the main character for his entire range of movements, but still had a smooth set of animations to show for it.

The slow-mo

My “bullet-time” plan turned out to be a huge plus for the game. I put a great deal of care into it, making sure that the project was geared towards the eventuality of slow motion from the moment I started throwing in the object code. The tweening for character models went hand-in-hand with this. Instead

of being forced to draw extra frames to keep animation fluid when the slow motion struck, I was able to simply sit back and let the tweening do its work. My life was made much easier as a result, and things generally looked prettier.

Specific areas of the game were also heavily laden with explosive barrels and the like - when struck in the midst of bullet-time, the resulting explosions looked far more spectacular, especially if the player had to dodge fiery death in the middle of it. Other game aesthetics were geared towards this sort of effect, such as air flow from vents or elaborate rocket trails.

I also fiddled about with the practicalities of having slow motion, aside from the greater response time that it afforded players. I eventually settled with the simple option of allowing the fire rate to remain the same even after the rest of the world had slowed down. I advertised this feature in the game’s tutorial level, and otherwise just put my faith in the fact that the player would catch on to this benefit after some experimentation.

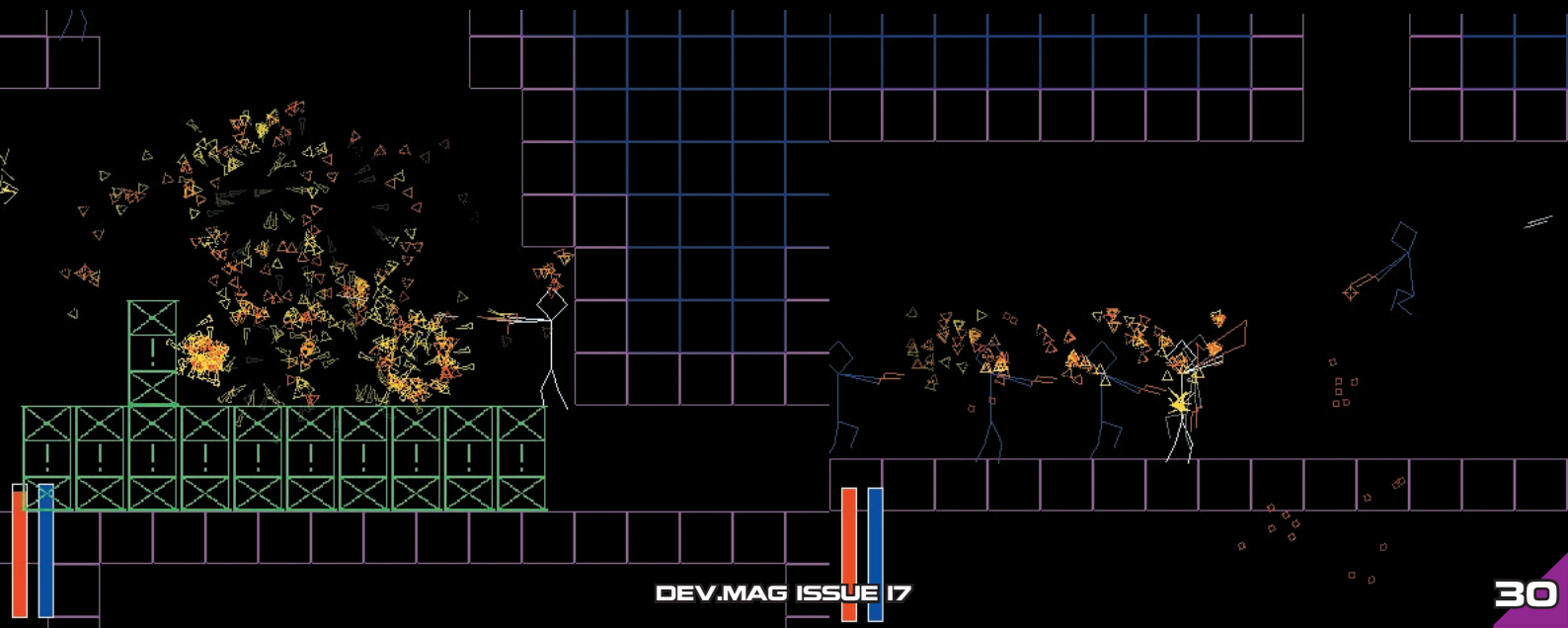
Player motivators

The slow motion also tied in with one of my other game goals: fast-paced action. Yes, the irony of putting those two phrases near each other is cringeworthy, but my reasoning is valid. Upon defeating an enemy in Line Wars, a small energy pod would be dropped, which needed to be retrieved quickly before it disappeared. This recharged not only health, but slow-mo “power” as well.

The idea was inspired by a similar dynamic present in American McGee’s Alice, and one which I instantly wanted to put into my own game. It granted the player the incentive to not only kill enemies rather than simply running past them, but to also charge in with guns blazing rather than picking enemies off from a distance or while hidden behind cover. If they didn’t grab the pods in time, they didn’t recover energy. Considering that I made these items the only source of health in the game, the need to go in for close contact became quite considerably heightened.

Did this make the game a mindless shoot-and-hope fest? Not at all. All that players had to do in the midst of a heated conflict was hit that handy bullet-time trigger and make sure that they collected the pods in good time.

Additionally, I had about eight different weapons programmed into the game (although not all of them are available in the currently released demo), most of them geared towards aesthetics as well as functionality. Although the weapons acquired in the later stages of the game were superior in terms of damage done per second, I tried to make ordnances with fun and varied uses. Grenade launchers could send bouncy (and deadly) projectiles through narrow openings or over walls to gib enemies on the other side. A flame thrower was included to set enemies on fire and stop them from shooting. A sniper rifle could deliver multiple headshots with one bullet. Even the lowly shotgun, if it struck an enemy with full force, could send unfortunates flying across the game screen. All of these effects were aimed at mixing usefulness with player satisfaction.



What went wrong

Programming flaws

The tweening process involved in making smooth animation ultimately cost me. Heavily. It made characters look smoother, and allowed for some humble physics and ragdoll-esque effects, but I did it poorly and without due regard for consequences. This was the most mathematically heavy aspect of the game, and in a tragic turn of events my game got slowed down whenever too many enemies came on-screen.

In this one instance, I honestly regretted having used a framework like Game Maker to create Line Wars. Most of my coding was done in a high-level GML interpreter, and I wrote a lot of custom lines without fully understanding the implications of such. Trig functions and way too many processes involving the default floating-point mathematics created a bottleneck which I struggled with for a while before giving up on. It's likely that I could have imported an external Game Maker library to solve the problem for me, but I feared that this would involve a radical restructuring of the game. Considering that I only discovered the extent of this problem after a good four or five levels had already been created, I was understandably reluctant to start over on core game mechanics.

I lost a significant amount of my drive to finish the game after realising that I couldn't

get around the issue. One of the ideas that I had envisioned at the outset was to have swarms of enemies in the game that could be mowed down by heavy firepower. This was no longer a possibility. It also interfered severely with level design, which now had to work around clustering too many enemies in the same area.

No design document

Foolish as I was, I decided not to put my game on paper before I put it on the screen. I had weapons, basic storyline, slow-mo dynamics and even bosses neatly organised in my mind. I believed this to be enough, and set about working on the first few levels without even realising that there was a gross portion of the project that I had NOT catered for.

As a creator of games which are very often small in size and scope, I tend to skip over the design doc process at times. My reasoning behind this is that I have the entire game in my head already - I only want to make five stages with two types of monsters in them, and the game is of a type where there isn't much room for error or finicky gameplay issues. This mode of thinking stabs me in the back when I turn to larger projects, and often makes for an unfinished game that basically has a nice starting point to its name.

My biggest problem was level design. Although the performance issues mentioned

earlier were a considerable thorn in my side, I started off with very little direction regarding how I laid out individual portions of the game. The result was that I expended most of my creative energy in the first few levels that I coded, but quickly realised afterwards that I was running on a proverbially empty tank. My level design soon became uninspired and generic, and I started moving from on section to another without any real plan of what I was doing. It became tiring and I soon became disinterested.

There were several other aspects that a lack of design docs tended to hamstring, but I felt that levels were the biggest issue.

Summary

I feel that Line Wars is a good attempt, but it needs to be reapproached. The game has a solid core dynamic and the demo is reported to be incredibly fun, but such an idea would not hold over for the long term without due consideration for its weaker areas. A better understanding of the tools I was using was definitely needed. A design document would have been invaluable. I had neither, and these two flaws in my war plan combined to drive me off the battlefield long before the project was complete.

Once those obstacles are overcome and I have a fair amount of time to commit, Line Wars shall truly strike with a vengeance. 🎯

GAME.DEV COMPETITION 15

The who, the how, the why, the winners

by Danny "dislekcia" Day

The Game.Dev organisation recently wrapped up its latest competition, focused on making educational games and offering impressive amounts of cash to the winners. After much pleading, begging, biting and scratching, we got the head of Game.Dev, Danny Day, to write us a few words about what's arguably South Africa's most high-profile set of game development competitions. This is his field report on Competition 15. Grab the popcorn and enjoy, and remember to hop on over to www.gamedotdev.co.za for more competitions, articles and important updates!

After NAG stepped in out of the blue last year and decided to sponsor R10000 for Competition 10 - which coincided perfectly with rAge 2006 - I've tried to make the idea of a yearly "big cash" competition a goal for Game.Dev.

Mindset had gotten in contact with me earlier in the year after asking around about educational games and how anyone would go about building them locally. After meeting with Dylan and co a couple of times, it was perfectly natural to posit the idea of them sponsoring a competition with rather unique



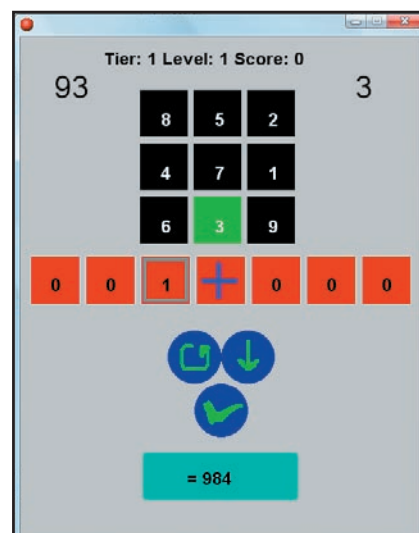
rules and see how things turned out. Unique rules because Mindset is an educational support organisation, they regularly send out curriculum support material via a customised content network to hundreds of schools around the country. They're also responsible for local educational DSTV channel 100.

After ruminating on good ways to get young developers keen on building something educational without having them get turned off by the horrendous legacy of "edu-tainment" (which is quite possibly one of the worst ideas ever), the idea of Guerrilla Learning popped up. Essentially, Guerrilla Learning is my own personal philosophy on how people learn while playing games married with the concept of taking that learning and making it relevant to more than just the current game being played. That's it. Players automatically learn how the game they're playing works, it's part of the reason we enjoy games as human beings (monkeys like learning, despite what school beats into you), if you make your

game mirror an aspect of the real world or an interesting/novel concept, that's Guerrilla Learning potential.

I was a little worried about the reception that GL would receive on the forums, but thankfully cold hard cash is a wonderful motivator and I think that a lot of Comp 15's entrants were rather intrigued by the idea of making a learning game. My fears proved unfounded as we had over 25 initial entrants that resulted in 20 entries with something to download and judge at the end of the competition. That's a new Game.Dev record.

The judging process really got Mindset into the picture and produced some very valuable insight into how educational content and games can come together. One of the more interesting (and sadly scary, from my perspective) results was that it became obvious that someone who was still in school was far less likely to understand the concept of GL than someone who was either finished with their schooling or in a tertiary institution. That shocked me: It's almost as though the current schooling system works very hard to make the idea of learning as unenjoyable as possible.



Worrying conclusions aside, Mindset were quick to label their Comp 15 experiment a success after we picked the winners. They were particularly keen on the idea of taking quite a few of the entries a little further and expanding on both the concepts that our intrepid community had come up with as well as helping to cement the learning information. The best part is that Mindset themselves really do get the thinking behind Guerrilla Learning, they don't want to make the games boring or confusing, they'd rather have a fun output with a slightly less immediately testable learning outcome, than a series of pointless quizzes with no gameplay just so that someone can "test" that you're learning something... If you finish a GL game, you've learnt something. End of story.

The Winners:

As with Comp 10, the Best New Entrant category was there to encourage new members to submit their games and ideas without having to feel like they need to compete against the more experienced community members. Kosie "ShadowMaster" van der Merwe's Typing Tower took the honours and the R1000. Mindset loved the idea of extending the typing mechanic in the game to cover slightly more curriculum-themed content like English drill and practice exercises that are otherwise insanely boring on paper, but with a little tension (escape the rising water!) quickly become much more enjoyable.

Third place and R1500 went to Robbie "Squid" Fraser for his prototype puzzler/ arithmetic game Math Attack. Keen to see it re-implemented as a cellphone game (which it's astoundingly perfect for), the Mindset judges were quick to see the potential in the rotation and manipulation of the puzzle mechanic to select the right number to make the sum make sense.

Second place, along with the R2500 prize, was the realm of Gareth "Gazza_N" Wilcock's Rockets! A game in which you're tasked with building and tweaking a functional rocket to hit various targets across different missions. All the while the forces of physics are there to aid or hinder you, depending on how you put things together. With a fantastically designed tutorial ("Aaaah! Soooo close!") and

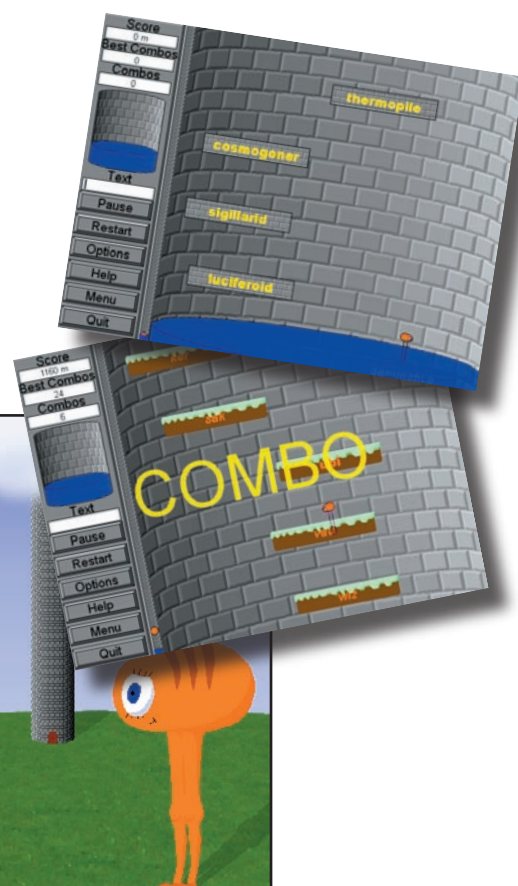
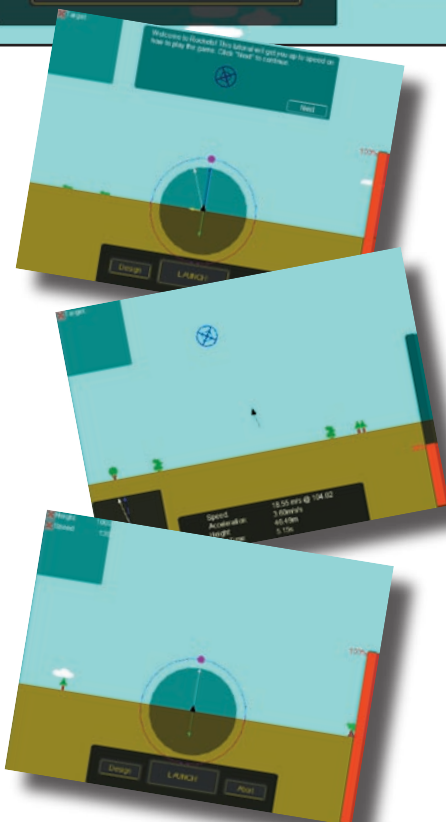
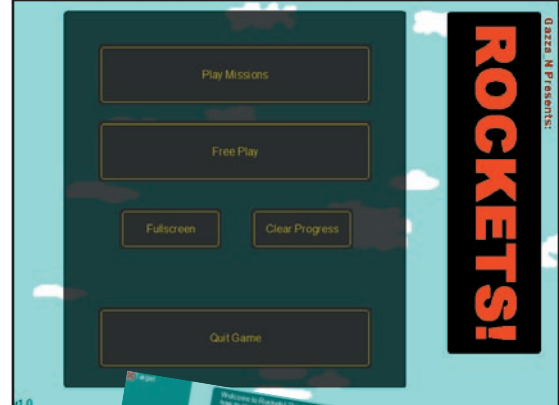
some smart onscreen information, Rockets! is a lot of bang for our buck... Forgive the pun.

First place and a whopping R5000 goes - for the second time - to Cadyn "Evil_Toaster" Bridgman, who won first place in Comp 10 last year with Fast Food in Space. His entry, Cartesian Chaos, built on the extended FFS engine, is a casual game clickfest with a Cartesian plane themed battleground. You have to select the right co-ordinates to blast simple monsters, create lines to defeat tougher ones and even point out answers to boss-monster equations. It's a hoot, especially with ET's trademark humour cropping up throughout the game. Polished nearly to perfection, Cartesian Chaos has everything you'd expect a commercial game of it's ilk to contain: Difficulty progression, different modes of play, combo systems, the works.

The next step:

Given the great quality of the games entered, I'm currently working on getting them some press in international gaming for educational uses circles, as well as simply popularising them because I think they're cool. Hopefully some of the developers will be able to work in partnership with Mindset to take their ideas a little further, that would be a dream come true for Game.Dev's goals. Not to mention pretty damn cool for the developers themselves!

I can't wait to see what the next competition creates. A ton of great ideas rise to the challenge each time I make a competition announcement, who knows what will emerge in the future? All I know is that this is a great way to grow game development in our country. 🌀



THE HISTORY OF I-IMAGINE

PART 8: "Round 2 ... FIGHT!"

by Luke "Coolhand" Lamothe

Once we had finally completed Chase after three years of hard work, we felt as though we had turned a corner for I-Imagine. We now had a completed and published game to our credit and a good relationship with a publisher who trusted us as a developer. To us, this meant that we would finally be past the point of having to fight publishers in order to be interested in what we were working on, and that we would finally be in a position to make game after game and perhaps even make some money!

Things did indeed start off with promise, as shortly after Chase had been completed, BAM! started to talk to us about their interest in porting it to both PS2 and GameCube. They saw the potential to turn Chase in a franchise which could be a valuable IP for them, and they wanted to start exploiting it as soon as possible. Right away we knew that we wouldn't be able to handle the PS2 version of Chase as the PS2 was quite a difficult beast to master, so BAM! arranged to have another developer whom they had worked with before begin to work on it. However, we felt comfortable with doing the GameCube version as it had a much more friendly SDK and hardware setup for first time developers. We received our development kits from Nintendo shortly afterwards and began to re-write our engine to make it multi-system friendly as it was currently only designed to work for Xbox.

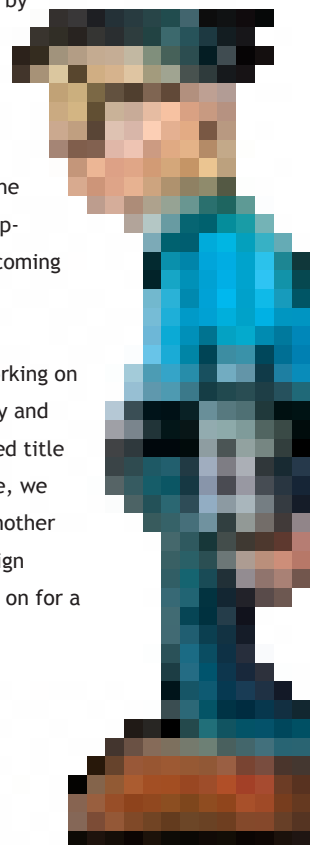
While the initial work for the ports of Chase was going on, BAM! also began to talk to us about starting work on a sequel to the game. They asked us to begin coming up with a design for it so that we could begin development of it as soon as the ports were finished. Unfortunately, bad luck managed to strike us yet again not long after these talks started as near the beginning of 2003, there was quite a big slump for all technology stocks and BAM!'s was no exception. Their stock weakened a lot over a very short period of time and this unfortunately meant that they just didn't have the capital in order to fund either the port of Chase or a sequel to it. By the end of 2004, BAM! would be nearly bankrupt and out of the video game publishing business altogether.

Meanwhile, towards the end of 2001 and long before Chase was nearing completion, another exciting project was underway at I-Imagine. This time, it was a 2D platform game for the Game Boy Advance entitled D.A. Dork. The premise of D.A. Dork was that players took on the role of a computer nerd who has to enter the virtual environment of a computer in order to rescue his computer generated girlfriend who was kidnapped by his arch-nemesis ... yes, another computer nerd! Inside of the computer, the player is aided by his trusty PDA sidekick whom he

needs to take control of at times in order to help solve puzzles necessary for progression through the game.

The game itself was actually very fun and it looked great to boot, but unfortunately, publishers were not at all interested in publishing original IP for GBA. In other words, if you didn't already have an existing successful IP or your game wasn't based on a movie or television license, publishers feared that your unknown product would just get lost amongst the glut of GBA titles out in the market. This unfortunately meant that after about a year and a half of hard work by our GBA team, a 95% complete D.A. Dork was retired to ROM heaven with the lack of a soundtrack being the sole remaining item stopping the game from becoming completed.

With our options for working on both the Chase property and our Game Boy Advanced title no longer being possible, we turned all focus onto another game idea that our design team had been working on for a



while on the side. The project was known as D4L which stood for Driving for a Living, and as you can probably guess, it was also a driving game*. We kept the design to be an arcade-style experience that put players into all kinds of various driving scenarios where they had to earn money to buy better cars, vehicle upgrades, and earn credibility in order to progress through the game.

In other words, players may start off in the game having to work as a pizza delivery driver until they had enough money and/or credibility to compete in street races. Eventually they would be able to move onto other such jobs as a limousine driver for the rich and famous or a camera truck driver for the local news station. What we tried to do in order to set this game apart was to create everything in the game around a quirky sense of humour that let everyone playing the game know that we weren't taking things very seriously at all. We had designed pretty much every aspect of the game as well as the characters in it upon caricatures of real-life people or pop-culture icons and events. So for instance, we had players street racing against a Britney Spears look-alike, running away from UFOs trying to destroy the city, and driving a limousine for Austin Powers and his entourage of femme-bots.

We worked on D4L up until E3 2003. Unfortunately, the response for it was lukewarm, and

we decided to sit back and re-evaluate our direction with the game design. This led us to realise that there were actually some very serious flaws with the design of the game, and that we would not be able to easily make the changes that were necessary to make the game fun to play. In other words, while we were confident that the idea of the game was sound, the mechanics of it were flawed, and unfortunately this meant that we had to make the decision to cancel the project entirely. It wasn't a complete loss however, as we were able to keep the technology that we had been building during this time and move forward with it**. Fittingly, the next project that we decided to undertake would be our own sequel to Chase.

Due to the publishing contract that we had signed with BAM!, we no longer had the rights to use the name "Chase" as it was now their IP. However, there was nothing in our contract against using "Hollywood Stunt Driver" as a title, so we began our project under that name. Before we began any work on the actual development of the game, we went through every single review of Chase that we could find and analysed the elements which people seemed to enjoy as well as the ones which they didn't. This enabled us to come up with a design which we felt kept the fun and easy pick-up-and-play aspects of Chase, while adding the addictive and challenging elements that a lot of people felt were lack-

ing from it. We also decided to incorporate as many "big action" set-pieces as we could in order to try and emulate what people find exciting while watching Hollywood movie stunt chase scenes, so action like the scene from Bad Boys 2 where the cars fall off of the trailer and the Ferrari dodges them would also be key to the design of the game.

We spent the next year or so working on a solid design for the game, great-looking technology***, and a really good demo containing 4 missions. This allowed us to receive some very positive interest from a few publishers who knew us from Chase and who liked the idea of doing a title that was familiar enough to people (it was still car driving) but with a unique enough angle to differentiate it from competing driving games (it was about doing stunts). The most interested publisher was Buena Vista Games, and as far as I can remember, we were actually pretty close to signing a deal with them. However, once again we were strung along for months and months as a publisher said "Keep working on it and show us some more when you have it!", and in the end, nothing concrete ever materialised from our communications with them.

While we were busy working on Hollywood Stunt Driver, another opportunity came along that we also decided to pursue. We somehow were put in touch with the management

* We wanted to continue to make use of the vehicle-centric technology that we had built to date.

**It was a re-vamped Chase engine which was now built around the RenderWare middleware platform and was running on Xbox, PC, PS2, and GameCube.

***Which now boasted complete liquid dynamics for water-based missions.





company that held the marketing rights for the ChampCar Series, which takes place in North America and is an open-wheel racing championship similar to Formula One. They were quite keen on giving us the sole mandate to create a racing game for them based around ChampCar, but we would still be responsible for funding it and finding a publisher in order to get the game to market.

We ended up putting together a pretty killer demo towards the end of 2003 of ChampCar

cars racing around Kyalami, but we were unable to convince a publisher to take us onboard as the ChampCar license didn't seem to be high enough profile for them, so we yet again had to abandon a project that we were doing quite well on.

In October 2005, a full two years after we were attempting to develop a ChampCar game, ChampCar announced that it had signed a deal with Jester Interactive Publishing to produce a game for them. However,

the deal apparently fell through and to this day, no games have yet been made with the ChampCar license.

The middle parts of 2004 were spent working on a couple of different pitches to publishers. In June we got a call from Codemasters who wanted us to pitch for a game built around a cops and robbers scenario where players had to do A-to-B racing while performing stunts and crashing through things. We pulled off an amazing demo for them in a short couple of months, but they ended up deciding to scrap the game idea from their production pipeline.

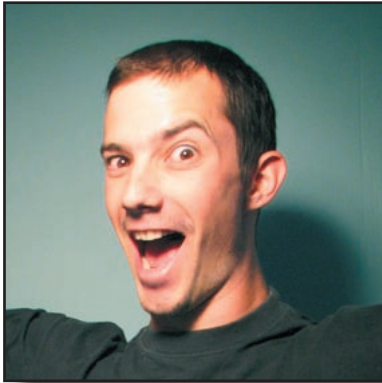
After doing this demo, we worked on doing a Need for Speed: Underground style night racing demo for another publisher who I can't seem to recall the name of (it was possibly Activision). Again, we were able to do a very good demo in a very limited amount of time, but the feedback from the publisher was once again not positive for us.

Finally, in October 2004, I-Imagine would begin to work on its final title during my time at the company. Perhaps prophetic in a way, the name of this game would eventually become known as Final Armada... 🎯



WE ARE LEGION

A smattering of game dev enthusiasts from all around South Africa. You're not alone!



Danny "dislekcia" Day (26)
Pretoria

When I first started going to school I started drawing mazes on the fly and giving them to my classmates to solve. They'd ask why I didn't join in and help them get to the end of the epic multi-page, item dependant, branching path mazes we'd end up with. My answer was that I enjoyed making the mazes far more than I ever enjoyed playing them. Even today, watching someone play a game I've made and having a blast brings a ludicrous smile to my face. I love that feeling... Watching people run with Game.Dev, seeing how individuals have grown and passing competition entries and things like Dev.Mag around, that's special. It's that ludicrous smile all over again.



Dominic "CiNiMoDZA" Lennon (18)
Durban

Well, it all started when I was born a very small, young child in Durban. I moved Estcourt, a town so small that not even its own mayor knows where it is! While there I stumbled across a group of nerds who kidnapped me and brain-washed me into doing nerd things, like game dev, chess, and dishes and stuff! After a while I moved back to Durban, and fell away from the spell of those evil red-headed nerd-mages, I started doing crazy things, like running, for fun!?! After being sent to Joburg to jump into a sand pit for KZN, and having a little rest from other daily activities(visiting the girlfriend, washing dishes, having a bath etc...), I found an old friend!!! Since I had some extra time, and NAG had just decided to sponsor a fair amount of money for some people to make some games, I decided to enter! After getting an award for Best-New-Entry, I was hooked yet again, and don't plan to leave this time! After discovering the gamedot-dev site, I was home!



Bernard "MushiMushi" Boshoff (25)
Durban

I am part owner of Indigo Child Studios [<http://www.indigochildstudios.blogspot.com>]; a media and sound studio in Durban, South Africa. We specialize in offering a range of solutions for all sorts of occasions. In the not too distant future we would like to diversify into a full service interactive marketing studio, offering our clients an interactive means of reaching their target market through the use of corporate games and online products. Our long term goal is to create a full fledged interactive online gaming experience codenamed S-T-O-M-P, which we hope to achieve by harnessing up and coming young local talent who are technically and graphically skilled. For the time being, I work as an investment and insurance broker in a large South African firm which will give me the financial means to fund my ambitions for Indigo Child. I also work on the DEV.MAG team from time to time and have been a part of the Durban Game.Dev Hotlabs initiative, which I wish to continue on my own capacity once I get settled into my new line of work.



William Morgan "cairnswm" Cairns (37)
Randburg

The first programs I can remember writing were games. Using old BasicA on a monochrome screen with no pixel capability I wrote racing car games, management games and even adventure games. At university I move onto Pascal and thereafter onto Delphi. I have contributed to one PC title so far. I have done contract work on a few other game development projects. The game that had the most influence on my wanting to develop games was Starcraft, once I had played it I knew that my final goal from game development was to be able to write an RTS. Over the Last 8 years I have mad on average R20 per day from my Game Development Hobby.



Simon "Tr00jg" de la Rouviere (17)
Stellenbosch

When I was 15, a hooded roach forced me into the art of game development. He said, "Young padawannabe, make games, or I will let David Hasselhof make a come back."

It was easy to accept, and here I am, still doing the thing I love! As you can see, humour is kind of my forte (depends on your taste). Roach Toaster (a unique turn-based strategy game) was my 1st success. It is totally awesome to do something creative and be able to entertain people. You should too (yes you), don't you think?



Kosie "ShadowMaster" van der Merwe (16)
Cape Town

OK... Hi I am Kosie van der Merwe and I am a... Game developer. I am 16 years old and currently live with my parents in the (sometimes) sunny Cape Town. As can clearly be seen from the previous sentences; I like jokes and all things funny.

My first venture into game development was modding Jazz Jackrabbit, i.e. making new levels. I later got on to writing games in Sphere, my only completed one is a Yahtzee clone. In grade 8, during exams, I learnt C++ and this year I finally wrote the first game I am really proud about, Typing Tower FTW! ShadowMaster out!



Luke "Coolhand" Lamothe (29)
Joburg

Since a young age growing up in Canada, I have always had a passion for video games. After graduating from high school I decided to pursue this love further by attending DigiPen, where I graduated with a Diploma in Video Game Programming with a Level of Excellence. After spending a year teaching at DigiPen, I moved to South Africa in 1999 to start I-Imagine Interactive, where I oversaw the development of both Chase: Hollywood Stunt Driver for Xbox and Final Armada for PS2 and PSP. At the end of 2006, after spending more than seven years at I-Imagine, I decided to pursue a new challenge. I ended up joining Luma Arcade where I now oversee the development of multiple game titles for various platforms.



Gareth “Gazza_N” Wilcock (22)
Johannes Burger (believed to be Joburg)

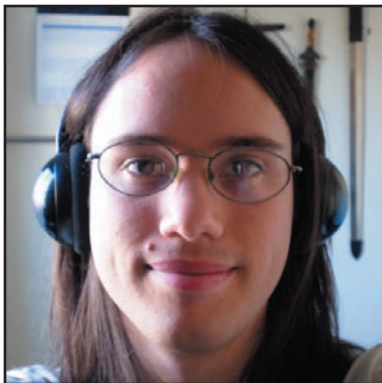
As of writing this, I’ve been part of Game.Dev for about a year. Now, most may think that my first ever foray into design was the first 2D Deathbringer prototype, but I’ve been attempting games pretty much my entire life. In fact, my first ever attempt was in Grade 6, where I whipped up a choose-your-own-adventure-style text game in QBASIC. Yes, I’ve had the game development bug for a loooooong time.



Rodain “Nandrew” Joubert (20)
Grahamstown

I like candles, trucks and hamsters. After those, game development. In fact, since about the age of ten I’ve been making terrible games and generally upsetting a lot of people in the process. Nowadays, I hang my hat at the Game.Dev community forums, shove projects in members’ faces and regularly set up threads that whine at them to write for Dev.Mag. I’ll whine at you too if you happen to stumble by.

That’s a choke collar around my neck. I don’t know why.



Ian “Thaumaturge” Eborn (24)
Cape Town

Magic. Sorcery. Enchantment. Thaumaturgy, for that matter. If it’s a creation of mine, there’s a pretty good chance that magic is involved. If I were to create a ‘mech game, or a space-based action game, there’s a reasonable chance that it will be a fantasy ‘mech game or space-based action game, and, of course, involve magic. (I actually half-created an instance of the latter once upon a time...) I do very much enjoy other genres. I just often prefer fantasy.

Oh, and I can (and do, at times) give what I think to be a fairly decent villainous laugh, of the “mwahaha” variety.



Ernest James “edg3” Loveland (16)
Johannesburg

(The poor soul typed all this from his cellphone. That’s commitment!)

My name is ernest james loveland, im 16 and ive been programming for about 2 years, starting seriously about 18 months ago. I taught myself visual basic, and then took IT (information technology or otherwise programming) at school. I have always been interested in game development, and after dabbling in game maker for a few hours decided i didnt have time for it. This all could actually boil down to me being an expert in the art of procrastination and disorganisation. Living in JHB and being the South African I am, i have one thing to say: “when life hands you a game idea, shut up and work on it!”



Tarryn "Azimuth" van der Byl

Age: Carbon dating suggests somewhere between 2 and 10 000 years.
Location undisclosed ... how mysterious

Subject demonstrates pronounced aversion to light, exceptional literary aptitude, a preference for caffeinated beverages, and an entirely unwholesome appetite for heavy metal music. Subject has previously begun development on a Flash-based remake of Space Quest II, but has since abandoned this in favour of playing other people's games for money. Subject will manifest immoderate hostility when confronted with sloppy grammar or spelling.



Ricky "Insomniac" Abell (20)

Pietermaritzburg

I'm currently a second year university student doing a BSc with a double major in Computer Science. One of the things I love most about game development is how it combines the creativeness of design with the logic of programming. When I'm not studying or making games I can usually be found jamming away on Guitar Hero II. I also find the field of cryptology interesting as it's like a big game of chess but instead of pieces and a board you have maths and the power of computers. Unfortunately I have yet to come up with an idea to incorporate it into a fun game but I'll keep trying.



Rishal "TheUntouchableOne" Hurbans (18)

Pretoria

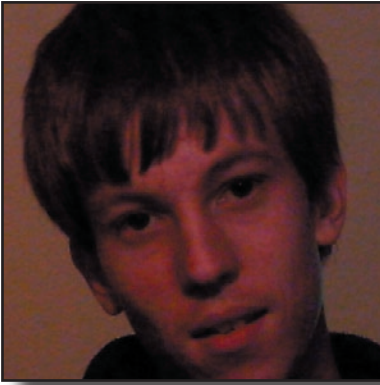
Well, I'm an everyday well-balanced type of guy with a zest for COMPUTERS, I love programming and creating great pieces of work from scratch drives me. I program in Delphi, Python and Basic. I am eager to learn Game Maker as it may be a simple tool but proves to be very powerful. Developing games interests me a lot as it provides a great way to channel your knowledge into producing something that can be productive yet fun. I'm supposed to write my interests here but too many things in the world interest and intruqe me so this is all you getting for now. For the love of Games, develop for the sake of fun!



Quinton "Q-Man" Bronkhorst (Ham)

Johannesburg

I'm a second year Journalism student and have a passion for writing! Although I know very basic programming, my interests lie more in the concepts and stories behind the games we play. I have a deep respect for the programmers and designers, but I believe that story is everything and can ultimately give a deeper and richer gaming experience! Nothing grips me more than a deep, complex plot, be it an epic tale where the unlikely hero must take to arms and save the world, or a dark psychological horror where the protagonist must face his inner demons, his past, and reach that epiphanic moment of truth.



Claudio "Chippit" de Sa (19)
Sasolburg

Besides my other hobbies, I like to spend my time creating things, ranging from drawn models to 3D models to Lego models to model games. At least, I like to tell myself the games are a model example of something, whether or not effort counts is disputable. When I'm not doing that, I'll likely be at the campus grounds (diligently?) studying towards my BSc in IT, or listening to some music from my varied selection.

Robbie "Squid" Fraser (16)
Centurion, Pretoria



Hi, I'm Squid. I like to frolic with wombats in the meadows... Also I love to make games. Some of my amazing talents include rock climbing, playing soccer and of course playing games. My ultimate goal is to own my own game development company with an in-house publisher, being a multi-billionaire is also on the list though.

Random tidbits:

Love heavy metal, especially Iron Maiden

Coded the Dev.Mag website

Like guitar although not a very competent player

Have a natural attraction to marsupials and large rodents

Have two cats

Can't stand seafood

Pretty lazy when it comes to anything non-game-development related

Sucker for green eyes, HDR and physics simulations

Christian

The name Squid was given to me by a school friend in an internet café in a ski resort in Serbia, pretty random eh?



Stefan "rman" Van der Vyver (he's the one on the left!)
Cape Town

I thought I could program games, so I bought Game Maker. I also bought the Blender Game Handbook from the Netherlands. Time became an issue. I quickly realized that I should focus my skills on doing 3D, teaching 3D and making music. So, I've got 7 guitars, 1 keyboard, 1 electronic drumkit and six computers. With that I create content for educational games, corporate multimedia content, television ads and original soundtracks. And that's about as close as I'm going to get to game programming.

CREATE. DEVELOP. EXPERIENCE.....**ONLINE**



DEV.MAG

CREATE • DEVELOP • EXPERIENCE



www.devmag.org.za