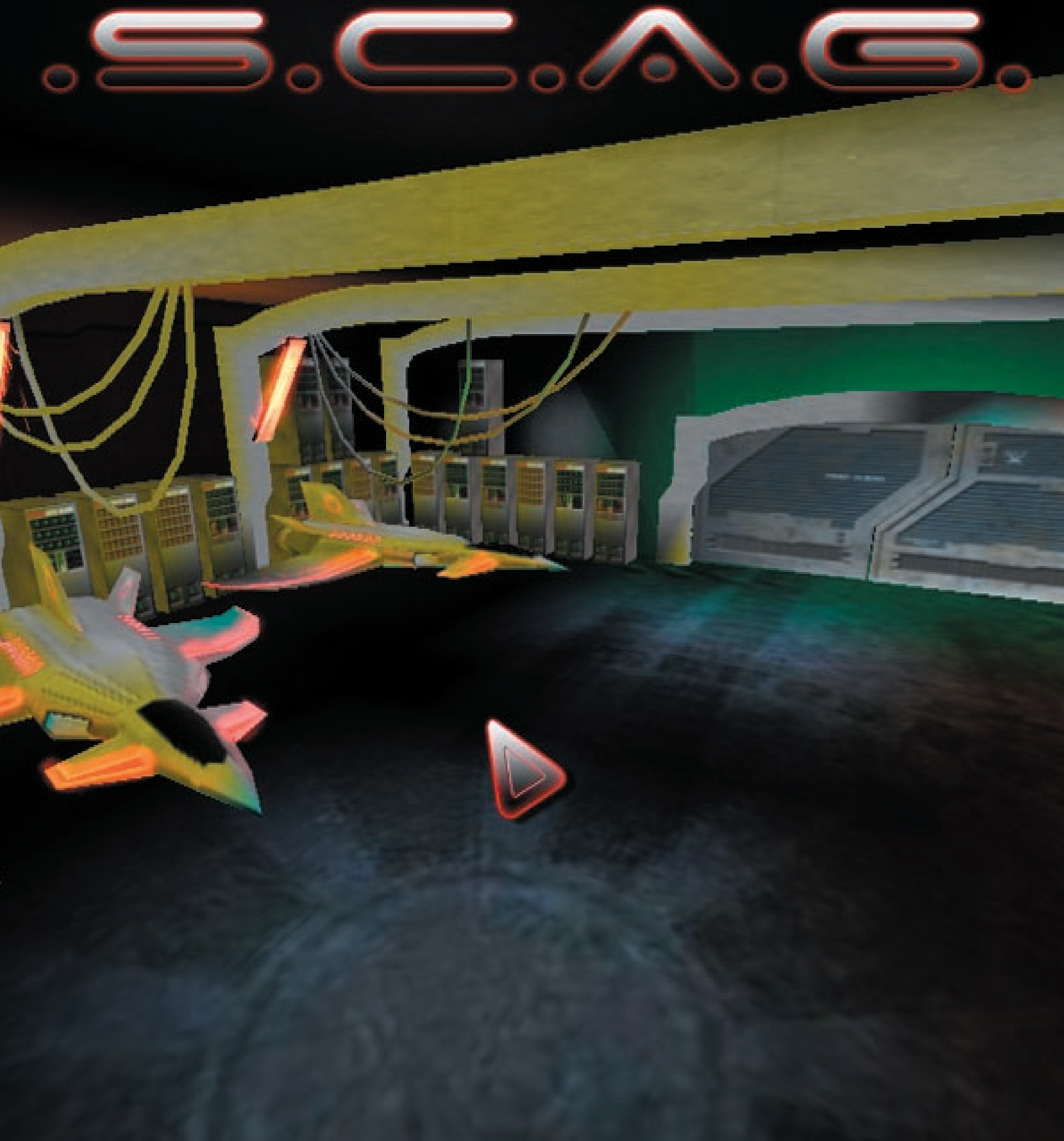
SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

JUNE 2007

DEV.MAG

CREATE ● DEVELOP ● EXPERIENCE



05

15

24

27

An interesting little thingie popped up on the Game.Dev forums the other day. Amidst all the hype, hubbub and competition entries for Comp 15 (hey, R10 000, remember?) that have been cluttering the community as of late, one of our members burst in on the latest Dev.Mag thread and drew our attention to the fact that he saw some Game.Dev ads on the telly. Pretty awesome, I must say! We're stoked that the group has come so far in this relatively short space of time, and kudos must be given to the community's founder and leader, Danny "I've-worked-my-ass-off" Day, known to most as dislekcia.

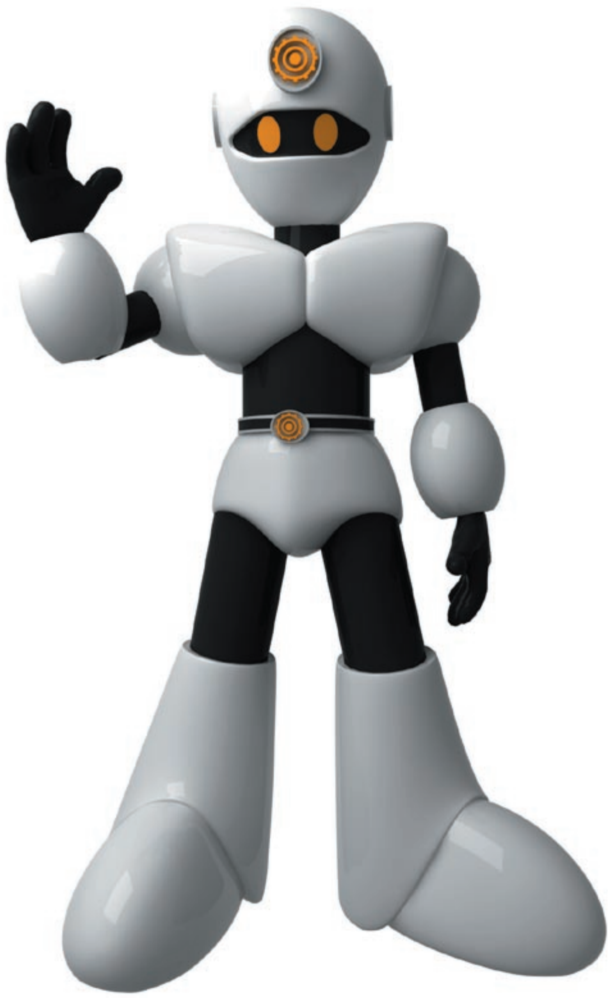
This kinda got me thinking that maybe this mag doesn't advertise its link to the Game.Dev group enough. It's always been great that the members have worked autonomously on this publication for more than a year now, and we've never been restricted by the high-ups regarding content decisions and how much extra icecream we get on Fridays (it's a perk for loyal staff members, I lie not). But, as flattering as it is to have some people forget that Dev.Mag is just a cog in the big Game.Dev machine, maybe it's time to pay homage directly to our alma mater over these next few issues. Thank you, Mommy. You've given birth to a beautiful baby 'zine.

On a similar note, some of our writers have brought up the concern that perhaps we're losing touch with our original ethos. No matter what else we put in our pages, this 'zine was ultimately created to serve as a guide those who were taking their first tentative steps on that scary game development road – so by golly, we're gonna stick with that goal! To this end, we've done three things for you, our readers:

1. We've introduced a "Blue Pill" article series in the Design section. Every month, we'll be taking a look at one game development product. We'll tell people what it is, how it works, where all the best tutorials can be found and why it's a good idea to get started with it.
2. We've established our own series of Game Maker tutorials to help anybody who still isn't convinced by what they hear about the product's ease of use. It's a blast, seriously!
3. We're now taking care to refer to back issues for the beginnings of article series or once-off tutorials, so that anybody confused by the stuff they read in here can easily flip back to a previous edition and be enlightened.

We're doing our best to make this work out as well as possible, so be sure to send us your feedback and tell us what you think of the changes!

Editor
Rodain " Nandrew " Joubert



WHOOPS!
Well, aren't we just silly sillies? Last edition, we failed to mention that this DB model (and, well, all of the other DB models you see) are done courtesy of one Geoff "GeometriX" Burrows. He did a pretty swell job on 'em, too. Thanks, Geoff!

This month's PGD contributors:
Michiel "NecroDOME" Gijbels
Frank "Mad Woody" van Gent
www.pascalgamedevelopment.com

This month's opinion columnists:
Rodain "Nandrew" Joubert
Simon "Tr00jg" de la Rouviere

DEV MAG
CREATE DEVELOP EXPERIENCE

EDITOR
Rodain " Nandrew " Joubert

DEPUTY EDITOR
Claudio " Chippit " de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNERS
Brandon " Cyberninja " Rajkumar
Geoff "GeometriX" Burrows

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

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

WEBSITE DESIGNER
Robbie " Squid " Fraser

WEBSITE
www.devmag.org.za

EMAIL
devmag@gmail.com

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

Doritos game design

http://www.unlockxbox.com/

The latest game-making initiative on the Xbox is being spearheaded by ... tortilla chips? Well, that seems to be the case, and it's an interesting challenge, to say the least. The premise is simple: submit a design idea for a Doritos-based game which has the chance of being developed into a Xbox Live Arcade game. If you happen to have a US citizenship lying around, take a stab at it yourself, or watch from the shadows to see what happens with what's proclaimed to be the first "user-generated Xbox 360 game".

Doritos PRESENTS UNLOCK XBOX. YOU DREAM IT. WE BUILD IT.

Reminder for Comp 15

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

That's right, a cheeky little reminder that the competition is still on and still accepting new entrants! Game makers extraordinaire have one month left to get cracking on their own masterpieces based around an educational theme and get their piece of R 10 000! Check out the Game.Dev website and forums for more details on entering.

game•dev
It's what you wish you were doing.

Game Addiction Discussed

http://www.gamedev.net/community/forums/topic.asp?topic_id=453210

The American Medical Association recently held a discussion about the severity of game addiction and whether or not it can be considered a formal psychological disorder. The final decision was against labelling it as such, but only because it was felt that more research needed to be done in the area. Gamedev.net has a link to the document outlining the issue, and it looks like the medical world is going to be paying much closer attention to gaming from now on.

AMA
HELPING DOCTORS HELP PATIENTS

Nintendo says "yes" to indie developers

http://gizmodo.com/gadgets/wiiiiiiiii!/nintendo-opens-wii-to-indy-developers-272717.php

Traditionally, Nintendo has kept its doors closed to third party and indie developers. However, a recent announcement by the company has turned that idea on its head and brought hope to Wii enthusiasts and avid developers. WiiWare, a game distribution service comparable to the one offered by Xbox Live, is set to allow smaller companies a chance to get their games to the public with a user-friendly sales system. Although it seems unlikely that this will open up Wii development to the man on the street, we're holding thumbs for it.

.S.C.A.G.

THE MAKING OF SCAG

by NecroSOFT

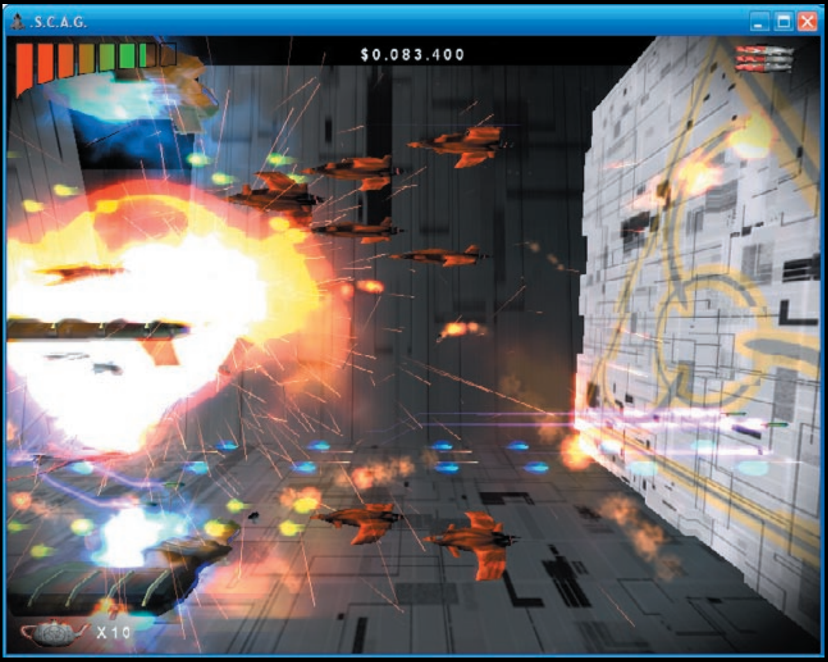
We just can't seem to get enough of those Pascal guys right now. This month, we sport a postmortem of one of the games built by the Pascal Game Development team known as NecroSOFT. They call it .S.C.A.G.. It's slick, it's fancy and it won 2nd place in the 2006 PGD "Big Boss" competition, so we thought it would be worthwhile to nag them for a juicy article. Of course, these guys don't think like most people, so what we ended up with was a cryptic list of Pascal code instead of the usual English. Fortunately for you, dear reader, we made a fair effort at translation ...

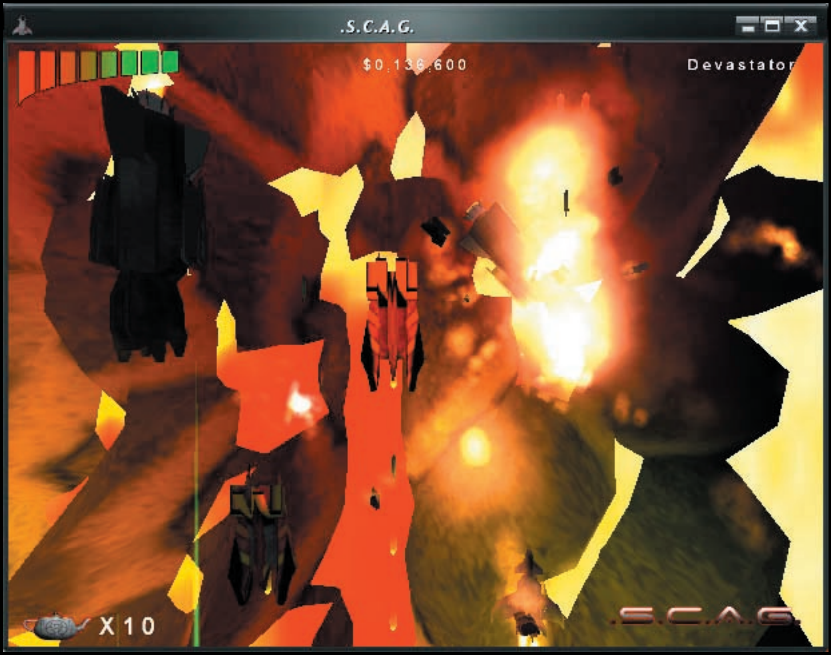
program The_Making_of_.S.C.A.G.;

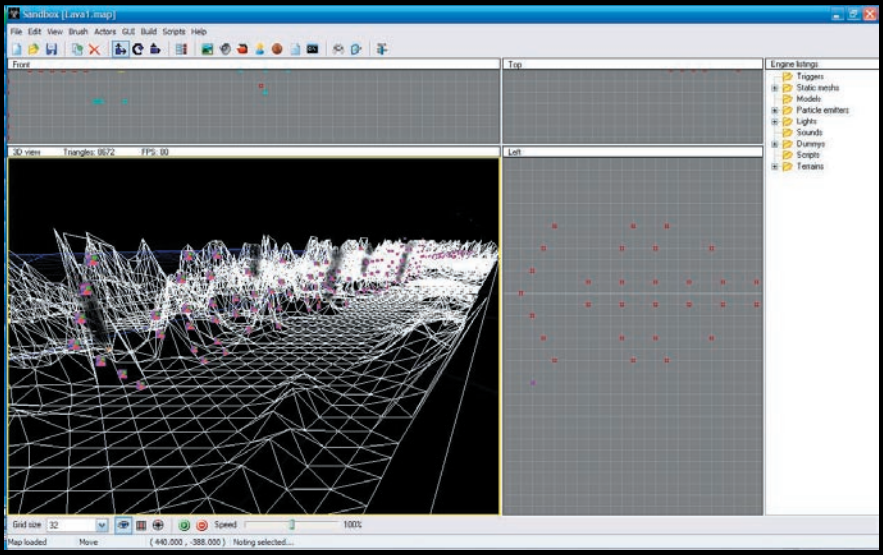
implementation

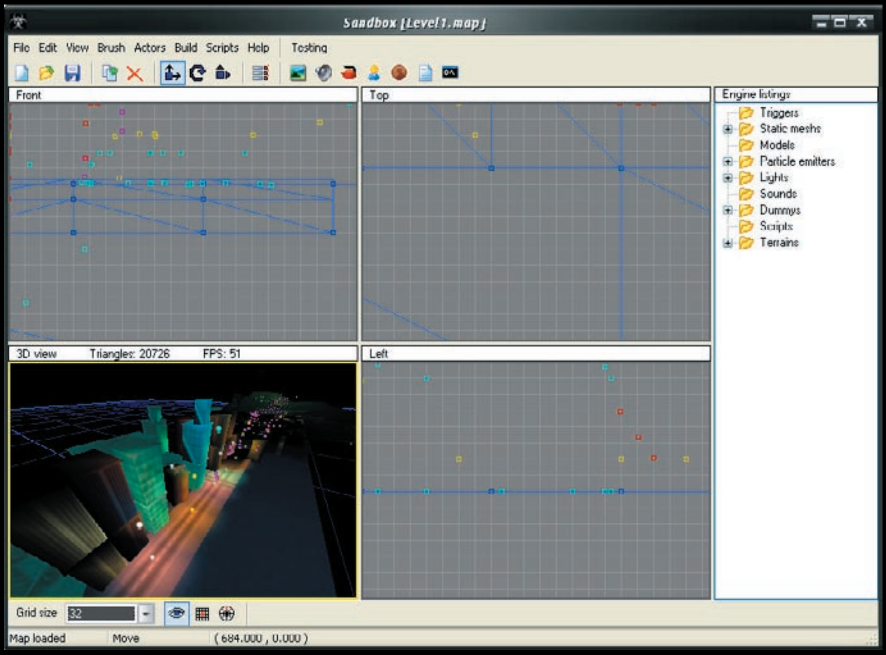
constructor Create();
begin
This article will describe in-depth how we made the game .S.C.A.G.. When we started with the engine a year and a half ago, the original idea was to make a first person shooter. When we finally had a user friendly editor, the PGD competition (Big Boss) crossed our path and we decided to create a game that looked like the old arcade game Raptor: Call of the Shadows.
end;

procedure Init();
begin
The first thing we did was to create a new project, and include our Necro3D game engine in it. This was a good test for us to see if we actually









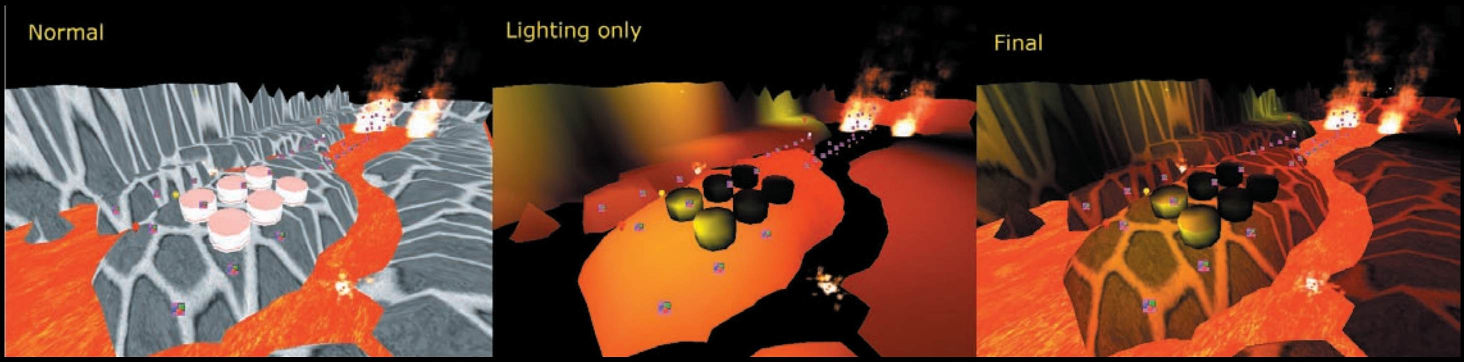
could make a game with the engine. We were also able to see if the editor was as user friendly as we wanted it to be. When our engine was successfully implemented, it was time to load a low-poly ship and a few cubes to get proper handling for the player's ship. In this phase we also got the great idea to do something more then a 2D Raptor game in a 3D engine. We decided to make a game that had a side-view, and a top-down view. With this in our minds, we started creating levels and enemies.
end;

function Level_Design()
begin

For the competition we had to make three levels. Well we made four just for fun! After the competition we expanded it to fourteen, divided into four different zones plus one bonus level.

The first level was a city. Because we had no terrain engine yet, we used planes to make the city road. In a few places in the level we added some big smoke particles for some really nice effects. This will give the player the idea that it's very windy in the town and that the city is deserted. It's easy to create, and gives a great environment to the game.

The top-down, and side-view cannot be chosen by the player, but are marked in the map. This is because we had to align the enemies on two different axes for the two view modes. All the enemies were placed with our editor. We can tell you this - it was one hell of a job to get it done. It didn't take very long before we created a copy and paste function for this in the editor.

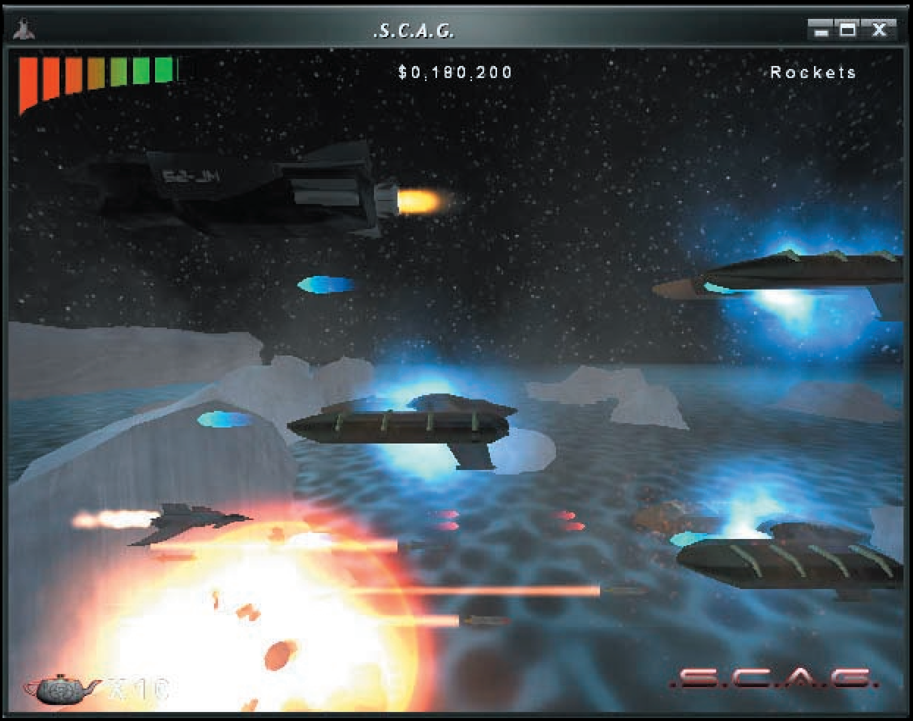


When the terrain engine was done, it was really easy to import height-maps and speed up the level design. A very important thing to create the right environment is light. Instead of using one global light, we used a lot of small lights. The most important thing is that all those lights we placed had different colors. For example, in the lava levels we used a lot of yellow and red lights to make the ugly grey levels more attractive. We had a small problem with this level because we had a lot of aliasing going on at the edges of the lava and the terrain. To save you guys a lot of time when you have this problem, it's caused by the Z buffer and the minimum camera distance.

Another fun thing about the lava level that we created after the competition was the rocks that are falling down. This is the only place where we used the Newton Physics Engine that we implemented into our engine.

In most of the levels we used multi-texturing. This means you can place two or more different textures on each other with a blending effect on it. In our editor we made a feature in the texture browser to easily do this. Just create an empty texture with a size and load the textures you want to use. You can see this effect in the ice level, as well as the mirroring on the buildings of the city level.

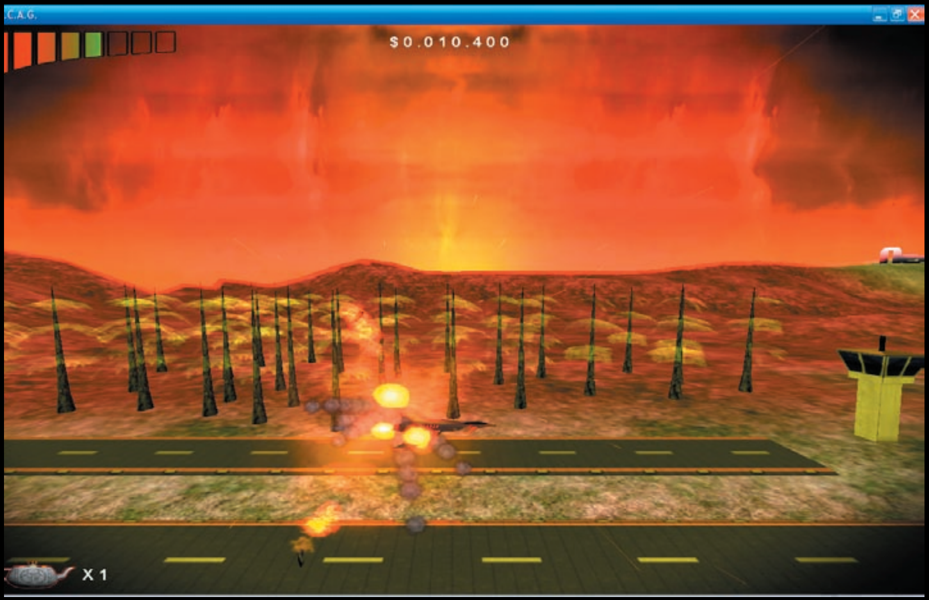
end;



function Weapons();
begin

The most fun part of creating a game is of course adding awesome weapons. The interesting thing of our game is that some weapons have two different behaviors. For example the rockets will fall down a little bit in side-view, and then in top-down view, they will go to out to the sides a bit on the screen. We also made a weapon called the Drunkenpod. It fires a bunch of small rockets that will fly all over the place --because they're drunk, of course. Each rocket leaves a small trail which creates a really cool effect. We made this by simply sticking some particles together.

end;



function Ship_Modeling();
begin

First, I have to say that we are not great designers. Our inspiration came specifically from 'Raptor'. We also looked for some ships on the internet for ideas and worked them out in 3D max, but with our own twist, of course. The idea of the Sharkfighter (yellow, red enemy) was created from an old Game Boy Classic game (don't remember the name). But during the creation we changed it so much it was a completely new fighter. After we created the models we had to do our favorite job - texturing. Well we made it, and looks good, but it's not the most fun part to do and very time consuming.

end;

function Soundtrack;
begin

We made four different soundtracks for

.S.C.A.G.. The first idea was to let a friend of ours make it with the help of his band. Unfortunately he didn't have the time to do it, so we decided to make it ourselves. We used a very old program from 1996 called NewBeat. It offers a lot of different samples divided into samples from 2, 4 and 8 seconds. With this you can make great songs if you use it right. It created just what we wanted, a retro style soundtrack. Only keep in mind that the samples shouldn't conflict with each other.

To play the soundtrack and sound effects, we had first used the Omega DirectX SDK for Delphi, but since the release of Vista we noticed it wasn't Vista compatible. So we then decided to use another SDK and as result, we can now say that .S.C.A.G. is Vista compatible. It now also supports the Doppler effect, but is not used. Yet... :)

end;

function Menu();

begin

For the menu we created a new menu engine that supports drag-able panels and custom fonts. Fonts in .S.C.A.G. are created at runtime to save disk space. The menu itself was created using buttons. Even the big caption logo on top of the menu was created using a button with no click event and two of the same images so you won't notice it when you hover your mouse over it. The scrolling text in the credits window was made with a render target that was placed on a button, which in turn was placed on the main credits panel. It gives a nice look to see credits scrolling inside a window.

end;

function The_New_Way();

begin

Our Necro3D game engine is great, but there is always room for improvement. That's why after two games in the Necro3D game engine, we are now busy with the Necro3D game engine version 2. The main improvement of this new engine is that it will be fully dynamic. We are going to use separate modules that are loaded in the engine like a plugin system. Examples of some of the modules are; the Render module, Sound module, Physics module, and so on. With this structure we can easily change, for example, the rendering with DirectX 10 or OpenGL, without rewriting the whole game engine. This also means that we can easily upgrade a game with a new physics engine. It's a little more complex to make, but when it's done we'll have a greatly improved, fully dynamic engine.

The work of this new engine progresses well and we are already able to load 3D worlds and compile shaders --something the old engine couldn't do. Also the editor is going to be very different. Instead of the four view sections --top, left, front and 3D-- we are now going to use one single 3D view. We are very excited about our new engine and hope to be able to show you the first games made with it soon.

end;

procedure Cleanup();

begin

With .S.C.A.G., we made a great game and finished second in the Pascal Game Development Annual Competition. We can say that we had a lot of fun making .S.C.A.G. and are planning to try and sell this wonderful game. Keep in mind that this is not the end, but only the beginning. We will continue making games and with the new engine, we hope to be able to create more quality games then ever. And who knows, maybe you'll see a .S.C.A.G. 2? ;)

Print('End of line.');

end;

// Give us a visit at
// http://necrodome.homeftp.net/

end.

"THE SANDTRAP"

Yeah, okay, so we all stand in awe of creativity and freedom in videogames, right? I mean, games are evolving from the droll, go-here-do-this linear hum-drum of yesteryear into the flexible, free-form sandboxes that we're met with today. They're more like the toys that they should be, right?

Riiiiiight.

Popular opinion seems to be that games like The Sims, Black and White and, to a lesser extent, Fable are ushering in the golden age of gaming - an era of virtual worlds which have activities and objectives that go beyond the basic goals of the game itself. In more extreme cases, user-defined goals serve as the only objectives in the game. So, woohoo! You have ultimate power over your destiny! Ain't that just dandy?

Well, while this sounds great and all, I think I'm going to step forward here and risk sounding like that proverbial old man on the porch. I don't think that this new wave of games are all they're cut out to be. They may appeal more to a new mass market, but as far as I'm concerned, games are defined by rules and goals set by the developer - much in the same way that movie plots are written by those blokes in Hollywood and not the people sitting in the cinema. Heck, even a joke has a defined punchline. Imagine how dreadfully boring it would be if your friend walked over to you and said, "Knock knock, you get the idea now. Finish it yourself."

I've played The Sims myself. Got rather bored after a while, so I went on to buy an expansion pack or two (especially since one of them had those adorable little pets in it). Those didn't last much longer. I eventually clued up as to why I wasn't enjoying the game - there was no defined challenges for me. If I got rich, I got rich. If I became fitter or got new employment, good on me. There simply did not seem to be enough of the game saying, "Hey great! You did a sweet job!"

Heck, I also played Fable - went and bought a house in-game, even got married. The result was that I now had a wife with a little heart above her head whenever I decided to step back into that one particular part of the game. And for what purpose? She didn't help me kill monsters, she didn't unlock any bonuses, she hardly even appeared unless I went out of my way to find her. It all seemed merely for the kudos of having a wife. Are modern gamers really drawn to that?

Don't get me wrong. I think sandbox games are important, and most games contain some concepts of user-defined goals. Still, I think it's important for us to pay heed to the advantages presented by a well-scripted and well-ordered games that provide rewards which are ... well, truly rewarding. Just a thought.

NANDREW

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

"Can Game Developers learn from Genetics?"

In my quest to discover why certain "original" games get the limelight while other "original" games get lost in the recesses of the web, I stumbled upon the answer.

It happened while I was watching my favourite show, Heroes. If you don't know, Heroes is all about seemingly normal people trying to cope with their special powers in their everyday lives. The narrator of the show speaks about the concept of evolution a lot. So being the person that I am, I wiki'ed Heroes and Evolution. I discovered an interesting and true phenomenon in evolution.

When there is a huge number of a species (like humans), any new mutation will be literally drowned out by sheer number of people who simply do not have that mutation. The chances that this new mutation will be carried to new generations are incredibly remote. A new mutation will have a higher chance to get carried on to new generations in small closed societies.

I immediately drew a comparison to games - what are the chances that an original game gets noticed and accepted as something "great"? Very small, simply because it is different to everything else out there.

I can hear you asking, "But if a game is original, doesn't it then stand out among the masses?" Yes it does, but again we draw a comparison to genetics. If someone with a new mutation occurs, we would be instinctively wary. An extreme example is the case that someone mutates a third arm. Look at all the possibilities and advantages of a third arm! But let's be serious here - who would go out with someone who has a third arm?

So I come back to my original question. Why do some original games fare better than others?

I'll explain it by another example. If Superman (who has awesome powers) gets tired of Lois Lane, he will have plenty of girls willing to have his babies. It is instinctively viable.

Now how do we distinguish between third-arm games and Superman games? Is your original game third-arm or Superman? If it's a Superman game, you wouldn't be reading this - you'd be taking a Sunday drive in your Ferrari. I don't know the answer, but I have the solution to get your third-arm game or Superman game noticed.

The great difference between genetics and game developing is that we can choose these new mutations in our games. If more and more people mutate third arms, the phenomenon will become accepted in society and be carried over to future generations.

So, what is the solution here? If you have an original game that does not seem to get noticed, you should keep flogging it everywhere and produce sequels with the same gameplay. If more and more people see your original game and its variants, it WILL become recognised and talked about.

Take Team 17's Worms as a brilliant example. All they have done is develop Worms after Worms. Who remembers the first worms when it came out? Back then, no-one really knew about it, but now it's grossing huge sales with its latest incarnation on X-box Live Arcade. They just kept going.

It's as simple as that. Just keep going.

TROOJG

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

STOP PRESS

MAKE A GAME AND GET YOUR PIECE OF R10 000

Here at Game.Dev, we believe that games are more than just entertainment or a silly fad — games can (and will) change the world and make a difference. Mindset Learn are keen on doing just that, making a difference. So they're challenging us – and you – to produce the next generation of entertaining and meaningful games. That challenge takes the form of:

R5000 for first place,
R2500 for second place,
R1500 for third place and
R1000 for the best new entrant.

So all you have to do now to get a shot at that prize money is put together a game that's fun and focuses on a concept you enjoyed or found challenging at school. You don't need to produce a fully fledged "educational" experience, in fact we're looking for quite the opposite: fun games that are enjoyable because they're well-designed and entertaining, you just happen to be playing within rules or a setting that match something you've learned. Think "guerrilla learning" and you're on the right track!

The competition begins on 1 June 2007, and the deadline for entries is 31 July 2007. That's two months that you have to make the ultimate game and win some big money!

For more details, head on over to the Game.Dev website at http://www.gamedotdev.co.za

game•dev

It`s what you wish you were doing.

BLENDER INTERMEDIATE TUTORIAL: Creating explosions with Blender 3D

(If you're new to Blender, check out Dev.Mag Issue 5 for the beginning of our Blender tutorial series!)

As with previous tutorials, I need to stress the fact that the basic Blender interface has already been discussed within previous releases of Dev.Mag. I will not therefore explain every single step of the way. There are plenty of interface-related resources on-line.

Blender 3D has an integrated particle system that can render production quality special effects. One possible use for this particle system is to render a particle system that simulates an explosion through animated colour and transparency parameters.

The rendered sequence can then be compiled / composited as a movie clip, animated GIF, or in any way that your specific software package will handle images with sequential numbers. Hence, an animated explosion.

The field of special effects is bigger than a short tutorial, so expect to find:

- The basics for setting up a particle system
- Basic animation of colour parameters
- Basic introduction to transparency for Blender particles

Particle systems can produce spectacular effects, depending on the time you are willing to spend tweaking it.

So, first things first. We have to plan this thing, since it has to fit into a game somehow.

I suggest doing an explosion:

- That will be an explosion for a scrolling top-down shooting game
- That starts and completes the animation within two seconds (I will work with 25 frames per second)

Two seconds of animation amounts to 50 frames at 25fps, and I will plan the colour changes of my particles to fit into that time frame.

I will set my colour animation to go from initial BLACK smoke, to bright hot RED, to flaming ORANGE, and ending off with GREY smoke. Timing will be set as indicated in the image below.

We will create:

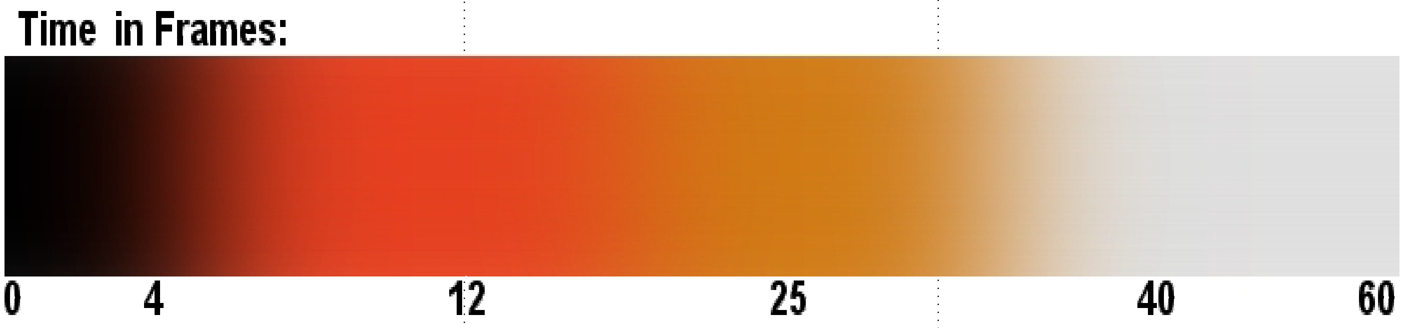
- An object that will shoot out particles
- Create a texture for the particles (mainly a smoky, see-through type of effect)
- Keyframe the colour changes
- Position a camera to document the explosion
- Render 50 frames

Particles will be generated (in this case) from a mesh object. We will shape the object so that the particles do not simply fly out in a straight line. Speed of the particles is determined by the menu settings.

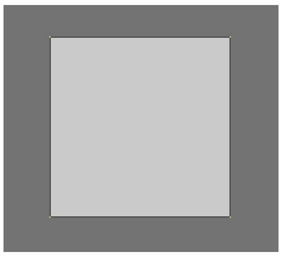
Of course we also have to tell the particles when to appear, how long they will last, and how long they will be emitted for.

For example, we might say that the particles will be thrown out of the object (emitter) within the first ten frames, which will create the actual explosion, after which some flames and smoke will billow outwards for the remaining 40 frames.

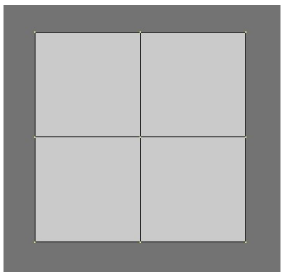
Start up Blender, if you're up for the challenge!



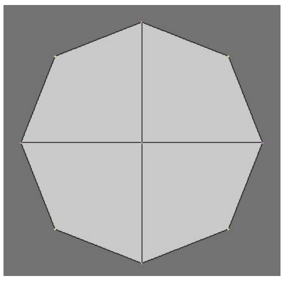
Create a mesh 'plane' .



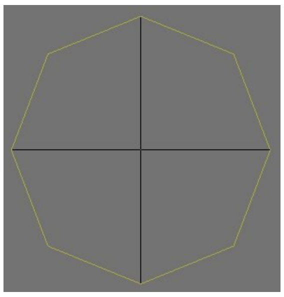
Subdivide the mesh plane (WKEY)



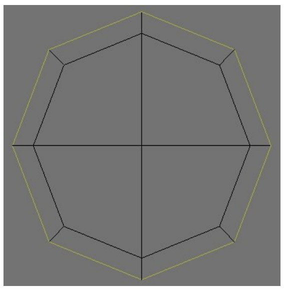
Select the four corners and scale them inwards



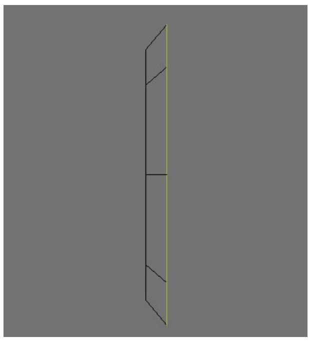
Go to Edge mode (Ctrl-TABKey) and select (RMB) the outer edges. I also switched to wireframe view (ZKEY) for easier viewing.



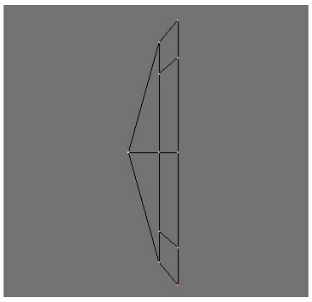
Extrude the edges (EKEY), but press LMB (left mouse button) immediately. After pressing the LMB, press SKEY. This will put you into scaling mode. Scale the edges outwards.



Now you have to change your view to a side view, which will allow us to add the shape that will make the particles fly out in a wider area. Move the selected axis according to this image:



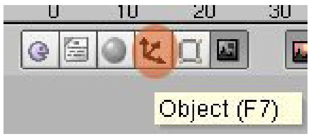
Select the center vertex of the initial mesh object, and move the vertex so that you an image resembling this:



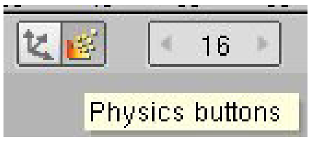
TABKEY leaves edit mode. This object will be the emitter for the particles.

It's time to add the particle system. Ensure that you have enough screen space around the emitter, so that you may see the effect of any of the changes you may make.

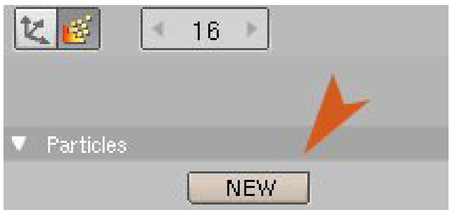
Click on the “Object” button.



Then locate the “Physics” button and activate it.



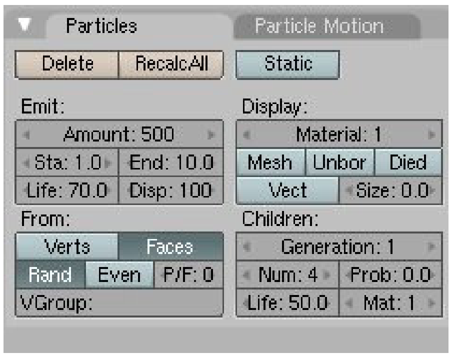
Next, create a “NEW” particle system.



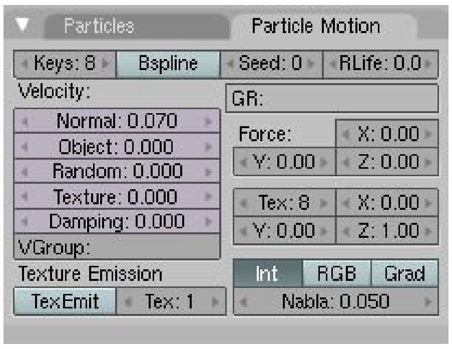
A new menu becomes available. Note that there is a tab for “Particles”, as well as a tab for “Particle Motion”.

Under “**Particles**”, you will find settings that define the creation and lifespan of particles, while the “**Particle Motion**” tab controls movement of the particles.

My suggestion would be to set the parameters as follows, after which you can start tweaking:



Particles | Particle Motion
Delete | RecalcAll | Static
Emit:
Amount: 500
Sta: 1.0 | End: 10.0
Life: 70.0 | Disp: 100
From:
Verts | Faces
Rand | Even | P/F: 0
VGroup:
Display:
Material: 1
Mesh | Unbor | Died
Vect | Size: 0.0
Children:
Generation: 1
Num: 4 | Prob: 0.0
Life: 50.0 | Mat: 1



Particles | Particle Motion
Keys: 8 | Bspline | Seed: 0 | RLife: 0.0
Velocity:
Normal: 0.070
Object: 0.000
Random: 0.000
Texture: 0.000
Damping: 0.000
VGroup:
Texture Emission
TexEmit | Tex: 1
GR:
Force: X: 0.00
Y: 0.00 | Z: 0.00
Tex: 8 | X: 0.00
Y: 0.00 | Z: 1.00
Int | RGB | Grad
Nabla: 0.050

You will notice that I only changed about six or seven parameters. If you want the particles to fly more randomly, try adding some “Random” under the “Velocity” heading. These values can also be adjusted negatively to change to the opposite direction.

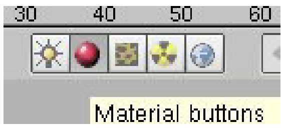
Play your animation (mouse over viewport and ALT + AKEY), or use the arrow keys to look at the animation of the particles.

Now for the texture of the particles. This part is quite challenging if you are new to this, so <u>allow some time to figure this out</u>.

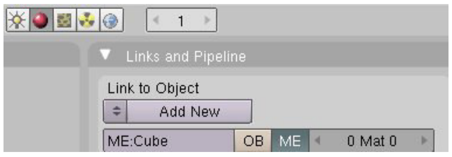
Texturing

We will use the solid color RGB area for primary color adjustment, and load a texture that will affect the smoky, transparent look of the fire and smoke.

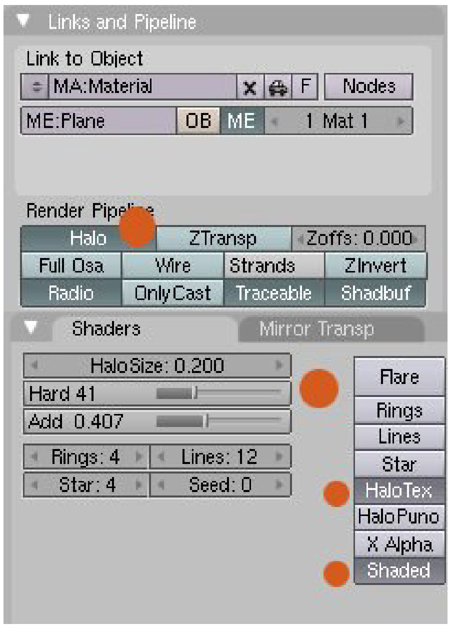
Click on the “Material” button.



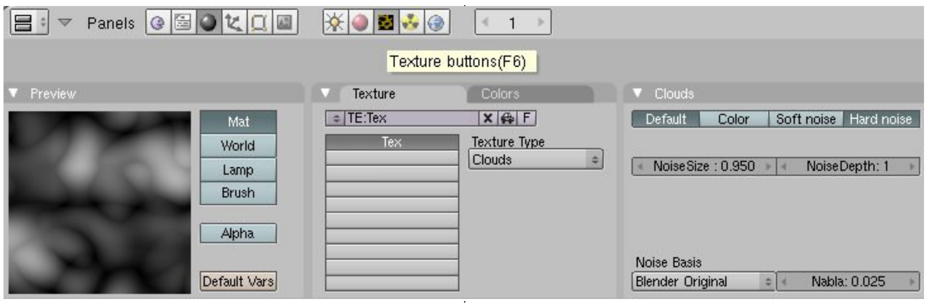
Assign a new material to your object. Click on “Add New”.


Links and Pipeline
Link to Object
Add New
ME:Cube OB ME 0 Mat 0

Firstly, we will set the material to create a “Halo” material. This is used along with particle systems and can create starry, or smoky effects. Click on the “halo” button. A new set of buttons will appear. Set those according to these settings:

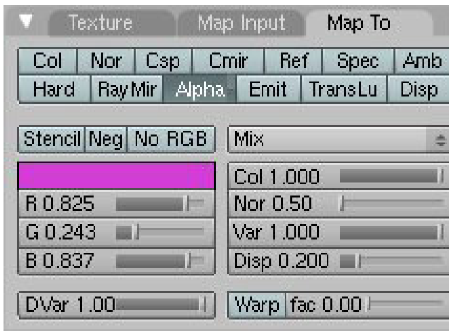

Links and Pipeline
Link to Object
MA:Material X F Nodes
ME:Plane OB ME 1 Mat 1
Render Pipeline
Halo ZTransp Zoffs: 0.000
Full Osa Wire Strands ZInvert
Radio OnlyCast Traceable Shadbuf
Shaders Mirror Transp
HaloSize: 0.200
Hard 41
Add 0.407
Rings: 4 Lines: 12
Star: 4 Seed: 0
Flare
Rings
Lines
Star
HaloTex
HaloPuno
X Alpha
Shaded

Next, we will add a texture, and use the mapping options to get that texture to influence the alpha channel of the particles. Click on the “Texture” button (that's the leopard skin one). Add a new texture, make it a “Clouds” type, and then set the settings as shown below.


Panels 1
Texture buttons(F6)
Preview
Mat
World
Lamp
Brush
Alpha
Default Vars
Texture Colors
TE:Tex X F
Tex
Texture Type
Clouds
Clouds
Default Color Soft noise Hard noise
NoiseSize : 0.950 NoiseDepth: 1
Noise Basis
Blender Original Nabla: 0.025

The size of our particles, along with the size of the scene, will influence what the particles look like. Tweak the particle size settings, as well as the noise settings (after we've done the mapping), to get the desired effect.

Transparency (alpha channel) will be based on 100% transparent for black, and fully opaque for white, with shades of gray offering various levels of transparency.


Texture Map Input Map To
Col Nor Csp Cmir Ref Spec Amb
Hard RayMir Alpha Emit TransLu Disp
Stencil Neg No RGB Mix
R 0.825 Col 1.000
G 0.243 Nor 0.50
B 0.837 Var 1.000
Disp 0.200
DVar 1.00 Warp fac 0.00

Click back to the Materials button. You should see the mapping options towards the right-hand side of your screen if you have a default screen layout. Activate the “Map To” tab, and activate the “Alpha” button. This tells Blender to use the colours from the texture to calculate an alpha channel. Don't worry about the pink color swatch. We are overriding that with the texture. One thing remains - to animate the color parameters on the particles. Don't get confused here:

We added a texture to control transparency, not change the actual colour of the particles! (You can see that the “Col” button is not activated in the “Map To” tab)

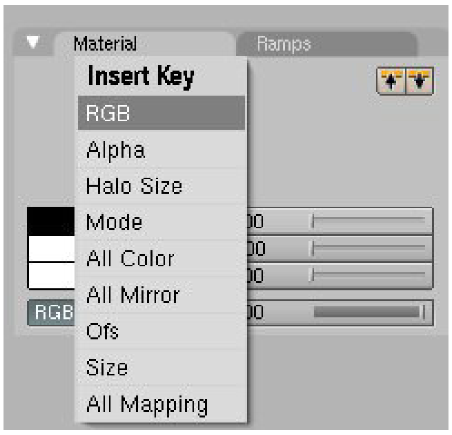
We will keyframe the RGB colors using the IKEY.

Ensure that you are on FRAME 1.

Again.

Ensure that you are on FRAME 1. You are going to struggle if you leave out this step!

Set the color to black, hover the mouse over the black swatch, and press the “IKEY”. A menu pops up. Choose “RGB”.



Now advance your frames to frame 4. Change the colour swatch to a bright red, and insert a RGB keyframe.

Advance to frame 12. Keyframe that same red color again.

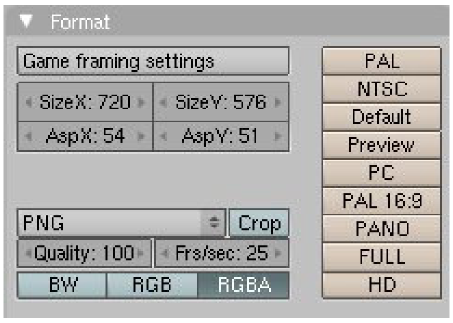
Move on to frame 25, keyframe a bright orange RGB value.

At frame 40, keyframe a light grey RGB value.

Last, but not least, use the same method to keyframe two values of the “A” (short for 'Alpha'):

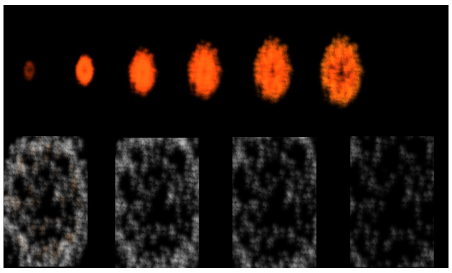
Value of '1' at frame 44;

Value of '0' at frame 70.


Format
Game framing settings
SizeX: 720 SizeY: 576
AspX: 54 AspY: 51
PNG Crop
Quality: 100 Frs/sec: 25
BW RGB RGBA
PAL
NTSC
Default
Preview
PC
PAL 16:9
PANO
FULL
HD

You can now set up your camera and lights, and start rendering your sequence. My suggestion would be to render PNG files with the alpha channel switched on. You can then composite your explosion on top of other images.

?RMAN



TAKING THE BLUE PILL FOR ...

GAME MAKER

Get Game Maker:

www.yoyogames.com/gamemaker

Tutorials and examples:

www.gamedotdev.co.za
www.yoyogames.com/make/tutorials
64digits.com/games/index.php?example=1

Game Maker Community:

gmc.yoyogames.com

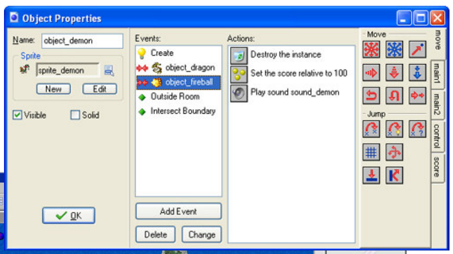
If you ask a triple-A game developer what is required to create a quality game, he'll almost certainly include 'a talented programming team' in his list of prerequisites, leaving many a budding game developer to wallow in sorrow. Thankfully, a Dutch computer scientist named Mark Overmars decided that complex programming needn't be a hurdle for producing quality games. He created Game Maker, a true boon to the game development community.
Anyone who ever had even the most fleeting thoughts about a potentially interesting game idea will find himself at home with Game Maker's simple, yet powerful, tools. These very tools also happen to be free. Well, most of them at least. The program comes of two versions: The unregistered ('Lite') version, which is free of charge, and the registered ('Pro') version. The Pro version, at the meager price of €15 (roughly R150), includes support for advanced drawing effects and functions, an effects system, and support for .dll imports, among other things.
Game Maker's strength lies in its innovative drag-drop interface, which allows multiple game creation opportunities with little or no programming required. As such, the world of game development is now open to those without intimate knowledge of computer systems and programming languages.

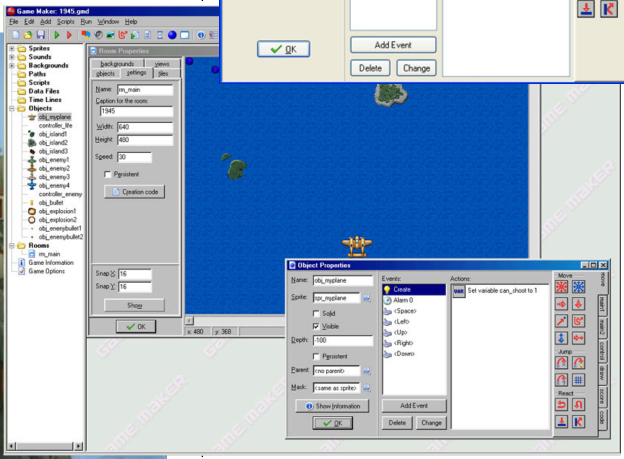
But before any of you hardcore game programmers write off this 'measly' program, know that, for those inclined to delve deeper, Game Maker includes a robust scripting language that will allow even the most complex interactions to be created. The scripting language, known as Game Maker Language or GML, increases the appeal of Game Maker beyond the non-programmer crowd. Together with the scripting language, high quality games can be created. Remember *The Cleaner,* a game we reviewed a while back? You guessed it! That was made in GM. As was *Ninja loves Pirate*. (http://www.ninjalovespirate.com/)

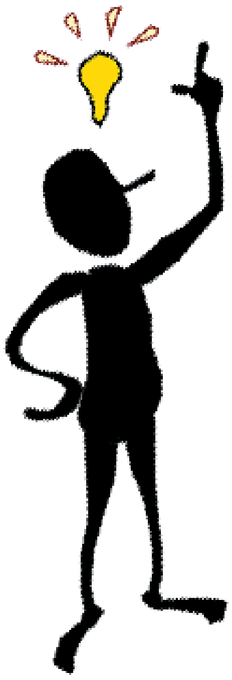
Game Maker also boasts a huge community base, providing assistance and tutorials for anyone eager to learn. The official Game Maker Community is likely the largest of these, but many more are abound. A previous Dev.Mag issue (Issue 9) includes a more exhaustive list.

So wallow no more. Go grab Game Maker and start making games. And once you're done, we'll be more than eager to see what you've made.

CHIPPIT


Object Properties
Name: object_demon
Sprite: sprite_demon
[✓ Visible] Solid
Events: Create; object_dragon; object_fireball; Outside Room; Intersect Boundary
Actions: Destroy the instance; Set the score relative to 100; Play sound sound_demon


Game Maker: 1945.gmd
Sprites, Sounds, Backgrounds, Paths, Scripts, Data Files, Time Lines, Objects: obj_myplane, controller_life, obj_island1, obj_island2, obj_island3, obj_enemy1, obj_enemy2, obj_enemy3, obj_enemy4, controller_enemy, obj_bullet, obj_explosion1, obj_explosion2, obj_enemybullet1, obj_enemybullet2; Rooms: rm_main; Game Information; Game Options
Room Properties
Name: rm_main
Caption for the room: 1945
Width: 640
Height: 480
Speed: 30
Persistent
Snap X: 16
Snap Y: 16
x: 480 y: 368
Object Properties
Name: obj_myplane
Sprite: spr_myplane
Solid
[✓ Visible]
Depth: -100
Persistent
Parent: <no parent>
Mask: <same as sprite>
Events: Create; Alarm 0; <Space>; <Left>; <Up>; <Right>; <Down>
Actions: Set variable can_shoot to 1



A BEGINNER'S GUIDE TO MAKING GAMES

PART 1

Welcome to the first in the Beginner's Guide to Making Games. Each month a single important concept required for making games will be discussed in detail. This month we're starting with moving objects, and next month we'll look at collisions. These have been chosen as the first two concepts because a good understanding of them will allow you to start building your own unique games.

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 code 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 assumed that the reader of the article has completed the first official Game Maker tutorial and is able, therefore, to create sprites and objects, it is quite possible to follow the article without having done so. The article is structured not only to introduce a new programmer to the concept, but should be of some value to the intermediate level programmer as well.

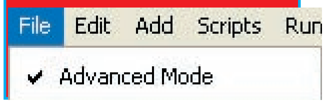
Moving an object – GM tutorial

The vast majority of games require movement within the game. Some games such as word puzzles get away without using moving objects but these are the exception rather than the rule. To keep things simple, we'll only look at moving an object in 2D space, while a later article may look at adding the third dimension.

Today's article will contain the following sections:

1.A brief Game Maker tutorial showing the basics of getting an object to move, both with the keyboard and with the mouse.
2.This is followed by a discussion, in detail, of the various Game Maker actions available to you.
3.Game Maker Language (GML) is a powerful way for the developer to get closer to the action, and contains numerous functions for moving objects.

HINT: When using Game Maker always set the Advanced option on. This makes it easier to see and access all the functions.



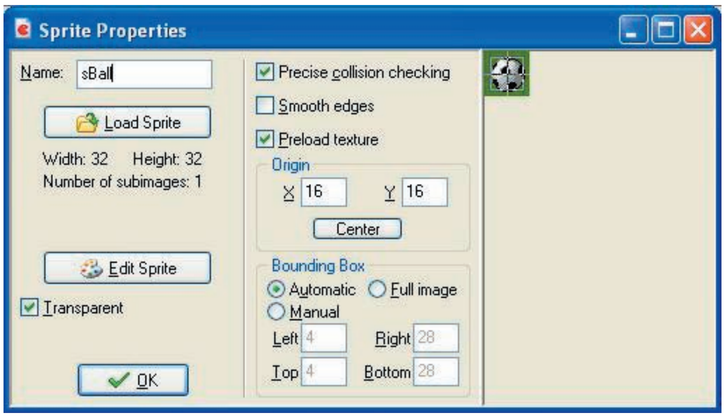
Game Maker Tutorial

This tutorial is going to create a room that contains two balls. The first ball will be controlled by the arrow keys on the keyboard and the second ball by clicking the mouse.

This tutorial is not a game as such, but is designed instead to show various methods of making objects move. A few small changes could make this into a fun game. Some possibilities might include spreading various objects around the screen and scoring as they get eaten by the balls, creating a two player Pac Man game where the balls must rush through a maze, of even a chasing game where one player must try and catch the other.

Step 1 – Create sprites

In Game Maker, add a new sprite. Select an image for it, such as one of the balls in the default graphics. The sprite screen should look like the image over the page:


Sprite Properties

Name: sBall

Width: 32 Height: 32

Number of subimages: 1

☑ Transparent

☑ Precise collision checking

☐ Smooth edges

☑ Preload texture

Origin: X 16 Y 16

Bounding Box: Automatic [✓] Full image / Manual

Left 4 Right 28

Top 4 Bottom 28

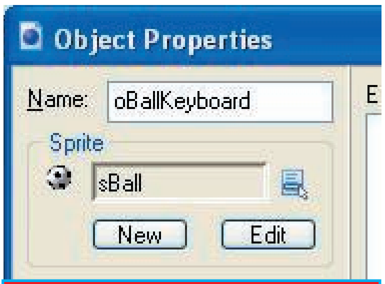
While not important for this tutorial, it is often a good idea to set the origin of a sprite to the centre of the sprite to assist with collision detection. To do so press the Centre button on the Sprite screen.

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

Step 2 – Making an object

Game Maker uses a sprite to store the graphics for the game, and the objects in the game are represented by a Game Maker Object. Create a new object using the previously created sprite.

Name this object oBallKeyboard to show that this ball will be controlled by the keyboard.


Object Properties

Name: oBallKeyboard

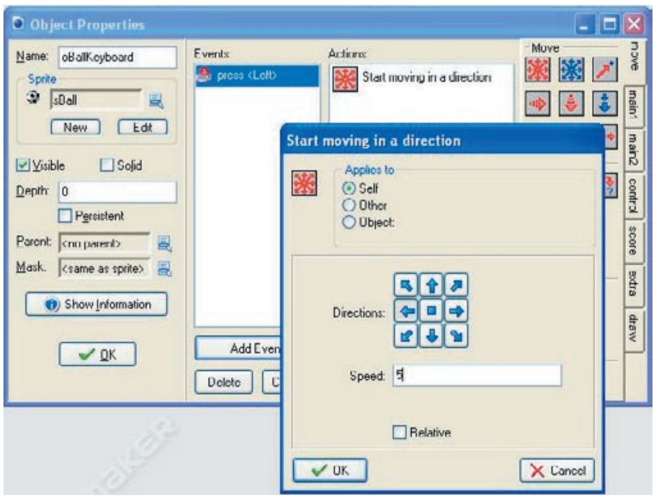
Sprite: sBall

HINT: Always rename the items you create in Game Maker. It is always easier to remember that something called sBall is a sprite containing the image of a ball than to try and remember that sprite125 is a 16 frame animated image of a vampire firing a bazooka!

To move an object in Game Maker we need to create events and allocate actions to the events. For the keyboard we need to create four events (one for each arrow key direction) and allocate a "Start moving in a direction" action to each.

On the object screen select the Add Event button, and in the following dialogue box select Key Press and then choose the Left option from the menu. This creates an event that will occur whenever the player of the game presses the left arrow key.

To move the object on the screen we now need to click and drag the "Start moving in a direction" action into the actions list. This is the first action on the Move action screen. For more details on how the "Start moving in a direction" action works, consult the next section.


Object Properties

Name: oBallKeyboard

Sprite: sBall

☑ Visible ☐ Solid

Depth: 0

☐ Persistent

Parent: <no parent>

Mask: <same as sprite>

Events: press <Left>

Actions: Start moving in a direction

Start moving in a direction

Applies to: Self [✓] / Other / Object

Speed: 5

☐ Relative

Whenever an action is added to the Actions list, a dialogue box will be displayed where you can set the parameters for the action. In this case we click the arrow pointing left and set the speed to 5. (Note that this speed is good for testing things, but can sometimes be too fast for actual in-game events).

Step 3 – Adding our object to the room

Rooms are where everything happens in a Game Maker game. Without a room, objects can't do anything. Rooms can also be set up to contain pretty backgrounds and can transition in fancy ways.

Create a new room in Game Maker. On the left tab ensure that the oBallKeyboard object is selected to be added to the room. If it isn't, click on the selection box under "Object to add with left mouse" and select it from the list. Click with the left mouse button anywhere in the room.

Step 4 – It works

Run your game by pressing the green "Run Game" button. The beauty of Game Maker is how your game runs every time without fail.

HINT: A good habit to get into is saving your game every time before you run it!

Press the Left arrow key and your ball will go flying off to the left. Unfortunately there is no way of changing its direction or stopping it at the moment. To be able change the direction of the ball repeat what was done above but for the other three directions of the "Key Press" event.

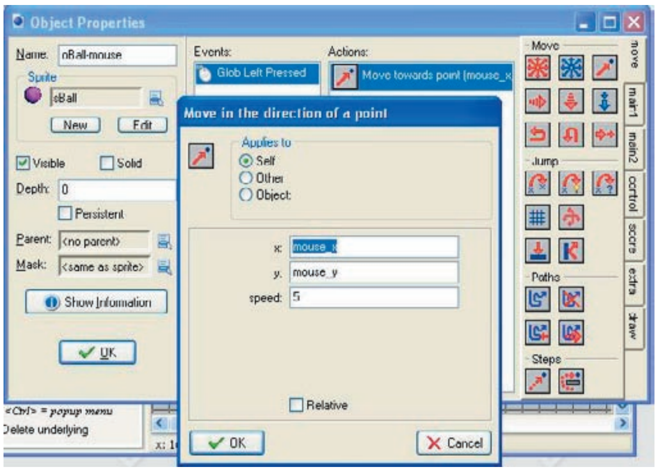
Step 5 – Mouse movement

Moving objects using the keyboard is just one method of moving the object on the screen. Another frequently used method of controlling object movement is to use the mouse. Making an object move toward the mouse is in fact very easy.

Follow the previous instructions to create a new sprite (using a different ball picture), and a new object called “oBallMouse”.

Create a new event for the next object, select the “Mouse”, “Global Mouse Event” then the “Global Left Pressed”. Mouse events in Game Maker can apply to the object individually or on a global level. Mouse Events such as “Mouse”, “Left Button” will only apply if the player left clicks on the actual object. The “Global Mouse Events” will trigger for the object irrespective of where the player clicks on the game screen.

When using the Game Maker actions, it is possible to use the internal Game Maker functions as parameters. In this case we want to make the object move in a direction toward the mouse. Fortunately there is a Game Maker action called “Move in the direction of a point”. Using the internal Game Maker mouse information, we can just set the x and y coordinates to mouse_x and mouse_y which represent the current position of the mouse.


Object Properties
Name: oBall-mouse
Sprite: sBall
[✓ Visible] Solid
Depth: 0
Persistent
Parent: <no parent>
Mask: <same as sprite>
Events: Glob Left Pressed
Actions: Move towards point (mouse_x

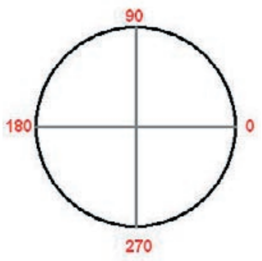
Move in the direction of a point
Applies to: [✓ Self] Other Object:
x: mouse_x
y: mouse_y
speed: 5
Relative

Add the new object into the room. Run the game again and click randomly on the screen and watch how your new ball follows the clicks around the screen.

Moving objects with actions

Game Maker is cool because its easy, and Actions are some of the most important things that make Game Maker such as easy tool to use. To get the most out of the Game Maker actions, you need to understand what they do, and what alternate options are available for each thing you want to do.

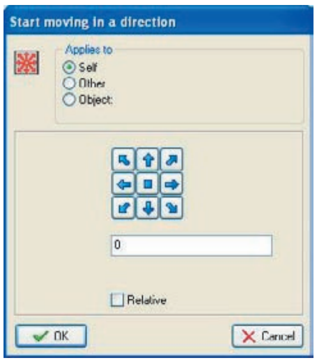
Often in Game Maker there are alternate options to achieve the same thing. However by understanding what the various options do, they can be better used to achieve the effects that your game requires.



Movement in actions consists of two elements, direction and speed. Speed indicates how far the object will move in each step, so the higher the speed value, the faster the object will move. Direction shows, in degrees, the direction the object should be moving starting at zero being right and moving anti-clockwise. A direction of 90 indicates a direction directly up, while a value of 180 would mean directly left.

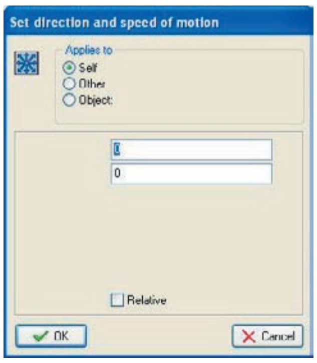
A nice trick to understand in Game Maker is the use of the Relative checkbox. When the relative check box is checked the values entered into the parameters boxes are added to the current values. So for example if an object is moving at a speed of 5, and a relative value of 1 is selected the object will start moving at 6 thereafter.

Start moving in a direction


Start moving in a direction
Applies to: [✓ Self] Other Object:
0
Relative

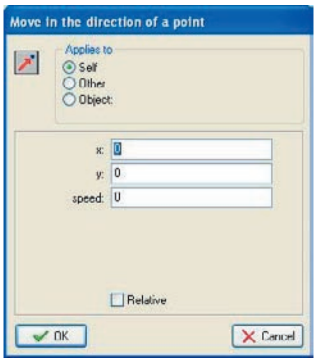
The most basic of the movement actions allows you to choose the direction you want the object to move without needing to understand what degrees and direction mean. This action is best used when you are designing a game that uses clearly defined horizontal and vertical lines, for example a maze game.

Set direction and speed of movement


Set direction and speed of motion
Applies to: [✓ Self] Other Object:
0
0
Relative

This option allows you to move an object in any direction and at any speed. To use it properly you need to understand how the various degree points work. Being able to set any direction for movement gives the developer a very large range of movement options. Being able to move in any direction is typically used in games that use an open game environment, where an object needs to move in any direction between points, such as enemy fighter planes in a shooter game.

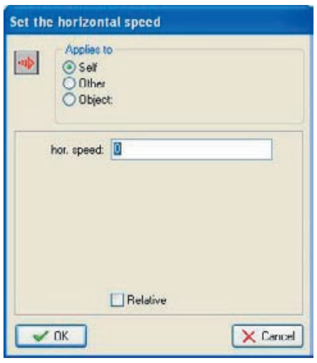
Move in a direction to a point



Rather than forcing the game developer to work out the direction an object would need to move in to get to a point, Game Maker has supplied an action to just define the point to which an object needs to move. Internally, this action uses the objects current position and the destination point to calculate the direction the object needs to move in.

Moving directly to a point is very useful in games where one object moves toward another to attack it, such as enemy bullets being shot at the player.

Set the horizontal speed, set the vertical speed



These very simple options allow you to move an object in a horizontal or vertical manner at a set speed. Note, however, that these options do not cancel the other directional speed that has been set. This means that if an object is moving up the screen and a Horizontal Speed is set the object will begin moving in a diagonal direction and not just horizontally. These actions are useful in games that need some sort of acceleration or deceleration in a direction option rather than specific speed settings.

Some examples

Here are some examples of doing the same thing in many different ways.

1.Moving an Object Left
   a.Use "Start moving in a direction", choose Left and define a speed
   b.Use "Set direction and speed of movement", with a direction of 180 and a speed.
   c.Use "Move in a direction to a point", set the new point's X value to "x-100" and set the required speed.
2.Move toward the mouse
   a.Use "Move in a direction to a point", set the X to mouse_x and the Y to mouse_y.
   b.Set the direction to point_direction(x,y,mouse_x,mouse_y) and the speed

There are almost always various different ways to make an object move the way you want it to.

Moving objects with GML

Game Maker actions are short cuts to Game Maker Language constructs. While that sounds complicated, it basically means that actions are the easier way of doing things that can be done in Game Maker Language.

Before we start discussing the various options for making objects move, here is a quick breakdown of how movement in Game Maker works.

All movement is based on a direction and a speed. Each step (frame) makes the object move a number of pixels on the screen as specified by the speed in a specific direction. Another way of looking at this is that the objects move a set speed in a vertical and horizontal distance. All these values are properties of the object and can be manipulated directly.

Property	Description
hspeed	Horizontal component of the speed.
vspeed	Vertical component of the speed.
direction	Current direction (0-360, counter-clockwise, 0 = to the right).
speed	Current speed (pixels per step).

Now it's important to understand that these properties adjust based on the value set in the other properties. So setting the direction and speed of an object also changes the hspeed and vspeed of the object.

Calling a script

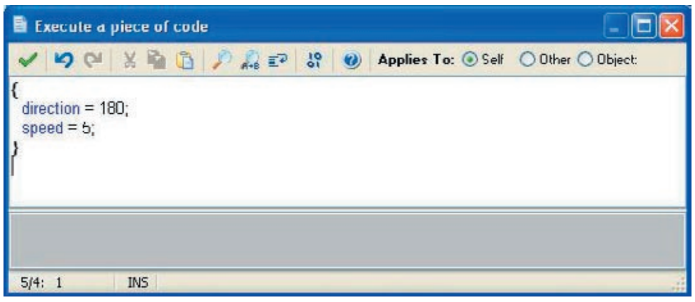
To call a script for an object, add an event to the object and add the "Execute a piece of code" action to the action list. In the popup text editor, add the commands you want to be executed.

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

Remember also that it is possible to access properties of other objects. So it's quite possible, for example. to do things like moving toward other objects or using the position of other objects to define the movements of the object.

Moving the object

In GML moving an object in a specific direction is as easy as setting the direction and speed properties of the object. So by creating a "Key Press", "Left" event and executing the following code:



```
{
direction = 180;
speed = 5;
}
```

The object will move toward the left just as if the "Start moving in a Direction" action was used to move left at a speed of 5.

The other movement actions can similarly be defined in code. To swap the horizontal or vertical movement the following statement can be used:

```
vspeed = -vspeed;
```

And to make the object move in exactly the opposite direction to which it is currently moving, try:

```
Direction = 360 - direction;
```

Special moves

GML contains some special movement functions. The various jump functions are being excluded as they will be covered in a future article. This article looks only at the functions that affect the movement of the object.

move_towards_point(x,y,speed)

Does the same as the "Move in the direction of a point" action. So in a global mouse left pressed event a piece of script

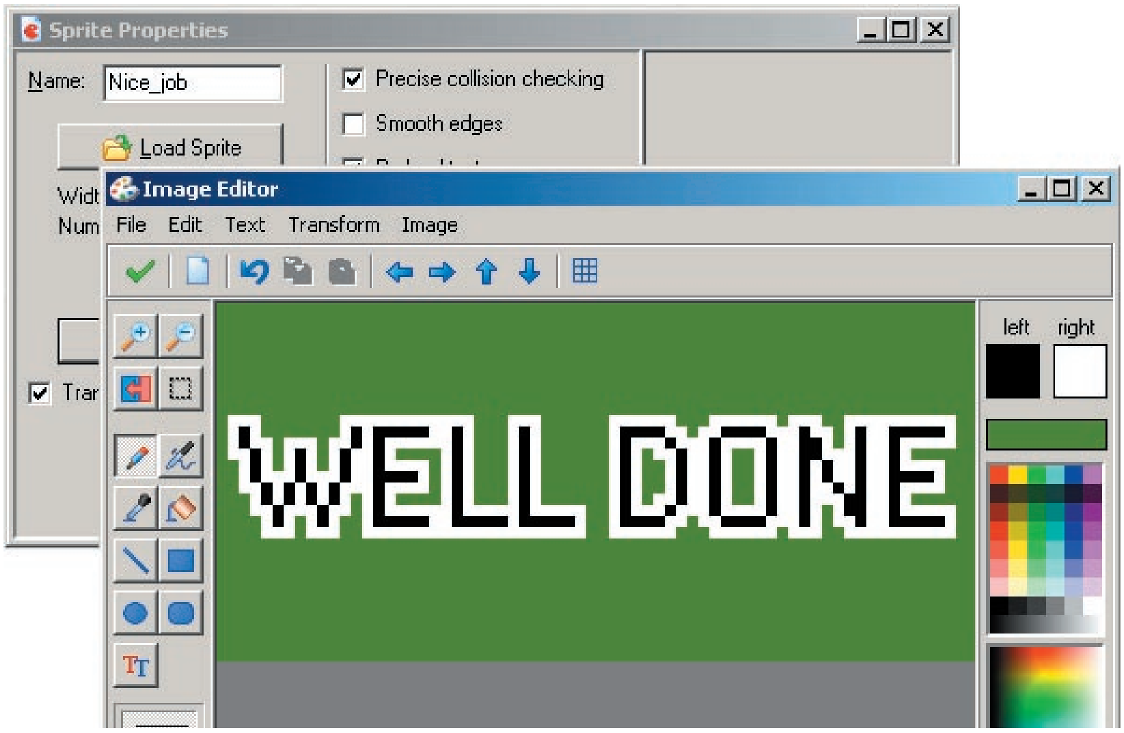
```
move_towards_point(mouse_x,mouse_y,5);
```

will move the object towards the position of the mouse.

The power of scripts

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

CAIRNSWM



ROACH TOASTER 2: PICKING OUT THE BUGS PART 5

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

So. Roach Toaster 2 has, as of June 2007, been a year into development from initial design to beta. I am very very proud of what I have been able to accomplish. There have been lots of things that nabbed my attention from Roach Toaster 2 - school, games, game dev competitions and just life in general.

As I am sitting here, Roach Toaster 2 is ready for public testing. This testing phase is basically for testing the full gameplay. Does it work? Is it fun? The next phase will be the meticulous balancing/polishing phase. I want to have a full-blown first beta thrown out everywhere. I even want to flaunt it to possible publishers. The big thing, however, is that I am stuck without a proper internet connection!

Yes, the bane that is South Africa's telecommunications did not plan well enough. The region I live in currently has a shortage of telephone points. They are in the process of upgrading it. The way I currently connect to the Internet is not so cost effective. It is about R2 per megabyte!

Without my proper internet connection, I won't be able to post, read feedback, and upload updates without chalking up a considerable amount in telephone bills. So, during this hiatus, I am doing five things concerning Roach Toaster as a whole.

1) I'm getting the art/sprites for Roach Toaster 2 done. I've already got an awesome main menu page from a local artist.

2) I've always wanted to remake Roach Toaster 1. This is because people might want to play Roach Toaster 1 as an extended demo for RT2, and because of Roach Toaster 1's poor production quality, I am working on a remake. This remake will be made in XNA for a competition. Due to XNA's bad sharing capabilities, I might just port the graphics to the Game Maker file in the end.

3) I already have the gameplay and story development in mind up to Roach Toaster 4, and I need this time to develop a character that will appear in the upcoming games. This might seem odd, but when Roach Toaster 2 gets released you will know exactly what I mean by this.

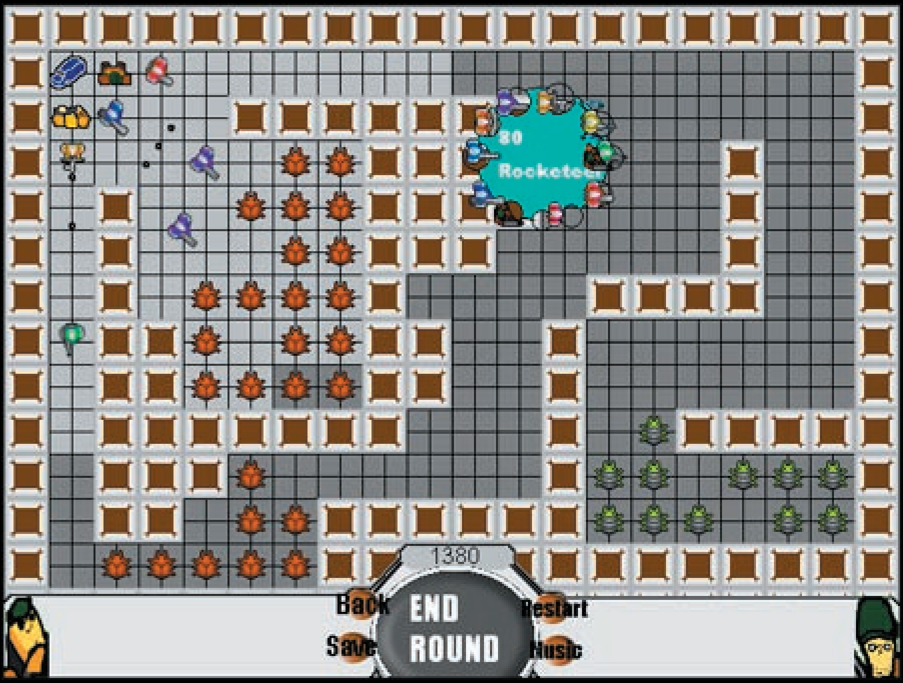
4) If you don't know, the official Roach Toaster website is up! http://www.roachtoaster.com. I plan to revamp the site a bit, i.e. add pictures/screenshots everywhere, and just pretty it up.

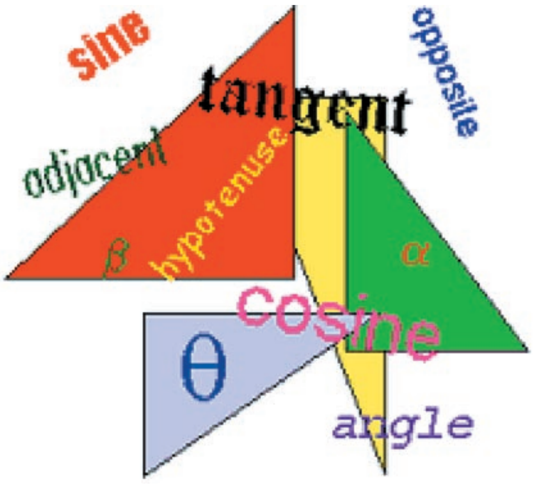
5) I also plan to sort out the online payment options. I have no clue how this works, and I have signed up for several e-commerce providers just to get a better idea about it all.

My original intention was to release the beta in December 2006... Oh, how wrong I was! As you can see, I have no idea when Roach Toaster 2 will be ready for release. With the way it has developed thus far, I'd say (with no certainty) it will be finished round about October 2007 (yes, rather set too far into the future, than too soon).

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

TROOJG





GAME CODING WITH TRIGONOMETRY PART 1

Trigonometry. The mere mention of it strikes terror into the minds of bewildered high school students. Confusing ratios, bizarre diagrams, trying to find the sine of the cosine of the tangent of theta - many people struggle to wrap their minds around it and eventually give up in desperation to do something easier, such as ending war or achieving cold fusion. Well, despite what your high school teachers may have said, trigonometry can actually be useful! Yes, in the vast sea of unintelligible terms is a valuable and powerful mathematical tool that can aid you immensely in the humble task of game design. The aim of this series is to explain to you just how simple trigonometry really is, and how you can use it to solve some very common programming problems. Hopefully it will make the job of coding your next blockbuster game just that little bit easier.

First Principles - The Triangle

Before we start writing any trig in our code, it's necessary to understand all the nasty mathematical stuff behind it (just joking, it's not that bad!). First, we should take a look at how trigonometry works at the most basic level. To do that, we'll need a triangle. More specifically, we'll need a right-angled triangle (a triangle that contains a single 90 degree corner). Here's one I found lying around:

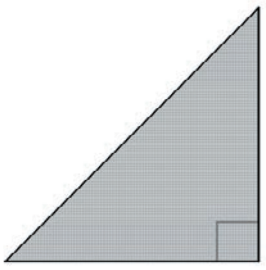


Figure 1: A boring right-angled triangle

Firstly, it's important that we name the sides of the triangle so that we can identify them when we build our equations later. In trig, we name sides based on their relationships to the corners of the triangle. Take this triangle as an example:

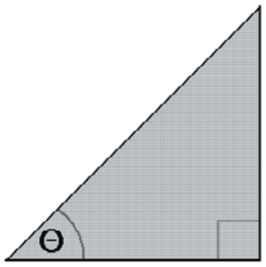


Figure 2: Boring right-angled triangle with angle theta

In this triangle, we've defined one of the non-right-angles with the variable theta (Θ). For now, this is simply a variable to identify the corner - don't worry about about how the numbers fit in just yet! Now that we've defined Θ, we can use it to identify our sides.

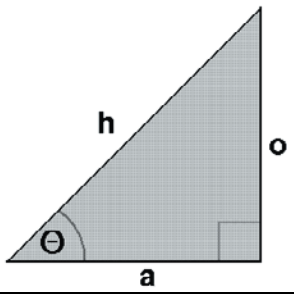


Figure 3: RA triangle with angles and sides defined

Now we've designated our sides. In trig, sides are always designated relative to the angle that we're working with, Θ in this case. o is the side opposite to the specified angle. a is the angle adjacent (next to) the specified angle. h is the hypotenuse, which is always the side opposite to the 90 degree angle. Clear? Well, if not, let's go through the same exercise with the other angle in the triangle. Let's designate that one Alpha (α), and take a look at how this affects our sides.

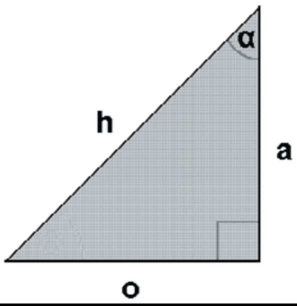


Figure 4: Right triangle with angle alpha, and sides

Note that this triangle is still the same triangle. We've just renamed the sides according to which corner we're viewing them relative to, in exactly the same way that we did with angle Θ. You'll note that h is still assigned to the same side. As I mentioned, this is because the hypotenuse is always the side opposite to the 90 degree angle, and so won't change if we switch corners. If this still seems a little confusing, don't worry - next we'll be seeing how all of this ties together!

Trigonometric Ratios

Trigonometry is not a new thing. We owe a lot of our geometrical knowledge to ancient Greek mathematicians, such as Pythagoras, who dedicated a lot of time to figuring out the relationships between the sides and angles of shapes. One of the many things that the Greeks managed to determine was that the lengths of the sides of right-angled triangles were subject to certain ratios, extrapolated relative to triangle's corners in the way that was discussed in the previous section. From this, they derived common mathematical functions to use these ratios for their calculations. For ease of use, they gave names to the six functions that they discovered. We know them as sine (sin), cosine (cos), tangent (tan), cotangent (cot), secant (sec), and cosecant (cosec).

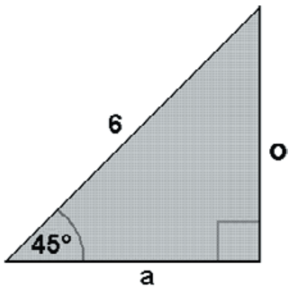
The ratios that the names describe are as follows. Note that the Θ after the ratio names denotes which angle we are taking these sides relative to.

$\sin\theta=\frac{o}{h}$	$\cos\theta=\frac{a}{h}$	$\tan\theta=\frac{o}{a}$
$sec\,\theta=\frac{h}{a}$	$cosec\,\theta=\frac{h}{o}$	$\cot\theta=\frac{a}{o}$

This is all good and fine, but the real magic comes in when we slot numbers into those variables. Time for some good old-fashioned maths!

Basic Calculations

Let's take the triangle that we've been working on and plug some numbers into it:



Now we have numbers to work with! However, you'll notice that side o and a haven't been given numbers. We have no idea what the lengths of those sides are. That means, of course, that we'll have to calculate them!

Calculating anything using trigonometry requires three steps:
1.Determine what numbers you have, and what numbers you want.
2.Determine which ratio to use, and build your equation.
3.Substitute the known values into the equation and solve for the unknown.

That may seem like quite a brainful, but once you see it in action you'll appreciate its simplicity!

First, we take a look at what we have: we have values for Θ (45°) and h (6). We want to calculate o and a. Next, we determine which ratios we need to use. Generally, we decide that based on the formula

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

Let's calculate o first. We'll use h as our known value. As a result, the above will look like this:

$$Ratio(\theta)=\frac{o}{h}$$

Looking familiar? Look back to the table where I outlined the ratios. If you read carefully, you'll see that the ratio corresponding to o/h is sin. Our equation now looks like this:

$$\sin(\theta)=\frac{o}{h}$$

And when we plug our numbers in and solve for o:

$$\sin(45)=\frac{o}{6}$$

$$0.707=\frac{o}{6}$$

$$4.424=o$$

Note: You can easily use the Windows calculator in Scientific mode to work with trig ratios. Simply ensure that the calculator is set to "degrees", then punch in the angle (45 in this case) and click the "sin" button to get the result above (0.707). I've rounded the answer off to three decimals for convenience.

And there you have it! We've calculated o to be 4.424 units long! Now try calculating a (hint: you'll be using cos as your ratio - check the list again to see why). You should get an answer of about 4.243.

What? Out of space already?

That's all for this month, but despair not! Next month, we'll be looking at how to calculate unknown angles, and you'll also get a taste of how you can start using trig methods in your games. Until then, practice calculating unknown sides using different ratios and values. I'll see you next month!

GAZZA_N

THE HISTORY OF I-IMAGINE

Part 5: The First 9 Months

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

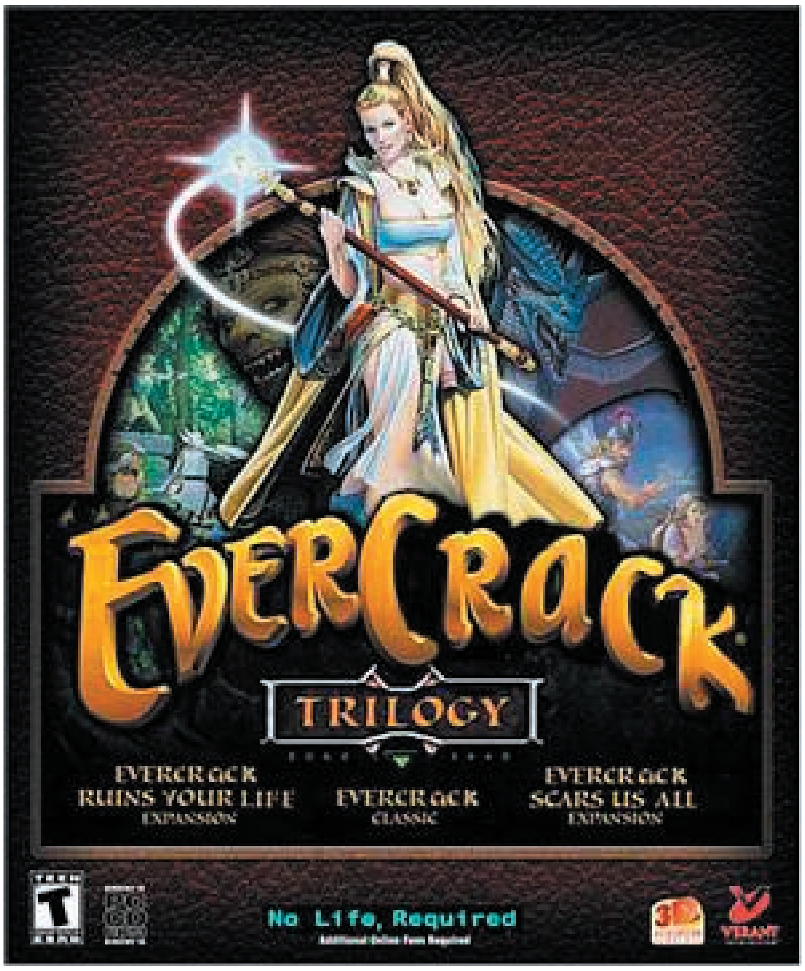
I'm pretty sure that not many of you have had the pleasure of experiencing the 18 hour non-stop flight between New York and Johannesburg. For those of you who haven't, re-read the previous sentence and replace "pleasure" with "misfortune", and add "from hell" directly after "flight" if you would like have an idea as to what it really feels like. Actually, I could fill an entire separate article just describing how those 18 hours feel more like 3 ½ years, and how I no longer fear what may await me in the afterlife. But I digress... Looking back on what is nearly eight years now, the first few months that I-Imagine spent on developing what would eventually turn into Chase: Hollywood Stunt Driver were filled with one growing pain after another. The relatively easy part of the job had been done, since we had decided upon a concept to go about developing our first game around. Now we were at the stage where we had to dig in and actually produce the goods, which was not going to be an easy task given that we had such a small team consisting of only four programmers, two artists and a single designer.

The programming tasks ended up being broken down quite easily with Dave's main task being to handle all of the rigid body physics (for vehicles as well as dynamic objects), Matt's main task being to create plug-in tools for 3DS MAX as well as the vehicle AI, and Dan's main task being the rendering engine as well as assuming the mantle of Lead Programmer. Seeing as how I was the least experienced programmer in the team, I was pretty much left to help out wherever I was needed, as well as to tackle all of the little filler tasks such as the GUI (which isn't really a filler task at all...) and various small tools.

The art tasks were pretty much divvied up evenly between Hoang and Felix with both of them being equally responsible for level and object building, although Hoang took on all of the conceptual artwork creation as well. Lastly, although the game was very much a design-by-committee production, it was left up to Kenny to fill in a lot of the blanks, as well as to create and maintain our design document.

The rest of 1999 was filled with a lot of hard work for the team, but it was also a lot of fun. I think that we all quite enjoyed the challenge and responsibility of having gone out on our own to create what we hoped would turn out to be a hit game. But before we would have the chance to possibly reach that lofty goal, we would have to navigate some rough seas... the first of which was not very far away at all.

The tale of what would ultimately be Felix and Kenny's short time at I-Imagine began almost as soon as they had arrived in South Africa. Unbeknownst to the rest of the team, Felix and Kenny had a bit of a problem... actually it was more like addiction, and it wasn't long before it came to the forefront and started to cause an issue between them and the rest of the team.

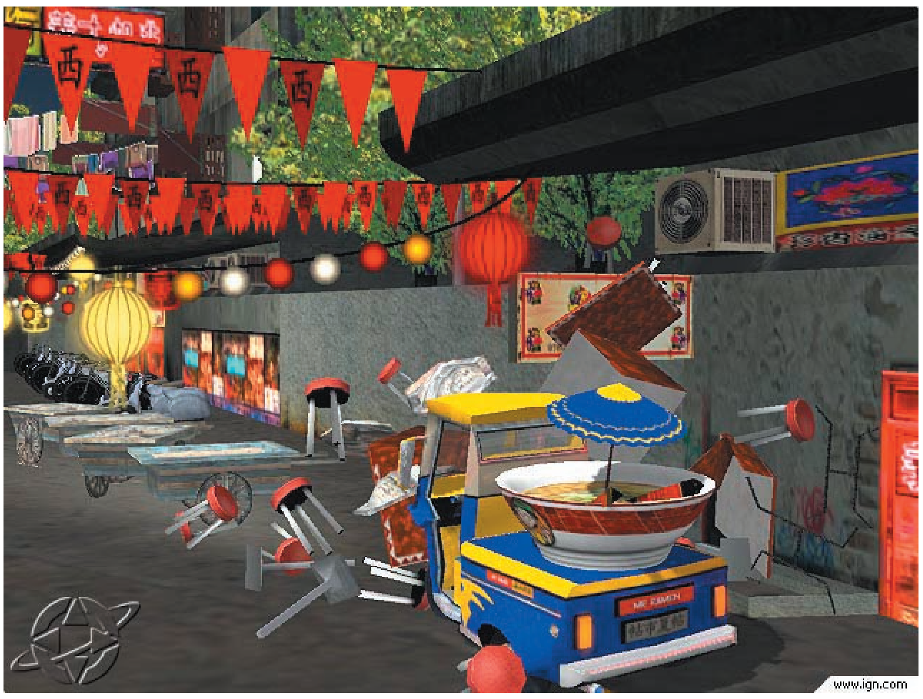

EverCrack
TRILOGY
EVERCRACK RUINS YOUR LIFE EXPANSION
EVERCRACK CLASSIC
EVERCRACK SCARS US ALL EXPANSION
No Life, Required

The offending substance wasn't alcohol, drugs, sex, or even oiled midget wrestling. No... it was something far more sinister, devious and life threatening than any other substance known to mankind. Although I am sure that it was originally christened with a different name (which was unpronounceable by human tongues) in the fiery pits in which it was spawned, to us mere mortals it was known simply as EverQuest.

I actually shudder when I think back to how utterly dependant those two poor souls were when it came to the sweet, sweet siren call that beckoned them to her keyboarded bosom. Every day around 4pm it was though the pied piper would begin to play his magical notes and Felix and Kenny would drop all of their work, march up to their rooms, close their doors, and log onto whatever EverCrack server they belonged to in order to join up with their friends who had just woken up on the east coast of America, only to also be once again driven to their keyboards by the ever persistent harpy song playing in their subconscious.

Now, as you can imagine (from my completely down to earth and not dramatised in any way depiction of the curse that followed Felix and Kenny from day to day), their actions could not but have a negative effect on the team and the work that needed to be done. With most of their working days being made up of four or maybe five hours of actual time in front of their computers, it was going to be difficult in the least for them to continue along this path and still remain a part of the team. This meant that Dan was left with no choice but to sit down with them and have a long talk. Initially they responded well, but unfortunately it wasn't long until they fell back into their old pattern.

Dan had yet another talk and once again, things got better, but only for a short time. Eventually Dan had no choice but to give them an ultimatum: either they shape up and become effective and committed members of the team, or they would have to part ways with us. I'm not sure as to their exact reasons (my belief is that they truly missed a lot of the things that they were used to from their lives in America), but instead of trying to shape up they decided to ship out. Felix actually couldn't wait to get back to the US, and subsequently bought plane tickets and left just a few days after Dan's final meeting with them, which was around the middle of March 2000.



Even though Felix and Kenny didn't leave us under the best of circumstances, it wasn't as though any work didn't get done during the time that they were still a part of the team. In fact, most of the demo that we ended up putting together for E3 2000 was done before they had left. Our first goal was to have a fun vehicle simulation up and running with nice arcade-style physics. We achieved this by the middle of January or so with our first demo, which was a free roaming arena filled with obstacles such as ramps, half-pipes, and loops where players would try to score as many points as possible by performing rolls, flips, spins, and various other stunts in a given time limit. We knew pretty soon that we were on to something as the entire team ended up wasting way too many hours "play testing" this demo once it was done!

Now that we knew that we a physics system that would handle the type of gameplay we wanted our players to experience, we moved on to creating levels that would show off the core gameplay experience of action packed stunt driving. Our first level was designed around a post-apocalyptic setting in the vein of "Mad Max", which had the player infiltrating an enemy compound to rescue someone before escaping and having to drive onto a moving articulated truck. The gameplay here was almost platformer-like in nature, as we wanted to try something that hadn't really been done with vehicle-based games before.

Our next level was set in an Asian city where the player drove a three-wheeled ramen delivery vehicle. Upon completing a delivery, the destination building would explode and the player would have to escape chasing police officers who thought that he was the cause of the explosion. This gameplay was more straight forward A-to-B driving but it took place in a fully populated city complete with traffic. The catch that we had with this level was that the delivery vehicle had an umbrella in the back of it that bobbed around like a spring as the player drove, and everyone who saw this immediately wanted to drive it.

Soon enough the middle of May was upon us which meant that it was time for E3, so we all headed to LA to pitch our game to publishers. We rented a small stand in the Kentia Hall, which for the E3 uninformed was the smallest of the three show-halls and it was usually home to much smaller developers trying to get lucky. Throughout the show, we managed to garner a lot of interest from various publishers even though when all was said and done, we weren't able to secure a publishing deal. However, a senior producer from Infogrames (this was before they bought the rights to use the name Atari) whom we had met at the previous year's E3 came by to see what we had done and he seemed very impressed with the game - so much so that after E3 ended, we kept in contact with him and eventually sent him our design documentation for Infogrames to evaluate the potential of for publishing. But that is another story best told at a later time.

Once we returned from E3, we focussed on furthering the game in order to show more stuff for the next big show (ECTS), which was happening at the start of September. After much testing, we decided to ditch the gameplay style that we had designed for our post-apocalyptic level as we didn't think that it was paying off enough in the fun department. The platformer style gameplay was just too slow paced to keep players interested, and we decided to re-focus all of our energy into our Asian world in order to create six fast-paced missions which would keep players constantly busy and engaged, thereby creating a more solid demo of our game for publishers to take back with them and evaluate properly.

However, we would have to do most of this work without our last experienced artist, as Hoang would soon be leaving us. Unfortunately, in order for us to get everything done that we needed to do in time for our trip to E3, Hoang was put under enormous pressure. This was due to Felix having left, along with the fact that we had only managed to find and hire a single South African artist to replace his many years of experience. Due to this lack of foresight on our part, Hoang pretty much crunched 12+ hours a day for the 2 months leading up to E3, and he was quite burnt out afterwards. So, once we arrived back from E3, he was given about a month or so off, which he used to relax and travel around Africa while we hoped that he would feel rejuvenated and be ready to come back to work when he returned. Unfortunately, when his vacation was over he told us all that he wasn't ready to continue working, and that he wanted to return to the US to pursue something other than video game development for the time being.

Reluctantly we had to say goodbye to Hoang and push on towards our next deadline which was only a few months away. Luckily we had hired 2 more South Africans just prior to E3, and although they weren't able to help Hoang with his push for that deadline due to their inexperience, he was able to help train and get them up to speed before he ended up leaving us.

Now, it was up to an entirely new art team to help carry I-Imagine forward, and hopefully secure a publishing deal for our game in what would eventually turn out to be our most important and productive show in the history of I-Imagine.

COOLHAND

JOHANNESBURG DEVLAN

GATHERING FOR FUN AND ARTIFICIAL INTELLIGENCE

Occasionally, a few of the Game.Dev members decide to get together in a non-Hotlabby kind of way and duke it out with some code. The devLANS are a proud example of how local game developers are able to combine a chilled attitude with a thirst for learning something new. One of the attendees of the latest such gathering describes the experience.

The topic of May's devLAN in Johannesburg was Artificial Intelligence, and we are very fortunate that we have Danny "Dislekcia" Day living up here who provided us with a talk and some advice on the subject. Although the speaker was unavoidably late due to family matters, Danny nevertheless gave a good definition of AI in relation to game development and a breakdown of the phases or steps which an AI must go through in a game, as well as various systems available which the developer can use to create an AI.

William "cairnswm" Cairns opened his house to the local members for the devLAN as he had become partly immobilised due to a running accident in which he broke his leg, and having the devLAN in his home meant he didn't have to travel far and could participate.

The group was quite small this time around with just 5 of us, but it still seems the game developers in Johannesburg are taking their hobby seriously as half of us had an existing project which we were already working on where we needed some form of dynamic interaction. We were hoping to use the devLAN as the perfect opportunity to get this done.

Danny attested that "AI is the art of making things happen dynamically", and an AI can fulfil multiple roles like helping the player manage his resources and make the world he's playing in more believable than just being a bot the player can play against. We then examined the steps an AI must take while operating which are: Information Gathering, Information Weighing and Information Output; after which we then explored various systems which a developer could use to let the computer reach a decision. We looked at Fall-through Logic, State Machines, Heuristic Systems and Decision Trees and the various benefits and complexities of each.

It's noteworthy that William finished a simulator using a state machine where the player fought against other AIs in a "grow yourself or attack others" scenario. His quick grasp of the concepts and carefree attitude to flashy graphics meant he focused all his energy on the problem space and in the final testing of his game, he had quite believable characters to play against. All too suddenly the evening set in, and with other commitments to attend to the members quickly and efficiently broke camp and headed home, promising each other to meet again soon and to upload their results of the day's endeavours when the projects reached a presentable state.

I hope we have another devLAN soon as a lot of work gets done on various projects at these events and it's the perfect time to ask questions of other developers, show off what you're working on, and learn a little bit more about the vast field of game development.

FENGOL

To keep up to date about happenings concerning future devLANs, be sure to regularly check up on the Game.Dev website (www.gamedotdev.co.za) and the Game.Dev forums.

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

DEV.MAG

CREATE ● DEVELOP ● EXPERIENCE

www.devmag.org.za