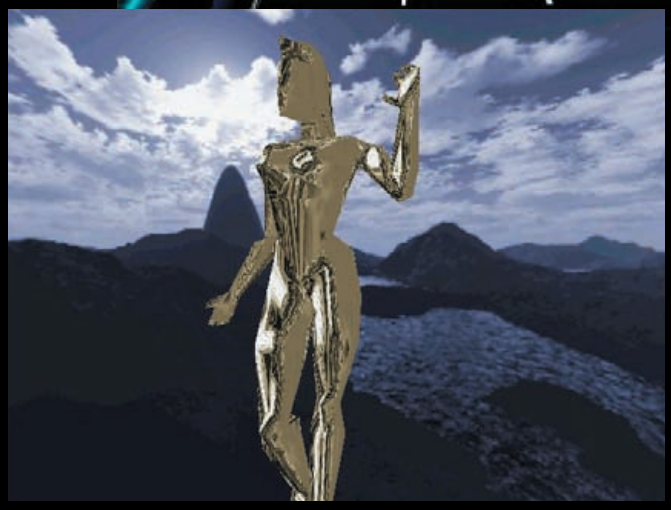
SOUTH AFRICA'S PREMIER GAME DEVELOPMENT MAGAZINE

February 2008

DEV.MAG

CREATE • DEVELOP • EXPERIENCE



THE "I" EDITION

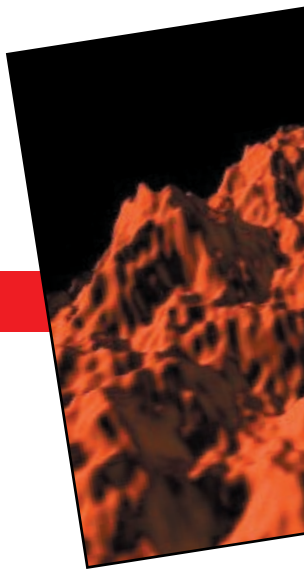
IRRLICHT, IGF AND INNOVATION

A MASSIVE **50** PAGES THIS ISSUE!

REGULARS

FEATURE

REVIEWS



TUTORIALS

DESIGN

TAILPIECE

DEAR READER ...

Hello and welcome to the new year! Sure, the greeting has probably gone a bit stale by now, but this is our first opportunity to hail you after the annual Dev.Mag siesta, so why not?

A great many things have happened over the past two months and we're practically beaming with joy at this release. As with last year, the Dev.Mag team really outdid themselves in an effort to make this first edition of 2008 into something truly special. No, really, did you see the big flashy proclamation on the bottom corner of this month's cover? This issue has broken the 50-page barrier, officially making it the biggest Dev.Mag to date. And that equals a lot of juicy content for our beloved readers.

The game development world is going crazy right now, and we're doing our best to keep up with all of the most awesome stuff. For every article that's made it into this issue, there's another one lying around that had to be either cut or delayed so that we could actually release this edition in (relatively) good time. Thanks in particular to all those readers who have submitted their own articles for the open Opinions section over the past month — unfortunately, we haven't been able to accept any at this time, but we encourage the community to get involved and we do hope to publish some work in the near future.

Oh, but one charming little paragraph left and I *still* haven't told you about what's in this issue. To be frank, though, that would be really difficult to do even if I had a full two pages to work with. My advice at this stage is to just flip through and see what you like — with this much content at your disposal, you're bound to stumble across something interesting.

Best of wishes, dear reader, for the year ahead. Keep on devving!

RODAIN "NANDREW" JOUBERT
EDITOR

Quite a few of the Dev.Mag staff have been involved in the open beta testing of Audiosurf, one of the contenders for the grand prize at this year's IGF. Simply put, we want the retail version. Now. It's an awesome game, and music just isn't the same without it. Find this gem at http://www.audio-surf.com/

DEV.MAG
CREATE • DEVELOP • EXPERIENCE

EDITOR
Rodain "Nandrew" Joubert

DEPUTY EDITOR
Claudio "Chippit" de Sa

SUB EDITOR
Tarryn "Azimuth" van der Byl

DESIGNER
Brandon "Cyberninja" Rajkumar

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
James "NightTimeHornets" Etherington-Smith
Gareth "Gazza_N" Wilcock
Sven "FuzzYspo0N" Bergstrom
Kyle "SkinkLizzard" van Duffelen

WEBSITE ADMIN
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**

Fun fact: this issue is HUGE. In fact, if this magazine was actually printed out, you probably wouldn't be able to carry it JUST because it's so big. How awesome is that?

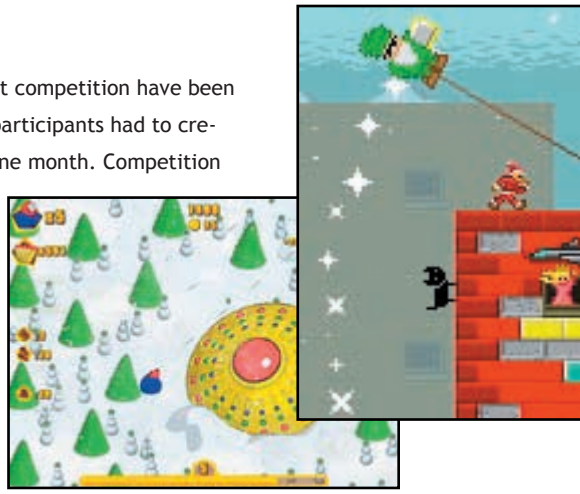
ADVENTURE GAME STUDIO 3.0 RELEASED

http://www.adventuregamestudio.co.uk/

If you're a fan of this popular (and free) adventure game creation tool, then you're in luck. AGS just recently upgraded to version 3.0, promising many new and useful features including better hardware acceleration support, a revamped interface and the removal of many previous hard-limits on development aspects such as views and GUIs. Adventure Game Studio has long been a favourite of many game developers, and classics such as A Tale of Two Kingdoms, the Chzo Mythos and the more recent Art of Theft were all created with this tool.

YOYO WINTER COMPETITION WINNERS

http://glog.yoyogames.com/?p=51



The eagerly anticipated results of Yoyo's first competition have been released! Using the Game Maker software, participants had to create a winter-themed game in the space of one month. Competition was pretty stiff and a lot of high-quality entries were submitted, but eventually the top three games were selected to receive cash prizes (starting with USD $1000 for first place), accompanied by a list of honourable mentions. More details are on the blog.

TOP 20 INDIE ARCADE GAMES

http://www.indiegames.com/blog/articles/index.php?c=ac&y=2007&gid=0

For a list of some awesome games from 2007, you can do far worse than taking a look over here. With titles such as Trilby: The Art of Theft and Frozzd populating the list, it looks like the new year has quite a bit to live up to in terms of indie gaming. All of the games on this list are freeware, and a few of them will be appearing in Dev. Mag's pages within the next couple of issues.

COMP 17: "END!"

http://forums.tidemedia.co.za/nag/showthread.php?t=3073

Game.Dev has just launched its first competition for the new year, centred around playability and brevity. The aim is for players to construct a game that goes on for no longer than 10 minutes. Whatever genre, whatever goal you may have, the only rule is that the game has to end before that ten-minute mark. If you're in South Africa and want to try your hand at a quick and fun game development competition, hop onto the forum and give it a whirl. The competition is open until the end of February.

GAMASUTRA'S "20 MYSTERIOUS GAMES"

http://www.gamasutra.com/view/feature/3485/game_design_essentials_20_.php

An interesting snippet from one of our favourite news sites, this feature has a look at all manner of games which have some sort of "mysterious" or hidden element in them. Everything from secret blocks in Super Mario to complex behind-the-scenes algorithms which determine the very fate of your in-game character all count as being mysterious in some ways, and this article deals with everything from the curious to the bizarre. An interesting read which allows readers to think about game design in a new way.

GISH 2 DEVELOPER'S BLOG

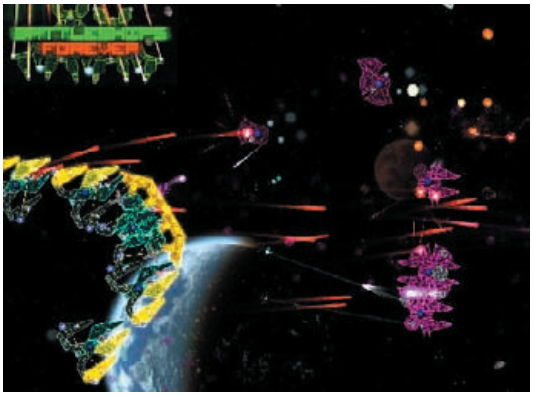
http://crypticsea.blogspot.com/

The heading says it all, really. Creators of the highly popular ball-o'-tar platformer Gish are furiously working away at a sequel. If you're keen on checking out some interesting news posts, or just want to see what these guys are up to, then take a look at their blog for some more info. If you don't have the original Gish yet, word is that the game is available on Steam for just less than USD $5 ...

AUDIOSURF COMING SOON

http://www.audio-surf.com/

After two beta sessions, Audiosurf has accumulated a ball of hype and excitement, with eager players lining up to get their hands on a full release. Audiosurf's blog regularly makes note of any progress or major changes made, and the beta forum is already open to those who want to chat about the game or check out a nice bunch of fan art. Audiosurf is due for release in February and is currently a contender for the 2007 IGF Grand Prize.



BATTLESHIPS FOREVER ON GAMASUTRA

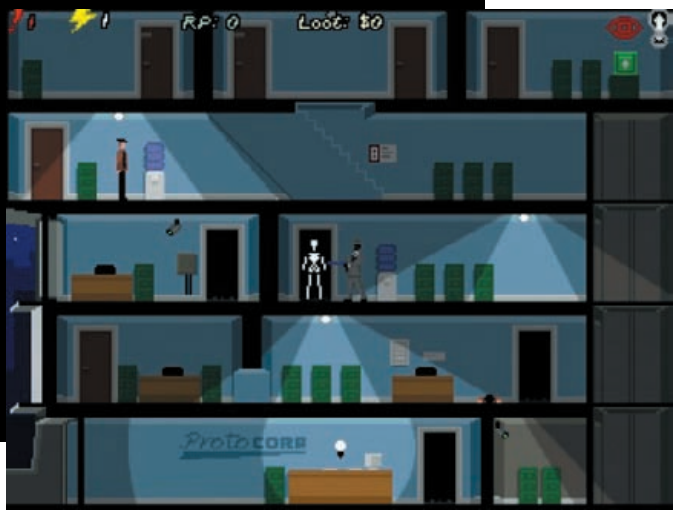
http://www.gamasutra.com/php-bin/news_index.php?story=17202

One of the Game Maker entrants in this year's IGF finals, Battleships Forever has stormed in with its innovative gameplay and in-depth look at strategy. This Gamasutra interview takes a look at creator Sean Chan's history, his motivation for the project and what he feels is most important to consider when creating a game like Battleships Forever. A worthwhile read, and if you want to try the game for yourself be sure to look at this month's Tailpiece for a link and brief description.

TRILBY: THE ART OF THEFT

http://www.escapistmagazine.com/content/games/yahtzee/artoftheft

Ben "Yahtzee" Croshaw seems unstoppable as of late. His latest offering is a cunning little thievery game starring one of Yahtzee's recurring game characters, the objective being to loot and sneak your way through Chapow City. An impressive title which becomes even more impressive when one considers that it was made using the Adventure Game Studio - a prime display of the tool's flexibility. Expect a full review next issue.

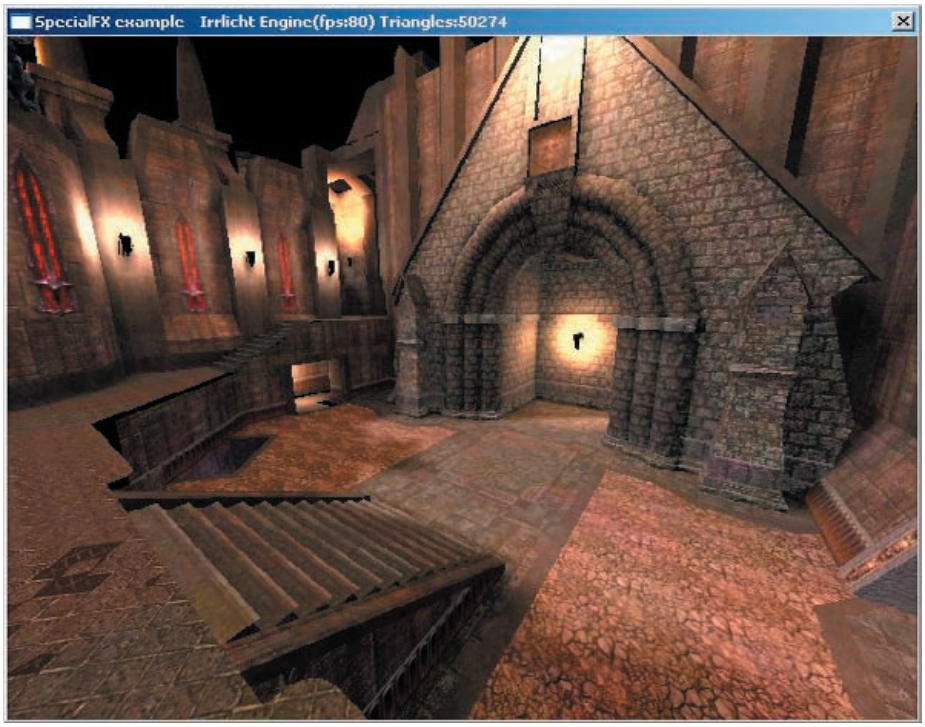


MR GEBHARDT, I PRESUME?

DEV.MAG TALKS TO THE MAIN MAN OF IRRLICHT

by Sven "FuzzYspoON" Bergstrom

Interestingly enough, there are people out there that give stuff away for free. They say nothing is free, but in the cutting-edge realm of software development there are always a handful of developers that break the mold. Nikolaus Gebhardt, creator of Irrlicht3D, is one of those developers, and his projects are most certainly groundbreaking. We decide to throw a few questions his way.

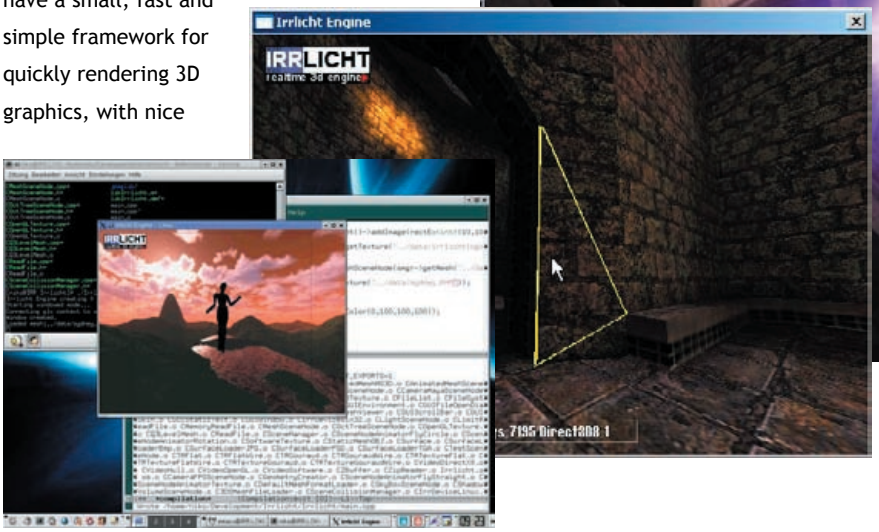


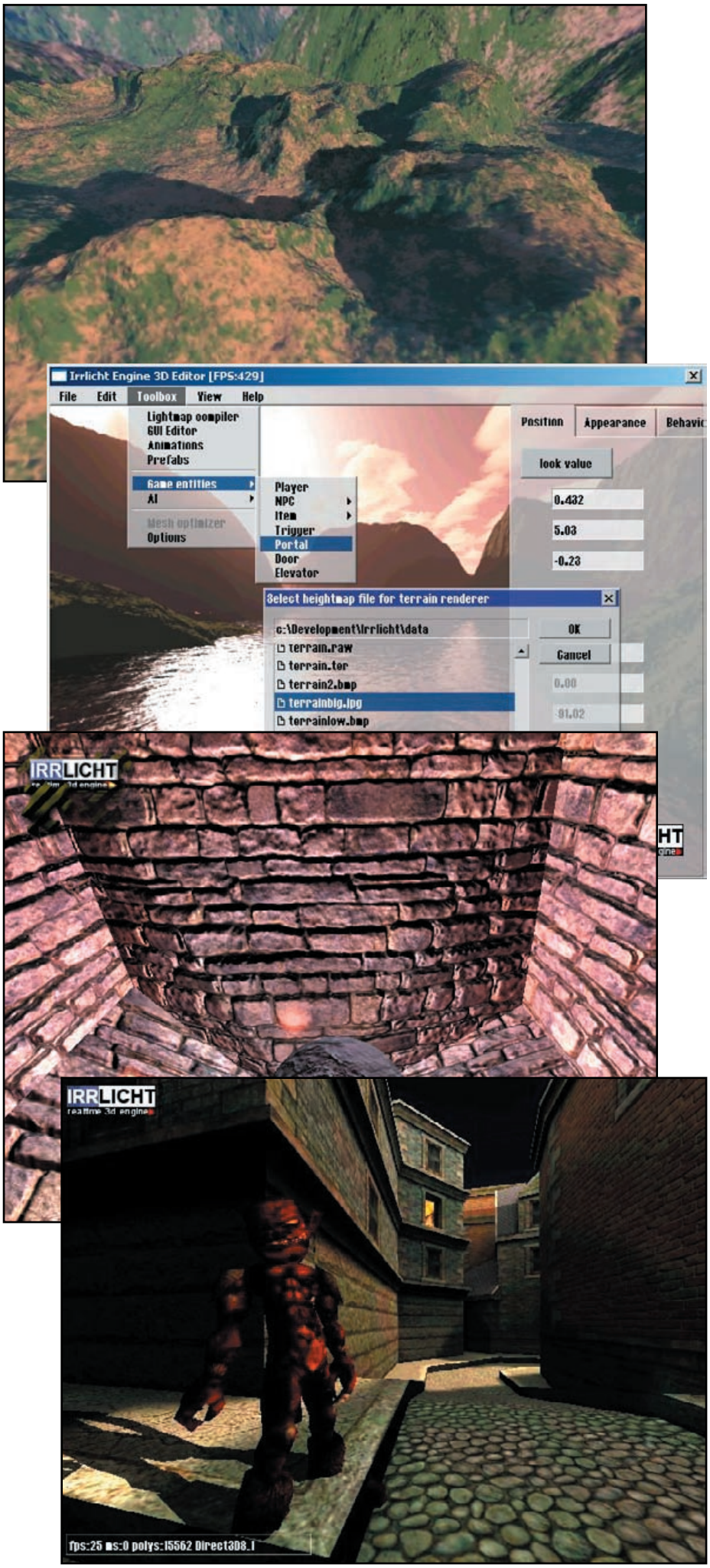
Mind telling us a bit about yourself?

My name is Nikolaus Gebhardt and I am living in Vienna, Austria. I'm a software developer and with my company 'Ambiera', I'm supporting other companies to develop computer games. Most people know me from my work on a software library named 'Irrlicht', now a huge project which I started some years ago.

As the creator of the Irrlicht engine, can you tell us how it came about?

In 2002, when I was employed at a game development company, I used to work with various commercial 3D engines and was dissatisfied with nearly all of them. I wanted to have a small, fast and simple framework for quickly rendering 3D graphics, with nice





documentation and which should also work on more platforms than just Windows. So I decided to sit down and write such a library myself. After some months, I made it open source, named it 'Irrlicht' and published it on Sourceforge. And because people apparently liked the library, it became a success.

Is Irrlicht becoming/being what you hoped for?

No, Irrlicht hasn't become what I hoped for at all: I never would have thought that Irrlicht would become as popular and liked as it is now. We now have a huge user base with more than 8000 registered members in the English forum alone and about 400 downloads of the Irrlicht Engine SDK per day. I'm getting flooded with feature requests and other mails regarding Irrlicht, and I have to spend at least one hour per day just to answer or at least read those mails every day and do other Irrlicht-related administrative tasks, so that I don't have that much time for development anymore. Fortunately, we are now a team working on Irrlicht, and Christian, Gaz, Luke, Tom and Dean are doing a great job at improving Irrlicht, and Alvaro, Jam and others are taking care of the Forums, giving me a bit more time.

What are your future ideas for Irrlicht?

There are a lot of new features in discussion, like adding the newest and most popular 3D rendering techniques and bringing Irrlicht to other platforms (PocketPC for example). Unfortunately, Irrlicht is a hobby project and done in the free time of all team members, and that's why we usually don't speak that much about features to come and release dates. It is not always possible for us to really uphold these planned features, and in order not to disappoint users, we prefer not to speak that much about things like these. But one thing is sure: We will continue to develop and improve Irrlicht where we can, and it will stay open source and free as long as we live

You also created Ambiera, the company that makes Irrlicht-related/independent tools for free. How did this start?

I founded Ambiera in an attempt to support Irrlicht a bit more. Irrlicht is completely free and I don't get any money from it, although a lot of companies use it in their commercial products. Ambiera provides some additional libraries and tools which can be bought for commercial use (they are still free for non-profit or free games and applications) and more importantly, it gives me a bit more free and flexible time for Irrlicht development. As a full-time employee I wasn't that flexible.

What drives you to make all your awesome tools for free?

That's a good question. I think I could have made quite some money if I had developed Irrlicht as commercial library from the beginning. But on the other hand, Irrlicht would not be that popular if it wasn't open source. The decision to release Irrlicht as an open library in 2002 was basically made because I wanted to give something back: I'm using so many open source tools, libraries, and even open source operating systems. The other tools like the audio library irrKlang and the editor irrEdit are free for non-commercial use so that Irrlicht users - which are about 90% hobbyists - may use them without problems.

If you had advice for indie developers, what would it be?

I'd say "give something back": There are so many indie developers who are using free and open source tools to make profit, but are only rarely contributing their changes and additions back. But in most cases they would even benefit from this. Fortunately, there are some exceptions.

DB SAYS ...

Hey, interested in giving this tool a shot yourself? Check out our Irrlicht starter series, beginning this issue. Also be sure to visit Irrlicht's home page at http://irrlicht.sourceforge.net/

QUICK QUOTES

Console or PC?

Never been a console user, maybe because I'm a programmer: I'm used to the fact that sometimes things don't work and that I have to find a solution myself.

Linux, Windows or Mac?

If I had to choose between Linux, Windows and Mac, I still would definitely choose Windows. I am using all three, but my main development platform is still Windows. Not only because most of the users are using Windows as well, but because it is more comfortable, especially for programmers. Although claiming to be a developer OS, Linux still hasn't a decent IDE like Visual Studio, and MacOSX is a bit too insecure for me personally.

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

Turbo Delphi

Delphi for Win32

Lazarus

Aquaria

http://www.bit-blot.com/aquaria/

by Claudio "Chippit" de Sa

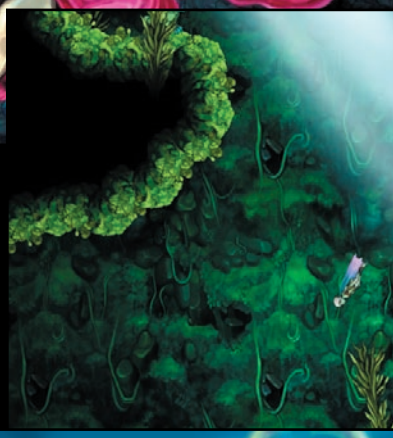
A stunning setting, an involved story, and a gargantuan world to explore. This is the promise of Aquaria, an indie title nearly 2 years in the making and winner of last year's IGF grand prize. Diving into this action-adventure styled game is definitely an exciting prospect.

The idyllic land of Aquaria is a vast undersea world consisting of deep abysses, massive open caverns and narrow passageways, inhabited by hundreds of aquatic beings, some peaceful, some hostile, all very unique. You'll find yourself drifting through the waters, admiring how light filters through cave ceilings, how the entangled reefs sway as you swim through them, how certain creatures are drawn to Naija as she sings, and how right everything feels.

Naija, the mermaid the player controls, begins the game with no memory of her past and little ambition for her future. The appearance of a mysterious figure prompts sudden curiosity in her existence and she sets out in a massive quest to discover who she is, whether or not she has any kin, and why the world doesn't feel quite so right immediately outside her home waters.

Possible routes of exploration branch out exponentially once Naija penetrates the open waters outside her more familiar personal boundary which also, conveniently, serves as the boundary of the rather extensive trial version. In fact, the sheer size of the world is rather overwhelming and this fact is only compounded by the general lack of direction supplied by the game. You may find that, because of an unfortunate choice of paths, you'll reach many avenues where you are unable to proceed due to lack of correct abilities. This results in occasional frustration with the lack of apparent progress. However, persistent exploration will usually yield a path where secrets can be discovered or advancement can be made, and the feeling of frustration is usually replaced with a new sense of awe at the discovery of a new locale.

Among the areas Naija will visit, impressive both visually and in scale, are ruined cities of mysteriously absent civilizations, massive reefs, huge open waters, and a deep, gloomy abyss as dark as a void. Most of these regions will yield new powers to Naija as she explores them, progressively unlocking more areas in the process as well as enticing her to venture further and further in pursuit of the secrets of the underwater world. These ventures often culminate in a major boss conflict, usually for the most important song of the region. As she explores and discovers new powers, Naija experiences

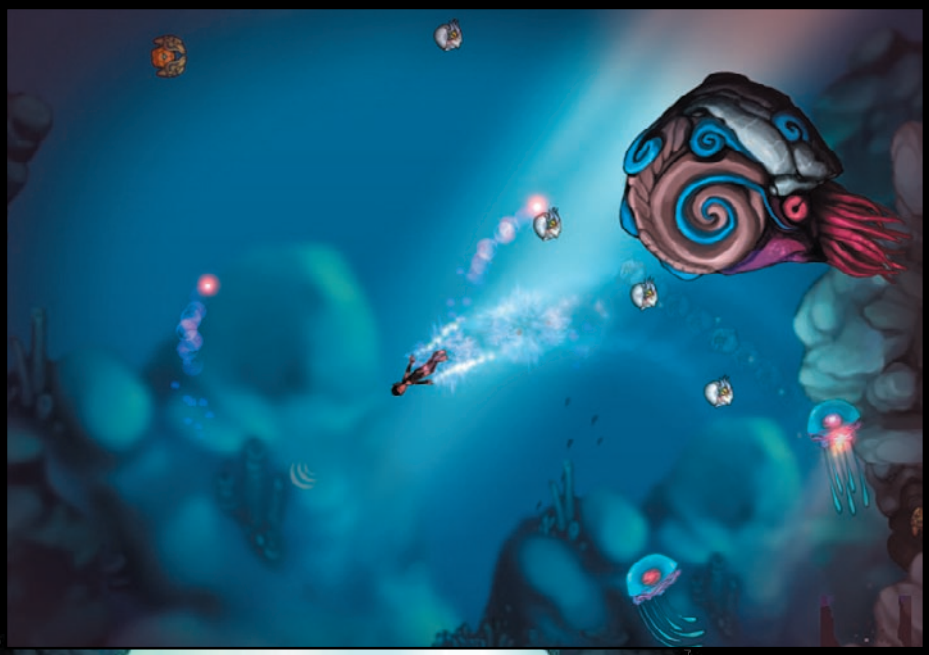


flashbacks of her own history or the history of the area and its peoples. These serve to advance the story, as well as provide a reward for the player's efforts.

Naija's greatest and most unique asset, her singing voice, grants her the ability to manipulate the magical energy of Aquaria, which she refers to as the Verse, allowing her to move massive objects, create an energy shield impervious to projectiles, or even completely change her physical form. To perform a magical feat, Naija needs to sing a specific string of at least 3 notes from a circular 8-note scale. On their own, single notes also have other uses including cracking open certain plants, attracting sea life of the same 'colour' as the note, or for solving other miscellaneous puzzles. Alternate forms grant her special unique traits, including offensive attacks, the ability to traverse very narrow passageways, thicker skin and other, more exotic, powers. These new transformations represent the progress in the game, with new abilities granting Naija access to previously untraversable environments, once again exponentially increasing potential avenues of exploration.

Another unique ability in Naija's inventory is her ability to create special treats from various ingredients scattered throughout the world. While the system is open to experimentation and clever trials may afford new recipes, the major source of new formulae is simply the act of finding the item in question. As soon as Naija discovers a new food type she will intuit the ingredients required to make it, whether they are other complete foods, raw ingredients, or a combination of the two. More powerful items are usually created by combining three ingredients at a time. Such foods cannot be created out in the open ocean like simpler concoctions and must be prepared in special kitchens. This item creation system adds an extra layer of motivation for world exploration, since many rarer ingredients and recipes are hidden in secret or hard-to-reach areas.

All in all, Aquaria is a highly polished representation of what the indie scene is capable of, and, despite a few minor design flaws, particularly with the massive scale of the world, it is an incredible experience. Whether or not it warrants its near triple-A price tag is likely a debate that will rage on for months to come, but it certainly is one of the most impressive games ever to be created by a team as small as Bit-Blot's.



H-CRAFT CHAMPIONSHIPS

http://www.irrgheist.com/

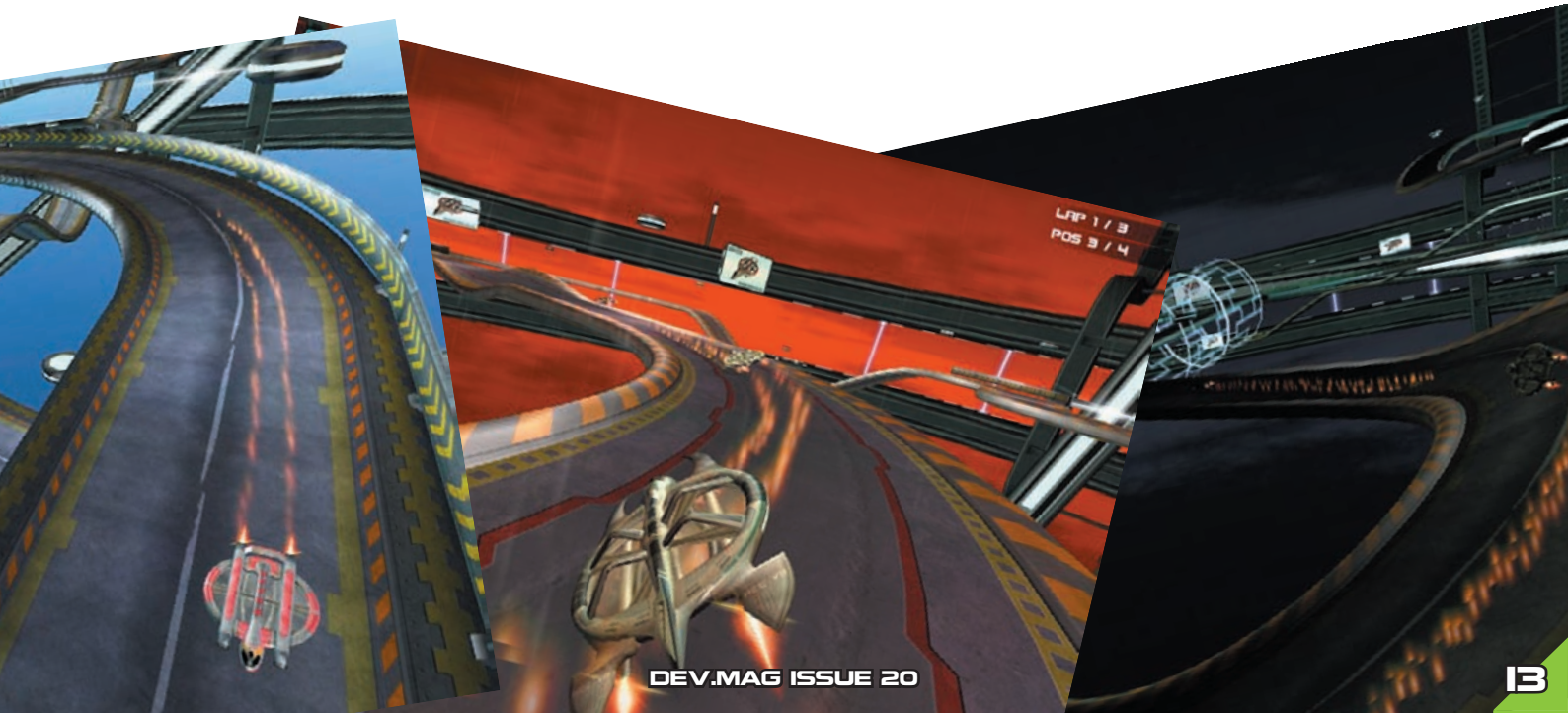
by Sven "FuzzYspoON" Bergstrom

Obviously a fan of indie games can never miss out on seeing another racing game, but a futuristic space sci-fi racer is a unique title to see coming from the two-man development team in Germany called Irrgheist.

This game is quite a challenge considering the unique, self-made physics system which breaks all egos on first impression. With the large number of maps that come with the full game, it seems like an endless supply of fun mixed with some interesting AI. The AI gets a head start for 3 seconds, which forces you to increase skill rather than finding shortcuts to beating the game. The game's AI has been commented on as being too difficult, but it is far from impossible to finish H-Craft. With the ability to record all your races and stack up against your friends, there's also a great opportunity present for you to challenge each other to a time attack. The developers hold challenges with times based on what they achieved while developing the game, which gives you even more to aspire to.

The demo leaves the feeling that the game may get monotonous after a while, but the full version adds a large amount of intriguing maps and interesting challenges. Utilising the Irrlicht engine for rendering graphics, the team introduces some interesting techniques for effects. Irrgheist use the cross-platform OpenAL for sound, XML-based settings and files, and utilise a mixture of common formats to create a map format that is simple but complex at the same time. They have built in-house tools to integrate the map meshes into their engine, as well as some great tools for making AI which helped their development process a huge amount.

Indie developers creating great games are becoming more and more common nowadays. It's advisable to keep an eye on this company, as they have great plans for more games of even higher quality.



TIGSource

http://www.tigsource.com/

by Simon "Tr00jg" de la Rouviere

Where would you go to get your dose of indie-gaming news? Where would you go to have a discussion with fellow indie developers? Where would you go to learn about the creation of monocles? Where would you go to find out who the sexiest indie developer out there is? You can find all this, tigersauce and more at www.tigsource.com.

Let's start with what TIGSource is. It stands for "The Independent Gaming Source" and is a website/blog dedicated to bringing you info on the independent game development scene. It started humbly in early 2005 and has since grown a lot, eventually passing hands to Derek Yu, the artist behind the incredible indie game Aquaria.

So what sets TIGSource apart from other blogs? The posts are, first of all, well-written and make for plenty of chuckles. Instead of just telling us the news, it adds its own flair.

Another thing that TIGSource has going for it is a great community which includes a host of well-known indie developers. TIGSource's forums are jam-packed with great threads about indie games, indie life and random awesomeness. The Feedback section is a really great place to get decent feedback from other developers and the master of critique himself, Guert. TIGSource's community is always willing to help wherever they can.

Amongst the above, TIGSource regularly hold interesting competitions like "Sexiest Indie Developer" and "A B-Game competition" (the latter being one where you purposefully have to make a shoddy game) and recently an interactive fiction competition.

I highly recommend for any developer to visit TIGSource to either have their dose of indie gaming news or just be a part of a great community. After visiting TIGSource, it leaves one with a feeling that indie games are the best thing in the universe and that it truly is something to be a member of the indie scene.

5 DAYS A STRANGER

http://www.fullyramblomatic.com/

by Ricky "Insomniac" Abell

5 Days a Stranger is a freeware point-and-click adventure game developed by Ben "Yahtzee" Croshaw. "Yahtzee" has a become a bit of an internet star lately with his weekly video reviews called Zero Punctuation, where he gives a funny and sarcastic view on the game he reviews that week. It was a great surprise for me to find out that he also made games, and thus I had to give 5 Days a Stranger a whirl.

Going into the game, it's easy to expect something with loads of sophisticated humour (given the reputation of the developer), but players will instead find a very well-designed and engrossing horror title. The game is played as Trilby, a mysterious man who describes himself as a gentleman thief. Things start off like a regular gig for Trilby - the abandoned DeFoe Manor is filled with many valuables ready for easy picking by the talented burglar. Unfortunately things don't go too smoothly and just after breaking in Trilby finds himself trapped in the manor. However, he soon discovers that he isn't alone, as there are a few other people who also seem to be a prisoner within this mysterious place. Now it is up to the player to explore the mansion and find a way out within 5 days.

As mentioned before, this is a horror game, but not in the usual ooh-look-everyone-has-turned-into-a-zombie-who-wants-to-sample-my-brain sense. Instead, what makes this such a captivating horror is the atmosphere. It doesn't have shiny graphics at its disposal to scare you but it has many other things. The sound effects - such as footsteps in the background when you're in a quiet, empty room - can easily send chills down your spine. The story and gameplay also pulls you in and has you hesitating before you click a door in fear of what will be behind it. The further you play, the more interesting things get, and you can't stop playing until the mystery is solved.

The gameplay is your fairly standard point-and-click fare where you talk to people, collect items along the way and also combine them to solve problems. Items you pick up, such as a diary, can sometimes progress the story and give a deeper insight into the characters - a nice touch. The only thing that seemed to mar the whole experience was a point in the game where reacting incorrectly in a tense and fast-paced situation would kill the player and force them to go back to a manual save point - a pain if you don't happen to save your game often.

Overall, 5 Days a Stranger doesn't do anything new or break the mould but it tells a great, fascinating story with style and for an adventure game you can't ask for much more. If you're new to adventure games, go ahead and give this a try, though you may need a walkthrough on standby rather than risk giving up at some point and missing out on an intriguing story.

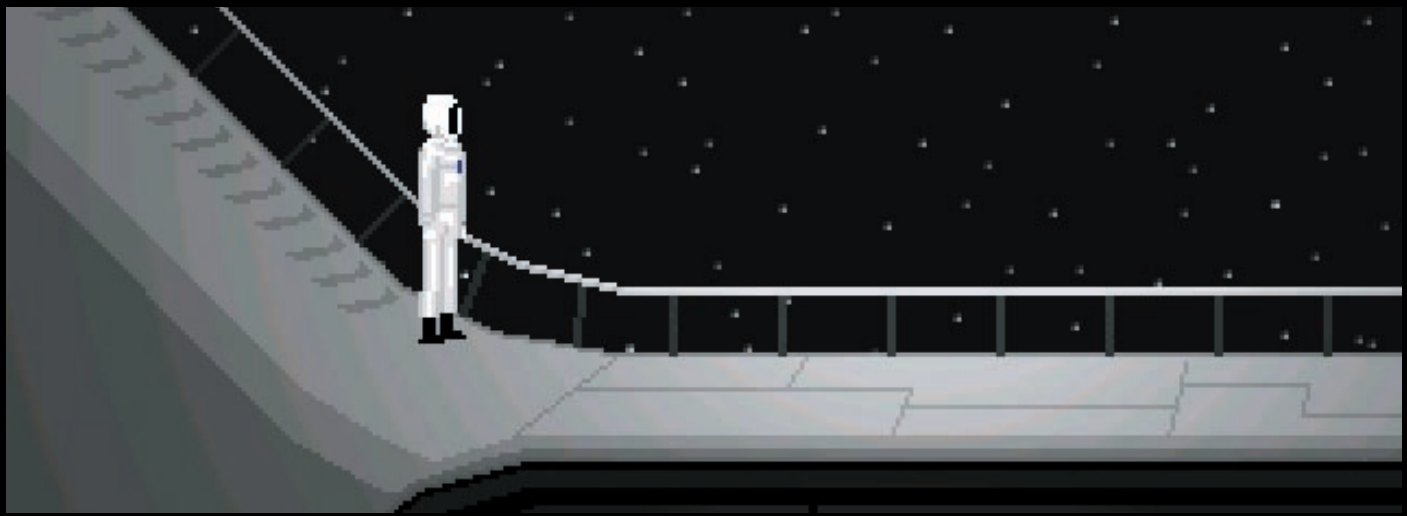
http://www.fullyramblomatic.com/

by Gareth "Gazza_N" Wilcock

It's a quiet day in the Caracus Galaxy. The starship Mephistopheles sullies forth across the sea of stars, its six-person skeleton crew boldly scouting what no six-person skeleton crew has scouted before. Of course (being a horror game and all), trouble is inevitable for the intrepid crew, and it comes their way in the form of a nondescript metal box floating innocently through the void...

Built using Chris Jones' free Adventure Game Studio software, 7 Days a Skeptic is the second game to be released in Ben "Yahtzee" Croshaw's Chzo Mythos series, and deals with the (literally) far-reaching consequences of the events covered in its predecessor. Set roughly 400 years after 5 Days a Stranger, the player is placed in the role of Doctor Jonathan Somerset, the Mephistopheles' resident psychologist, who along with the rest of the crew is subjected to a series of inexplicable, and increasingly violent, paranormal events. It isn't high literature, but it isn't meant to be. 7 Days, like its predecessor, attempts to emulate the tension of ye olde slasher movies of yore, where a set of characters is confined within an isolated environment while an unpredictable and unknown killer stalks the corridors. Fortunately, Yahtzee hits the nail on the head in that regard, producing a surprisingly unnerving atmosphere by using both the setting as well as some very unsettling dream sequences/visions. Fortunately the writing and plot progression are solid given the premise, and the story includes some twists and turns that keep you interested throughout.

Gameplay-wise, the game is structured very much like 5 Days a Stranger in that the plot takes place over the titular seven days, with each "day" consisting of a series of puzzles that needs to be solved in order to progress to the next day. The interface is an expanded version of the classic verb coin system. Right-clicking on objects brings up a set of verbs and inventory items, which, when clicked, perform the relevant action or use the relevant item. This system does tend to give interaction a very staccato feel, since



the menu needs to be invoked constantly for every item the player wishes to interact with rather than allowing a default action to be performed. Nonetheless, the interface is clear and functional and allows the player to see at a glance all the interaction options that are available, rather than forcing them to dredge through inventory screens or cycle through endless verb cursors. The graphics are rudimentary, but sufficiently detailed to serve their purpose, and you'll quickly find yourself ignoring the MS Paint-y look of the artwork as you play.

Naturally, the game isn't without its niggles. While the puzzles are for the most part logical and solvable with a bit of brainwork, the game has one or two instances where the solutions are outright cryptic. Worse still, these tend to take place during high-pressure chase scenes which, while succeeding in heightening the tension, start becoming rather tiresome after your n'th death. Yahtzee is also rather fond of hide-and-seek style puzzles where one is forced to comb every nook and cranny of the ship in order to find a relevant NPC. Admittedly, it creates a good incentive to discover the layout of the ship towards the beginning of the game, and makes the characters feel like they actually live on the ship rather than being superglued to their posts, but after a while the frequency with which you're forced to go through this exercise starts to feel more like padding than proper game progression.

Don't let the above deter you, though. Overall, 7 Days a Skeptic is a solid point and click game that is well worth your attention, especially considering its price (free, or $5 US for an expanded special edition). It's also an oft-cited (and dual-AGS-award-winning) example of what can be accomplished with Adventure Game Studio given sufficient time and effort, making it a worthy reference if you plan on using the package for future projects. If atmospheric point-and-click horror is your thing, you could do a lot worse.

BLENDER TUTORIAL

Mapping pictures onto a 3D cube

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.

This tutorial will show you how to use Blender 3D to map pictures onto a 3D object. The tutorial uses a simple cube. By sticking pictures on the cube we will make it resemble a skyscraper. This skyscraper can then be 'photographed' in 3D from any angle to be used in a game scene.

I am using a picture from www.yotophoto.com as a texture. The license for this picture states that it may be used free of charge for private and commercial use. Download the picture to your computer and save it where you know you will find it in a moment. The Blender file and picture map is provided along with this tutorial on the Dev.Mag website.

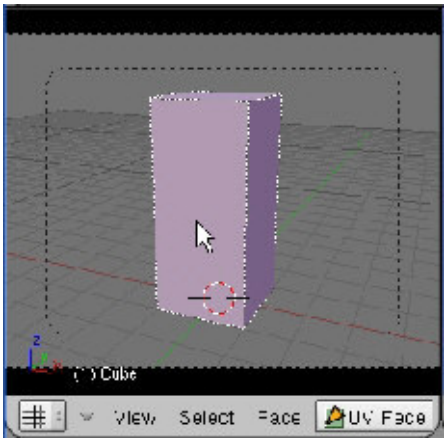
In essence, we will select each side of the cube and place it on the picture of the building. The way that we place it on the image will determine the way that it is displayed on the cube.

Open the blender file that is provided along with this tutorial. On opening the file, you will see a screen with two viewports, with a button window below. On the left is the view from the camera. The window on the right is set to a UV/ Image Editor view.

We'll start right away by selecting the building. Use RMB (right mouse button) to click on the building in the camera view. When selected, the building will turn pink (as shown in the image below).

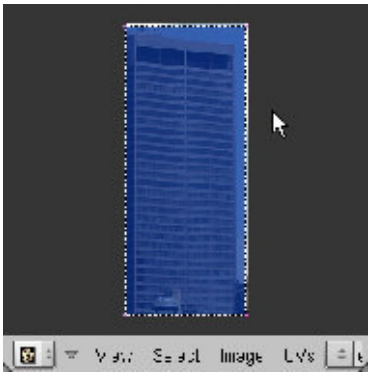
Press the FKEY. The view of the model changes slightly. This is the UV Face Select view which we will use to select the different sides of the model.

Press AKEY (select/ unselect all) until you have all the faces (sides of the model) selected.

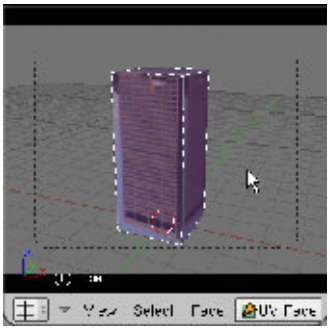


Now shift you attention to the window on the right. Clicking on the Image button on the menu bar allows you to open the image that we want to use as a picture map. This will be the picture of the building that I have provided.

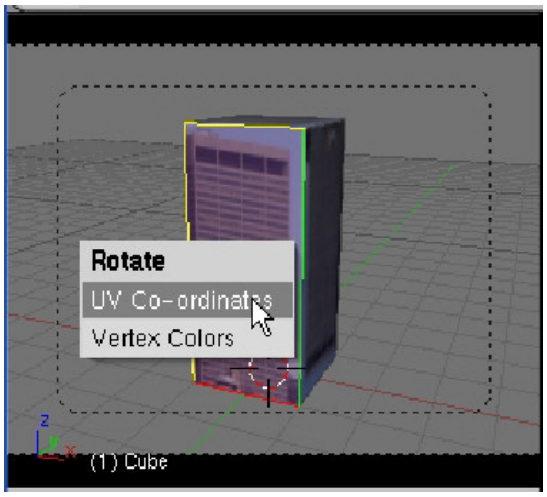
This is what the UV/ Image Editor view should look like when the picture has been successfully loaded.



After you have loaded the image of the building in the UV/ Image Editor window, move your cursor to the left viewport, and press ALTKEY + ZKEY until you see the image of the building displayed on the model.

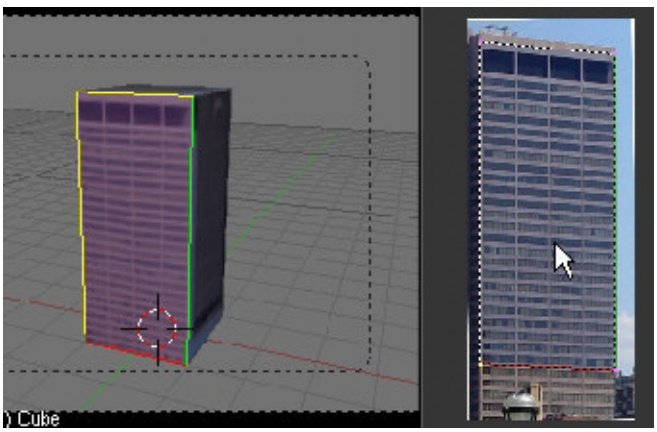


As you can see, the picture of the building is upside down, and we can see blue sky along the edges. We will sort that out with the next step. With your mouse cursor over the camera view on the left, RMB select the front face of the model. Press RKEY and use the popup menu to rotate the UV-coordinates until the picture of the building is the right side up.



Now we are going to use the 4 points that define the "face" of that one side of the model, to make the correct part of the building fit onto the model. We do that by moving the points (vertices) inside the UV/ Image Editor window.

With the single side of the building selected in the camera view, shift your attention to the right hand view. RMB select a single point (vertex). Then press GKEY (grab) and move that vertex in the window. LMB (left mouse button) click to confirm where you want it. You will see that the picture on the model is stretched accordingly.



By positioning the vertices on the picture of the building, we can position (map) the picture of the building onto the model. This will even allow us to map a building that was photographed at an angle, like this one.

In the picture at the bottom of the page, I have positioned the vertices in the UV/ Image Editor Window so that the correct part of the building is mapped onto the model. The next step is to RMB select one of the other faces on the model. Then do exactly the same with the vertices in the UV window.

To see the building from a different view, position the mouse cursors over the camera view, and press:

NUMPAD1 for front view,
NUMPAD3 for side view,
NUMPAD7 for top view,
NUMPAD0 for camera view.

Alternatively, use the MMB (middle mouse button) to pan around the window. Holding down SHIFTKEY while you use the MMB will allow you to pan the whole viewport. If you get lost, refer to the online Blender manual that can be found at www.blender.org.

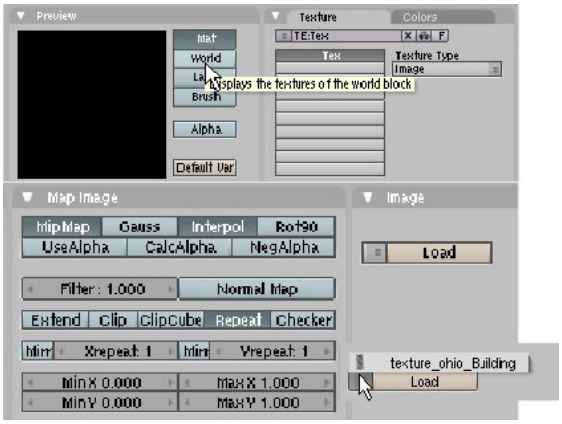
Remember that you only need to map the sides that the camera will see. Don't waste time on doing more than you should to get the right effect. I'm sure that there are more elements of your game/ project waiting.

When you are done mapping all the sides that you want, position the mouse cursor over the left viewport and press FKEY to leave UV Face Select mode.

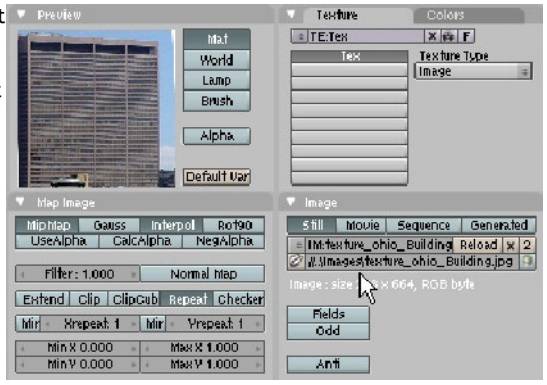
Now we need to add a material to the scene. I have already set up lighting and a camera. Ensure that the bottom

window is on the materials view. Click the Add New button. This adds a new, standard, gray button to the model.

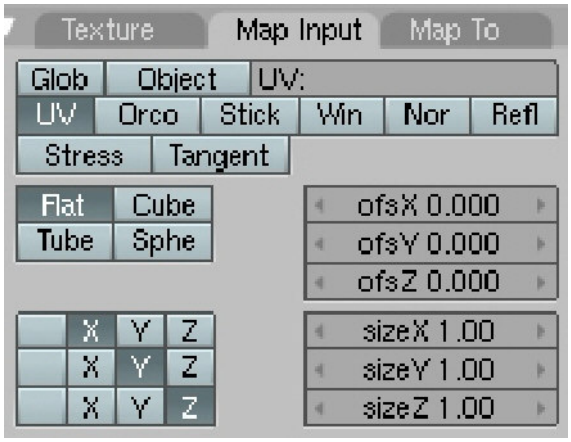
We need to add the picture to this material, and tell it to use the UV coordinates to place the picture on the model in the correct way. Select the leopard skin icon in the menu bar. This adds a texture to the current material. In this way we can add the picture as a map. Click the Add New button. Then click on None and select Image from the menu. Your buttons window should now look like this:



When you click on the double arrow icon next to the word "Load" on the left-hand side, you should see the picture file as an available option. Load the picture file. When you have correctly loaded your image map, the texture button window should look like this:



Go back to the materials window. On the right hand side of the materials window, you will find three tabs: Texture, Map Input and Map To. Select Map Input. Once you have selected Map Input, you should see the following menu:

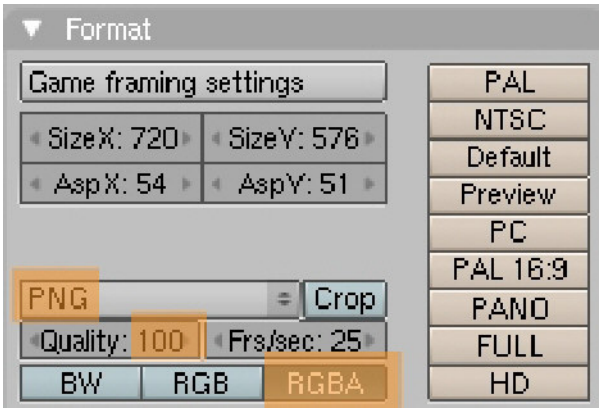


Note: In Blender you can scale menus by using the - and + key by the NUMPAD. If your menu runs beyond the border or your window, simply scale the menu smaller.

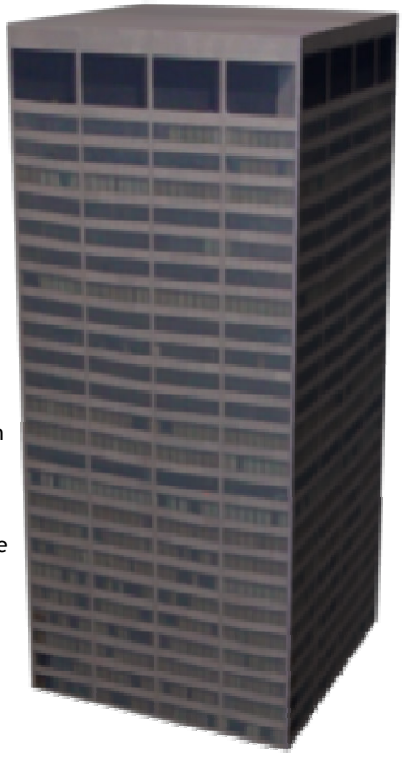
Make sure that you activate the UV option, as shown in the picture. You can now hit F12KEY to make a render of your building.

I would suggest that we bring down the shininess (specularity) of the material. That will probably yield a better render result. Do this by clicking on the Shaders tab and bring the Spec value down to zero.

The last thing we need to do is to set the render format so that you may render an image with a transparent background. This will allow you to compose more than one render together in an external image editor, or layer an image on top of another using the transparency (alpha channel). Go to the render button and ensure that they are set as in the image alongside.

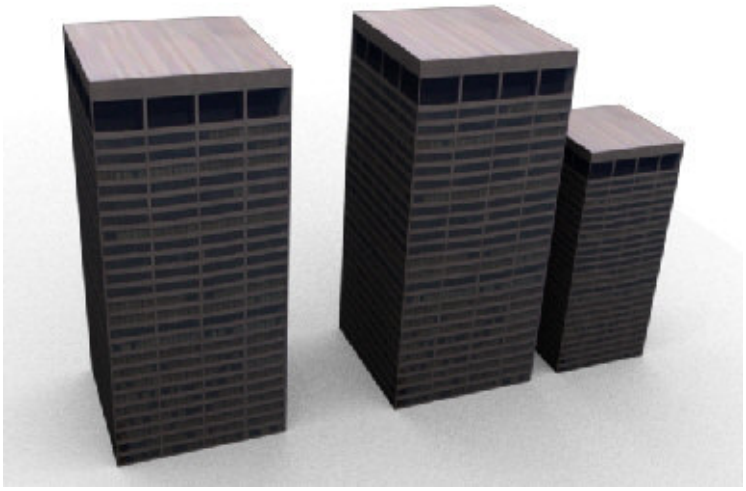


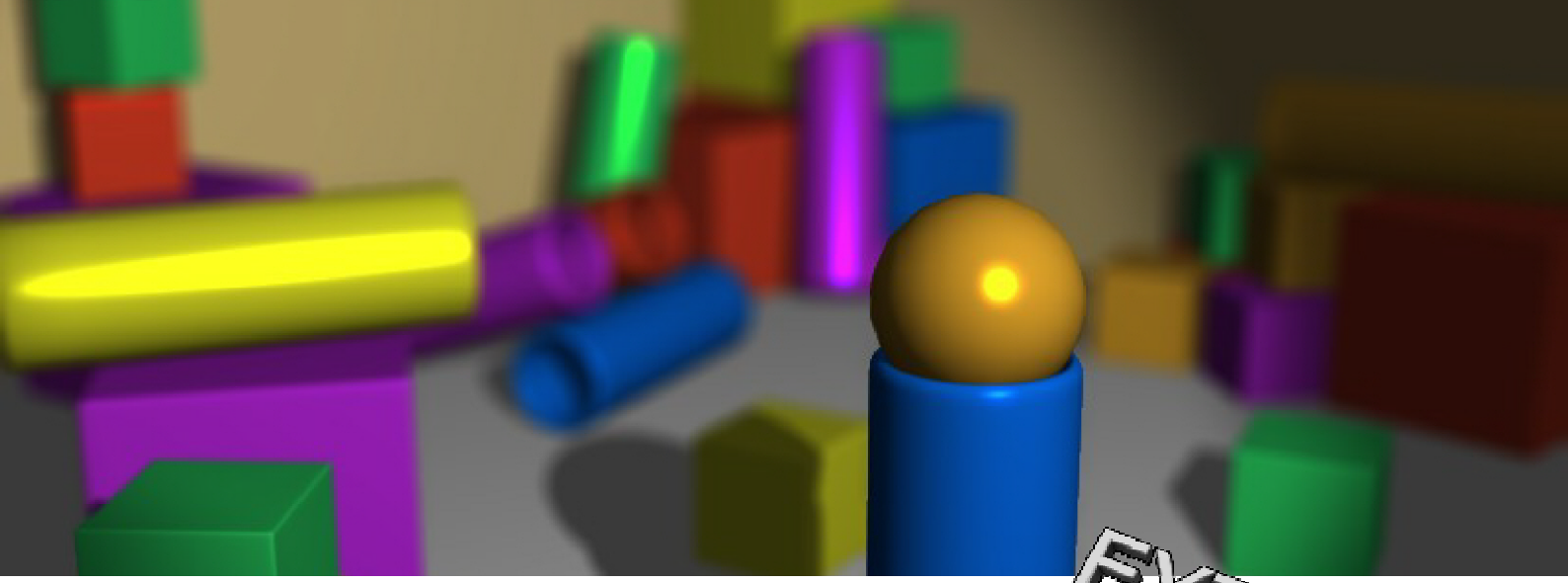
The PNG format can hold transparency values, while the RGBA setting tells blender to render colors (Red, Green, Blue) as well as an Alpha channel. Press F12 to render the building model. When the render is finished, you can press F3KEY to save the image.



Using this process it is quite feasible to build an impressive cityscape with a couple of hours, with buildings placed entirely at your own will.

Best wishes for the use of Blender 3D in your game development. This software is a great tool that yields professional results, provided that you take the time to investigate it.





BLENDER TUTORIAL EXTRA

Compositing

By Claudio "Chippit" de Sa

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.

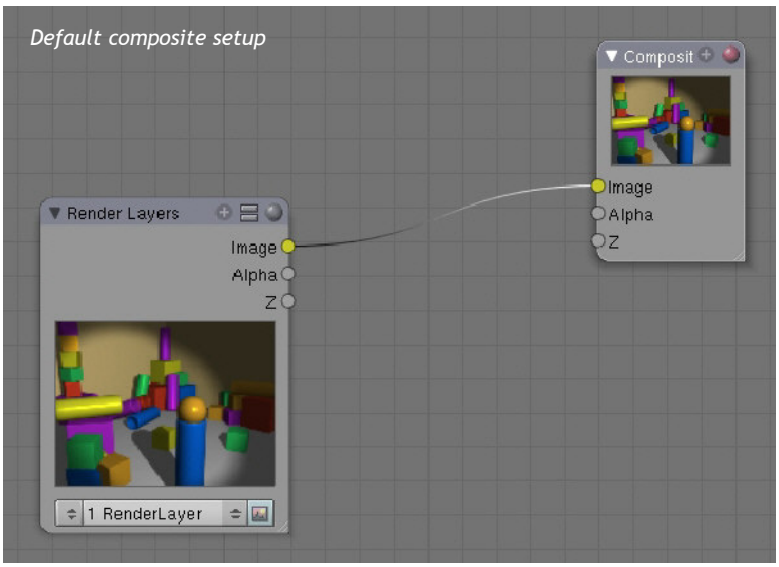
This supplementary Blender tutorial will guide you through a lesser-used, yet flexible and powerful, Blender feature: The post-process compositor. The compositor has the ability to apply special filters and effects onto a completed blender scene. Effects range from simple depth of field and bloom to complex effects only really limited by imagination. While it is possible to directly apply this tutorial into any existing Blender scene, I recommend you download the scene available on the Dev.Mag website to begin with.

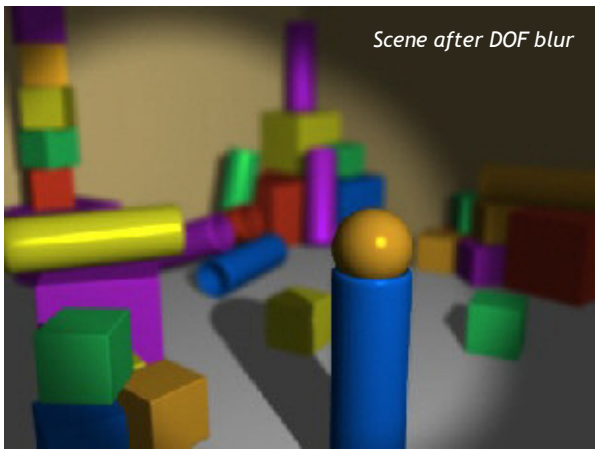
To get started, we need to tell Blender that we're going to be applying post-process effects with the compositor. To do this, switch to Scene buttons and click the 'Do Composite' button on the Anim tab.

To set up composite effects you need to use the node editor. You can split the workspace, but since we won't really need the 3D view you can dedicate your primary window to the node editor. Since you've probably never used the node editor before, it warrants a quick explanation. The node editor allows for advanced editing techniques for materials and compositing, allowing you to chain together complex effects in any order to produce the required end result. Any explanation of its use can be distilled down to adding different objects to the chain, linking their input and output nodes together to produce the required effect.

Like most Blender windows, the controls in the node editor are equivalent to the 3D view, with the exception that left-click selects nodes, and the right-click brings up the add menu (similar to the function of the spacebar in 3D view). When first opening the node editor it will default to material editing mode. We're not going to be using that now, so switch over to Composite mode by clicking the button at the bottom of the node editor window. Then click 'Use Nodes' to enable the composite nodes. Now we're all ready to go.

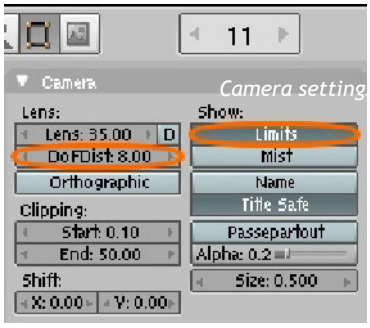
The node editor should already have two objects visible. Render Layers, and Composite, with the output 'Image' node of the Render Layers object connected to the input node of the Composite object. If not, hit spacebar and add the two nodes (Render Layers under input submenu; Composite under output) and connect them accordingly. To connect nodes, drag the output node over to the input node you want to connect it to, then release the mouse button. A line will be drawn from the one to the other signifying that they are linked.





The first thing we're going to be doing to this render is adding a depth of field effect to provide a focal point to the otherwise rather busy scene. Add a Defocus filter in the Node Editor, then link the Image output of the Render Layers object to the input of the Defocus object, and the output of the Defocus object to the input of the Composite object. The Defocus object also takes one more optional input in the form of the Z-Buffer. The Z-Buffer contains the depth value of every pixel on the screen so that the Defocus filter can determine how much to blend each part of the image. Connect the Z node from the render layers object to the appropriate defocus input node and disable the 'No zbuffer' option to tell the filter to use the zbuffer provided.

You'll notice that rendering the scene now won't yield any changes. This is because we have yet to set the focal length of the camera as well as a variable that tells the filter how much to blur objects as they move further away from the focal point. To set the focal length of the camera we need to go back to the 3D view. Select the camera object, and, in the edit buttons window,



Camera setting

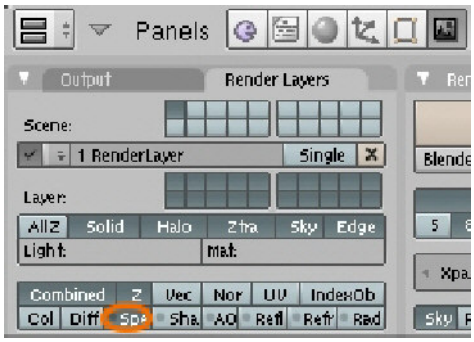
adjust the DoFDist value to adjust the focal point of the camera. Make sure that you enable 'show limits' so that the focal point will be displayed in the 3D view. Set the value so that the focal point is approximately at the blue cylinder object near the front of the scene (Best done from top view).

Back in the node editor, notice the fStop value in the Defocus filter. This value represents how much objects will blur as they become more distant from the camera. A lower value will increase the amount of blur, with 128 being perfect focus. Because our scene has generally very little in the form of depth a value of around 4 will provide best results. Be sure to disable the preview option in the filter as well, to provide a much higher resolution blur.

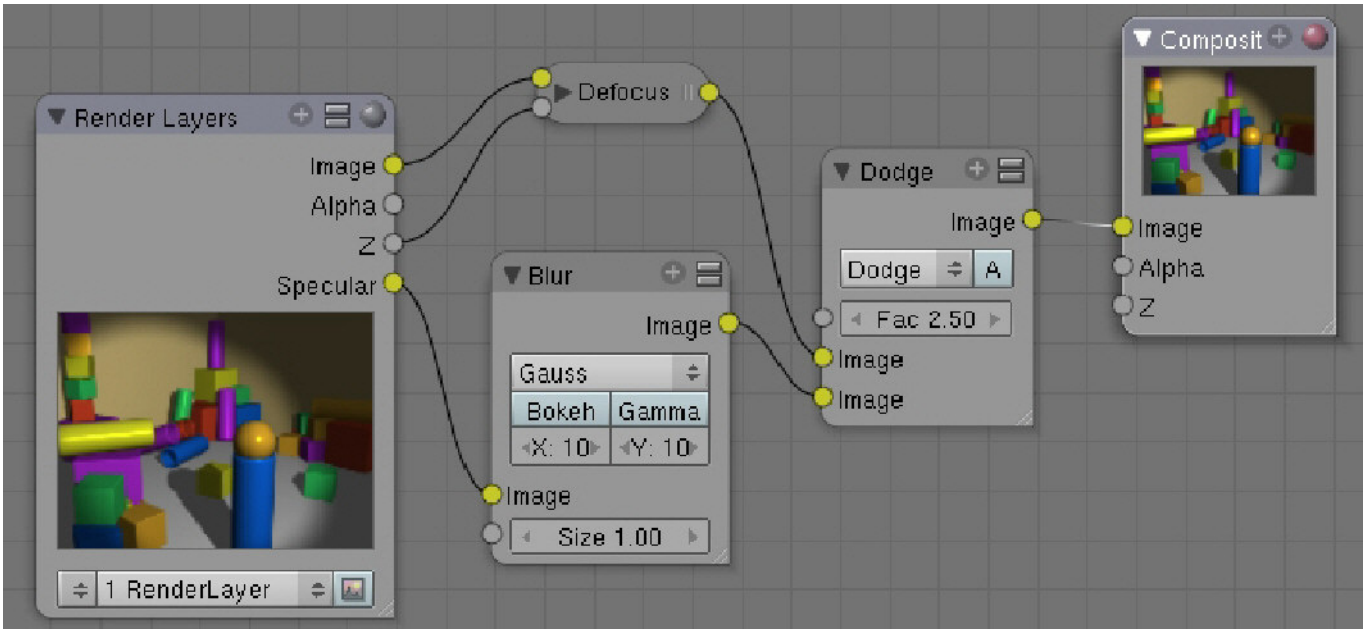
Defocus filter settings

That's it! You have now applied a depth of field effect to this scene. If you have already rendered the scene, you don't have to render it again to view the results. Because compositing is a post-process effect, Blender simply reapplies the effect onto the scene you've already rendered. Press F11 to display the results of the filter. Notice how a simple depth of field effect dramatically changes the points of interest in the scene.

The second effect we'll be adding to this scene is a filter to make the specular areas 'glow'. To do this, we'll need to grab an additional input from the renderer, the specularity pass, to use in post-processing. In the scene buttons, under the render layers tab, click the 'Spec' button to send an additional render pass to the compositor.

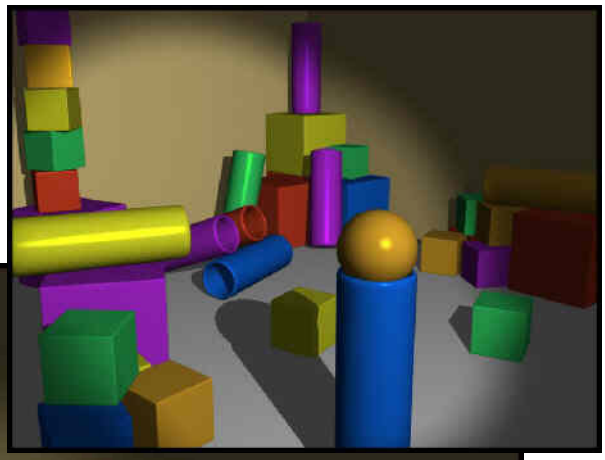


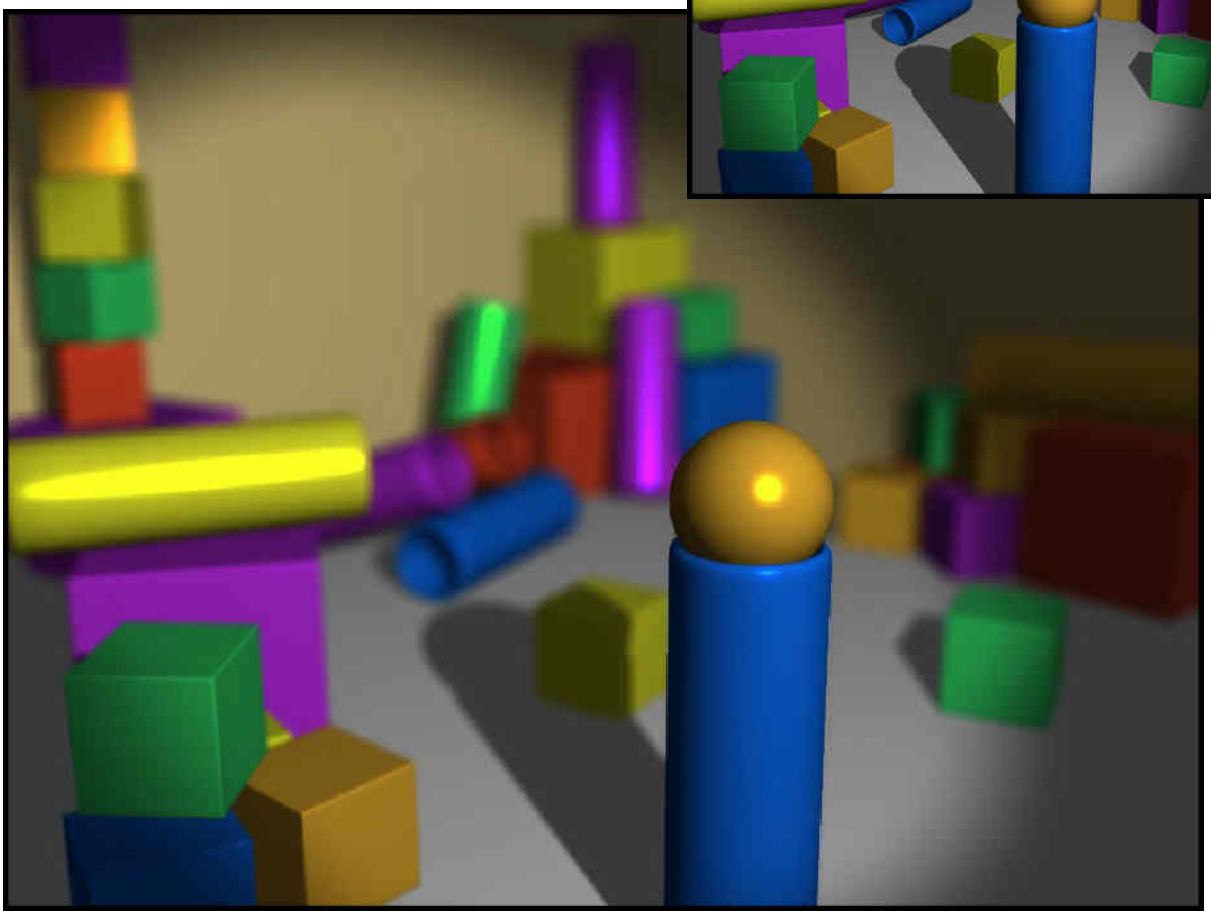
You'll notice an additional output node, Specular, has now appeared in the Render Layers node object. This represents the render pass that highlights objects. You'll need to render the scene again to provide this pass to the compositor. Once you've done that, if you're curious, you can connect the Specular node to either a new Viewer object (under Output) or to the Compositor node to get a preview of what information is contained in that node. We're going to take that specular information, blur it significantly and then blend it back into our scene.



Add a Blur filter object to the Node editor and connect the Specular output to it. The X and Y values in the Blur filter represent how much it will blur the image in a certain direction. Set them to 10 each. You can also change the blur type from Flat to Gaussian to produce a higher quality blur if you like, but it is a slower process. Finally, we need to blend this on top of our defocused scene for the final output. To do that, we'll need another node object. Add a Mix object (Under the Color submenu) to the Node Editor, connect the output from the defocus filter to the first input of the Mix filter, and the output from the blur filter to the second input. Change the drop-down box to 'Dodge', and the factor value to 2.5. Finally, connect the output on the Mix object to the Composite node. Pressing F11 will display the final scene, with a defocus filter and a slight bloom effect applied.

The final blender file will also be available from the Dev.Mag content section, as always. Happy Blending.





GAME GRAPHICS WITH ADOBE PHOTOSHOP

GAME GRAPHICS DESIGN

Part 4: Backdrops

By Rishal "TheUntouchableOne" Hurbans

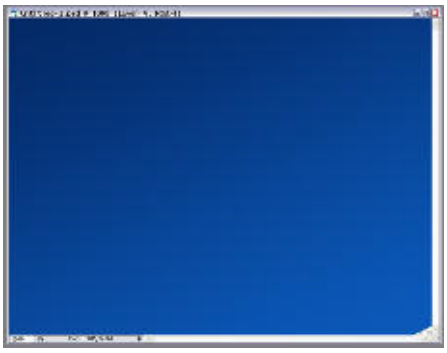
This article refers to resources available at the "Contents" section of the Dev.Mag website (www.devmag.org.za). It is recommended that you visit the site and download these resources.

You're well on your way to creating a complete set of graphics for a game. You have already learned how to create basic vector sprites and how to apply some basic but effective animation. The next step would be the character's surroundings. You can use the methods explained in tutorial two to create some neat vector objects for your game. We have mainly created vector images and haven't at all touched on raster images. Well this skill is particularly useful when creating large images that may contain detail and needs excellence. Images that are usually not resized are images suitable to be raster images. Think about a game such as Aquaria. The backgrounds used in that epic game are very detailed and are truly beautiful. We need the same for our games. We need awesome, detailed backgrounds. Jaw-dropping backdrops!

In this tutorial you will learn how to create useful backgrounds for you games. This tutorial will be a basic introduction to raster images and how they may be used and manipulated to produce impressive graphics for your games.

As usual we will need to start by creating a new empty image to work on. Select File>New...

Since we usually know the resolution of the window that our game will run in, we can use this as the size of our image. Let's make the image 800x600. Set the width as 800 and the height as 600. In the game where this background will be used, the window will have an 800X600 resolution.

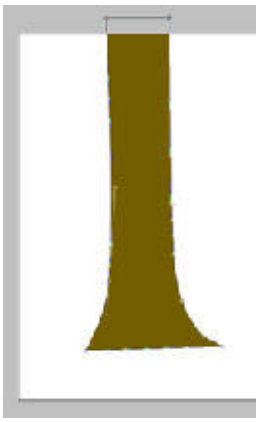


Now as you stare into the white space you might wonder what theme you would like to create for your game. It doesn't really matter what theme you aiming for, the general methods and uses of the tools in Photoshop are the same. There are a number of tools within Photoshop and a variety of ways to apply them to create the exact effect you are desire. We will create a forest type background, a forest scene in the moon-light sounds good.

Before you start on the image itself, brainstorm some ideas. What elements can you use in the image? Think about what elements of the background can also be incorporated into the game's playability itself. Trees are useful-items can be placed on them, lush grass is good-small enemies could emerge from it, a full moon and a some stars would also provide a good atmosphere. Now we can start with the actual design of the background.

The first element we will start with is the sky that would be at the extreme back of the image. Choose the gradient tool and select a dark blue going into a slightly lighter blue. Then, while the background layer is selected, left-click and drag the tool from the top to the bottom of the page. You should have a gradient on your image as shown to the left.

The next element that we will create is a tree. Select a shade of brown and use the pen tool to create a shape resembling the trunk of a tree. Make the size of the tree range from the bottom of the page to the top (these are very large trees). Tune the shape using the convert point tool to make your tree look its best.



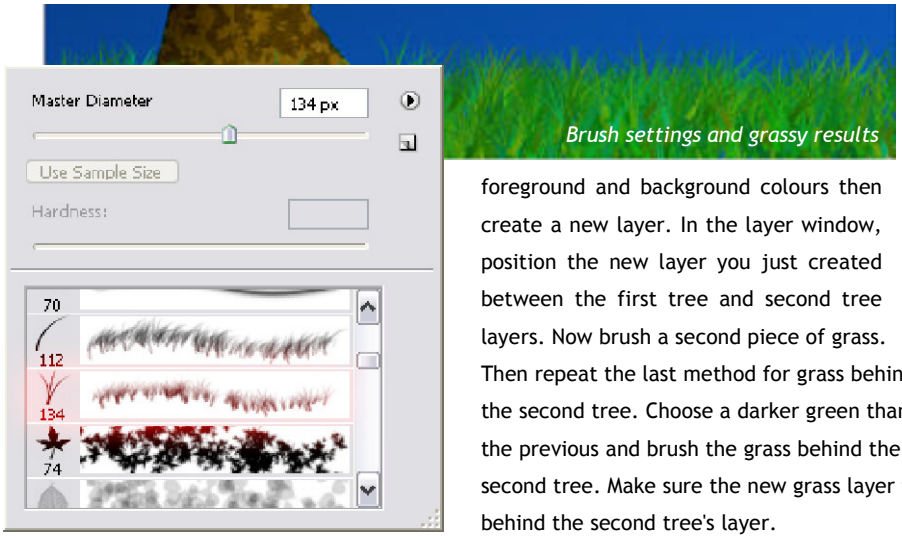
Once the you are happy with the shape of the tree, right-click on the tree's layer select Rasterize Layer. The image is no longer a vector image, it is a raster image. The tree needs some texture, we cannot draw each and every detail on the tree so we will use one of the filters in Photoshop. Select Filters>Artistic>Sponge... Set the required values as follows.

Note: Experiment and play around with the filters in Photoshop, they have a host of effects that are very useful in manipulating raster images.

Now we can use the burn tool to give the tree some depth. Select the Burn tool in the tools palette and apply it to the tree. Use it to your discretion and create shadows on the tree. The tree should look similar to the figure below.

I think the next element should be the front layer of grass, choose a medium dark green as the foreground colour and a lighter green as the background colour. We need to create a new layer for the first piece of grass. Select Layer>New>Layer. Now select the brush tool. In the options menu at the top of the screen, select the “grass” brush type and set the size value as 75.

Brush the grass across the bottom of the page. Start just before the bottom of the tree and continue to the far bottom of the page. This would be the most effective way to make the grass look its best. This makes the grass look like it’s firmly planted into the ground.



Brush settings and grassy results

The background is looking good already but we need more trees. Instead of going into the trouble of creating more trees from scratch, we will use our current tree, right-click on the tree's layer and select Duplicate Layer. Move it next to the first tree. Now we can resize the second tree and manipulate it to look slightly different to the previous tree. We will make the tree thinner and we will make it look further away from us. Select Edit>Transform>Scale to make the tree thinner and smaller(for depth distance) then Select Edit>Transform>Flip Horizontal, this will make the tree look distinguishable from our first tree. Your image should look similar to the example to the right.

We need another piece of grass between the first tree and the second one. Select the brush tool again, it should already be on the grass brush type. Now, select a darker green as the foreground and background colours then create a new layer. In the layer window, position the new layer you just created between the first tree and second tree layers. Now brush a second piece of grass. Then repeat the last method for grass behind the second tree. Choose a darker green than the previous and brush the grass behind the second tree. Make sure the new grass layer is behind the second tree's layer.

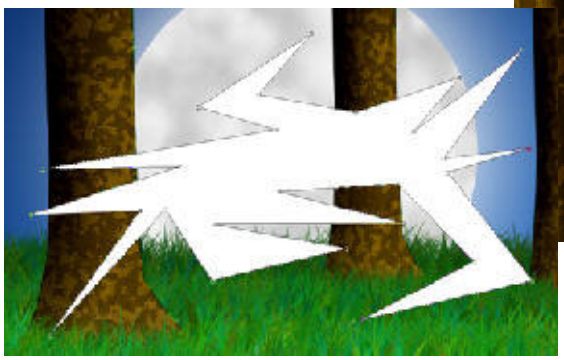
The background is coming along quite nicely, it is starting to look more polished but we need it to be more interesting. Lets add a big moon in the background on the horizon. Select the Ellipse tool with a foreground colour of white and draw a circle, it should be rather large as we want the moon to stand out somewhat. The moon layer must be behind the last piece of grass.

The moon needs texture and some glow. Remember the Blending options we worked with in tutorial two? We will use these again now. Select the blending options of the moon layer and choose outer glow, make the glow white and increase the size of the glow to give a brighter effect. We will also choose a pattern overlay for some texture. Browse through the patterns in the Blending Options, you should find one that looks similar to the surface of the moon. Use the scale slider to scale the pattern to the size that looks right.

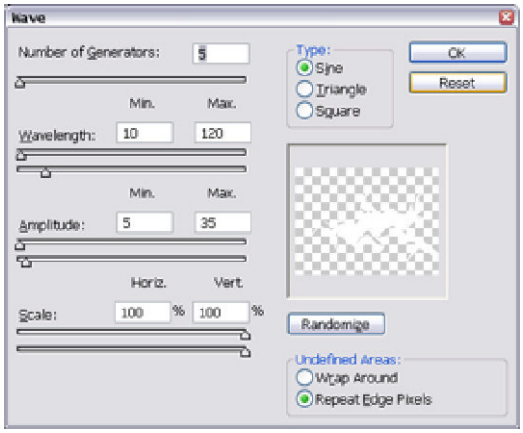
Two trees in a forest seems pretty empty. We need more trees. Two more will do the trick but we cant draw new trees, we will just duplicate one of the trees we have. Once you have duplicated one of the trees, resize it and find a suitable spot for it. You can do this for as many trees as you want.

Some stars in the sky would also be nice. Select white as the foreground colour , there should be a brush type in the brush options that looks similar to a star. Create a new layer for the stars and go wild with them.

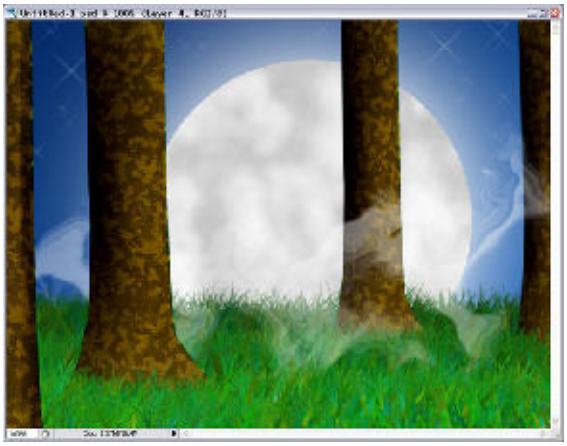
The background looks fairly complete. To add an awesome effect to the background, I thought fog. So to create some nice looking fog/smoke effect we need to create a new layer. Select white as the foreground colour then select the pen tool. Draw a sharp edged shape, similar to the figure here.



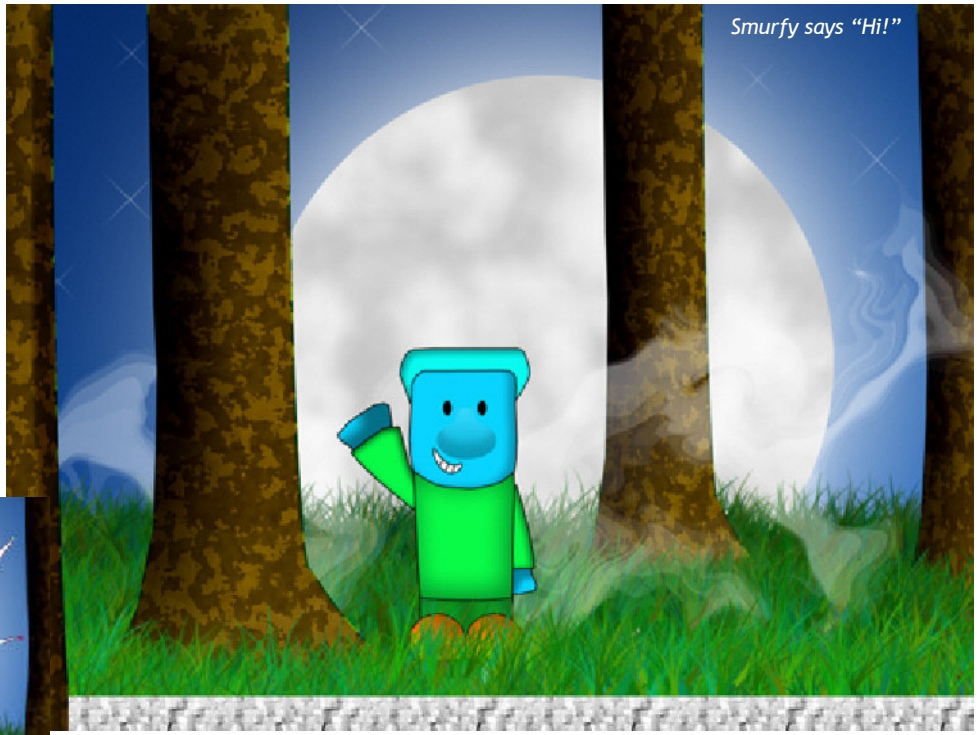
Now rasterize the shape by using the method explained earlier. This doesn't look good at all at the moment but I can guarantee you that the final fog will look excellent. We are going to use the Photoshop filters once again. The distort filter will help us a lot. Select Filters>Distort>Wave... Then set the values as shown in the image alongside.



Select Edit>Fade Wave... Set the fade at 50%. Repeat the Distortion and the Fade a few times until the white shape looks like fog or smoke. Once you have successfully completed this, set the fog layer's opacity to 40%-50%. Position the fog layer in between the first tree and second tree layers. The image should now look similar to the figure shown.



The final background can be used effectively in a game now. You can now go out and create your own master pieces for your games. Apply the knowledge gained here to any idea that comes to mind, practice a bit and you will soon be producing professional looking polished background for your games.



IRRLICHT

Introduction to Irrlicht

by Sven "FuzzYspoON" Bergstrom

Looking for a free 3D engine that can do all the latest tricks when it comes to game development is quite a task, but finding one that gives you the freedom to add, change or even recreate its whole core is even harder. Stumbling across Irrlicht3D is surprising as well as exciting; the engine is fast and flexible, free as well as open source.

Don't drop your jaw just yet. Irrlicht sports a wealth of features, a wealth of format support and a wealth of active and willing community members thriving on each other and the incredible backbone, Irrlicht3D. Boasting about features is what Irrlicht is good at without inspection, but it certainly holds a large number of great things it can do, and create. The creator and his team are a selection of incredibly skilled people who dedicate time to the engine to bring great features across to the end user. Niko, who founded Ambiera, has created a bunch of tools that utilise the Irrlicht engine to help with developing games, architectural software and pretty much any 3D application that requires a fast engine for 3D rendering, 3D sound and even well managed XML tools.

The feature list is quite long, but here are a few notable features.

- Cross platform, Cross language
- Powerful and fast 3D rendering pipeline
- Built in material library with vertex and pixel shader support
- Skeletal character animation and morphing control
- Built in special effects, particle system and scene optimisation
- Complete built in GUI
- PSD, JPG, TGA, BMP, PNG, PCX native texture support
- 3DS, B3D, OBJ, CSM, XML, DAE, DMF, OCT, IRR, IRRMESH, X, MS3D, MY3D, MESH, LMTS, BSP, MD2, STL mesh format support
- More at http://irrlicht.sourceforge.net/features.html

Above all the time that could be spent on the features of the engine, this series is rather aimed at using the engine to create games for free, and to create 3D applications with overwhelming ease. This first part is an introduction to the very basics behind using Irrlicht, and what it takes to use the engine for your game or application.

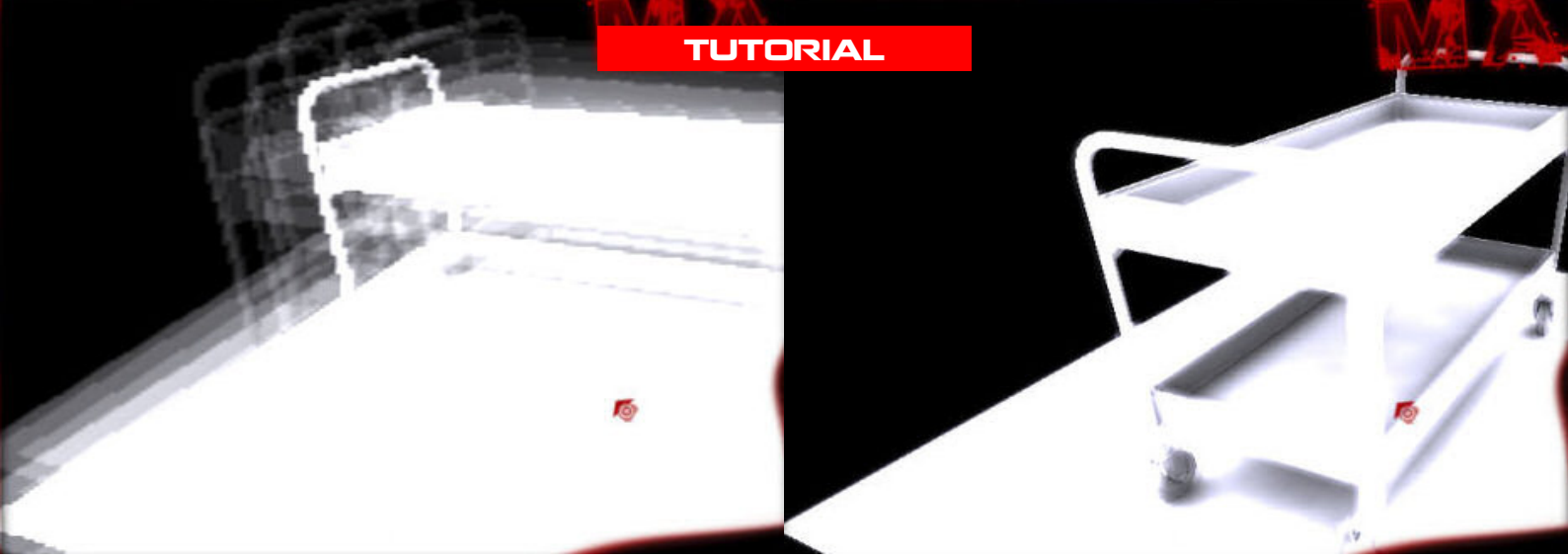
Weigh up your options before you start using Irrlicht. Don't ask on forums which engine to use for your first 3D game. Don't ask which is the best free engine available, I would even say that you should rather try each engine before even registering on a forum. Try them because each one is unique and each one has strengths and weaknesses. View each engine objectively and weigh up what you want from the engine, whether you can add or remove features and whether you can even make the project you are embarking on. A few questions to consider when choosing an engine can include the following:

Is the engine capable of what I need, if not am I able to add what is missing?

Is the engine too bloated for my miniature game?

Is the engine too complicated for me and my team to complete a project?

If the Irrlicht engine is one you could see yourself using, and you are confident it can help you learn and create a product you hoped you for, this series is aimed at you. If you are not new to game development but have never embarked on a 3D project before and like how Irrlicht works, this series is also aimed at you. If you are just interested in how to create simple games with the Irrlicht engine, this is also for you.

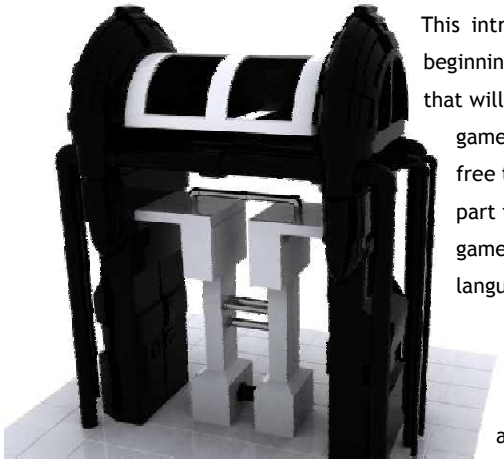


The other important ones of course include the video driver, as well as the scene manager, GUI manager and the GPU programming services. All of these interfaces are available through the IrrlichtDevice and each one manages its respective areas well. For example, you can use the scene manager to load a mesh, add a button in the window with the GUI manager and handle the events with the event receiver, all within a few lines of code.

Let's look at the basic outline of a standard Irrlicht program.

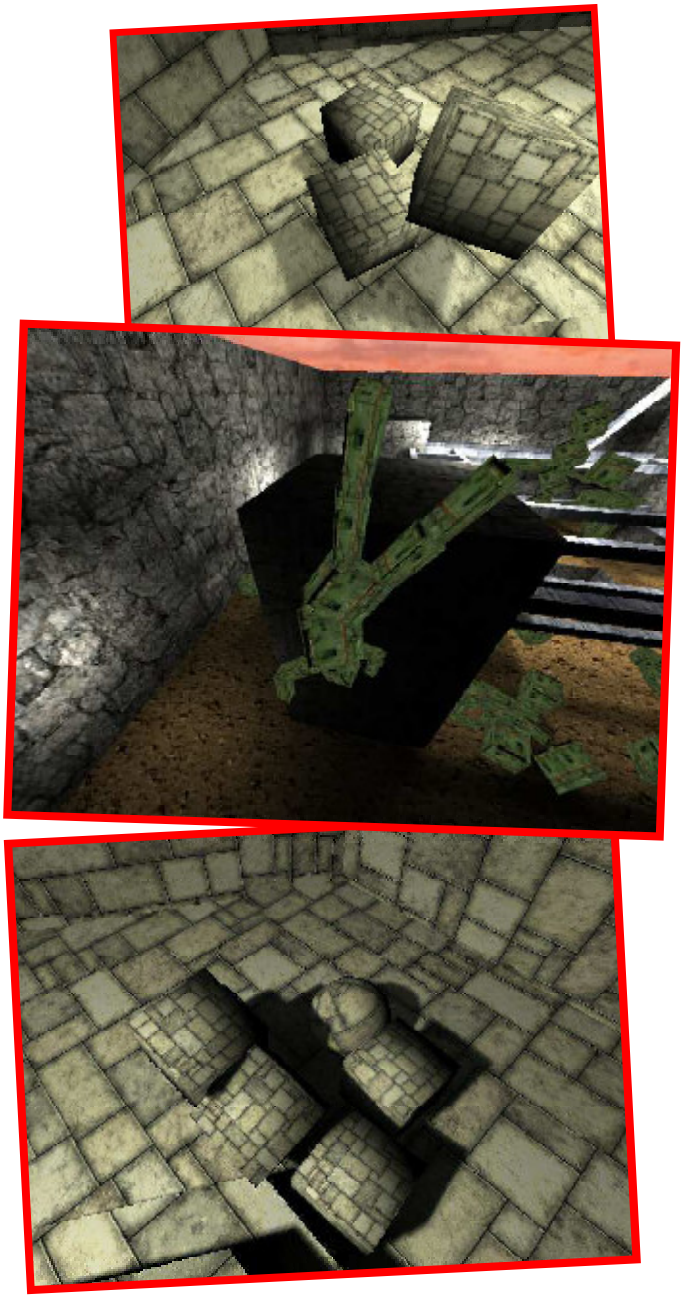
```
CreateDeviceEx(deviceParamaters) //hand the device the
parameters for the device
LoadScene(sceneFileName) // Loads a scene from irrEdit
into the scene manager
AddCameraSceneNodeFPS() //Add a built in first person
camera that handles keys

While (deviceIsRunning)

Draw all scene related stuff
Draw all GUI related stuff

End while
closeDevice
```

This is how easy it is to create a full 3D scene from an IRR file which is created by irrEdit, or saved by Irrlicht to a file. The built in camera class is a simple key controlled FPS style camera. There is also a simple camera and a Maya style camera available. Of course, you can make any kind of camera from a simple camera too.



This introduction is only the beginning of what will be a series that will show you how to create games using this easily-learned free tool. Look out for the next part in the series, a simple 2D game complete with multiple language implementations. In the mean time, head on over to http://irrlicht.sourceforge.net and download the latest SDK, or get connected to the SVN server for a more up-to-date version. There are plenty tutorials in the wiki and included in the download to get you started.

This draws the line in part one of this series, keep watching Dev.Mag for more Irrlicht tutorials, and start making full 3D games sooner than you thought.



HOW TO USE PERLIN NOISE

Your game textures will never be the same again ...

by Herman Tulleken

Perlin noise is the foundation of many procedural texture and modelling algorithms. It can be used to create marble, wood, clouds, fire, and height maps for terrain. It is also very useful for tiling grids to simulate organic regions, and blending textures for interesting transitions. In this article I will explain how to implement Perlin noise, and how you can use it in your games.

1.Implementation

Written in its concise mathematical form the Perlin noise generation seems daunting, but it is actually easy to implement. There are two steps:

1.Generate a number of arrays containing “smooth” noise. Each array is called an octave, and the smoothness is different for each octave. (See the first 7 images in Figure 1 below).

2.Blend these together. The result is the last image in Figure 1.

That’s all there is to it! Now let’s look at the code necessary to work this out.

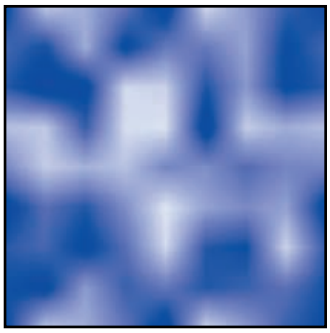
a) Generating Smooth Noise

First, you need to create an array with random values between 0 and 1. This array must be the same size as the array of Perlin noise you need.

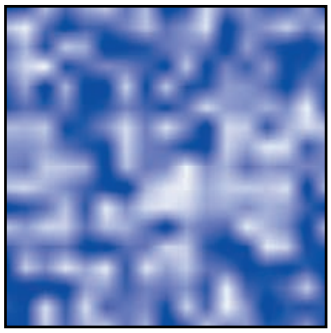
```
generateUniformNoise()
{
  baseNoise[][]; //an array for our uniform noise

  for(i = 0; i < width; i++)
    for(j = 0; j < height; j++)
      //random float between 0 and 1
      baseNoise[i][j] = random();

  return baseNoise;
}
```

For creating the *k*th octave, sample the noise array at every point (i*2^k, j*2^k), for all i, j, and interpolate the other points linearly. The value 2^k is called the **wave length** of that octave, and the value 1/2^k is called the **frequency**.

FIGURE 1

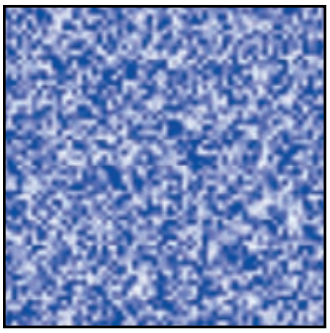


wave length = 64
frequency = 0.015625

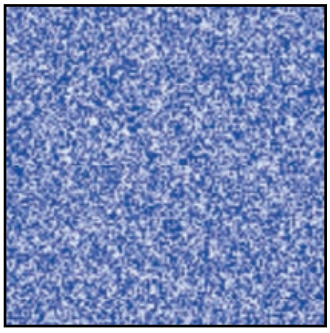


wave length = 32
frequency = 0.03125

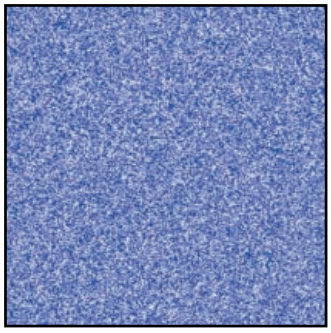
wave length = 16
frequency = 0.0625



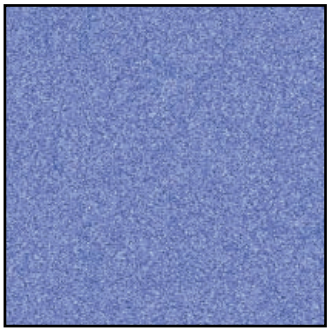
wave length = 8
frequency = 0.125



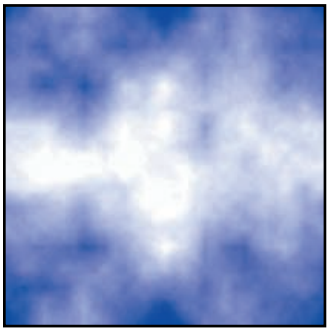
wave length = 4
frequency = 0.25



wave length = 2
frequency = 0.5



wave length = 1
frequency = 1



Perlin noise

The following pseudo C snippet shows how the *k*th octave is generated:

```
generateSmoothNoise(baseNoise, k)
{
  samplePeriod = 1 << k; // calculates 2 ^ k
  sampleFrequency = 1.0f / samplePeriod;

  for(i = 0; i < width; i++)
  {
    //calculate the horizontal sampling indices
    sample_i0 = (i / samplePeriod) * samplePeriod;
    sample_i1 = sample_i0 % width; //wrap around

    horizontal_blend = (i - sample_i0) * sampleFrequency;

    for(j = 0; j < height; j++)
    {
      //calculate the vertical sampling indices
      sample_j0 = (j / samplePeriod) * samplePeriod;
      sample_j1 = (sample_j0 + 1) % height; //wrap around

      vertical_blend = (j - sample_j0) * sampleFrequency;

      //blend the top two corners
      top = interpolate(baseNoise[sample_i0][sample_j0],
        baseNoise[sample_i1][sample_j1], horizontal_blend);

      //blend the bottom two corners
      bottom = interpolate(baseNoise[sample_i0][sample_j1],
        baseNoise[sample_i1][sample_j1], horizontal_blend);

      //final blend
      smooth[i][j] =
        interpolate(top, bottom, vertical_blend);
    }
  }

  return smooth;
}
```

The following function returns a linear interpolation between two values. Essentially, the closer alpha is to 0, the closer the resulting value will be to x0; the closer alpha is to 1, the closer the resulting value will be to x1.

```
interpolate(x0, x1, alpha) //alpha lies between 0 and 1
{
        return (1 - alpha) * x0 + alpha * x1;
}
```

There are a variety of interpolation schemes; the best one to use depends on your application.
See http://local.wasp.uwa.edu.au/~pbourke/other/interpolation/ for some common interpolation schemes.

b) Blending the Arrays

To make the final array, you add weighted values of all the smooth noise arrays together. The weight used for each octave is called the **amplitude**. Any values can be used for the amplitudes, with different effects. A good starting point is to use a weight of 0.5 for the first octave, 0.25 for the next octave, and so on, multiplying the amplitude with 0.5 in each step. In this scheme, the value 0.5 is called the **persistence** of the noise. After you have added all the noise values, you should normalise it by dividing it by the sum of all the amplitudes, so that all noise values lie between 0 and 1.

```
generatePerlin(baseNoise, octaveCount)
{
  smooth[]; //an array of 2D arrays containing
  persistance = 0.5f;

  //generate smooth noise
  for(i = 0; i < octaveCount; i++)
    smooth[i] = generateSmoothNoise(baseNoise, i);

  perlinNoise[][]; //an array of floats initialised to 0
  amplitude = 1.0f;
  totalAmplitude = 0.0f;

  //blend noise together
  for(k = 0; k < octaveCount; k++)
  {
    amplitude *= persistance;
    totalAmplitude += amplitude;

    for(i = 0; i < width; i++)
      for(j = 0; j < height; j++)
        perlinNoise[i][j] += smooth[k][i][j] * weight;
  }

  //normalisation
  for(k = 0; k < octaveCount; k++)
  {
    perlinNoise[i][j] /= totalAmplitude;
  }

  return perlinNoise;
}
```

In practise smooth noise arrays are not actually created; rather, the noise is calculated and added to the final array on the fly. How this is done is not shown here - check out the link at the end of the article for a link to example code.

Perlin noise can sometimes look diluted. For many applications this is fine. However, sometimes you may want something a bit more dramatic. This can easily be achieved in one of the following ways:

— Increase the contrast as a post-processing step using image editing software.

– Use a higher persistence (0.7 is good for 6 octaves), and skip the normalisation step. Just make sure to clamp your final values to 1.

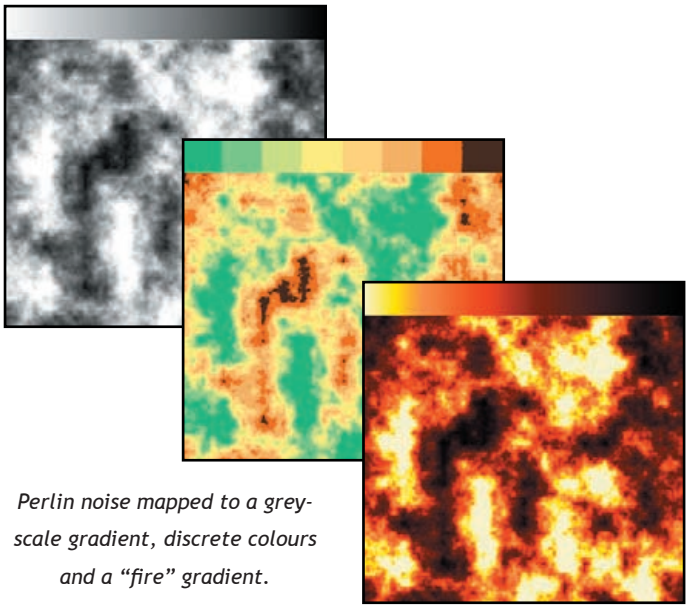
The first method is better when you want tight visual control. The second method is more convenient as it is part of the creation process and can hence be better automated.

2.Applications

a) Textures

One of the simplest uses of Perlin noise is to map it with a gradient. This can be used for attractive maps or cheesy fire effects (which you can animate – explained in another section) as shown in Figure 2. You can do this with your image editor, or programmatically. For the latter approach, you need a gradient function that returns a colour given a number between 0 and 1. This function is then called for every element in you Perlin noise array to obtain a colour, which you can store in a separate array, from which an image can be created.

FIGURE 2



Perlin noise mapped to a grey-scale gradient, discrete colours and a "fire" gradient.

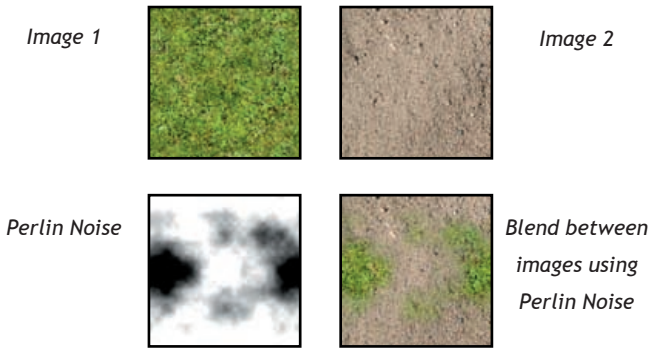
Here is a code snippet showing how it works:

```
getColor(x) //x is a number between 0 and 1
{
  //Replace this with your own code
  //This is a gradient from white to blue
  return Color(255 * (1 - x), 255 * (1 - x), 255);
}

mapGradient(perlinNoise[][])
{
  image[][]; //an array of colours

  for(i = 0; i < width; i++)
    for(j = 0; j < height; j++)
      image = getColor(perlinNoise[i][j]);
}
```

Perlin noise can be used to blend between two textures, as shown in Figure 3. You should use Perlin noise with very high contrast to prevent textures from looking fuzzy. The following code snippet shows how to blend two images using Perlin noise.

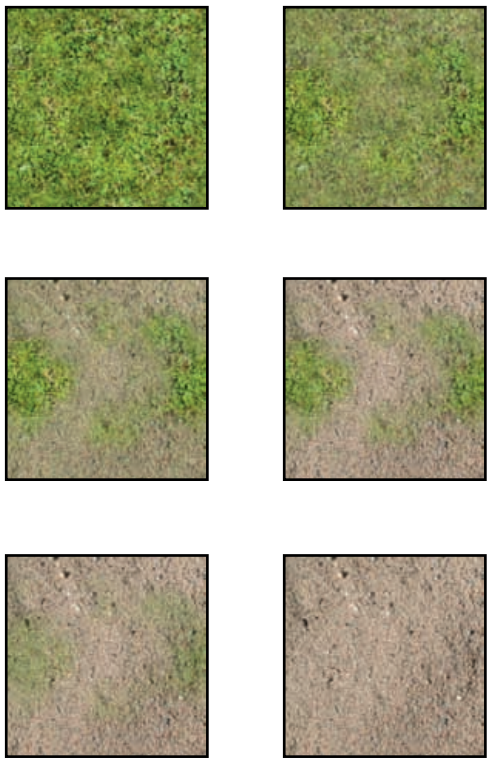
```
blend(image1[][], image2[][]; perlinNoise[][])
{
  image[][]; //an array of colours for the new image

  for(i = 0; i < width; i++)
    for(j = 0; j < height; j++)
      image[i][j] = interpolate(image1[i][j], image2[i][j],
        perlinNoise[i][j]);

  return image;
}
```

FIGURE 3



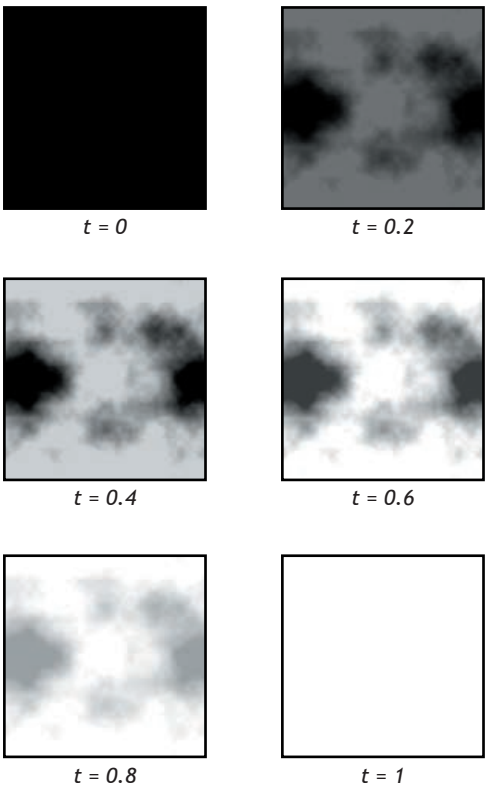
Of course, you would not create textures this way, but it can be used for interesting real-time transitions. Figure 4 shows very cheap plant growth using only three textures and appropriate blending.

FIGURE 4.1



Real-time transitions using Perlin blending

FIGURE 4.2



Effective blend texture

The following function blends the images as shown by calculating a new blend factor alpha, depending on the value t. The value t runs from 0 to 1, and is calculated from the elapsed game time.

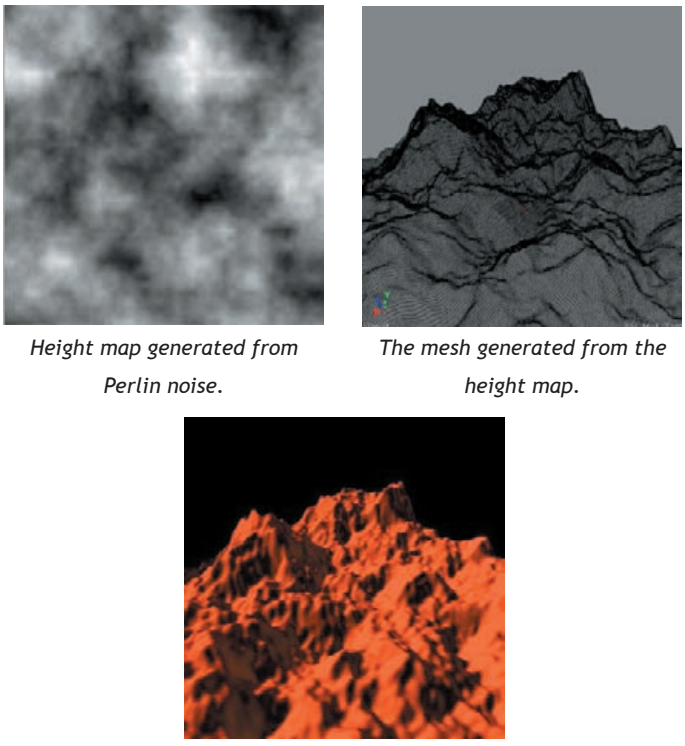
```
blend(image1[][], image2[][], perlinNoise[][], t)
{
  image[][];

  for(i = 0; i < whidth; i++)
    for(j = 0; j < height; j++)
    {
      //calculate new blend factor alpha
      if (t < 0.5)
        //blend PerlinNoise with black
        alpha = perlinNoise[i][j]*t/0.5;
      else
        //blend PerlinNoise with white
        alpha = perlinNoise[i][j]*(1-t)/0.5 + (t-0.5)/0.5

      //blend images using alpha
      image[i][j] =
        interpolate(image1[i][j], image2[i][j], alpha);
    }

  return image;
}
```

b) Landscape Generation

When Perlin noise is interpreted as a height map, an interesting terrain can be created (Figure 5).

Softimage Mod Tool is a free 3D modelling and animation application especially suited for games. One of it's features is a built-in landscape generator that can use Perlin noise, among several others, to generate landscape meshes (http://www.softimage.com/products/modtool/).

FIGURE 5



Height map generated from Perlin noise.

The mesh generated from the height map.

The rendered mesh.

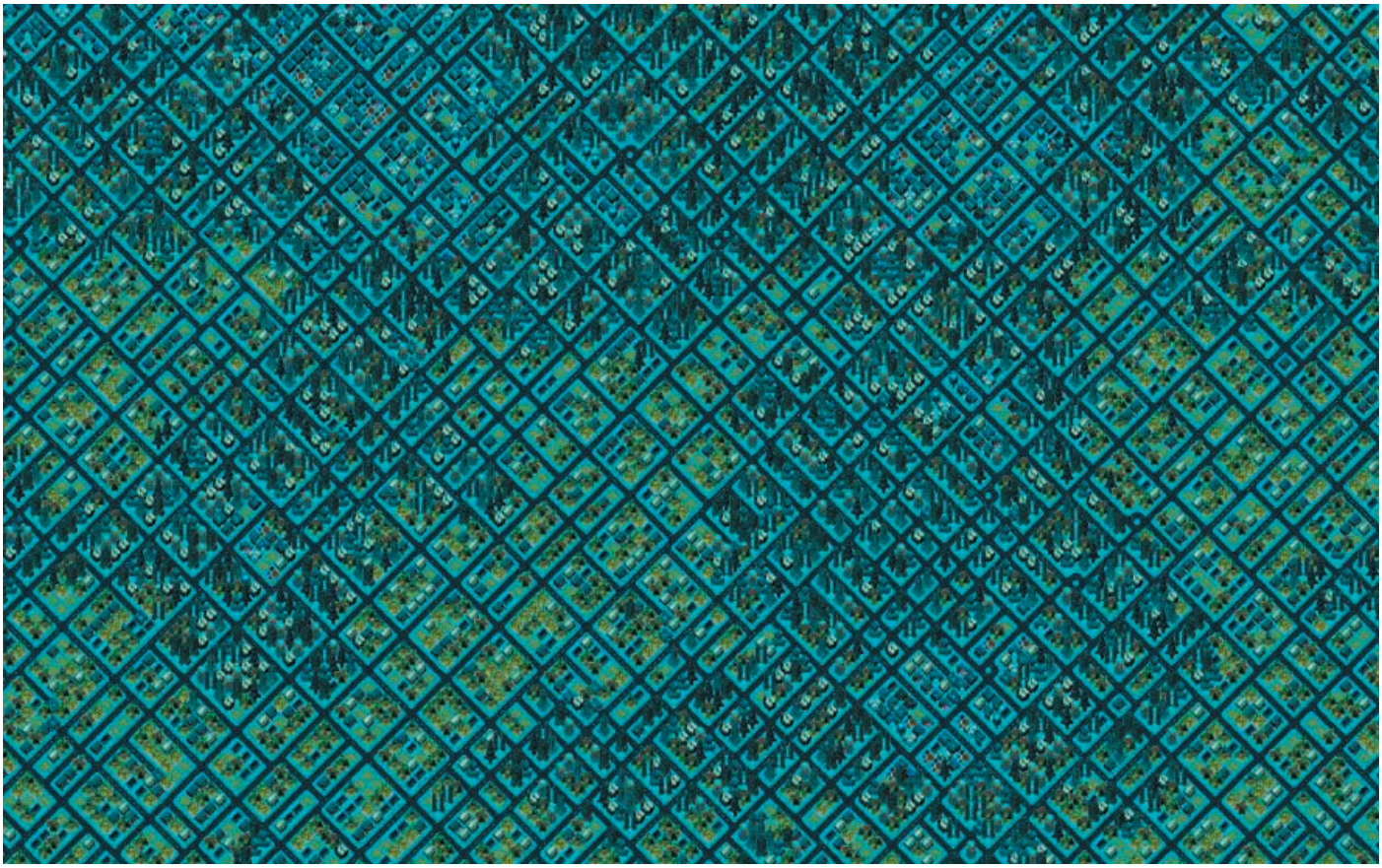
c) Object Placement

Perlin noise can also be used to place objects on a grid more naturally than can be done with uniform random placement. To do this, follow these steps:

(1) Define *k* sets of similar-looking objects.
(2) Create Perlin noise large enough to cover your grid. Each pixel of noise should correspond with one cell in the grid.
(3)For every cell in the grid, find the corresponding pixel of noise, and use the following formula to decide from which set you should choose an object: i = floor (n / (1.0 / k)). You choose objects from set S[i], usually randomly.

You can use this method even if you do not use a grid to place objects. Just define a grid so that there is roughly, on average, one object per cell. For every object to be placed, first determine the cell it corresponds to, and then proceed as above. If this requires a too large grid, you can use a smaller grid, so that there is more than one object per cell. Use linear interpolation to obtain a Perlin value for an object. You won't get the Perlin pattern inside a grid cell - but it will hide the fact that there is a grid.

FIGURE 6



In large worlds, the Perlin patterns can easily be seen. Here three sets of objects have been used: city, suburban, and industrial. (Art: Chris Cunnington)

For you to see the characteristic Perlin pattern, the world has to be quite large (Figure 6). However, you can benefit from Perlin placement even for small worlds. In a small world, there will be much more variation between successively generated worlds than there would be had they been generated by another method, as the images in Figure 7 illustrate.

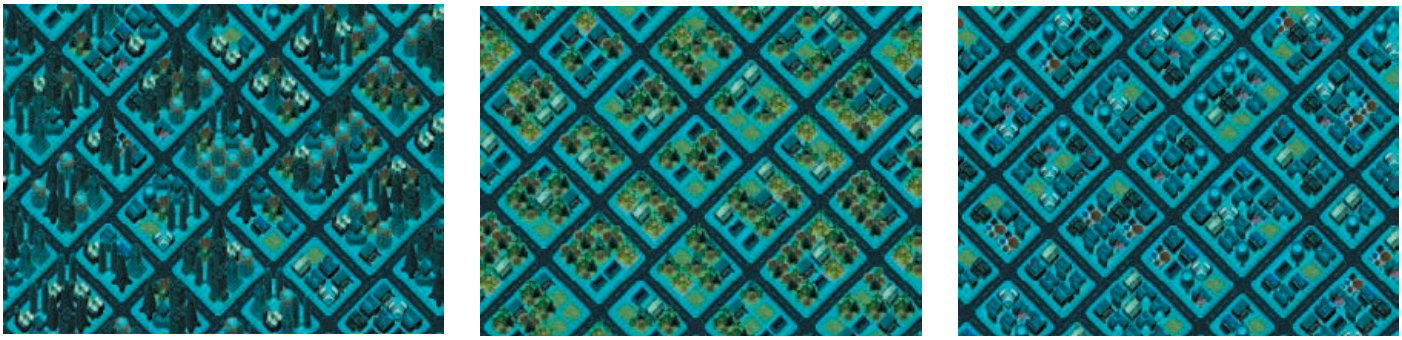
Sometimes you can construct your sets so that a certain property maps to the set number in an obvious way. For example, for sets of buildings, it would make sense to let the shortest buildings be in set 0, and the tallest buildings in the last set. Arranging your objects in this way ensures that very short buildings are never next to very tall buildings. This can enhance the illusion of "spatial progression".

Sometimes there is no relationship between the set number and any property, and you benefit nothing from one set of objects always being close to objects in the same or adjacent sets. In this case, you can increase the amount of variety between different worlds by permuting the set indices, i.e., every time the algorithm runs, i = 0 will select objects from a different set. This can easily be accomplished by creating a shuffled array m containing the integers 0..k-1, and then choosing objects from the set S[m[i]].

It is very noticeable when two adjacent objects are the same, and this happens much more frequently with Perlin placement than with uniform random placement. There are three ways in which you can reduce or eliminate this effect:

1.Create more objects. To get the same number of adjacent objects in a grid as you would get with uniform random placement, you would

FIGURE 7



A big variety of small worlds. (Art: Chris Cunnington)

need as many objects in a set as you would need in the entire collection for uniform random placement! If you have *k* sets, you will need to create *k* times the amount of art!
2. Another nifty trick is to further subdivide each set into "white" and "black" objects. The grid is then treated as a black and white checker board, and objects are placed so that "white" objects are always on "white" squares, and "black" objects always on "black" squares. This trick can be expanded to more than two colours, and non-periodic grid patterns, but is the subject of another article.
3. In some games, you can reduce the jarring effect by introducing small random transformations (rotation, scaling, mirroring, and for certain types of objects, shearing). This is especially effective for plants and other organic objects. You can also introduce small random colour variations, although you should take care not to destroy the overall colour and lighting of the game.

For this application, you might find that the first and last sets are under-represented. This has to do with the way Perlin noise generates more greys near the centre than it generates whites and blacks. The easiest way to rectify this problem, is to duplicate the under-represented sets.

The more sets you have, the smaller contiguous regions will be, and the more interesting your grid will be. If you have a small number of sets, you might wish to duplicate the sets for an more interesting world. Make sure that duplicate sets are not next to each other (for example, S[0] and S[1] must contain different elements) – otherwise this step will have no effect.

Be careful when tweaking a world generation parameter: remember that Perlin worlds look quite different from each other with the same parameter settings, something that can easily throw off any "trends". Always run the algorithm a few times with the same settings to make sure what you perceive as a general occurrence is not an isolated anomaly.

d) Perlin Noise Tiles
Perlin noise can be made tileable by using a power of 2 for the array dimensions (128 × 128, for example).

You can also create a set of tiles by using a set of base noise arrays with the same first row and first column. All tiles in the set will tile smoothly with each other.

e) Perlin Noise Animation
The algorithm for generating Perlin noise is easily modified to make animation sequences. The basic idea is to generate a block of 3D Perlin noise, cut it in slices, and use each slice as an image of the animation sequence. If you use a power of two for the time dimension, the sequence will loop smoothly as well.

This sequence can be used for object placement to simulate transitions that will add life to your game.
The 3D version of the algorithm is quite slow – on my machine a lazy-man's implementation takes about thirty minutes to produce 256 frames of a 256 × 256 image sequence.

3. Download

You can download implementations for these algorithms from
http://www.luma.co.za/labs/2008/01/20/perlin-noise/
There are implementations in Java, Python and GML (GameMaker).

DB SAYS ...

If this isn't enough to get you going with Perlin Noise, here's a few other sources which may help you understand the topic better!

Gustavson, S. **Simplex noise demystified.** *staffwww.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf*
Explains classic Perlin noise, and the improved simplex noise. The treatment is technical, but comes with very readable source code.

Atkins, M.; Barad H.; Gerlitz, O.; Goehring, D. **Real-Time Procedural Texturing Techniques Using MMX.**
http://www.gamasutra.com/view/feature/3098/a_realtime_procedural_universe_.php
More texture generating techniques using Perlin noise.

Elias, H. **Perlin Noise.** *http://freespace.virgin.net/hugo.elias/models/m_perlin.htm*
A very good introduction to Perlin noise.

Perlin, K. **Making Noise.** *http://www.noisemachine.com/talk1/*
The creator of Perlin noise's website.

Zucker, M. **The Perlin noise math FAQ.**
http://www.cs.cmu.edu/~mzucker/code/perlin-noise-math-faq.html
A deeper look into the mathematics of Perlin noise.

PUMP IT!

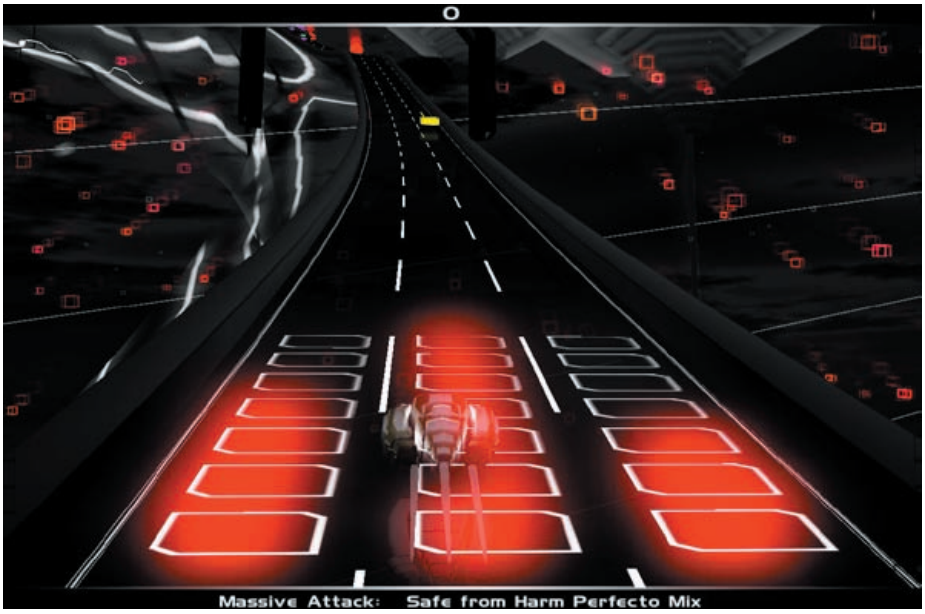
Sound-based games and the developers who love them

by Rodain "Nandrew" Joubert

A common habit of most hobbyist game developers is to fuss about with the gameplay and graphics in their work, getting those things all nicely done up before looking for an arbitrary sound package on the Internet and slotting in sound effects which sound good enough to fit in with the game that's been cranked out.

While this may not be all that bad (and in some cases should suit the developer's needs just fine), there's definitely a lot more work that can be done with sound in most people's games ... moreover, there's a lot more potential that can be squeezed out of the sound aspect if the game's designer gets sufficiently creative. Audio doesn't have to be a purely aesthetic aspect of one's game. It can serve as a very real and very functional aspect to the point where it becomes an integral part of the gaming experience.

Today, IGF Grand Prize nominee Audiosurf is one of the prime examples of what sound can do for your gameplay. The game's developer, Dylan Fitterer, has spent several years working on his creation using a mix of Visual C++ and Quest3D, and recent beta releases have shown players a very promising title which proves to be both fun and original. "The ability to use your own music is the main draw of the game, and I really think that's what players love about it," Fitterer says. "It's like a big sandbox - you can create whatever experience you want to create."



Sound and rhythm games haven't always been as sophisticated as they are nowadays, but from their earliest incarnations they've been a steady and accessible source of fun for many gamers. The genre itself is comparatively new - most well-known audio games are less than a decade old, while the mainstream gaming industry has been around for at least 30 years. The most likely explanation for this is the limitation of technology: busting a move to the beeps and blurps of an old NES system, after all, isn't what most people have in mind when they want something to get their groove on. Nowadays, in the era of massive MP3 collections, music visualisers and an ever-expanding gamer base, it's not hard to see how music and rhythm are having a much more profound effect.

"It took years of tuning to get the game to be fun across all kinds of different types of songs. That was a lot of hard work but it's really paid off because I'm so happy with where the game is today. It's great to hear people having fun across their entire musical collection."

— Dylan Fitterer on the challenges of making Audiosurf

Audio and rhythm - the difference?

One of the first things to get out of the way is that a game based on sound is not necessarily the same as a game based on rhythm. Some games may opt to alter the game environment for the player - for example, generating tracks in Audiosurf based on the background music - without requiring an actual sense of rhythm on the player's part. Games such as Rez and Lumines are other examples of sound-based games where the player doesn't necessarily have to be skilled in rhythm games to enjoy.

Rhythm games tend to require careful, regular timing from a player, which may not be the case for all titles in which music is a dominating factor. DDR and Guitar Hero are easily branded as rhythm games. Karaoke titles are not.

Rhythm games are by far the most common type of sound-based games due to their familiarity and ease of use. They're a handy option for interested developers to take if a sufficiently novel or addictive premise can be

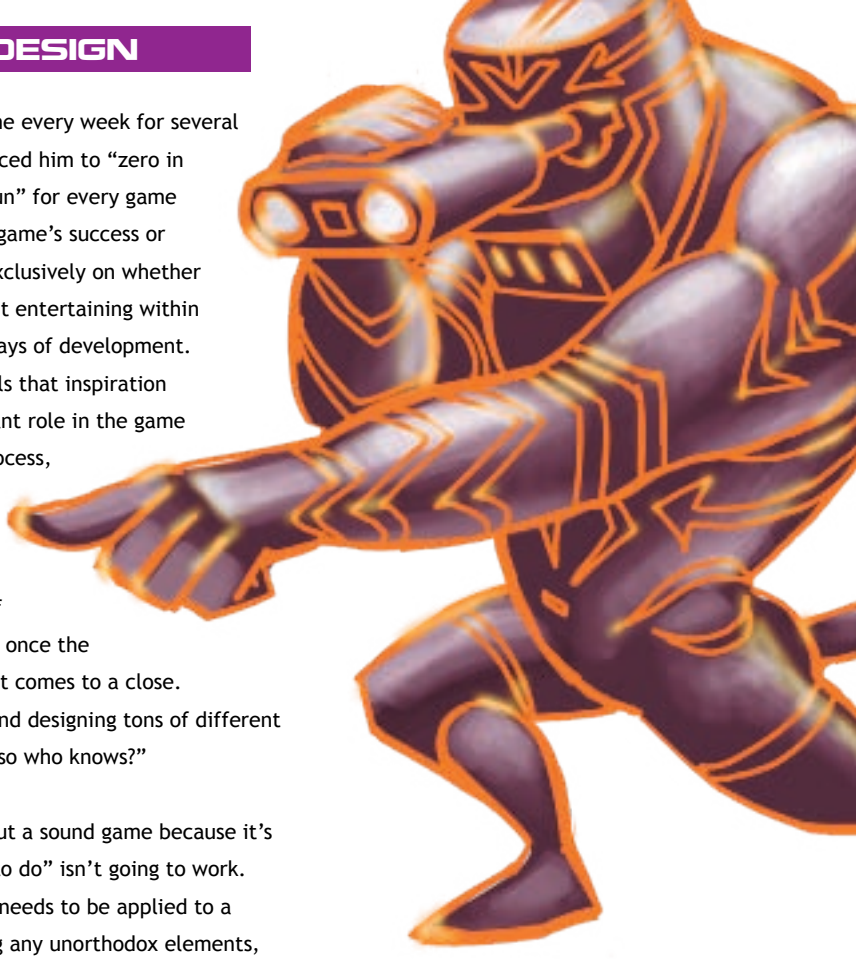
found, and inspiration for these sort of titles can be found everywhere. Indie equivalents for a lot of mainstream rhythm games already exist: take, for example, Stepmania (a popular DDR clone) and Frets on Fire (an indie incarnation of Guitar Hero).

However, although non-rhythm sound games are less common, the fact that the genre is considerably undeveloped when compared to the heavily saturated realms of FPS titles and RPGs means that new developers have a lot of opportunities to explore brand new territories.

Not a walk in the park

Sound-based gameplay may be a fertile land of opportunity, but that doesn't mean that things are going to be as easy as throwing out a random audio title and raking in an inevitable mountain of praise. A lot of work goes into these sort of titles, not least because of all the associated idiosyncrasies. The potential of audio games may be tempting for the avid developer, but actually finding a game concept that works and can successfully be molded into something entertaining is no small challenge.

Audiosurf's Dylan Fitterer explains how he got his development experience a long time before working on his flagship title by developing a free game every week for several months. This forced him to "zero in on the core of fun" for every game he made – each game's success or failure hinged exclusively on whether or not he made it entertaining within the first seven days of development. Fitterer also feels that inspiration plays an important role in the game development process, and he'll quite possibly be doing a different type of game altogether once the Audiosurf project comes to a close. "I love playing and designing tons of different kinds of games, so who knows?"



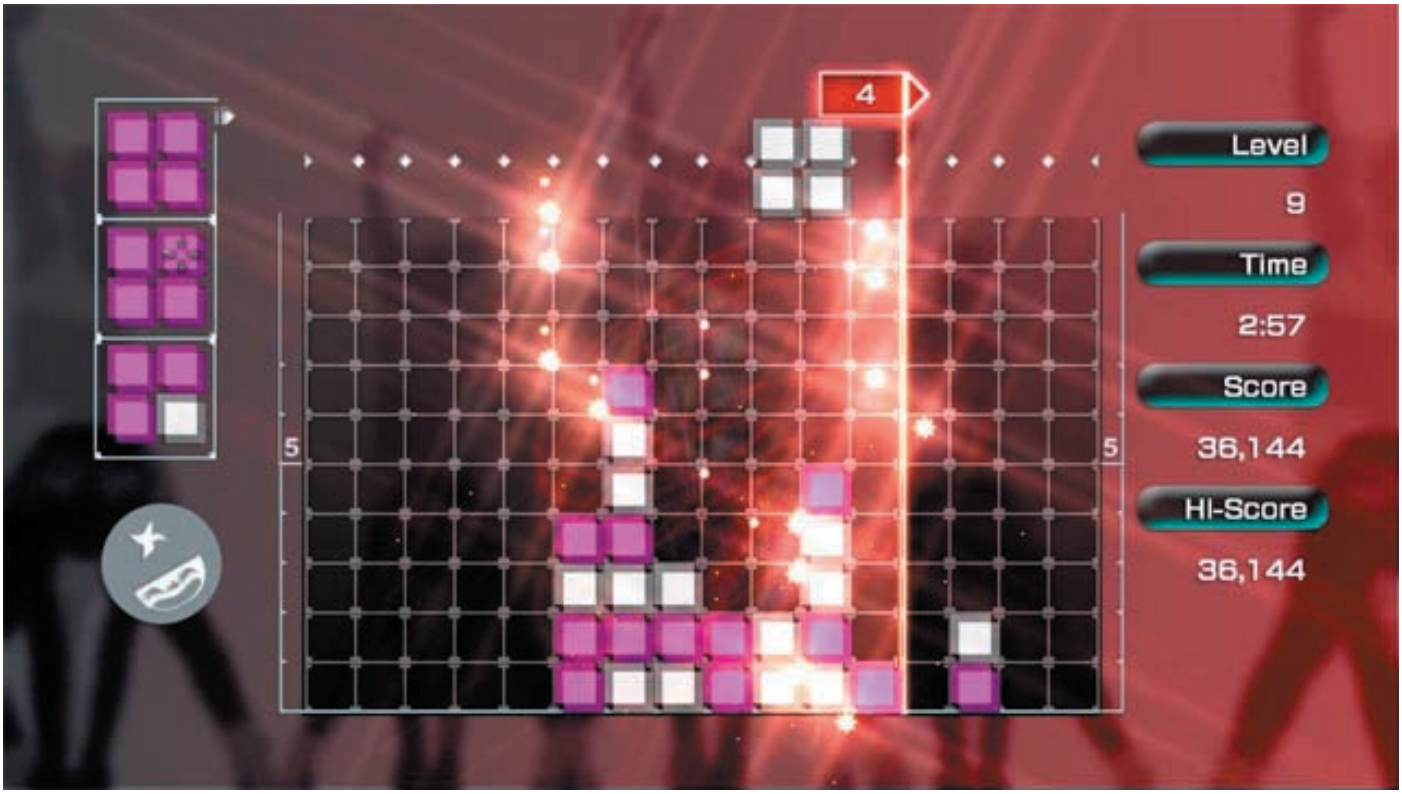
Simply forcing out a sound game because it's the "cool thing to do" isn't going to work. Careful thought needs to be applied to a project involving any unorthodox elements, and a great deal of rapid prototyping is a better idea than simply jumping head-first into a major project.

What to do

So, your heart is set on developing something with audio-based gameplay. What steps can you take to ensure that it turns out as fun as possible?

Practice, practice, practice

Prototyping is your friend. Focusing on small projects also helps. In fact, as a good developer you should already be trying to get as many experimental games under your belt as possible. Seeing how people react to these sort of projects is key to establishing what's fun, what your capabilities are and what direction you should be taking. This is



particularly valuable in areas like sound and rhythm.

Learn from others

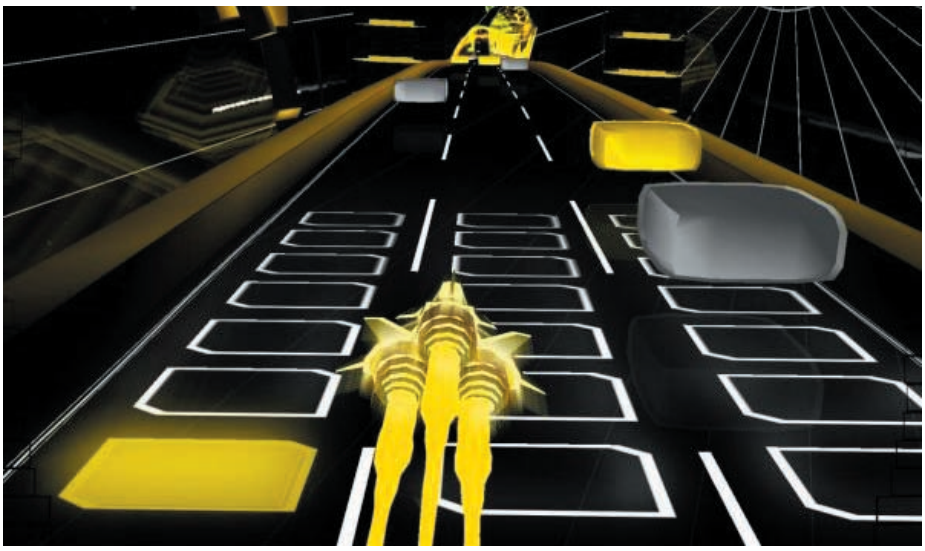
Fitterer talks about the games that inspired Audiosurf. “Rez is a big inspiration, and I also loved a 3D visualizer Wild Tangent put out several years ago where a ship flew above beautiful terrain that was shaped by the music.” He based the creation of his game on these titles, and Audiosurf is frequently compared to many other sound games as well. Esentially, if a game like Lumines is released and enjoys massive popularity, chances are that it’s doing something right. Part of any developer’s design process should involve analysing these games properly and isolating the fun factor. Better still, have a look at flawed games and take note of the mistakes made. Errors and muck-ups are the best learning opportunities, after all.

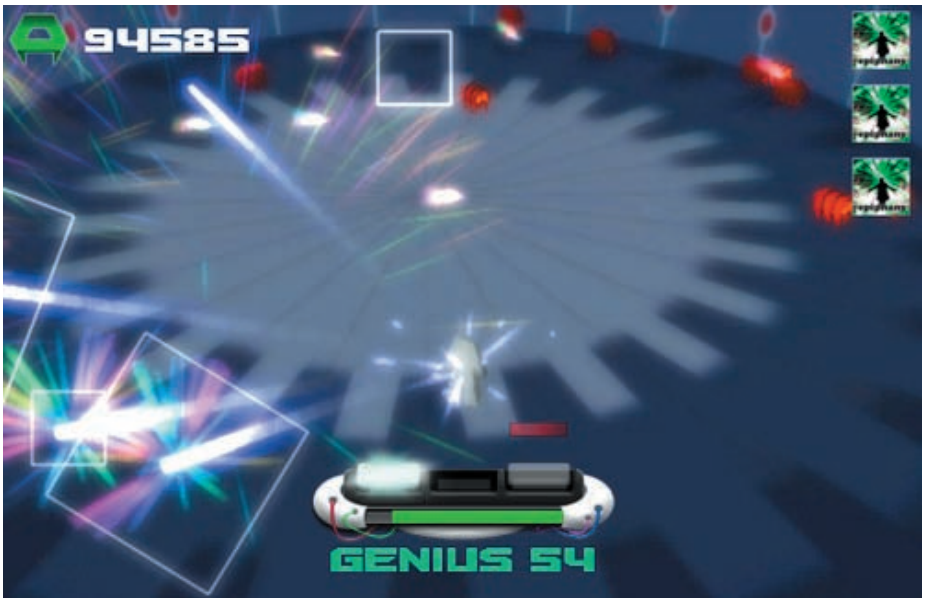
Understand sound

Most developers know well enough how to throw sound into their games – in most tools, it’s simplified to a sound_play() option or something similar. However, the challenge increases monumentally when you want your sound to do a little more than just play passively. Audiosurf, for example, requires a full analysis of the sound provided to make sure that the racing track has bumps, twists and cars in all the right places. Go onto the Internet and do some research on digital audio – it’ll grant you some extra flexibility and help you deal with errors when they crop up.

Test it!

“It’s great to hear people being so passionate about Audiosurf and wanting more!” says Fitterer about his first public beta. In his case, the amount of feedback and suggestions that he received in one weekend was phenomenal compared to that which he could expect when keeping the game largely private. He believes that putting other people’s ideas into the mix was definitely a positive move. Perhaps you don’t want to go as far as an-





“Remember how personal music is to everyone ... classical, rap, heavy metal, ambient, or whatever ... allow for a broad range of experiences with your game that allows players to feel a full spectrum of emotions.”

— Dylan Fitterer on good development practice

nouncing a public testing session, but it's a good idea to at least get a regular, reliable group of testers at your beck and call. Releasing test versions and requesting feedback from communities is not only a great way to see how actual players react to your game, but people have an incredible knack for finding errors that never occur to the developer. In the case of a game that you want to market, open testing is also a great excuse to give your project more attention.

To wrap it up

Audio-based gameplay is an exciting and reasonably fresh approach to game development that is starting to receive an increased amount of attention from both players and indie development groups. The IGF, for example, has an entire awards category dedicated to excellence in audio, and if you want to see some other games that do credit to the genre, be sure to have a look at finalists such as Synaesthete and Fret Nice for additional inspiration.

These sort of games require work like any other, but the reward potential is great. Just remember that novelty, expression and intuition all have their place in a sound-based game, and that if you want players to have fun, these games have the potential to connect them to the experience in a way that no others can.

FIRST IMPRESSIONS: AUDIOSURF BETA

by James "NightTimeHornets" Etherington-Smith

The concept is simple: allow players to choose a song from their personal music collection, then generate a 3D racing course which reflects the tempo and style of the chosen song. And it just goes up from there.

After playing Audiosurf, simply listening to your music will no longer suffice. Not after you've found yourself in control of a colourful and zippy little futuristic racing craft, hurtling down a beautifully-crafted race track and which twists, bumps and turns in tune with your favourite piece of music. The layout of the race course itself reflects the overall tempo of the song chosen, with upward climbs for calmer moments, plunging drops for the intense areas and some corkscrews thrown in for good measure. On a smaller scale, the track will pulse with the beat of the music, allowing you to visually experience every bass kick, drum snare or synthesiser note. Obviously no music visualisation game would be worth its salt without a liberal amount of seizure-inducing flashing colours, and Audiosurf is more than happy to oblige. The track changes colour as the song progresses and a large amount of psychedelic polygons are hurled at the screen throughout.

If you thought that was more than enough visualisation for one person, then think again. The game challenges the player to collect coloured blocks, referred to as 'cars', which also correspond to the music; softer colours for the slow moments and vivid colours for the fast-paced areas. The player has a 7 x 3 grid in front of their vehicle in which they scoop up these blocks. Matching three or more colours together in any combination will generate points, the more colours together the better. Filling the grid beyond capacity results in a loss of points or the inability to gather blocks for a few seconds. All of this isn't as easy as it sounds when you are plunging down a virtual cliff on an undulating track, but it sure is fun. This leads to the game element that will probably keep you coming back for more - the ability to upload your scores to the internet. Through the in-game menu system you can search for your favourite songs and see who else in the world has logged high scores. You can also add friends who are playing, compare scores and then try to best one another. Audiosurf will even send you an e-mail taunt to notify you that you have been knocked from the top score position of your favourite song. If all that is too much competition for the faint of heart, the game has a relaxed cruising mode in which you can drift down the visualisation highway and just enjoy the scenery.

Audiosurf does a terrific job of keeping things interesting by providing three difficulty modes, each with its own distinct visualisation environment, and an array of different vehicles each with their own speciality and tricks. The game supports two players with two racing craft on the screen, one controlled by the mouse and the other by keyboard. This is great if you have a nagging friend at your house or simply wish to go cross-eyed trying to manoeuvre two vehicles by yourself.

Overall the game is immensely fun and extremely addictive. Fans of all musical genres, from the heaviest metal to the most ear-pounding trance and everything in between, will find themselves at home with the game as the tracks generated always provide an accurate representation of the style. It is almost futile to describe the game with words as it is a visual extravaganza that must be experienced for oneself.

Audiosurf is due to go retail in February. Keep up to date by visiting http://www.audio-surf.com/

PROJECT SCR

A work in progress

by Sven "FuzzYspoON" Bergstrom

Project: Shopping Cart Revenge (known as SCR for short) is a destruction racing game where the objective is to crash into other players. Originally just a minigame, the creators have decided to expand its potential and are now working on a fully-blown title, with an arena-style system for multiplayer and several challenge modes to be added. We won't be seeing any releases just yet, but here's a project story from the lead developer to tide you over for the meantime ...

This project was an interesting find in the shadows of my mind – I always look for something that either doesn't make sense, or has been done before but not in the way my mind explains it. I have always loved games that express an interesting way to destroy something, either subtly or directly. With the likes of demolition derby and all other crash racers, I wanted a game where you can simply drive your heart out. Being my first fully 3D game, I've decided to equip SCR with all the latest tricks in graphics, optimized multiplayer, 3D sound and all the interesting ideas I can pack into a feasible deadline.

Project Description

A crash racer can be a lot of fun, but what would make it that much more fun than your average crash game? More violence? More blood? Or an interesting twist that makes it amusing rather than savage? I found out that it could be a unique vehicle that would both be ridiculous and impossible, or fast and out of control. The fast and crazy has been done in a few games, and I spotted a unique vehicle in the form of a shopping trolley. This, in my mind, was an instant winner. Being a great premise for unique gameplay and interesting single player missions sparked something in the brain part of my head. Imagine riding with your friends in a multiplayer game of shopping trolley mayhem. Throwing groceries at unsuspecting players and flying into walls as you hit into players, knocking others over with your face at high speeds all constitute as ridiculously amusing.

Project Details

The team started out as me and my C++ compiler. I started planning the game design and plotting my code structures in a text file, as well as testing my texture baking and lighting ideas. I noticed some great results so I looked at the next option, how would I have approached map loading and the physics. I started testing with a Newton wrapper that worked easily for what I wanted to test, and ran efficiently on slower machines. I started testing the game with my engine of choice, Irrlicht3D gives me an easy platform to test graphics and game play. So, I started making a prototype in a test map, giving me a handle on testing the game play and how the trolley would "drive", how the trolley would react in the map, and how the graphics would perform at high stress.

The first trolley I tested was a hefty 68 000 polygons, which would surely break the record of "most ridiculous poly count in a game", but luckily I knew the free model I had downloaded was not going to be in my final game. I quickly optimised the trolley to 12 000 polygons, which would do adequately for tests in the engine I was attempting to create. I spawned a couple of them in my test engine, a mere 30 – 40 of them. This was purely testing the engine to see what it could handle, and it gracefully dropped the

frame rate by 10, running with full screen, vertical sync and a full screen bloom shader with a 1024x1024 render-to-texture target, which is larger than the screen resolution I was testing. I was actually surprised that it ran smoothly - though it definitely was on a "good" computer.

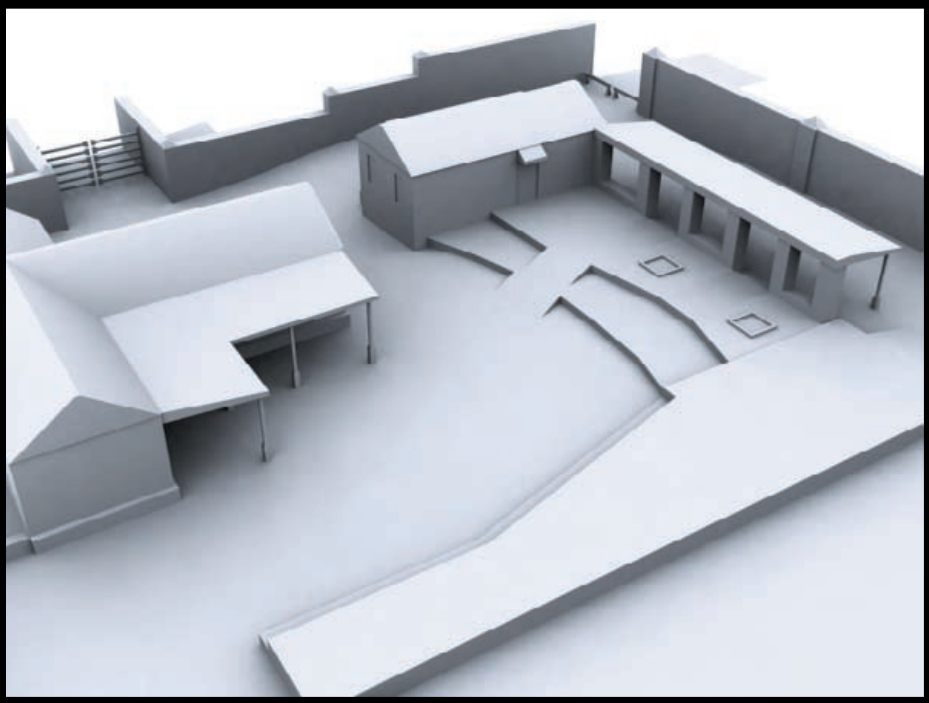
Above the stress tests, I tested the 12 000 poly trolley with all these high settings enabled and got smooth results on a horrid PC as well. I now faced my next problem, multiplayer.

Project Team

I decided a team wouldn't be a bad thing, though I have never really worked on a game with a team of more than two, I decided to see who I could recruit that would prove reliable and helpful, while helping me learn and not giving me code but giving me knowledge. I started talking to a great guy named Varmint, who is now one of the Irrlicht team on the Mac side. Turns out he knows way too much about networking, which wasn't a bad thing at all - in fact, it was just what I needed. As he took time out of his day to explain a lot of useful and great techniques, we eventually decided that he would be helping with the networking and the AI.

The artist, on the other hand, wasn't as coincidentally recruited. I have been talking to ReVerthex quite a while back, when we were frequenting ultimategfx.co.za together. We have had some talks and I have used a few of his great models as tests in my endeavour to find interesting texture mapping techniques. As we discussed SCR, he decided to work on a map that I would include in the first demo as a test map, leaving me more time to tweak the code side of things while he made the map. From then we decided to make a certain number of maps, and he would do a certain amount and I would work on another bunch. This is why he is now a team member, as he continues to make cool maps for the game, and has brought my pitiful trolley into a 3000 polygon made-from-scratch wonder.

The newest member is a friend from the Irrlicht scene, nicknamed Blindside. Amongst forum members there are always guys that stand out with quality contributions, this guy



being one of the really great submitters to the community. Having stood out for effects wrappers, shadowing techniques and even implanting great-looking graphics into not-so-good-looking maps, he certainly raised my interest as a developer. As we discussed the techniques he employs for soft shadowing in Irrlicht, we decided it would be beneficial to have him on the team for effects and eye candy. My tweaked bloom shader would fit his effects for realistic water, and other great effects we planned to employ.

Project Pitfalls

The few things that have tripped me up were so simple in my mind, but ended up costing hours of wondering and waiting. Some of these things were unexplainable, but with some insight from the more professional amongst the beginners, I managed to find code that helped me out. Such pitfalls that took away time of mine were the 3rd person camera. Surprisingly enough, my math did not step up when I couldn't figure out the quaternion and Euler problems I was facing, but a friend on IRC managed to scratch the code into a pastebin that helped me fix the math problems.

Other challenges included large texture files, my test map was really small and square but I was facing a 65 MB file size on the final map. This, of course, was one of the first things I looked into, having a bit of experience with 3D and textures. Lowering the textures to a mere 3 MB with the same level of quality didn't take much, but will definitely help in the development of a fast, good-looking game.

Project Breakdown

This project is purely experimental and for fun. It is not the next game to hit the shelves, it is merely a project to learn from. Creating a game that people can enjoy and can learn from is part of my motivation for ProjectSCR. The open source policy will not change on this project, and all team members are in agreement that more people can learn from it than what we could make out of getting recognition for the game.
We are making this game for the people, as this serves as an example to all the people who feel elitist out there. Somewhere along the line, somebody helped you get somewhere. Let's do something that is not about ourselves for once - it's not like we won't get the credit for it.

DB SAYS ...

Interested in this project? Take a look at the following sites for updates:
http://irrlicht.sourceforge.net/phpBB2/viewtopic.php?t=25288
http://www.owned.co.za/fail

The team members can also be found stomping about on the Internet if you fancy hounding any of them:
http://ReVerthex.deviantart.com
http://newclearfuzzy.deviantart.com
http://thermal.freehostia.com/wordpress/
http://www.krabbit.com

Ultimate Quest

The ultimate postmortem

by Claudio "Chippit" de Sa

Ultimate Quest is a tongue-in-cheek text adventure with a retro take on graphics and a charming sense of humour. It's also really strange. And you can lick sheep. We managed to rope in one of the development duo to talk some more about it. Brace yourselves.

It started in IRC. A normal day, though discussions of the recent and crazier than usual Game.Dev competition rules were abound. Everyone was nursing ways to exploit the 'text-only' limitation for the competition. As is the norm, of course, no-one active in that discussion actually made their ideas a reality. At least, not until Tarryn "Azimuth" van der Byl joined the conversation, caught a whiff of potential for a text-only adventure, and plainly declared that we 'make it!'

Ultimate Quest was born in that moment, and, as it turned out, the only collaborative project I'd ever been involved in that was actually completed. The very conjunctive nature of the project was the largest contributing factor to the completion of the game. Since I didn't have to worry about where the visuals of the game would come from, I was able to devote time to getting the game to work as it should as well as creating the content that was expected from a game of this kind. Azimuth contributed a significant amount of both artistic and literary skill to the project; a large majority of the dialogue, as well as every graphical element in the game is attributed entirely to her talent. The fact that I had feedback in more stages of development than I would normally have had - as well as input into decisions that I would have made myself - was instrumental to the game's development and progress.

Text Parser

After the initial sharing of ideas, basic themes and defining features, I set about concocting a small text-parser demo to see whether I could create one which had the functionality that would be integral for the game. The parser I created was quick and simple, yet fairly effective. It would simply search the input string for known verbs (or their varied synonyms; the final game had many of those, for both verbs and objects), and then find the object onto which the verb will be applied. It then passed the verb over to the object for processing, and the object itself would handle any action if such a response existed for the specific object/verb pair. This text-parser hardly changed its implementation since incorporating it into the game, barring verbs that were added in as time went by. Unfortunately, this implementation had a particular limitation that I was only aware of near the end of development: the parser only searched for the first object it could find, and, in fact, disregarded word order or additional words in the phrase. As such, it could not process commands involving two objects, such as 'Use scissors on sheep'. I worked around this issue by having commands that were reliant on a secondary object take the nearest object in game-space as a parameter, which masked the limitation quite effectively since it was natural to walk towards whatever you want to interact with.

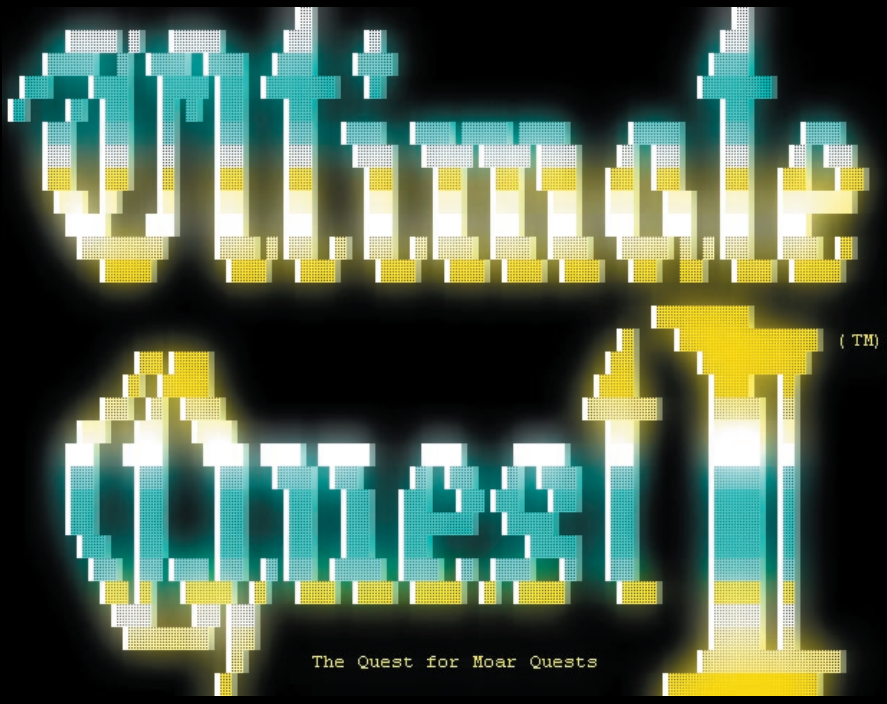
To do this, as well as some other actions in the game, I had to forge a somewhat hasty and sloppy connection between the text parser and the game's physical space, since, initially, the text-parser had no knowledge or link to the graphical elements of the game and could run entirely independently of it. In the final version of the game, the text-parser could find the object that was closest to the avatar, as well as have access to certain specific objects in the scene as necessary.

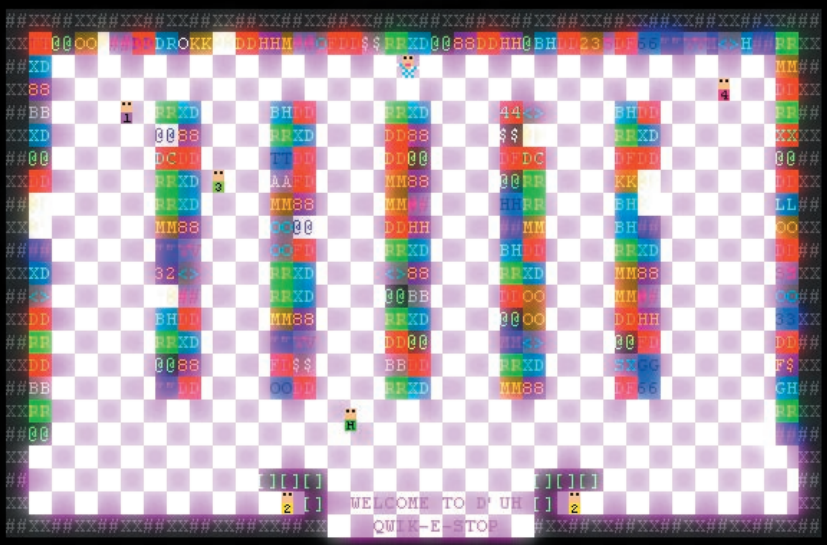
However, despite a few sloppy additions to the parser system near the end of development, it still worked to the degree that it was originally intended, and, since it is the only means of interacting with the game world, I felt that it effectively served its purpose.

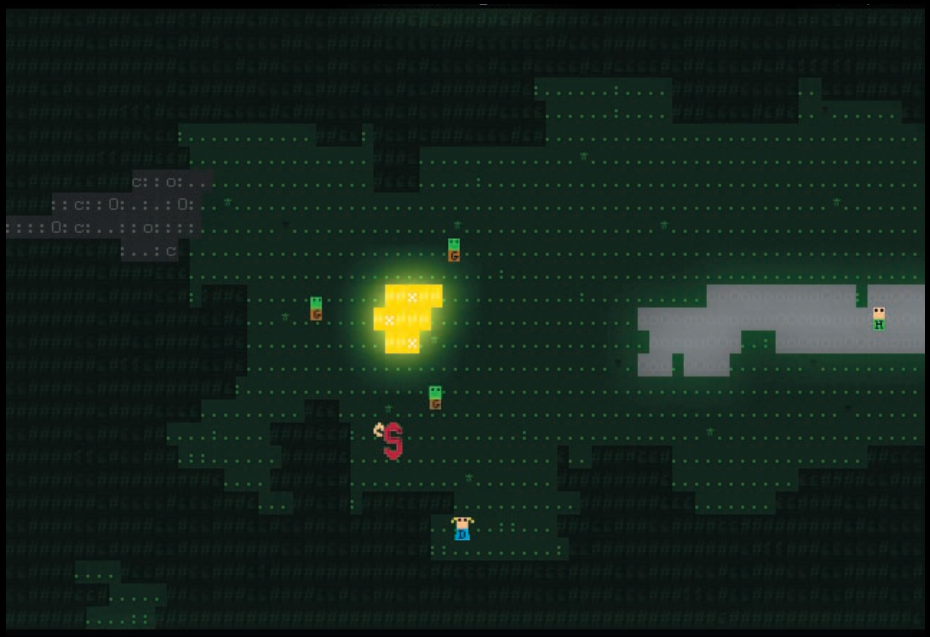
Graphics

One of the first things we discussed was how we would actually represent the game world. Simple text-based bitmap images were a logical choice and would result in more visually appealing graphics, but images are large and static. My original plan was simply to store all graphics as their respective letters and positions, since we were limited to text anyway. This reduced the eventual size of the game quite significantly, as the only graphics resources are the actual 8x16 letters (with the notable exception of a handful of special sprites, all but one of which were also 8x16), and had a beneficial side-effect that allowed me potentially to animate each character on the screen individually. The problem: since I was not actually going to be creating the art myself (if this was the case, I would probably have hardcoded all of it. Not a pleasant prospect, in hindsight), I needed a way to give Azimuth the tools she needed to make them in a format I could import into the game.

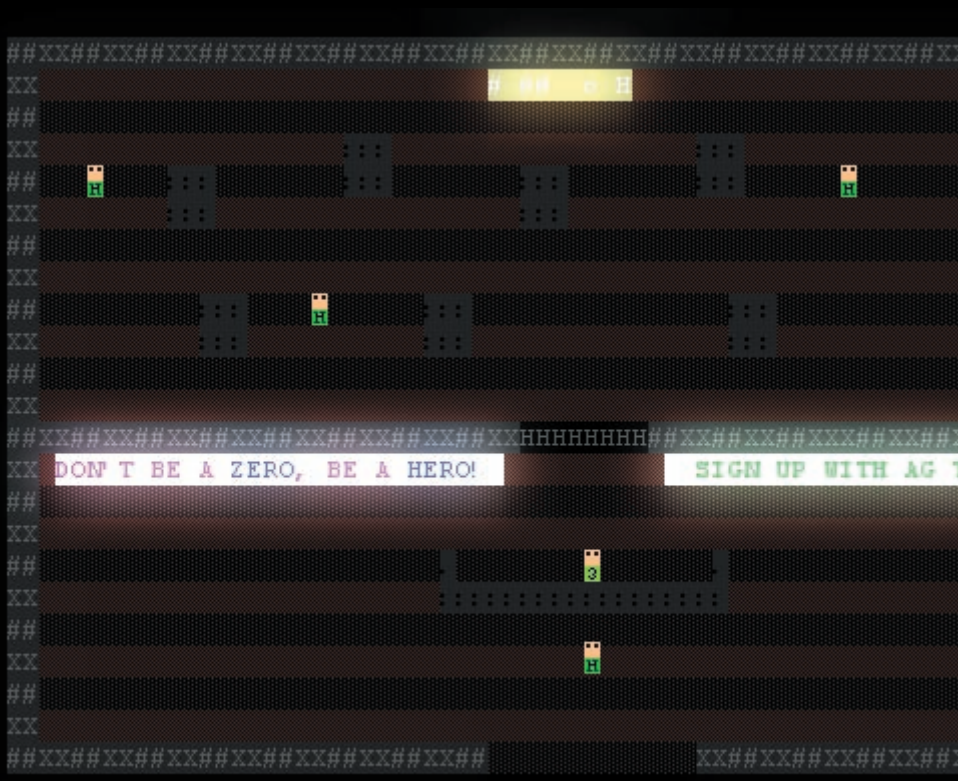
Time constraints meant that I couldn't create a program to do it easily and efficiently, nor would it have the features necessary to make the work less cumbersome. I wasn't actually certain if such programs even existed, so I turned to everyone's trusty friend, Google, where I eventually came upon a program that seemed like it would work perfectly: an open source ASCII art creation program called TundraDraw. It supported 24-bit coloured text and backgrounds and any of the 256 extended-ASCII characters, which seemed perfect for our uses. The biggest consideration now was whether I could get the game to understand TundraDraw's file format without too much hassle. As it turned out, I quite easily dissected the files and was able to load scenes straight from TundraDraw into the game thanks to a simple, logical file structure.







TundraDraw imposed one inherent limitation upon the game, however. While the program was unlimited in vertical dimensions, it had a fixed 80-character width field to work in. 80 characters conveniently translated to 640 pixels in the font size I was working in and, while that was fairly acceptable, I would have preferred to work with smaller characters for a bit more graphical fidelity. Azimuth managed to make the most of the limitations, though, and that is showcased quite clearly from the very first game screen.



The advantages outweighed any limitations, though. The transition effect was one I was proud of, and I rather enjoyed writing the algorithm that would transition any screen of characters to any other, regardless of character count or whether or not the characters match, and then watching it in action. The way I represented everything internally meant that, theoretically, I could transition between any visual elements with any conceivable effect. A single letter could transform into a screen full of characters and vice-versa. Because it occasionally became rather tedious to see letters fly for the umpteenth time, some of the transitions and animations were cut for aesthetic reasons, but the basic effect still remains in major scene transitions and in how message boxes appear only to have the characters fall off the screen when the player is done reading.

Ultimately, and not disregarding a spontaneous decision to apply a post-process bloom filter to the graphics to create glowing, stylized text, the overall graphical feel and style of the game was consistent and visually acceptable. It contributed well to the complete package.

Oversights, problems and lacking features

Ultimate Quest was always intended to be a parody game, poking fun at whatever it could, including at itself. This occasionally allowed us to simply 'write-off' some minor features that we couldn't complete in time and that most games would be expected to have. There were some glaring omissions that were difficult to glaze over, such as the lack of saving and loading capabilities (offset somewhat due to the length of the game), and, most noticeably, the lack of sound that would have made the game considerably more complete.

Near the end of development, I also discovered that a few users were having problems with the game. At the time, I didn't really bother trying to isolate the problem since it appeared to be happening sporadically and would detract from time that I could spend on completing the project. After the competition was over, and after the game had started receiving a little more attention, I dedicated some time to rooting out the cause of the problem (only occurring on Vista systems, hence why I had not noticed it in my own tests), and eventually traced it to an encoding error mysteriously embedding itself in a .NET function call that would fail on Vista systems in certain rare cases, which I duly fixed.

As a whole, though, the development cycle of Ultimate Quest flowed smoother and quicker than anything I had done before. For less than a month's work, Azimuth and I had created a fairly complete (albeit short) adventure title, made completely out of text, that had succeeded in doing what it was intended to do.

DB SAYS ...

Ultimate Quest is a quick and entertaining play which never hesitates to poke fun at the adventure game genre. If you're interested in looking at a satirical game that doesn't take itself too seriously, check out:

http://gamedev.openhazel.co.za/filecloset/download.php?id=286

SUMMIT OF ACHIEVEMENT

Independent Games Festival 2008

By Claudio "Chippit" de Sa

The Independent Games Festival is an annual festival held at the Game Developers Conference, originally founded in 1998 as an initiative by the CMP Game Group, who are also responsible for Gamasutra.com and Game Developer magazine.

The 2008 festival, 10th in the history of IGF, will be held at the Moscone Center in San Francisco on 20 February 2008, and will award prizes totalling $50 000 split between the main, mod and student competition categories.

The Main Competition comprises of the following seven categories:

- **Audience Award** ($2 500). Previous winners include *Savage: The Battle for Newerth*, *Alien Hominid* and *Castle Crashers*.
- **Best Web Browser Game** ($2 500), which replaced separate downloadable and web game categories which were awarded in 2004 and 2005. Past winners include *Samorost 2* and *Dad 'N Me*.
- **Technical Excellence** ($2 500). Winners include *Savage: The Battle for Newerth*, *Alien Hominid*, *Darwinia* and *Bang! Howdy*.
- **Design Innovation Award** ($2 500). Notable winners include *Bontãgo*, *Gish*, *Wik and the Fable of Souls*, and *Everyday Shooter*.
- **Excellence in Audio** ($2 500). Past achievers include *Everyday Shooter* and *Weird Worlds: Return to Infinite Space*.
- **Excellence in Visual Art** ($2 500). Including *Castle Crashers*, *Darwinia*, Wik *and the Fable of Souls*, and *Alien Hominid*.
- And, finally, the **Seumas McNally Grand Prize** of $20 000. Previous notable winners of this award include *Aquaria*, *Darwinia*, *Gish*, *Wik and the Fable of Souls*, and *Savage: The Battle for Newerth*.

Here's a roundup of all the finalists in the 2008 competition.

Audiosurf

If you cannot contemplate what Wipeout would be as a rhythm game, Audiosurf is just that: a racer that lets you ride a procedurally generated embodiment of your music, with all the highs, lows and traffic synchronized perfectly to the music.

Nominated for: Seumas McNally; Excellence in Audio; Technical Excellence

http://www.audio-surf.com/

Axiom Overdrive

Set within an asteroid mine, players are challenged to utilize unique physics-based gameplay in a 3D omni-directional scrolling action/puzzle game.

Nominated for: Technical Excellence

http://www.axiomoverdrive.com/

Battleships Forever

Using innovative new techniques and ideas coupled with long-standing RTS systems, Battleships Forever is a real-time space combat simulator sporting ships that you can literally break apart piece-by-piece.

Nominated for: Design Innovation

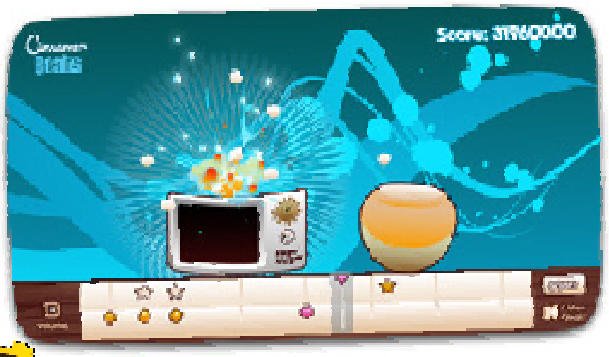
http://www.wyrdysm.com/

Cinnamon Beats

An interesting twist on music-based games, Cinnamon Beats is a puzzle game where the goal is to use the physics engine to generate your own music.

Nominated for: Excellence in Audio

http://secretexit.com/games/cinnamonbeats/



Clean Asia!

With a clean, line-based representation, Clean Asia! is a vertical shooter with innovative new features, and two completely different ways to play the game.

Nominated for: Excellence in Visual Art; Excellence in Audio

http://www.cactus-soft.co.nr/

Crayon Physics Deluxe

A unique physics-based puzzle game (an improvement on the original Crayon Physics prototype) that will take any drawn figure and bring it to life using its remarkably flexible physics engine.

Nominated for: Seumas McNally

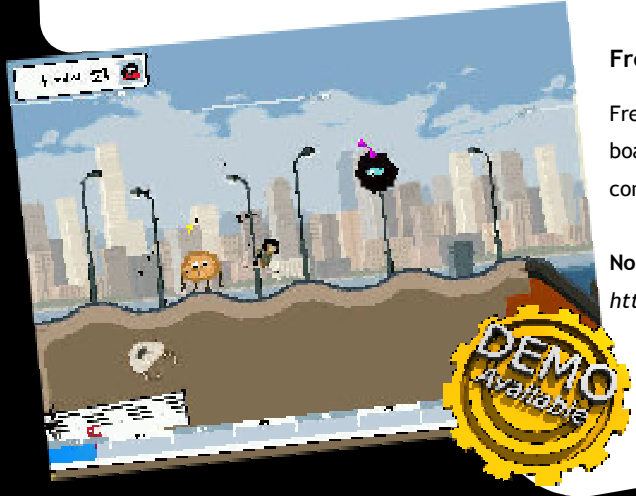
http://www.kloonigames.com/crayon/

Fez

While initially appearing to be a standard 2D platformer, Fez adds a new dimension to the genre by allowing the player to flip the world and project it into 2D from a different angle, allowing the player to progress in a unique and innovative way.

Nominated for: Design Innovation; Excellence in Visual Art

http://www.kokoromi.org/fez



Fret Nice

Fret Nice is a rhythm game disguised as a platformer, and boasting the unique trait of being designed for use with a guitar controller.

Nominated for: Design Innovation; Excellence in Audio

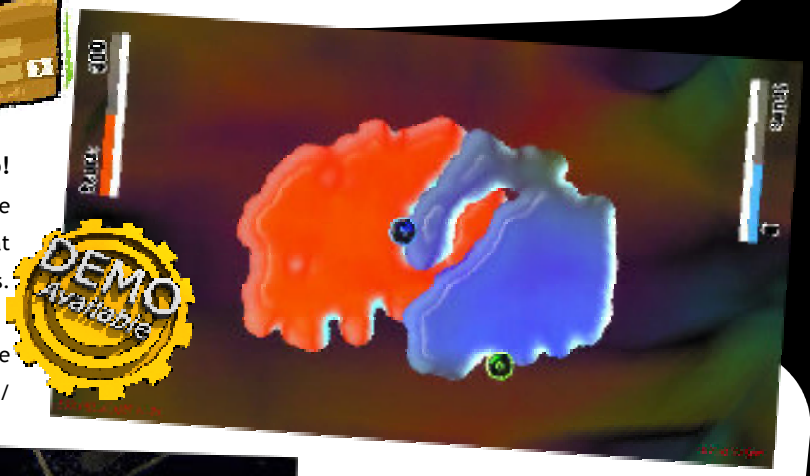
http://www.fretnice.com/

Globulos.com

Globulos is an online multiplayer mini-games environment, boasting over 20 different games for 2 or 4 players.

Nominated for: Best Web Browser Game

http://www.globulos.com/



Goo!

Goo! is a single or multiplayer game inspired somewhat by the board game Go, except that players do battle with physically simulated liquid armies.

Nominated for: Technical Excellence

http://goo.pillowfortgames.com/

Gumboy Tournament

A follow-up to the critically acclaimed Gumboy: Crazy Adventures, Gumboy Tournament adds a much requested multiplayer feature which allows competition against AI or others players on LAN, split-screen or over the internet.

Nominated for: Technical Excellence

http://www.gumboytournament.com/

Hammerfell

Appearing mysteriously out of Russia, Hammerfall is a 2-dimensional, combat-oriented action game with interesting physics-based combat mechanisms.

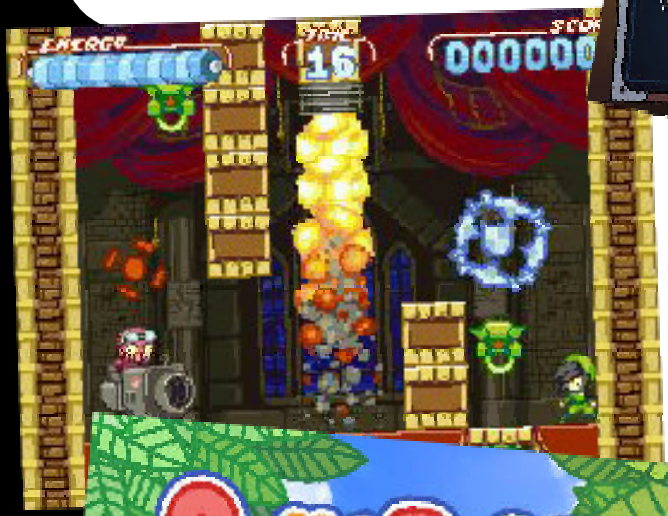
Nominated for: Seumas McNally; Excellence in Visual Art

Iron Dukes

A highly stylized and humourous seafaring adventure where the goal is simply to accumulate as much treasure as possible through your pirate endeavours, looting enemies and gaining better equipment along the way.

Nominated for: Best Web Browser Game

http://onetonghost.com/



Noitu Love 2: Devolution

Noitu Love 2 is a frantically paced action-platformer with an innovative new mouse-based control scheme.

Nominated for: Seumas McNally

http://www.konjak.org/

OokiBloks

OokiBloks is another sound-based action puzzle game, where the player controls a monkey named Ooki in a colourful and musical world.

Nominated for: Excellence in Audio

http://www.ookibloks.com/

Snapshot Adventures: Secret of Bird Island

Snapshot Adventures is a casual photography game where the premise is to travel across the world and collect equipment, photographs of exotic birds as well as unravel a mysterious disappearance.

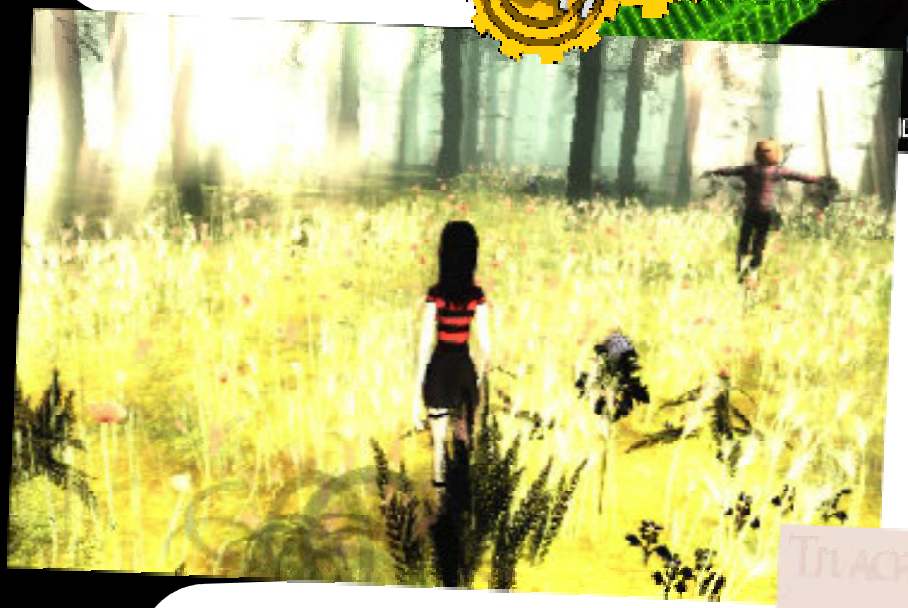
Nominated for: Design Innovation

http://www.largeanimal.com/games/deluxe/snapshot-adventures-secret-of-bird-island

Synaesthete

Synaesthete is a unique rhythm game that attempts to fuse graphics and audio and create a synergised experienced of colour and music.

Nominated for: Excellence in Visual Art
https://typo3.digipen.edu/index.php?id=986



The Path

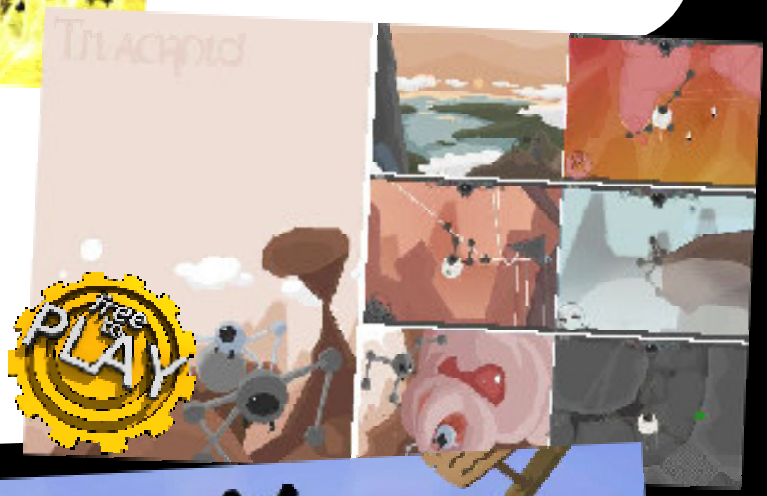
The Path is a visually stunning horror representation of a common fairytale, Little Red Riding Hood, promising a unique, immersive experience.

Nominated for: Excellence in Visual Art
http://www.tale-of-tales.com/ThePath/

Tri-Achnid

Tri-Achnid is a platformer with a unique control scheme that entrusts the care of a three-legged spider and his eggs to the player.

Nominated for: Best Web Browser Game
http://triachnid.com/



World of Goo

World of Goo is a charming physically-simulated construction game, based on the original prototype on Experimental Gameplay.

Nominated for: Seumas McNally; Design Innovation; Technical Excellence
http://2dboy.com/