ISSUE 8 2006

DEV.MAG

CREATE • DEVELOP • EXPERIENCE

THE GAME MAKER'S APPRENTICE!

Interview with co-writer Jacob Habgood

XNA IS HERE!

MOBILE GAME DEVELOPMENT: TOOLING UP WITH NETBEANS

SOUTH AFRICA'S FIRST GAME DEVELOPMENT MAGAZINE

THE GAME DEV. POWERHOUSE

REVIEWS : PYGAME. FEATURE : XNA. GAME DEV AND DEV MAG AT RAGE!. BLENDER TUTORIALS

Cover: The Game Maker's Apprentice http://book.gamemaker.nl/

CONTENTS

ED'S NOTE

Hey all, due to fiendish circumstances, our editor has been out of action for this month. No fear, however, as Dev.Mag will proudly march on regardless! We have some special stuff lined up for you this month, including the opening of a new Projects section in the mag.

Something that we've been thinking about recently is the idea that some of our content can follow the "lead by example" philosophy – the idea that entire games, rather than just components of their build, could be examined and evaluated as they progress.

To this end, Dev.Mag has incorporated two new articles this issue, one of them being an offshoot of last month's feature known as a postmortem (in which a game's creators reflect upon the successes and failures of completed projects) and a new series that follows the building process of the game Nonex 2, with the author highlighting triumphs, challenges and downright pitfalls as the game progresses from chicken scratch to completed product.

We hope you like what we've done with this issue, we've done our best to revamp a lot even given oppressive circumstances, and I hope this is reflected in the enjoyment that you, the readers, gain from this jam-packed edition.

Game on!

Deputy Editor
Rodain "Nandrew" Joubert

DEV.MAG
CREATE • DEVELOP • EXPERIENCE

THE STAFF

RANKING OFFICER
Stuart "GoNzO" Botma

SECOND IN COMMAND
Rodain " Nandrew " Joubert

DESIGN SQUAD
Brandon "CyberNinja" Rajkumar
Paul "Higushi" Myburgh

CEREBRAL SOLDIERS
Simon "Tr00jg" de la Rouviere
Ricky "Insomniac" Abell
William "cairnswm" Cairns
Bernard "BurnAbis" Boshoff
Danny "dislekcia" Day
Andre "Fengol" Odendaal
Yuri "knet" Oyoko
Heinrich "Himmler" Rall
Matt "Flint" Benic
Luke "Coolhand" Lamothe

WEB WARRIOR
Claudio "Ch1ppit" de Sa
Robbie "Squid" Fraser

WEBSITE
www.devmag.org.za

To join, make suggestions or just tell us we're great, contact: devmag@gmail.com

This magazine is a project of the NAG Game.Dev forum. Visit us at www.nag.co.za

DIGITAL STOMPIES

NEWS

Books: some good reading ...

http://www.next-gen.biz/

Next Generation has assembled a collection of about 50 books related to game development that are recommended for anybody going into the field of game creation, or anybody who's already in the game industry, for that matter. Topics range from theory to design practice and even include sociological texts, with an aim of accommodating absolutely everyone in the business, up to and including lawyers and part-time testers. Browse through this handy little library – you could probably find something that takes your fancy.

Swiss army chainsaw

http://www.gamasutra.com/

This Gamasutra feature is a handy little guide to tool development, what challenges to expect, and what to do while dealing with them. A great guide for anybody who wants to improve the usability of their products (or games) and learn how to work more efficiently, especially with other people. Even if you're just a lone ranger doing bedroom development, it helps to know when to smile and "fend off bad voodoo".

Moondance releases Independent Games Vol 2

http://www.moondancegames.com/games/

Moondance Games, media sponsor of the Independent Games Festival, has just retailed its second compilation pack of indie games featuring titles from the festival. Game titles include Cute Knight (Kishi Kawaii), Tube Twist, Unipong, Morning's Wrath and many others over a wide range of genres. For indie gaming at its finest, take a look at getting your hands on this set. It sports 21 impressive indie titles and is available for $19.95 from Amazon.com.

How game artists get their jobs

http://www.gamecareerguide.com

Are you a budding 2D/3D artist, environment designer or animation specialist looking to get into the industry? Take a peek at this 8-page feature on Game Career Guide, which takes an in-depth look at your portfolio requirements, how to approach companies for a job and just what sort of qualities companies are looking for in potential employees.

Talent can bring you most of the way, but a little smart thinking still needs to be applied when you're in such a competitive environment, making this article invaluable for those in the field.

Quantum leap awards – RPGs

http://www.gamasutra.com/

Much like their previous feature on revolutionary first person shooters, Gamasutra has asked its readers what they consider to be the game which marks the greatest leap made in the RPG genre.

With so many to choose from, some interesting results have reared their heads, along with actual quotes from readers explaining why they felt like they did regarding their chosen game, and just how much it contributed towards changing the RPG world. Consists of the top five games and several honorary mentions..

Mobile game contest

http://www.actionscript.it/mobilecontest/mob_rules.html

January 29, 2007. A new date to diarise, marking the deadline for a brand-new mobile development competition run by mobile.actionscript.it. The premise – make a game, any game, using either Flashlite or J2ME.

Prizes are awarded for each category of tool used, including books, Flash Studio 8 and, naturally, several high-end mobile phones.

This is the second contest from the organisation, the first having brough forward entries of an exceptionally high quality and from all over the world. Time to put some more local names in there ...

Blog: Tales of the rampant coyote

http://www.rampantgames.com/

Blog hunters, hark! Here's another indie gaming blog to whet your appetite, complete with a particularly interesting post made entitled "You can't design fun on paper". The author takes an in-depth look at design documents, how they work as a "second brain", how they sometimes don't work and how getting too detailed too quickly could be a danger.

An interesting perspective that stands contrary to most established ideas, coming from an established game developer.

A guide to XNA development

http://learnxna.com/pages/XNABook.aspx

For those who want to learn a little bit about using XNA, this isn't too bad a site to be looking at. LearnXNA.com has XNA tutorial videos, news posts concerning the XNA toolkit and its own downloadable book, currently standing at four chapters, which strives to cover an ever-greater variety of topics as new chapters get released. Stop by here if you fancy learning a little about this exciting new tool, covered extensively in this month's Dev.Mag feature.

FEATURE

It's all about XNA!

I am very excited to tell you about XNA Game Studio Express; Microsoft's offering to students and hobbyist game developers for building games on Windows AND the Xbox 360.

The XNA GSE (Game Studio Express) is a set of tools for building managed games together with the XNA Framework. XNA GSE is based on the free Visual C# Express 2005 and to install game studio on a desktop, users will need Visual C# Express 2005, the latest DirectX runtime (or the August 2006 DirectX SDK if users want to use the DirectX audio creation tools) and the XNA Game Studio Express plug-in (http://www.microsoft.com/xna).

Visual C# Express 2005 can run alongside Visual Studio 2005 and will not interfere with your other development projects. The game studio only supports C# but you can use the XNA Framework in any .NET language (although you won't then have the visualization tools that make the XNA GSE so worthwhile).

The XNA Framework is a completely different set of technologies from MDX (Managed DirectX) 1.1 and 2.0 and provides more functionality to help game developers create games productively (although there are similarities because it is still based on DirectX). MDX 2.0 beta has been unchanged since the DirectX April 2006 SDK release and there will be no further changes nor will it ever be released officially. The MDX 2.0 assemblies will be removed from the DirectX SDK once the XNA Framework beta is released.

(Taken straight from the Microsoft website)

Games built with managed code are not slower than game written in C++ because they are just-in-time (JIT) complied into native code when it is initially loaded by a process, prior to execution, and this allows hardware-specific optimizations unique to the PC and Xbox 360 architectures. Only the most intensive games would benefit from being written in C++. The three most notable features when using the

" XNA Game Studio Express; Microsoft's offering to students and hobbyist game developers for building games on Windows AND the Xbox 360."

XNA GSE (which, at time of writing, is currently in beta) are device creation, game loop and game components. As soon as you start your project you can run your game and see the default cornflower blue background that has become the hallmark

of the DirectX tutorials. You don't need to spend time writing code just to get a handle to the graphics device; and you can manage the device's properties at design time through a property window.

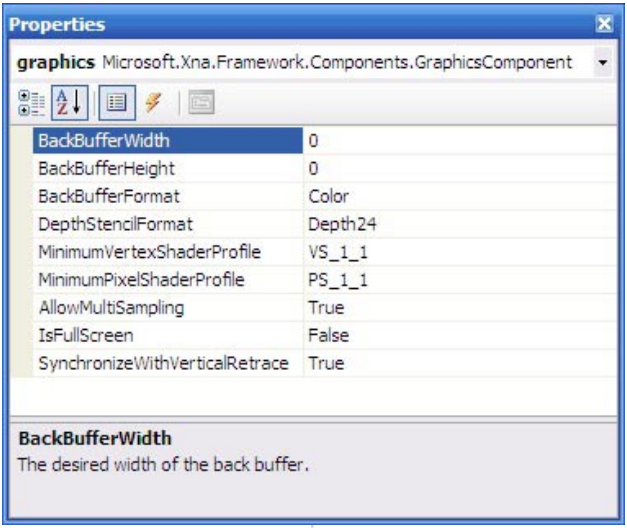
There's also a built-in game loop done for you so that your first lines of code are related to your game concept and not to the mechanics involved in making it run. Using properties in the designer you can even set a fixed time step (used to create a framerate) and sync it to the vertical retrace if required (don't have worry if you don't understand what I'm talking about, the power of the toolset is that the best defaults have already been set for you).

Thirdly (and my favorite feature) are game components, which allow you to break up your game into physical and logical parts and to manage them like you would custom controls on a windows form. Drag-and-drop your own game components or ones built by third parties onto your game and they're executed and managed automatically with properties developers can manage at design time. Game components can be connected together by marking relevant properties in the component as public and assigning them to other game component properties in the visual designer. Game components help move technical designs to a Service Orientated Architecture by being loosely coupled and autonomous.

There are lots of opportunities with the XNA Framework for developers who enjoy building their own frameworks and engines. Developers can focus on building unique game components (like physics, scene management or game-genre specific helper functions) which will plug easily into other game developers' projects with affecting or dictating how game developers make games. Now for the bad news – at the time of writing this, you can't develop games for the Xbox 360, that's coming out when XNA goes live (for security reasons Microsoft does not release beta software on the console). Game developers wanting to run their games on the Xbox 360 platform will have to join the Creator's Club (estimated at $99 a year) and download and install the XNA Framework on their console.

The XNA Framework makes use of a custom, native implementation of the .NET Compact Framework 2.0 CLR on the Xbox 360; debugging on the console is supported through a remote debugging connection from a Windows desktop running XNA GSE. Sharing your newly developed games on the Xbox 360 will also not be immediately available. Users wanting to play your

Below: Graphics Device



FEATURE

games will also need to have an active subscription to the Creator's Club and have the XNA Framework installed on their consoles. They will also need XNA GSE installed on a connected desktop because sharing of binaries is not supported and they will have to compile your source code themselves to debug on their box.

Eventually, you'll be able to distribute you game to other Xbox 360s, opening up a unique publishing avenue which will democratize game development on consoles (Another direct take from the Microsoft website), and Dev.Mag will do a write-up of developing games on the Xbox 360 when it becomes available.

Below: Sample code

```
namespace Simple3D
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    partial class Game1 : Microsoft.Xna.Framework.Game
    {
        public Game1()
        {
            InitializeComponent();
        }

        protected override void Update()
        {
            // The time since Update was called last
            float elapsed = (float)ElapsedTime.TotalSeconds;

            // TODO: Add your game logic here

            // Let the GameComponents update
            UpdateComponents();
        }

        protected override void Draw()
        {
            // Make sure we have a valid device
            if (!graphics.EnsureDevice())
                return;

            graphics.GraphicsDevice.Clear(Color.Black);
            graphics.GraphicsDevice.BeginScene();

            // TODO: Add your drawing code here

            // Let the GameComponents draw
            DrawComponents();

            graphics.GraphicsDevice.EndScene();
            graphics.GraphicsDevice.Present();
        }
    }
}
```

MANAGE YOUR PROJECT, NOT YOUR CODE

TORQUE X

Other companies are also making use of XNA GSE and the XNA Framework to build feature-rich tools for game developers. For example Garage Games (creators of Torque Game Builder and games like Marble Blast and Puzzle Poker) are porting key portions oftheir 2D and 3D technology and tools to XNA so that enthusiasts, indie developers, educators and commercial studios can develop games for the Xbox 360 using their tools (www.garagegames.com/products/torque/x/). Their tools include an environment for non-developers to make games, artist-friendly shader support, physics and collision detection.

Various communities devoted to XNA have also popped up with forums, tutorials, samples and articles on the subject. Microsoft also has a vibrant forum on MSDN (http://forums.microsoft.com/MSDNShowForum.aspx?ForumID=882&SiteID=1) which the XNA project team and various MVPs (Microsoft's Most Valued Professional) monitor and help with bugs, issues and problems.
The XBOX 360 Homebrew community... (http://xbox360homebrew.com/) even has a competition to build a game using the XNA Framework with first prize being one year's subscription to the Microsoft Creator's Club.

So to summarize the XNA GSE and XNA Framework are more productive to work with and easier to get into than previous versions of DirectX. The use of the XNA Framework in other professional toolsets and engines means wide support for game developers at any technical level. The game components provide powerful design and functionality possibilities, especially for building your own engines.

There is a growing community base using the technology and the ability to develop games for the Xbox 360

TORQUE X

will open new avenues of creativity, allowing game developers to reach eager audiences and bringing about the potential for building a viable financial revenue.

FENGOL

"The game components provide powerful design and functionality possibilities, especially for building your own engines."

FEATURE

Pygame

Some of you out there have probably used the Python programming language before. It's great and easy to use, provides excellent cross-platform support and automates a lot of tasks which programmers usually have to grieve over. But is it a suitable game development language?

Well, this website provides a definite "yes" as an answer – backed by the power of an awesome little tool known as Pygame.

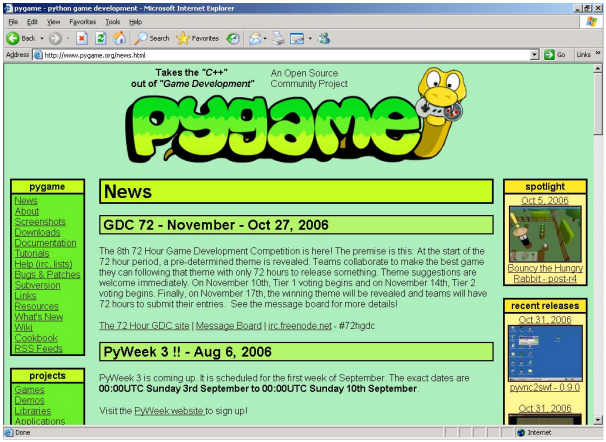
Pygame is a Python module which dedicates itself exclusively to the development of games. All the resources which you need to establish Win32- or Unix-based game applications are at your disposal, including tools for graphics, sound, timers and everything else that just isn't accessible through the core language. Combined with the ever-handy py2exe module (for compiling the code into a runnable, standalone program), Pygame has allowed its users to produce games of an impressive quality where none seemed possible before.

The website dedicated to this particular kit doesn't come entirely unequipped, and it's complete with the standard news reports, tutorials and all-important downloads which a user would normally expect. However, one is truly blown away by the list of projects in development as well as completed games which get released on a regular – sometimes daily – basis, and the competitions as well as "Pygame Weeks" that get run frequently serve as a motivation to draw in more enthusiasts and curious developers. An important thing to note for Python users is that as of writing this review, the latest version of Python did not have a corresponding version of Pygame which it supported. Should this be the case with you, revert to the Python 2.4 version and keep developing with that until Pygame support is brought forward. Take a look at this website, and make sure you get your hands on that Pygame module! After that, it's happy game development all the way.

NANDREW

Last known update: Continuous

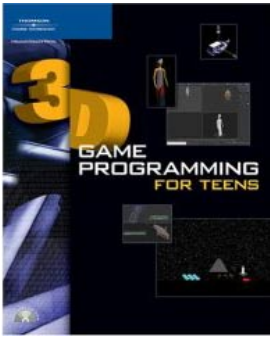
www.pygame.org



REVIEW

3D Game Programming for Teens

REVIEW



Author: Eric D Grebler

Title: 3D game programming for teens

Publisher: Thomson Course Technology

Category: Development/Programming

User-level: Beginner/Intermediate

ISBN: 1-59200-900-X

Average Price: R 350 - R 380

Available from: Exclusive books , Larger book retailers

Without experience in game development, programming and 3D graphics, this book will teach you to create a 3-dimensional game in as long as it takes you to understand and complete the training that this book offers. In short, the book covers everything in the game development process, from the beginning stage of acquiring the knowledge and the skill involved, to creating a full 3D game. As a relatively cheap computer book, it was quite surprising to see the range it covers, all with examples and free software to learn with. The book deals its content in 3 sections, the first being a theory based section about the tools, skills, and a brief history of games. In the second part, the actual understanding and programming is taught making mini-programs along the way, and finally in the last part creating a game based on the mini-programs you already have created.

The book itself is a good start for game developers who want to make that step out of game maker into the programming of full 3D games, even 2D games of course as it goes through all the phases using tools such as Blitz3D, 3Dstudio Max 8 and Corel Draw, which all come with the book as a starting point to creating functional 3D games. The trial versions of the software are quite functional considering the price of the book, and the content starts at the very beginning of game creation, all the way to creating a first person style game with the free tools, all while teaching the principle of creating games in 3D, from scratch.

With an interesting history on games and the development of games, it goes on to explaining gaining skills game developers usually have, such as programming, graphics, team, math as well as communication. Diving into game design theory and teaching with some examples, it goes from there straight into the anatomy of a game, and subsequently starts teaching you to use the tools provided to create a game. It covers 3D modelling, 3D theory, graphics theory (such as transparency, textures and coloring) and on the game side fills in the details regarding lights, cameras, game control, collision detection, sounds and music, physics, timing and other game creation techniques, all instantly codeable into the blitz3D engine supplied. The later chapters cover programming, the fundamentals behind creating working code and programming concepts, and finally using the rest of the knowledge to create a fully functional game in the last chapter of the book.

This book is recommended for beginners who are interested in creating 3D games, but dont have the time or resources to learn one of the very technical languages or frameworks 3D games usually use, and you as a reader can start creating games right away.

At beginner level this book is a worthwhile buy, but if you have had experience coding games in any language, and understand theory behind creating games, then this book might be a little primitive for your taste. As a game maker enthusiast who would like to get a 3D game under their belt and grab the reins right away, this book is a great start, and provides the tools to get going.

FUZZYSPOON

The Game Maker's Apprentice

SPOT LIGHT

For those who are new to the world of Game Maker, or who have more experience but want a more detailed look at what this package can offer, a new book entitled The Game Maker's Apprentice serves as a great resource for not only people looking to learn Game Maker, but also those who want game design hints and tricks in general. TROOJG has a chat with Jacob Habgood, co-writer of the book alongside Mark Overmars, so that our mag can pick his brain a bit...

Jacob Habgood

What were your intentions regarding The Game Maker's Apprentice?

We wanted to create an engaging and accessible beginners guide to game development. There are a number of books out there that claim to offer this, but we didn't feel it had really been achieved yet. We went to great lengths in order to create something that people would really enjoy owning, with colourful artwork and games that were actually fun to play. We were very pleased with the way it turned out and it ultimately exceeded both our expectations.

Are there any useful things in it for the expert Game Maker user?

We hope so. The game design chapters in particular are the combination of years of experience that we think would be useful to everyone. It's already been used as part of a Masters level course in the UK and we know it is being used in other university courses too.

What version comes with the book? 6.x and registered or not registered?

6.1A unregistered. We're currently considering hiring a blimp to display that answer on.

What market is the book aimed at?

Beginners of all ages, although you do need to able to read the book. To quote the introduction: "This book is not specifically for the young or old, but anyone who loves computer games and wants to have a go at making them for themselves. We've all painted a picture, written a story, and made a wobbly piece of pottery at some point in our lives, so it's now time to embrace the art form of the future and try making computer games too."

" We went to great lengths in order to create something that people would really enjoy owning, with colourful artwork and games that were actually fun to play."

What was the best part of writing the book?

The best part is actually getting feedback that shows you that you've created something that people appreciate. It makes all the blood, sweat and tears that bit more worthwhile.

Any planned future collaborations?

We did consider writing a follow up at one point, but it's difficult to see how either of us would ever get the time.

What you do for a living?

I'm currently writing up my PhD thesis in "The Effective Integration of Digital Games and Learning Content". It marks the end of a three-year career break for me, and I'm returning to the games industry as a Project Lead for Sumo Digital in a couple of months (Virtua Tennis, Outrun 2006:Coast 2 Coast, Broken Sword). Sumo are doing lots of cool stuff at the moment and I have some interesting projects lined up for when I get there.

What games have you guys developed?

Micro Machines (2002), Infogrames

Hogs of War (2000), Atari Europe S.A.S.U.

Premier Manager 2000 (2000), Infogrames UK Ltd.

Actua Soccer 3 (1998), Gremlin Interactive Ltd.

N2O Nitrous Oxide (1998), Gremlin Interactive Ltd.

Actua Soccer 2 (1997), Gremlin Interactive Ltd.

Judge Dredd (1997), Gremlin Interactive Ltd.

Re-Loaded (1996), Interplay Entertainment Corp.

Greenies (1995), F1 Licenceware

Do you guys still develop games?

Certainly will be.

Favourite next-gen console? Why?

Has to be the Wii, despite the name. Nintendo games rock.

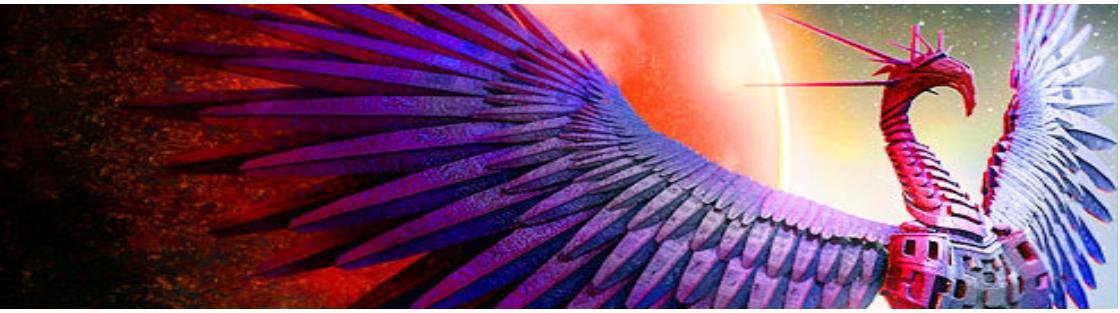
What impact does products like Game Maker have on the game development industry?

Interesting question. Probably not a great deal to date, but some of us are trying to change that. Have a look on Gamasutra or GameCareerGuide and search for articles by Jacob…

Where do you see the game development industry going in the future?

India, probably. :-) No idea what will happen really, but it would be nice to see more of an innovative casual gaming scene. Cheaper, shorter games where developers are able to take risks with gameplay. XBox live is an excellent idea in principle, and it'll be interesting to see how it plays out. Fewer MMORPGs too – if you have a hole in your life that big then you should do something constructive like developing your own games instead. Game Maker beats being an Elf any day of the week!

TROOJG



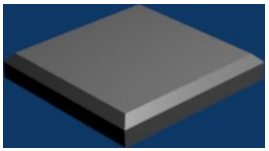
BLENDER TUTORIAL PART 4

This part of the Blender tutorial will use all the skills we've learned before to create a new scene from scratch. If you are not familiar with everything we've covered before, I advise you go back and do the previous tutorials again. The scene we will be making is a checkerboard.

The board

We'll start off with the board itself. For the sake of detail, we'll be modelling the blocks individually. Delete the original cube, switch to top-view, and create a plane. The default 2x2 size will suit us perfectly.

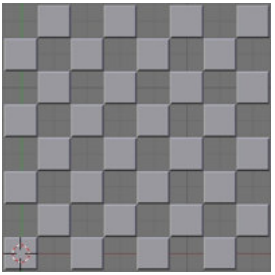
To give out block some depth, we'll be using a modelling technique known as extrusion. Extrusion essentially takes the selected faces and pushed them outward, giving them depth. Switch to side- or front-view and zoom in closer to the block. In edit-mode, press 'E' or select Extrude from the Mesh menu to bring up a pop-up menu. Select Region from the pop-up menu to start the extrusion process. Move the mouse upwards 0.2 units and click to accept the changes. We'll also taper the edge a bit, to make it look smoother. To do this, extrude the top vertices again, this time 0.1 units upwards. Accept the changes and then scale them down to 0.9 size. It should look like the following image when you're done.



A checkerboard block. Not much.

Now is a good time to mention the two different duplication methods available in Blender. Pressing Shift+D, or selecting Duplicate from the Object menu, will make an exact, but independent double of the object. Pressing Alt+D, or selecting Duplicate Linked from the menu, will create a duplicate that is linked to the original. Therefore any changes you make to one of the copies will affect all the duplicates. This includes changes in the model structure and texture, but not scaling and positioning.

To create the whole board, we'll need to do some duplication. Bear in mind that every alternate block is a different colour, so we'll need to be careful how we duplicate the blocks. Select the block in object mode, and create linked duplicates of the block until you've filled every other block of an 8x8 grid. Now create an ordinary duplicate of one of those blocks, and place it in one of the

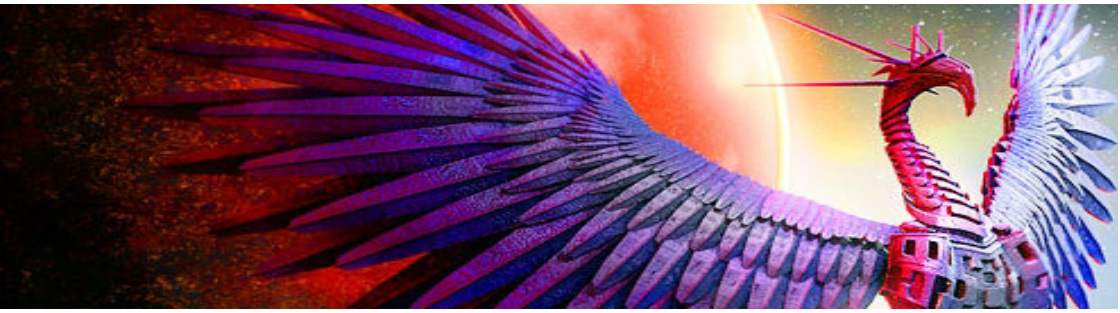


Half the board pieces are not filled in.

open spaces in the board. Now create linked duplicates of that block until the entire board is full.

The pieces

Now we have a board, but what about the pieces? We'll be making simple round pieces to populate our board, so start off by making a circle in Top-View. 32 vertices should suffice. Place the piece over one of the existing blocks and scale it until in fits neatly inside the block. I found a 0.6 scale value worked perfectly. Then in front view, place it so that it rests just on top of the block. Extrude this block upwards 0.2 to give it depth. You'll notice that there is no 'Region' option when you extrude these, so select 'Only Edges' in this case. This happens because the circle Blender makes isn't filled by default. We'll fill it in once we're finished with it.



Now we're going to give this block a little notch in the middle. This is easier if you activate wireframe mode with the 'Z' key. Still in front view with the top vertices selected, extrude the vertices again but do not move them. Immediately press 'S' and scale the new vertices to 0.9 size. Extrude these new vertices downwards by 0.02. Now we fill these vertices to create a new face. Press Shift+F, or select Faces, Fill from the Mesh menu. You can repeat this entire process to create a notch for the bottom edge too if you wish, but it is not necessary because it won't be seen in our scene. You can simply fill it with Shift+F and leave it at that.

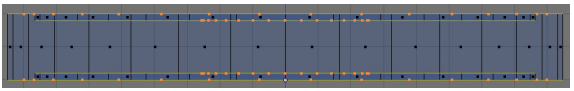
The piece will still have rough edges in the render though, so, in Object Mode, select Set Smooth from the Editing Tab. However, rendering the image now displays in some strange results.

Ooh, now that's not supposed to happen, is it?

This happens because the flat surfaces in our model do not respond well to Set Smooth. To fix this, we'll have to set the individual flat faces in our model to solid. So switch to Edit Mode again, and front- or side-view if you aren't still there. We'll use face selecting instead of vertex selecting for this because it will make our job a bit easier. Click this button on the 3D view menu to change to Face Select mode.

The Vertex-Selection, Edge-Selection and Face-Selection buttons.

We'll need to be careful to select the correct faces for this operation, and we'll be using box-select ('B') extensively for this. All our horizontal surfaces are the ones that need to be solid, so carefully draw a box around the squares representing the four layers of horizontal surfaces. You may need to zoom in a lot to do this. Your selection should look like the following when you're done:

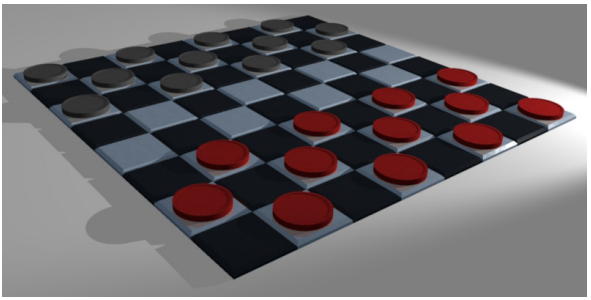


Above: Selection

Now that you have the correct faces selected, simply click the Set Solid button and the problem should be solved. When you're done, you can change back to vertex selection mode. Finally, by using the same duplication procedure described before, place the pieces on the correct places while remembering to make linked duplicates of the appropriate pieces.

And that's all I have space for this time. We'll finish off this scene next month by adding textures and cleaning up the lighting. In the meantime, feel free to play around with the scene on your own using the texturing principles learned in Part 3. As always, the completed scene will be available from the Dev.Mag website at www.devmag.org.za

CH1PPIT



Above: What the scene could look like when we finish with it

PART 4 :
Adding the Quality Touch 2

A well-polished game stands out when it is played. Not just because of the obvious things that have been added to the game through the implementation of a proper design, but also because of the time and care that has gone into making the game a pleasant experience for the user. Often these little things are unnoticed by the player but their very presence in the game makes the player more comfortable and secure in their new environment.

Mouse Control

By adding mouse control wherever possible, the user needs little effort to get immersed in the game. Allowing the user to use any input device of their choice will also make it easier for players to jump right in and play the game. While the player may not realise the effort that has been put into the game to allow this freedom of choice, they will be able to choose the input device they prefer to play the game with.

Pausing

A player who is busy playing may be disturbed by other members of their family -- in this case, they may just pause the game and come back to it later. By stopping all animations, game progress and time-based action within the game, the player can return to find the game in the same state as when they left it. A nice addition is to pause the game as well as the in-game music and sound effects when the game is minimised. This allows players to spend a small amount of time over a long period in the game. This will often allow people that are working from home to be involved in the game during their breaks and their time on the phone.

Friendly Environment

The game controls should impact the players experience as little as possible. The control panel and score display should never obscure the player's view of the game. By fading or moving the control panel as the players units come near it the player will always be able to control their units on the screen. Another method to ensure the players experience is as easy as possible is to enable an autofire feature on the players units, this basically means the player does not have to continually bang away at the keyboard while playing.

Sound Issues

A good rule of thumb is to never have multiple copies of the same sound effect playing at maximum volume at the same time, this creates a sound effect overload and will drown out other sound effects that might be relevant to the player in the game. Another mistake often made when polishing a game is to not fade out music between state transitions. While the game may fade between the relevant screen in an effective way, the music in the game continues loudly throughout the transition. By fading the music along with the screen a more consistent and less intrusive effect can be created.

Difficulty Settings

Every person that plays the game will have a different skill level. Certain players just don't do well in games, and other players just seem to know how the internal functioning of the game will affect their play. By implementing numerous difficulty levels into the game, all different levels of player can be accommodated. This can be done by allowing the player to select the speed of the items in the game, or by affecting the AI opponents' levels of intelligence. There are also players that would prefer being able to beat the game regularly on the easiest level than having to learn how to beat the top level AI. Don't forget a good Kiddie difficulty level, where enemies are very easy to beat a lot of extra pickups with amazing effects are implemented.

Other Players

A really well polished game will include little things like network play or a cooperative level of some form that will allow multiple players to sit together and play the game. Other ideas to keep in mind are autosaved games at specific points or time intervals, adding a sound effect to every in game action no matter how trivial, game progress indicators such as a map showing completed levels, in game customization of the players units, names and symbols, or even renaming other in game objects.

Nagging

Shareware games should also include a finely tuned demo restriction and nag screen to get people to buy the game. A nice feature is to have the "No thanks" button disabled for a while to encourage the player to spend the time to buy the game. Shareware games should also make it very clear to the player what additional features the purchased game will have over and above the features available in the demo version. By demonstrating and detailing the additional features the player will be encouraged to spend the money and purchase the full version of the game.

Very few free and even many shareware games do not have all the features mentioned in these articles. As such it is understood that all these features are not truly needed in a game. However, by adding some or all of these features the quality of the game will be increased and therefore the chance of retaining the player's attention for long periods of time is increased. By extending the player's interest in the game, the chance of getting the same player to spend some money on the game is increased.

Polish is not easy to quantify, but when you sit down in front of a game, the level of quality within that game is immediately apparent. Quality games improve player satisfaction and as a game designer, the higher level of player satisfaction you can achieve, the better chance you have of turning the game you make into a source of income.

CAIRNSWM

Good User Interface Design

Let's take a look at something that can often make or break a game or application. User interface design is vital to not annoy the living daylights out of users. A well-designed user interface is designed on three principles: Simplicity, Structure and Tolerance.

Simplicity

At all times, your interface should be simple and easy to understand. This means that your interface must be intuitive – in other words, if users can't find something via instructions, they should be able to find by making a logical, educated guess. Don't confuse users with jargon, and make sure everything is labelled neatly. You don't want buttons with a random picture on them and no explanation of what they do.

Structure

Always design your interfaces to be logical. It is important to be consistent throughout your design if everything looks and functions in a similar way then users can understand the interface based on previous experience. Use colour appropriately and consistently, you don't want overload your design with bright colours that don't go well together. You must also keep your colour scheme consistent and always keep the contrast rule in mind, since you always want everything to be clearly visible. Try to keep everything aligned, as bad alignment makes interfaces harder to understand. You must always try to group similar things in groups so that it easy for the user to find something according to the group it should logically belong to. If you are

designing an application, it is important to define the rules of how the interface works clearly. Relating back to simplicity, never create busy interfaces. Clusters of random buttons will chase people away.

Tolerance

A user interface should always give the user feedback in terms of what the user has just done, is about to do and can to. A good example is to make menu buttons glow when the mouse is over them. An interface should always be forgiving, so you should have things such as an undo and redo. If the user has done something wrong, let them know what they did and how they should do it in the future.

Your interface should never be cumbersome – a slow interface is one of the worst things you can put into a game. A good example of a cumbersome interface would be the Startopia menus that play a long annoying animation whenever you change an option.

Now that you know the basics of making an effective user interface, go out and make games with annoying, crowded, ugly menus! Also have a look at this for some more great pointers. **http://www.ambysoft.com/essays/userInterfaceDesign.html**

SQUID

The making of Nonex 2: Part 1

Welcome to the first installment of my new series of articles, where I document the trials and struggles I encounter as I (and two of my buddies) build and finish our new and exciting title, Nonex 2. The plan with this is that new developers can learn from our experiences, and possibly inspire others to do the same.

Unfortunately, it will be impossible for me to document absolutely everything we do, but we will at least cover the most fundamental areas as well as the areas where we struggled the most. So where to begin? Well, you need a plan of action. A certain order in which you want, or need, to do things. So, I had to make my plan for Nonex 2. In the past I usually found myself eager to climb straight into the coding and graphics, and before I knew it I had a half finished game that looked kind of good, but lacked a lot of depth and quality.

My last gaming effort, "Team Boss" was the exception to the rule. I first made a game design document, then made layouts and concept art. The result is that my game had all the planned elements in it at the end of the day. Granted, the game wasn't that great, but it did exactly what I wanted it to do.

Using this new-found knowledge, I set out to plan this venture. In an order of steps I planned the systematic creation of my game.

1 Game design document. (Wish list)
2 Game requirements document.
3 Storyline document.
4 “Make the game work on paper” document
5 Concept art.
6 Level design and level concept art.
7 Character development
8 Environment and villain development.
9 Final game design document with all concept art and character sheets included, that conforms to the game requirements document.
10 Creation of game engine.
11 Creation of assets. (Game graphics and sound)
12 Asset implementation.
13 Beta testing. (Official game testers)
14 Bug fixing.
15 Beta testing. (Official game testers)
16 Release demo.
17 Market full version game.
18 Final game testing (Official game testers)
19 Bug fixing.
20 Release full game for purchase on portal.

This doesn’t just look like a handful – it actually is. Steps 1 – 9 are indeed the most important steps of any game making experience. They will most definitely determine the quality and the style of your game. And the more time spent on this, the better. You can also see that I do in fact plan to sell this game when it is released. This means that the game must be of such a standard that I myself would want to buy it.

So what is my game going to be about? Well, I have forced myself to describe it in one paragraph.

“Nonex 2 is a top-down space arcade shooter with an RPG factor which includes 3 different ship classes, each with its own skill tree. You will gain levels and will be able to buy and sell items at vendors, and loot some in the process of ridding the universe of evil forces.”

Sounds complicated doesn’t it? Well imagine Nonex 1 (For those who haven’t played it can download it from my website http://www.apfstudios.com) with a DOTA bar at the bottom. A bigger inventory and skill tree than DOTA, but less than Diablo 2. And with ships that cast “spells”. All this with a 2 – 4 player co-op multiplayer mode, for added fun.

The game should be replayable and team oriented. But it must be playable and fun single player as well (unlike DOTA). Now, that is a description, next month you can expect a full Design document and Requirements document for your reading pleasure.

HIMMLER

FFS! (Fast Food in Space!)

I've played a lot of management games, but to be honest, the thought of actually making one never occurred to me until it was announced as the genre for the NAG Game.Dev Forum's tenth competition. Historically, I've had an eye on the goings-on there, but my involvement has typically been fairly limited.

When news came in that there was money to be made from game development, the greedy bastard within was motivated enough to allow the other part of me, which wanted to make games just for the sake of making games, to get along long enough to finally sit down and put forward a proper attempt at making a game. I had no idea what I was going to make, but by God, I was going to make it.

It may be surprising, but the concept of fast food in space is no stranger to me. It's been sitting in the back of my head for years ever since I had an idea for a shooter game in which the protagonist is desperately trying to defend his killed-by-pirates parent's fast food delivery company from evil space pirates. That game never happened, but ideas like that have a habit of lurking about until the right opportunity arrives for them to unfurl their wings. After musing over a variety of options, I arrived at the idea of making a management game based on running an intergalactic fast food restaurant empire.

The Design Process

The design of FFS was a rather interesting progression. My original idea (heavily influenced by my RTS obsessions) was to have an inner ring where you make buildings, and an outer ring populated by resources which you harvested in order to have the necessary supplies to run your restaurants. The idea of resources was dropped completely once I got it through my thick skull that in this context, there was absolutely no fun to be had of constantly building resource collection ships and dealing with supply shortages. Unfortunately, that took with it an interesting offshoot idea where you would be able to design

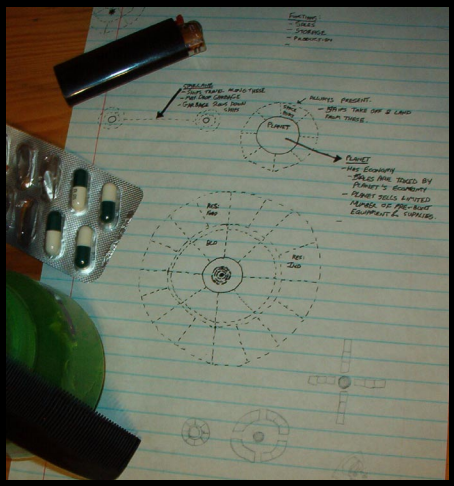
"It's a little known fact that there are traffic circles in space"

different meals based on the food resources you had available, but in retrospect, there probably wouldn't have been time to implement that feature properly in any case.

Soon after the core gameplay mechanics had been established, Louis got involved with the project, though I don't remember exactly how. One minute it was just me working away on it, and the next, Louis was there, exuding enthusiasm and just being generally interested. That rumour people pass around that two heads are better than one certainly proved true. We had countless debates on various aspects of the gameplay. These debates helped to eliminate many crazy unworkable ideas that we would independently come up with, and generally pushed the game's design to a level that simply would not have happened if it had only been me working on the game.

"One minute it was just me working away on it, and the next, Louis was there, exuding enthusiasm and just being generally interested. That rumour people pass around that two heads are better than one certainly proved true."

Game World Implementation



The game's design document

Creation of the game world itself went brilliantly. Implementing the planets, star lanes and background tiles went very smoothly. The ship movement was not easy to implement, but drawing from prior mathematics experience gained on a 3D Game programming course, I managed to put it together precisely as I'd seen it happening in my head, which in my experience, is a pretty rare thing.

Louis comes to terms with algebra

The Engine

The engine that FFS uses is a custom built 2D game engine that I'd pieced together from the various games I've worked on over the years. I had recently upgraded its core to an event driven architecture. In the first few weeks of development, I implemented a whole bunch of concepts that I'd been struggling with for years, for instance, a centralised component for creating, managing and destroying game objects. It was hacked together in two days, and it worked far better than I could have hoped. Overall, the functionality of the engine had far more pros than cons. There were many things it could not do, but what it could, it did well, and didn't fall short as the functional requirements of the game became more complex.

Game Data Abstraction

One of the strongest points of FFS's design is that every single game variable that mattered was abstracted to a definitions file. This made the task of balancing the game much easier. Louis, whose last major programming achievement was changing a mouse pointer green, was able to balance the game by editing values contained within a single file… eventually.

Crunch Period

The final 72 hours of development was a blur of frenzied activity. Considering we cut things so late, this final push could have easily ended in disaster. A record number of energy drinks were consumed as we finalised the core gameplay, fixed bugs, created graphics and balanced the game. In the lead up to cut off time, I was hacking in the game menu and credits while Louis was attending a family braai. He did however, come back afterwards and performed some truly Herculean last minute gameplay balancing with the assistance of my "why didn't I also get a mention for Herculean last minute gameplay balancing too?" brother Byron who acted as the official game tester and coffee maker.

Having lost the use of his other nine fingers, ET pressed on!

What went wrong?

Interfaces!

If anyone ever tells you that making decent interfaces for a management game is easy, they're probably

getting theirs done by cheap Indian labour, and have no idea of the complexities and problems than can arise. No doubt I'm being over dramatic, but I'd been blithely building the game, thinking that adding the interfaces would be a simple matter, and would take no more than a couple of days. Fate being the fickle wench that she is, threw me multiple curveballs of doom™.

Finding ways to get game data into the interface controls, and ensure that the data stayed up to date ended up being hacked directly into the interface objects themselves. That combined with a hastily put together hierarchy of interface controls resulted in an enormous amount of bugs and difficult to trace performance issues.

Finding that conventional techniques failed, ET tried other methods of persuasion

On the up side, I've learned many lessons and now have a far better ideas of how I'll be making interfaces in future.

unDelphiX

Using the unDelphiX graphics library was both a curse and a blessing. The library is fast, fairly well structured, and works on pretty much every machine I've ever tested it on. (With the exception of ones with very old graphics cards) A week into development, I went looking to see if there had been any new developments happening with the library. A newer version had been released, which I downloaded and installed. Two sentences back, I'd made a terrible mistake.

Do not under any circumstances, during the development of a project, trade a code base which you know to be working for one which claims to be newer, better, faster. I lost about a week of development discovering that the new version had some major bugs, previously working functionality just stopped working completely, causing a number of things in the game to break.
In the end, I rolled back to the previous version and continued development from there.

The unDelphiX library is basically a hardware accelerated version of the original DelphiX, and unfortunately, the conversion has many holes in it, and a number of inconsistencies. I would spend hours pouring through the game's code searching for a bug, only to find the problem originated from an unDelphiX function.

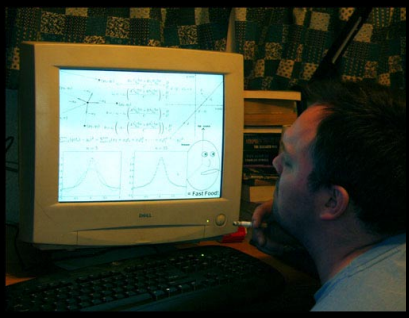
This little retelling should not be viewed as a black mark against unDelphiX, it's still a good library despite its flaws. Every graphics library I've ever worked with had its own set of problems and quirks.

Gameplay

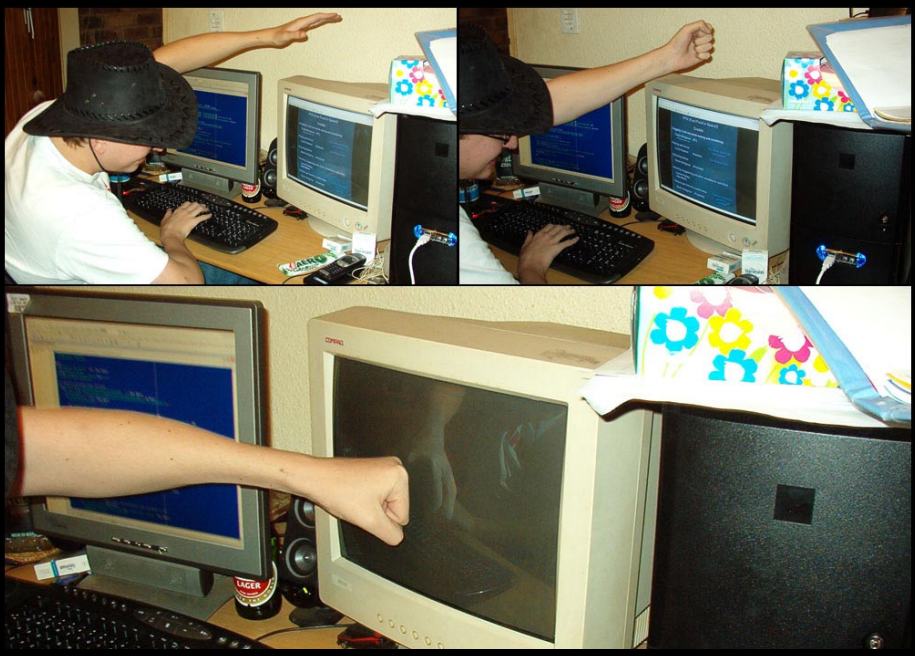
Why is the gameplay in the “wrong” section? Surely the game that won the competition shouldn’t have to worry about such things? Wrong. The debates that raged between Louis and myself were epic in scope and gargantuan in beer consumption terms. Jokes aside, there are two major reasons that gameplay ends up here.

A: Insufficient Game Testing - There simply wasn’t enough time to play-test and balance the game due to many core elements being implemented so late. As a result, the gameplay is not as balanced as it could have been.

B: Gameplay flat line - The gameplay in FFS reaches a plateau within about 30 minutes. The design did not include enough diversity to allow for the game to remain engaging for the full duration of a game.



Louis turned to performance enhancing drugs



No explanation necessary.

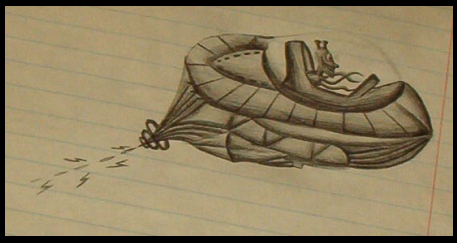
Time vs. Features

We had a little less than two months to create the game. I thought the people complaining that it was too much time were crazy. The amount of dropped features was extensive, and the number of elements that simply should be in a complete game, such as saving and loading, or sound and music, just aren't there.

In Closing

Overall, Louis and I had a great time developing FFS, and we're extremely proud of what we managed to achieve.As for the future of FFS, we view its win as an indication of a successful prototype. The game is certainly not complete, neither in features, nor in gameplay. The design for the next version is fully underway, and it will be simply bigger and better in every way. There are also plans in the works to have the completed game published. Let's see how that goes…

EVIL_TOASTER



Another satisfied alien customer

FAST FOOD IN SPACE! in action!

THE TECH WIZARD

An introduction to pathfinding

Pathfinding is one of the building blocks of AI programming. Whether it's a bunch of flying monsters coordinating themselves with one another in a 3D labyrinth or a couple of enemies finding their way to the hero in a Pacman clone, working a way around obstacles as opposed to through them is an important task for any reasonably smart enemy.

Although there are many techniques for pathfinding, this article will focus on a simple – yet still highly effective – algorithm designed for the job: A*.

Overview of the algorithm

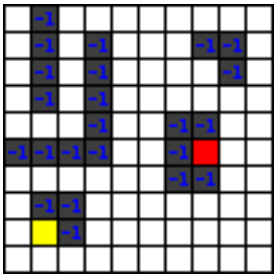
A* (pronounced "A-star"), also occasionally known as Floyd's Algorithm, is a simple pathfinding technique used by many programmers to acquire suitable pathing in any grid-based obstacle course. It involves a breadth-first search of the maze, starting with the entity's grid location and then proceeding to "flood" the grid blocks around it. The blocks themselves are elements of an array (which is essentially our grid), and the values assigned to these are steadily incremented each time the flooding procedure is run on the grid.Each block can only be assigned a value once – thus, by the end of the routine, these blocks should have a value corresponding to the number of steps required to get from the entity to any given block, and from here it's easy to get the AI to do what you want with regards to pathfinding.

Building a simple A* algorithm

To demonstrate the basic workings of A*, we'll use a 2D maze, a generic AI entity to use the code on and a generic target which this entity needs to reach. Obstacles will be thrown into the maze to make sure that only a smart path will work for the entity. Since this is just an introduction to the algorithm, we'll be using a static 10x10 integer array for our maze.

```
int grid[10][10];
```

The array elements should be initialised to zero. Throw in a few walls by changing the value of some elements to -1. Get a value for the entity and the target (use negative numbers, such as -2 for the entity and -3 for the target). Represented as a picture, your array should now look something like this:

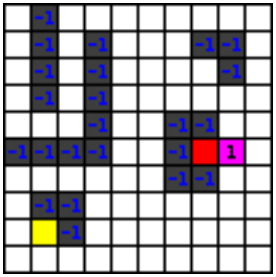


Now that it's initialised, we can take a look at messing about with its values to get ourselves a path. In other words, we run our first "flood" iteration. For this, we'll need to set up some loop structures:

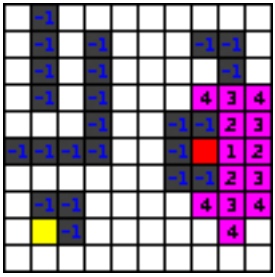
```
i = 0;
while s = true
{
i = i + 1;
  for x = 0 to 9
    for y = 0 to 9
    {
     //code to check array at position [x][y]
     }
//additional code follows as necessary
}
```

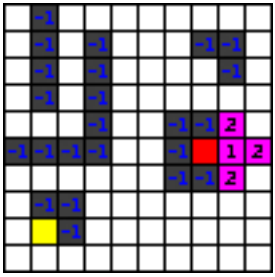
Variable “s” is any sentinel value initialised as “true”, to make sure that we run the flood as many times as we want. “x” and “y” are grid co-ordinates. “i” keeps track of how many iterations of the flood we’ve made already.As you may have guessed already, the first part of our code checks every block on the grid, from position (0,0) to (9,9). Since this is the first iteration, we want to be checking for a -2 (the grid position of our entity). Once that happens, we want to be checking the blocks immediately around it. In this example, our entity (red block) is at position (7,5). This means that we want to check the blocks (6,5), (8,5), (7,4) and (7,6).

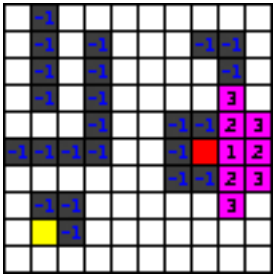
If there is a zero in any of these blocks, it means that this block hasn’t been touched by the pathfinder yet, and we can give it the value of i, which is in this case one. This indicates how many steps need to be made to get from the entity to the given block, and the number represented by i will increase every time the flood procedure is run. If a block already has a non-zero value (such as -1 for a wall), there is no new value assigned. Your first step should end with the array looking like this:



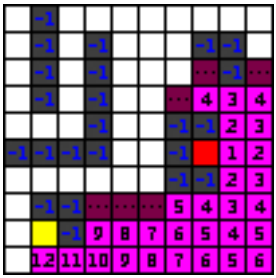
The second time you run the flood, your i variable will be increased by one, and from here on you’ll be searching for blocks with the value of (i-1). Thus, on the second iteration, i will have a value of 2 and you’ll be looking for blocks with the value of 1. Each block you find will have the four surrounding blocks checked for zeroes, and will have the value of 2 assigned if any are found. The same holds for the next iteration, except that i will be 3. Here’s what your flood should look like after the next few loops:







See what we mean by “flooding”? As the loop keeps iterating, the numbers will keep expanding across the grid until they reach their target, like so (keep in mind that the whole flood isn’t shown here):



Note that throughout your loops, you should be searching for the target when checking block numbers. Here, when i = 13, we’ve managed to find it. Thus, we save the coordinates of the target we’ve hit, terminate our main loops by switching the sentinel value “s” to false and entering a new section of code.

After this, things are easy – starting at the target block, search for a block next to it which has a value of one less than variable i (ie. 12). Store that block’s co-ordinates for use later (preferably in a list structure) and then, using it as a reference, decrease i by one and repeat the process so that you’re looking for 11.

Keep going until you reach the original block. And yes, this process works – there are multiple paths that can be taken, but all are of the same length:

And that's it!

You've just learned how to use a simple implementation of the A* pathfinding algorithm.

Using the principles here, it's easy to create your own pathfinding bot and expand upon it to make it more sophisticated.
This was a very simple example, and improvements could be made in the following areas:

– allowing for variable maze sizes (use of non-static structures or dynamically declared arrays)

– assigning "checking" co-ordinates to lists so that you don't have to check the entire grid on each iteration

– catering for multiple targets, preferred targets or unreachable targets

– anything else that you'll need!

Good luck, and happy coding!

NANDREW

TECH

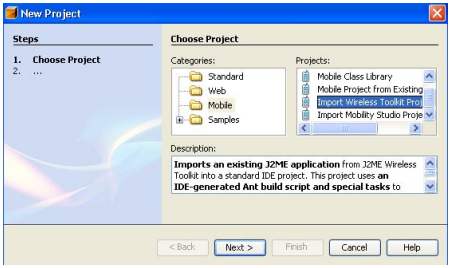
Tooling up with NetBeans

Working in a simple text editor with basic compilation tools is all good and well, but the development world has mostly moved on from that 'primitive' approach (with the exception of a few hardcore masochists, of course). Modern toolkits include richly featured Integrated Development Environments (IDEs) that incorporate syntax highlighting, debuggers, resource managers and more. This all sounds really fancy and expensive, doesn't it?

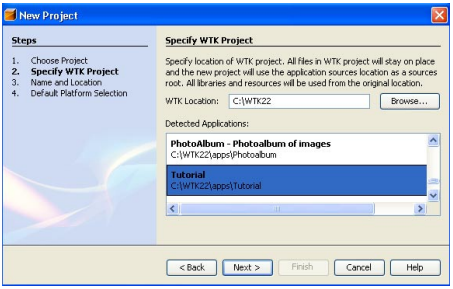
Well, the good news is that in the development world some of the best things really are free. One of them is the NetBeans IDE, which we will be using from now on. This edition of the tutorial series will focus on moving our project so far into NetBeans, so it won't involve much coding.

If you have not already done so, download and install NetBeans 5 and the NetBeans Mobility Pack from www.netbeans.org and install them. The installation process is simple, just run the installation executables and follow the instructions. If you do run into problems, the mailing lists to be found on the NetBeans website provide excellent support.

Now run NetBeans (the Mobility components will automatically be loaded). To invoke the wizard conveniently provided to import WIK projects, choose New Project from the File menu. Select Import Wireless Toolkit Project from the Mobile group and click Next. Ensure that the correct location of your Wireless Toolkit is specified (typically C:\WTK<version>), choose our Tutorial project in the list and click next. On the Name and Location page you may want to change the name of the project to simply Tutorial, and note the location in which the project will be created. Note also that your source files will not be moved to



Above: Import Wizard



Above: WTK Project Select

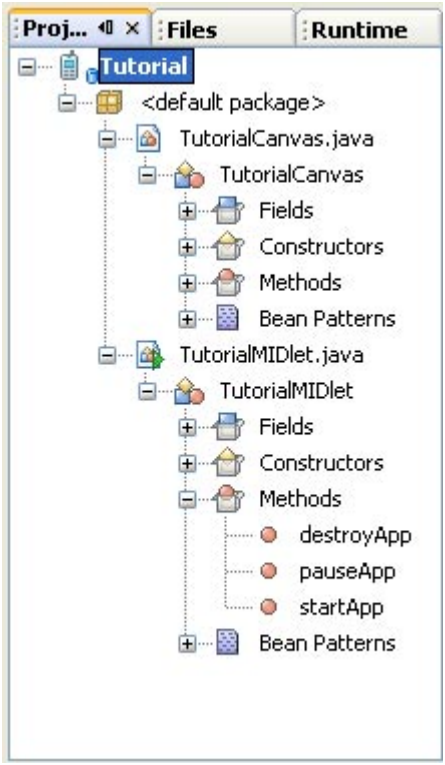
this location, this is just where the NetBeans project files will be created; if you had to create a new project from scratch, your source files would be placed in the same location as your project files. Make sure the Set as main project check box is ticked and click Finish. You will now see the Tutorial project in the project explorer in NetBeans, expand <default package> node to reveal your java source file. If you keep expanding the tree and double click on any of the elements you will see that this view allows to to jump to source files, classes, or even individual methods and member variables.

Now that we are using a tool that makes it easier for us to navigate files and classes, lets move our TutorialCanvas class into its own Java file, since this is proper Java practice. To do this, right click on <default package> in the project explorer, and choose New->Java Class. Call the class TutorialCanvas (remember it's case sensitive) and click Finish. NetBeans creates a simple class for you with a constructor, but we will just replace it with our own class. Double click the TutorialCanvas node under TutorialMIDlet.java to find our implementation. Drag-select all the code defining the class (from the class keyword up to and including the last closing brace) and cut it from that file (Ctrl+X). Now go back to TutorialCanvas.java, delete everything in that file, and paste what you just copied (Ctrl+P). You will now see many red blocks appear next to the source code.

If you click on any of these, they will take you to a line underlined in red. This is the IDE at work, identifying errors in your code before you even compile it- you have to admit, that's pretty cool! But didn't this code work before we moved it? Yes it did, however that was in a file that had its imports declared, so now when this file is compiled, Java won't know where to find classes like Graphics and Canvas. This is easily fixed, just go back to your MIDlet java file and copy all the import statements from the top of it (Ctrl+C).

Now go back to the Canvas java file and paste them at the top of the file, before the class declaration. Within a couple of seconds, all those angry red lines and blocks will disappear. These early indicators are extremely useful, so get used to using them early on. Now that we've neatened things up a bit, let's make sure everything still works. To run the MIDlet in debug mode, press F5.

Below: Project Tree



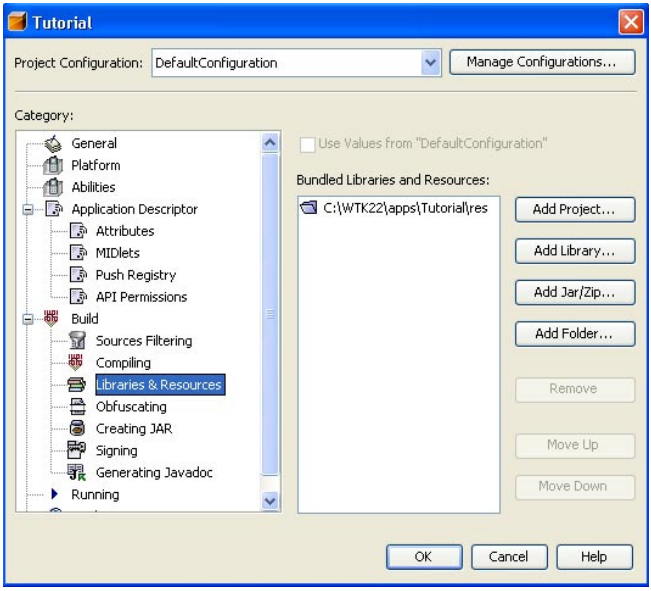
All files will automatically be saved and your project will compile and build and the familiar emulator will pop up ready to run your game. Close the emulator and lets do some final poking around in the IDE. We've had a look at the source code, and it's obvious where you need to look for those, but what happened to our sprite images? NetBeans allows us to specify files, folders or jars to be included in our MIDlet at build time.

This is useful for including third party libraries as well as for image, sound and other resource files. Right click on the Tutorial project and select Properties to open the project properties window. As you can see there are loads of options but for now open the Libraries and Resources page under the Build group. You will see that the import process automatically added your res folder to be included in the project. At this point feel free to explore the settings available to you in the properties window, and see if you can figure out what some of them are for. If you manage to 'break' something, you can always delete the project and re-import it. There is also context sensitive help on each of these options, just press F1 at any time to see help relevant to what you have selected.

That's all for this tutorial. Next time we will try our hand at the debugger, and have a look at how NetBeans makes it easier to target your game at the many different cellphones that are out there (including the one you carry with you). In preparation for this I would suggest you download the SonyEricsson SDK from http://developer.sonyericsson.com/site/global/docstools/java/p_java.jsp, and possibly whatever emulator matches your own cellphone.

Until then, enjoy digging around NetBeans to see what it offers.

FLINT



Above: Libs and Resources

"NetBeans allows us to specify files, folders or jars to be included in our MIDlet at build time. "

rAge report!

There were people. There were games. There were competitions. There were Game Devvers.On 29th September to 1st October, rAge struck. This was the biggest, baddest gaming and technology expo to hit South Africa this year, and Dev.Mag was all a part of it. Three days of blood, sweat and considerable amounts of tears went into making rAge 2006 a special event for Game.Dev, with this year sporting a bigger stand, bigger prizes and ... well, heck, a bigger everything, really. And as it turns out, bigger definitely equals better.

Getting started

As early as three days prior to rAge, members of the Game.Dev and Dev.Mag crew were busy setting up the stand. This sweet deal sported a fancy projector, some kindly sponsored computers from IT Intellect and, of course, some all-important beanbags for people to crash on when they weren't busy running about and looking important. It was awesome seeing an event like this building itself from scratch, especially when one had a role in setting it all up!

The morning of the 29th arrived, and we were still frantically getting our gear organised when the doors of rAge burst open and the people flooded forth in

Above: Danny " Dislekcia" Day

a torrent of anime lingo and game-themed t-shirts. Fortunately, those amongst our crew already present were hardy enough to withstand the storm, fielding the first few individuals who staggered over to this mysterious "Game.Dev" stand and telling them about the wonders and joys of making games (while doing their utmost to show off the wicked Game.Dev shirts provided by Luma).

Technical issues at the stand were resolved just in time to meet the greater flood which soon descended upon us, including the arrival of most of the rest of our team. Introductions and horrified realisations ("Oh my word, you're actually a guy?") left quite a few of the


members occupied for a while – especially owing to the fact that Game.Dev consists of members hailing from all sorts of places around the country. The surrounding chaos made the meetings a little bit less heartwarming, but still got people motivated to work together on some of Game.Dev's projects at rAge:

Talk, talk, talk ...

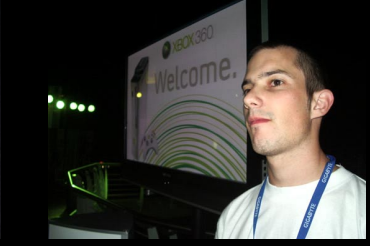
Game.Dev hosted at least a dozen talks and seminars throughout the course of rAge, delivered by a variety of people on a massive range of topics – these ranged from postmortems of locally-produced AAA titles to talks on the advantages of frameworks and even a seminar displaying the power of rapid game development tools such as Game Maker.

Big names who did some chatting in this regard were Dale Best and the fellows from Luma (responsible for Club Silo), I-Imagine's Dan Wagner and SA Developer .Net's Andre Odendaal. Of course, the rest of the crew and even the audience were allowed to join in when the discussion panel came around for people to get involved with.

While discussing how to move game development forward in South Africa, the microphone was passed back and forth like a hot potato, with experts giving their views on the topic – right alongside the not-so-experts (the esteemed author of this article included), who spent their time trying to tread intellectual water and somehow managed to sound smart in the process. Aside from the official talks, there were also countless impromptu sales pitches concerning the Game.Dev and Dev.Mag brands offered to clusters of excited newcomers, and many others were given hands-on presentations of how neat Game Maker was, or even got to see just how awesome some of the community's games really were.



TAILPIECE



Fun with the Xbox 360

Game.Dev (along with your ever-reliable Dev.Mag journalist!) naturally wasn't too busy to pass up the opportunity of checking out the big hoo-hah surrounding Microsoft's brand new console. Aside from a very fancy and very exclusive launch party at rAge's Dome itself (which follows the law of all exclusive parties – the less people allowed to go, the better the party has to be), there was the actual BT Games launch of the console at midnight on the 29th. This included giving away a nice big stack of premium 360s, free of charge and soul-binding contracts!

Of course, the game developers simply had to put their own spin on the 360 launch, and a demonstration of the new XNA toolkit – a means of easily developing games for both the PC and 360 platforms – was delivered by Game.Dev to anybody who was interested in learning about it. This talk was given by an actual Microsoft rAge exhibitor, adding significantly to the "ooh!" factor of the display. Next-generation stuff doesn't have to be complex when you know what tools to use!

Free stuff everywhere

No exaggeration. From itty bitty baby ITI pens to unwittingly "given" posters (yeah, they were that good), we had our hands full in the freebies department. This included fresh-from-the-oven Game Development DVDs, burned at the event itself and going like hotcakes whenever people passed by. These weren't harmless little DVDs either – each had over 4 gigs of indie games, educational videos, podcasts, websites, resources, toolkits, engines, samples, tutorials and crocheted sweaters at the wannabe game developer's disposal for absolutely no cost.

TAILPIECE

Giveaways of a competitive nature were also prevalent. One such event offering these was the Game.Dev Idols competition, in which several game development books sponsored by Intersoft were given away to people who could step up and relate plans for clever and original games which they'd come up with.

There was a surprising supply of ideas – concepts such as an inverse system of Lemmings and a multiplayer point-and-click adventure were thrown at our panel of game industry experts. Throughout the process, ideas that were genuinely interesting, original and applicable reared their heads, and their creators were duly rewarded. Creativity and inspiration is definitely alive and well in the local community, it just needs a platform to work from!

The big brother of the pack was, of course, the prizegiving for Game.Dev's Comp 10, for which participants were required to create a management game with whatever tools they had handy. There were some genuinely astounding entries, coupled with some genuinely astounding prizes – R10 000 was sponsored by NAG for this competition. Serious business!

The ceremony took place on the rAge main stage, which made the whole thing really slick, fancy and important-looking – as if the prospect of winning all that money wasn't important enough already.

The prizegiving for Game.Dev Comp 10 sponsored by NAG

In closing

The presence of Game.Dev at rAge has massively increased since last year, and more events (including next year's rAge) are already in the organisational pipeline for both Game.Dev and Dev.Mag. It's amazing fun taking part in the Game.Dev activities, and hopefully some of you are reading this after hearing the good word at the expo.We're continuing our efforts to expand and make game development even more awesome for South Africans out there, so if you're interested in advertising, sponsoring or assisting Game.Dev or Dev.Mag in any future endeavours, be sure to visit www.gamedotdev.co.za or e-mail Dev.Mag at devmag@gmail.com. As for rAge – if you saw us there, we hope you had an awesome time! If you didn't go, then make sure you do next time around, when it'll be bigger, better and even more devvy!

NANDREW

Comp 10 announced its winners at rAge 2006, in front of many enthusiastic game developers and general rAge attendees. Giant cheques were a feature, although they weren't quite as cool as the R10 000 in cash prizes that got handed out! The winners were as follows:

Best new entrant (R1 000)

Hotel manager – Darth Penguin and CiNiMoD

Third place (R1 500)

Cyberworkz – kRush and Geometrix

Second place (R2 500)

Fantasy Land – Cairnswm

First place (R5 000)

Fast Food in Space -- Evil_Toaster

UNTIL NEXT YEAR... GAME ON!


GAME.DEV at RAGE 2006
TALKS AND DISCUSSIONS
Er... Is this thing on?
BIG CHEQUES
FREE STUFF
AND EVERYTHING ELSE
DEV IDOL
GAME.DEV IDOLS
TO THOSE WHO MADE IT POSSIBLE...
WE THANK YOU!


www.itintellect.com
Durban
Ph 031 277 2000
info.dbn@itintellect.com
Bryanston
Ph 0861 484 484
info.gp@itintellect.com
Cape Town
Ph 021 421 8555
info.ct@itintellect.com
Richards Bay
Ph 035 789 3115
info.rb@itintellect.com
Pietermaritzburg
Ph 033 386 6057
info.pmb@itintellect.com
Bloemfontein
Ph 051 447 3635
info.bfm@itintellect.com
DON'T PLAY GAMES...
B A : DEGREE IN GAME DESIGN
B Sc : DEGREE IN GAME PROGRAMMING
N C I T : NATIONAL CERTIFICATE IN INFORMATION TECHNOLOGY
i.t. intellect
COMPUTER TRAINING SOLUTIONS
GAMING

O N L I N E

www.devmag.org.za