



# DEV.MAG

**CREATE • DEVELOP • EXPERIENCE**

**BACK OF A NAPKIN:  
PART 2: LEARNING ABOUT  
VERTICES**

**HOT  
LABS!**

**GAMEMAKER:  
WHY YOU SHOULD USE GAME-  
MAKER!**

**MOBILE GAMES:  
HOW TO CREATE GAMES  
USING JAVA**

SOUTH AFRICA'S FIRST GAME DEVELOPMENT MAGAZINE

**REVIEWS : SECOND LIFE FEATURE : HOT LABS  
LUDUMDARE 48HR GAME DEV COMPETITION**



# CONTENTS

## REGULARS

03 - ED'S NOTE

04 - DIGITAL STOMPIES

## FEATURE

05 - CAN YOU SAY "D-SHARP"?

## SPOTLIGHT

07 - PIETER GERMISHUYS

## REVIEW

09 - SECOND LIFE

## DESIGN

10 - WHY YOU SHOULD USE GAME MAKER

11 - BACK OF A NAPKIN PART 2: LEARNING ABOUT VERTICES

## MOBILE

13 - GAME DEVELOPMENT IN JAVA

## TECH

15 - A LITTLE BIT ABOUT O-NOTATION

16 - SORTING ALGORITHMS

## TAILPIECE

17 - SWARM OF GAMES

## COMIC

19 - DIGITAL MAYHEM



# ED'S NOTE

**F**or this editorial, I wanted to do some research on what a game developer is and how game development started. I looked and looked ... finding only complex answers along the way. I then thought: if I could only find complex answers to a simple question, what must people out of game development 'circles' think of game developers? So I began to question what they thought of us. To be honest, some answers were quite derogatory. Then a motion popped into my head... why should we care what other people have to say or think about us? For all we know, they might collect stamps or bottle caps. We're living in an interesting age, where gaming itself is being recognised as a sport and a valuable industry. Taking a note from that, we can then only begin to hope that the stereotype which surrounds us can be changed as well. But honestly, who cares? We're having fun making games and that's all that matters.

Now for more pressing matters:

As always, there's something new. In the constant quest to garner new readers, Dev.Mag presents the tech section. With its focus more on the code-wizard side of game development, it should always prove to be an interesting read for budding coders. This month will also see the start of our mobile dev section's tutorial series. Beginning with getting you set up, this set of tutorials will eventually have you creating mobile games of your own.

And finally, last but not least, I'm happy to announce (with the rest of the Dev.Mag team ... to their surprise) that the Dev.Mag website is up. A special thanks to Ch1ppit for heading up the site creation and making it all happen. This will hopefully solve the hosting issues that have been plaguing us for a while now, and give us a base from which to send news and updates to the community. So go give the site a look, at [devmag.googlepages.com](http://devmag.googlepages.com)

## Editor

Stuart "GoNz0" Botma



Why was Chuck born upside down?  
So he could roundhouse kick his way out!

## THE TEAM

### RANKING OFFICER

Stuart "GoNz0" 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

### WEBSITE

[devmag.googlepages.com](http://devmag.googlepages.com)

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

This magazine is a project of the NAG Game.Dev forum.  
Visit us at [www.nag.co.za](http://www.nag.co.za)

All images used are Copyright and belong to their respective owners.  
If Chuck is reading this magazine: All jokes within are for entertainment purposes and should not be taken seriously or acted upon ie: Please dont roundhouse kick us.



## GAMASUTRA GOES PODCASTING

Gamasutra recently released their first podcast, entitled "Gamer Demographics, Part One". About half an hour long, this podcast opens up a new way for Gamasutra fans to get their favourite news and views from the gaming world – great for long car trips or quiet afternoons in the office. Partnering with Tom Kim of Fatpixels Radio, Gamasutra is expected to keep up a high-quality series of podcasts which will not only inform, but entertain at the same time. The podcast, only 12 MB in size, can currently be found at [http://gamasutra.com/features/20060502/gamapodcast\\_01.shtml](http://gamasutra.com/features/20060502/gamapodcast_01.shtml)



## SHMUP-DEV COMPETITION

SHMUP-Dev ([www.shmup-dev.com](http://www.shmup-dev.com)) is holding its second online game-making competition, this time centred around the theme of dragons. SHMUP (short for "shoot-em-up") has brought in several sponsors for this competition and prizes including games, game development tools and other business tools are up for grabs in a variety of

categories. Although this particular competition kicked off at the beginning of May, the deadline is 1 July and new entrants are allowed to register as late as 21 June. Cool games, a cool community and some really cool prizes all await!

## MICROSOFT ANNOUNCES DIRECT PHYSICS

With AEGIA's PhysX processor causing tremors throughout the gaming community, Microsoft has wasted no time in announcing their new Physics API – termed as Direct Physics – being produced to take advantage of the surge in physics popularity. This API is expected to become a standard part of DirectX 11, further consolidating



physics accelerators as a viable marketing concept. It's hoped that the introduction of this standard will eliminate some of the confusion surrounding current physics tools and the dependencies on each (such as the well-known Havok). What will this mean for game developers? Well, hopefully some less painful physics handling.



## SUN JAVA CONVENTION

For two days in May, Sun Microsystems (the creators of the Java language) ran the last of their International Developer Days at the Sandton Convention Centre. While no game-specific sessions were held, the issue of game development in Java was continually brought up. The Mobile development sessions detailed upcoming changes for 3D and 2D game development. Some examples of alternate devices such as cars and robots that use the J2ME platform were also demonstrated. With the advent of 3D-enabled cellphones, 3D-based games on mobile devices will certainly become a growing industry.

## GAMEMAKER FORUM HACKED

The Game Maker forums (<http://forums.gamemaker.nl>) recently recovered from a hack involving a forum exploit. The hacker in question was able to gain administrator privileges and edited the forum's template, adding an iframe which downloaded trojan viruses to Internet Explorer users. The problem has since been resolved and the forum has been reset. Forum-goers concerned about the trojans should check the appropriate thread in the new forum.





# HOT LABS

## Can you say “D-Sharp”?

After the popularity and success of the Hotlabs in Johannesburg, the people of Durban finally got their own. Presented by Pieter Germishuys and hosted at IT Intellect Musgrave, the D# Hotlabs focus on managed DirectX and C#.

On 29 April, the first D# Hotlab kicked off with the topic “Setting up a window and Direct 3D”. A group of about 10 people attended, ranging from programming amateurs to hardcore coders, all with a great passion

for making games. At 8:30 the first eager attendees arrived, sat around drinking coffee and had some interesting discussions about game development. Although there were a few hiccups in getting all the equipment set up early on, in no time the problems were fixed and things got going.

Pieter did a great job in introducing the basic theory of programming in DirectX and, soon enough, the code on everybody's screens started making perfect sense.





(Above) The Hot Labs workshop held at IT Intellect in Musgrave. The workshop was presented by Pieter Germishuys a well known figure in the world of DirectX programming.

After setting up the window, we went on to create shapes, move objects around and then texture a wire-frame model.

Overall the day was amazing, especially the constant discussions about game development. For me, the frameworks versus programming debate was fascinating, especially hearing Pieter's view on the subject. Everyone went home impressed with what they could do in DirectX and I'm sure all who attended look forward to next month's event. Everyone in the Durban area is encouraged to attend.

INSOMNIAC

**"Everyone went home impressed with what they could do in DirectX"**



## D# WORKSHOPS.

### WORKING TOWARDS BREAKOUT

**DATE:** 24th of June 2006

Dropping blocks was fun but we need to get some real interaction going. Breakout was a game that allowed the player to control a paddle and to prevent the ball from dropping. In addition to not dropping the ball, the player had to hit blocks that resulted in bonuses dropping from the blocks that were hit.

### WORKING TOWARDS PACMAN

**DATE:** 29th of July 2006

Pacman introduced some very interesting concepts such as Artificial Intelligence and maps. Although Pacman only has one map, we still need to do boundary checks.

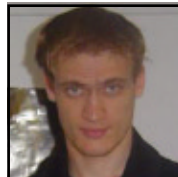
### WORKING TOWARDS SPACE INVADERS

**DATE:** 26th of August 2006

Space Invaders added an exciting twist. The player has a few barriers and has to eliminate all the enemies before they reach the bottom and take out the player. The enemies fly/march down a row and keep on speed up as they progress.

# DIRECT X MVP

**P**ieter Germishuys is a graphics/game developer, involved with an array of projects ranging from MDX info to answering questions on [www.gamedev.net](http://www.gamedev.net) and [www.graphicsdev.net](http://www.graphicsdev.net). He's also written tutorials and articles on C# and Managed DirectX. Best of all, he lives in South Africa! He presents the D# Hotlabs which occur in Durban on the last Saturday of each month. Dev.Mag were lucky enough to get an interview with Pieter after the D# Hotlab to obtain some wisdom and learn a bit more about him



**Could you please tell us a bit about yourself and what work you're currently doing involving game development?**

My name is Pieter. I'm a DirectX MVP and currently involved in [mdxinfo.com](http://mdxinfo.com), which is a collaborative effort between myself, Oliver Charles and Rim van Wersch.

**How do you feel the D# Hotlab went today?**

I felt everything went smoothly and very well.

**When and why did you start making games?**

I was 19. I like the idea of bringing my imagination to reality and being able to create my own worlds.

**What's your view on using a framework like Game Maker versus programming in DirectX?**

I'm a strong believer in learning the theory before using tools. Game Maker is a good tool to use if you don't want to jump directly into programming theory and it's good for prototyping games.

**You are the first person in South Africa to be recognised as a DirectX developer and to have been appointed a MVP DX. Please tell us a bit about how you achieved it and what it means?**

You don't aspire to be a MVP. The MVP status gets awarded to people that share [knowledge] through online and offline activities. It shows that we do have talent in South Africa when it comes to developing games using API's and hopefully people will realise it.

**Have you read Dev.Mag yet? If so, what do think?**

Yes, it's an awesome initiative from the game developers in South Africa.

**What do you think needs to be done to improve and expand game development in South Africa?**

I think people need to stop focusing on what they use and instead work together and share ideas to learn.

**Any words of advice to the SA game dev. community?**

Try be actively involved with things like Dev.Mag and the game development community. Also share your ideas and support each other.

**Thanks for your time and DevMag wishes you the best of luck for your future in game development and Hotlabs.**

You can find out more about Pieter and the work he's doing by going to [www.pieterg.com](http://www.pieterg.com) and [www.mdxinfo.com](http://www.mdxinfo.com).

INSOMNIAC



YOUR FREE E-MAGAZINE

[WWW.PRESSX.ZA.NET](http://WWW.PRESSX.ZA.NET)







# SECOND LIFE

**Do you like zombies? Do you like running them over with a car? There are a lot great things about Second Life and that is one of them.**

**S**econd Life drops you into the seat of a dead human. Just when you thought you would go to heaven, a necromancer grants you ... a second life. Unbeknownst to the necromancer, he has resurrected one of his arch enemies: a Human. He discovers you are a bit lost and takes you to his home, whereupon you discover that the humans have swept over the realm and are destroying all the wizards and magicians. The story is a bit flat, but it serves its purpose. It's funny at parts too.

The first thing that catches your eye about Second Life is the great style. The music blends nicely with the graphics to form an enjoyable environment. The sprites are well done, even though the world does seem a bit bare. There is also a nifty day/night cycle. Sometimes it even begins to rain and thunder. This really adds to the style and atmosphere.

The humour in the intro is quite good, although the intro itself is a bit long. After this, you are tasked to venture

into the wilderness and find a mushroom. The zombies arrive on the scene and the necromancer gives you a pistol ... The zombies are a bit slow, so the game is pretty easy. The controls are easy too. After you have mashed up some zombies and shot down some humans, you are tasked to get a car. The car is heaven. As you ride over zombies, blood splatters onto your bonnet and your tyre tracks leave a trail of blood! The UI is a bit bland and it is never explained what the bars do. The game is also buggy because you get stuck sometimes. If you did not save, you will have to sit through the lengthy intro again. Apart from the story mode, there is also a "Rampage" mode.

You are given guns and ammo and the task of destroying wave upon wave of zombies and humans. It is similar to Crimsonland, and quite enjoyable. Before we end off ... Second Life isn't quite finished. This is only the beta version, but in the end, the game still delivers to some extent. The atmosphere and style are the things that keep you playing, not the gameplay or story. All this game needs is a second life and it should really be enjoyable!

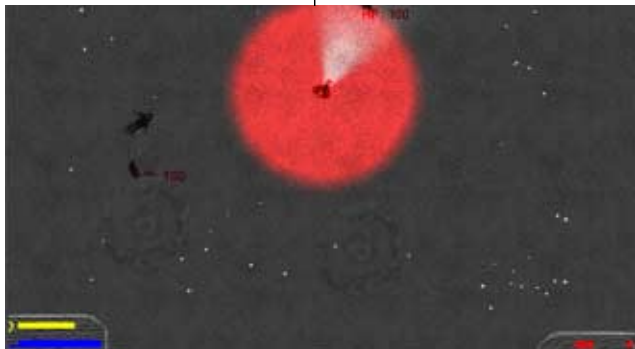
**TR00JG**

## FACT BOX

**DEVELOPER:** Unc1354am

**RELEASED:** November 2005

**CURRENT LINK:** <http://host-a.net/getfile.php?user=Unc1354m&file=SecondLife.zip>



**REVIEW**

# Why you should use Game Maker



**Game Maker is controversial within the programming community, to say the least. Purists hate it, engine designers look down on it and tutors laugh at it. At the end of the day, however ... well, it works.**

**F**or years, I dreamed of making my own games. I tried to learn BASIC on my spectrum, and then later I tried Pascal on my PC. I could make some graphical effects and add sound, but I could never make a game. I eventually gave up and moved on to other things, like dabbling with 3D modeling software and map editors in my spare time... Until a group of gaming friends wanted to start a modding team called "Apollo".

We were going to make the first Half-life 2 mod in South Africa and it all started with a bang. We did some modeling and scripting and imported them into the game. I was so proud to see my weapon in a game! The project died soon after because no-one was interested enough to go through the effort to learn the scripting language, this led to our demise.

However, in July last year I came across the NAG forums. Being an avid reader of NAG, I thought I could submit some suggestions on the forums. Then what did I find? Game.Dev? South African game developers? Apparently, these people thought it was easy to make games. Some school student told me about a product called Game

Maker and what you could do with it. Obviously this guy was a moron, right? How could somebody like this make games, when in the past I had tried so hard and failed? But when they told me how small it was and that it was free, I thought, 'Hell, I'll try anything that's 3MB and free'. Now for the amazing transformation part... Less than a year later I have 4 fully playable games under my belt and one RTS engine waiting to happen. Why? Because I didn't need to make an engine, learn a difficult programming language, or design collision-detection systems. I only needed to use my 3d modeling skills to make a few sprites, download some free sounds and actually make games.

Game Maker works and the proof can be seen in the quality products that can be produced with it. In our relatively small community, the quality and variety of our games is amazing. Why

struggle with the unnecessary and tedious task of coding a gaming engine when you can concentrate fully on making and building amazing and challenging games? That is why we are in this community. That's why we can have a game-making competition every second month! Don't get me wrong, I know there are people out there making games in other very popular programming languages such as Delphi, C#, and Pygame. This doesn't matter, as long as you are able to produce a game worthy of your imagination.

In short, don't be as negative and skeptical as I once was just because it sounds too good to be true. If you produce games, and not code and engines, it's all good. **HIMMLER**

(right) Game Maker, friend or foe?

**While purists hate it, it still is one of the most popular game creation tools used today.**





# ON THE BACK OF A NAPKIN: PART 2: VERTICES

Last time we went over some of the very basics of 3D graphics. This week we take a look at exactly what it is that vertices do for us and how they do it. We'll cover some more abstractions and end up explaining some of the single-letter terms that kept cropping up a few years ago.

**S**o, we know that to create a 3D scene on our monitors, we need triangles which are defined by three vertices. We know that most of the 3D data that is sent to modern cards is vertex information. But what exactly are these vertices?

The best answer is that a vertex isn't a set thing. Modern cards have many formats of vertex that they recognize, and programmers use whichever format best suits the trick they're trying to get right at the moment.

## POSITIONS:

The one thing that every single vertex will have is a position. After all, a vertex must describe a point in some

space so that we know where our (eventual) triangle is going to appear. It's this position that is turned into screen positions by our magic 3D cards. The actual format the position is stored in doesn't make much difference, as there are many different ways to do this. Typically vertices store x, y and z values (because we can describe 3D space using three axes, go look it up) but they also store a fourth value, w, that is pretty cool for lots of mathematical reasons. Honest.

## COLOUR

Vertices may or may not have some kind of a colour value, it depends on what a programmer or artist wants to happen. Vertex colours can be used to tint models, Diablo style. In 3D engines

that use Vertex Lighting (remember Quake 2 and 3) the vertex colour is determined by the interactions of the different lights in the scene.



(Above) **QUAKE 4.** The game makes a brilliant use of textures.

**TEXTURE CO-ORDINATES:**

Ah, textures. Textures are very important in 3D graphics. That's why 3D cards have such large amounts of RAM and why we need to be able to send data to them really fast:

Textures are big. But textures don't store any positional information, so we need to know where on a texture a vertex is. To do that vertices store a texture position (they don't store which texture they point to, that's handled by the engine so that different textures can be used if needed) but storing the individual pixel x and y values from the texture image wouldn't make much sense, so we use U and V scales in which 0,0 = the top left corner of the texture and 1,1 = the bottom right corner. The 0-1 scale is used so that textures can be different sizes without "breaking" the vertex. Vertices can store other data as well, but we're not going to go into that kind of detail now. Something else to remember is that vertex

shaders operate on vertex data, so some programmers prefer to add their own data to vertices that are going to be processed by their vertex shaders, but we'll cover that in another article.

**INTERPOLATION:**

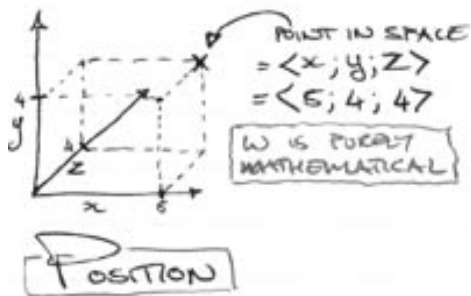
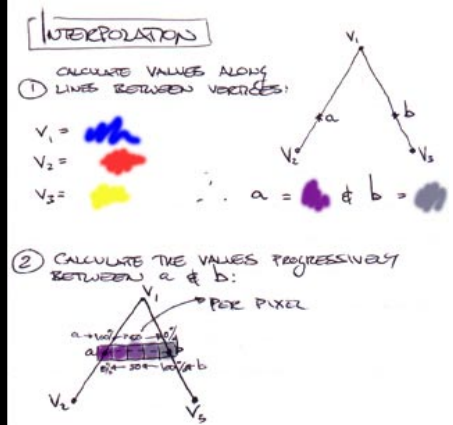
Great, a vertex provides us with a whole ton of positional information, but it's still an abstraction that's not tied to our final image on screen. How does the point-based information in a vertex get "spread" across the rest of the triangle to fill it with colour or a texture? The answer lies in a simple technique called Interpolation.

To interpolate between two values simply means to put each value on either end of a scale and to combine those values in different amounts based on where you are between them. In 3D graphics, we tend to use interpolation twice when drawing a triangle: First we interpolate all the values we need between the three vertices to create the

edges of our triangle; Then we interpolate horizontally over the triangle to get the values at each pixel in the triangle. This process of determining the actual pixel colours is called Rasterisation and determines the Fill Rate of your card (ie: How many pixels your card can colour in a second).

**SO, WHAT HAVE WE LEARNED?**

- Vertices tell us where things are.
- Programmers can pick what they want to store in vertices.
- Vertices will always have a position.
- Texture coordinates link vertex points and texture pixels.
- Interpolation "spreads" the data in vertices out to fill triangles.
- Colouring the actual pixels onscreen is called Rasterisation.

**DISLEKCIA****POSITION****INTERPOLATION**



# MOBILE GAME DEVELOPMENT IN JAVA



## PART 1: GETTING STARTED

This is the first in a series of tutorials that will introduce you to the wonders of creating games on your cellphone with the popular Java programming language. This series will not focus on teaching you Java – the official Java tutorial (<http://java.sun.com/learning/tutorial/>) already does that far better than I could ever hope to.

In this lesson, we'll get started by obtaining the tools we need and writing our first MIDlet (the name for a Java application designed to run on a Mobile Information Device such as your cellphone). In future lessons we will evolve our MIDlet into a fully-featured game that you can show off to everyone you meet. All the tools we'll be needing are free to download, you just need to go out there and get them. The following are what we'll be using:

**Java JDK:** This includes the Java compiler and certain core libraries.

<http://java.sun.com/j2se/downloads.html>

**WTK:** This has specialized libraries for mobile development, as well as some useful tools.

[http://java.sun.com/products/sjwtoolkit/download-2\\_2.html](http://java.sun.com/products/sjwtoolkit/download-2_2.html)

**NetBeans with Mobility addon:** This feature-packed IDE includes support for mobile development, and in later lessons we will use it to ease the process of building our projects. It can be downloaded from

<http://www.netbeans.info/downloads/download.php?type=5.0>,

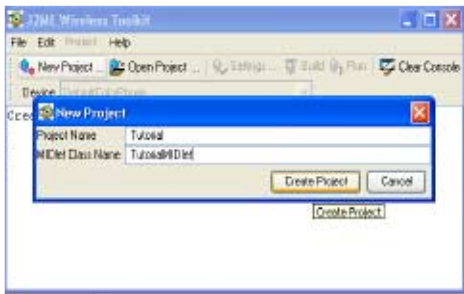
or you can request a free CD at

<http://www.netbeans.org/about/cd-form.html>.

Download and install each of these in the order listed, you can just stick to all default install settings. Now you're good to go!



*The default emulator displaying a list of available MIDlets.*



*Creating a new project in KToolbar.*



*Creating a new text file in Windows Explorer.*

### Code listing 1:

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

//Main MIDlet class
public class TutorialMIDlet extends MIDlet
{
    //Constructor
    public TutorialMIDlet () {}

    //Called when the app starts
    public void startApp ()
    {
        TutorialCanvas canvas = new TutorialCanvas();
        Display.getDisplay(this).setCurrent (canvas);
        canvas.repaint();
    }

    //called when the app pauses. eg. the phone rings
    public void pauseApp()
    {
    }

    //called when the app is destroyed
    public void destroyApp(boolean unconditional)
    {
    }
}

//Canvas class
class TutorialCanvas extends Canvas
{
    //paint the canvas
    protected void paint(Graphics g)
    {
        g.setColor(0xff00ff);
        g.fillRect(0, 0, getWidth(), getHeight());
        g.setColor(0xff0000);
        g.drawString("Hello World", 0, 0, 0);
    }
}
```

So, let's get your first MIDlet up and running!

Run the WTK Toolbar from Start->Programs->J2ME wireless Toolkit->KToolbar.

Click the New Project button, enter 'Tutorial' and 'TutorialMIDlet' respectively into the Project Name and MIDlet Class Name fields. Remember now and throughout the series that Java is case sensitive, and that includes class naming.

When you click Create Project, a settings form appears. Leave all the default settings and click OK. The KToolbar will print a couple of messages, including one similar to 'Place Java source files in <directory>'. Browse to that directory with Windows Explorer and right click in the empty space and choose New Text Document from the menu. Rename the new document to TutorialMIDlet.java.

Use Notepad to open the file, it can be found on the start bar at Start->Programs->Accessories->Notepad and enter the code in listing 1 in the file and save it.

Can you guess what the code does before you run it?

When you're ready to confirm your suspicions, click the Build button on the KToolbar, you should see a message confirming a successful build. **FLINT**



## A Little Bit About O-Notation

O-Notation is a term that computer scientists use to take a relatively simple concept and make it sound as complex as possible. The exact definition is that O-Notation: **Defines the asymptotic upper bound of an algorithm.**

**H**uh? Well, in plain English, it means that O-Notation is used to describe what the worst case running time of an algorithm can be. The most basic equation of O-Notation that exists is  $O(1)$ , where  $O()$  represents the operation or function being performed (ie. the algorithm), and '1' (or 'n' as we will see later on) represents the input parameter used to evaluate this function. The actual meaning of an  $O(1)$  O-Notation is that the algorithm described by it has a constant running time. That is, there are no parameters that can vary how much time the algorithm will take to complete. An example of an  $O(1)$  algorithm would be one that swapped the values of two given variables.

In most cases, the O-Notation of an algorithm can be determined just by looking at its structure. For example, an algorithm that linearly searches for the largest value in an unsorted list of numbers would be an  $O(n)$  sized algorithm, as it would need to check every number in the list for the largest one, with 'n' representing the size of the list being checked.

A more complex O-Notation of  $O(n^2)$  (read as n squared) would be used to describe a nested loop algorithm (such as an Insertion Sort). In this case, 'n' is the largest of the two loop extents as the process performed by the innermost loop will execute at most "iOuterMostLoopLimit \* iInnerMostLoopLimit" times.

Given those two variables, the worst case scenario would be if they both had the same value, which is where we get our  $(n^2)$  from. Comparatively, a binary based recursive algorithm (such as a Merge Sort) has a maximum running time of  $O(n \log_2 n)$ , where 'n' is the size of the list to sort. The complexity of this kind of algorithm is reduced greatly as the task is broken up into multiple steps using a "divide and conquer" approach. Don't be scared by the ' $\log_2 n$ ' as it is always going to be a smaller value than 'n' itself, and if you do the math for the O-Notation value of both of these algorithms where 'n' is the size of your list to sort, you will see that for whatever value of 'n' is use, the running time of the Merge Sort is always going to be less than that of the Insertion Sort.

Remember however that O-Notation is only used to generalize the worst case running time of an algorithm. It does not take into account such things as setup costs or the complexity of the operation(s) at each step of an algorithm, and therefore it is only one piece of the information required in order for you to be able to choose which algorithm is best suited for any given task. For reference, the most common O-Notation functions used for representing algorithms are:  $O(1)$ ,  $O(\log_2 n)$ ,  $O(n \log_2 n)$ ,  $O(n)$ ,  $O(2^n)$ ,  $O(n^2)$ , and  $O(n!)$ . **COOLHAND**



## Sorting Algorithms

I'm sure most of you can search through your data structures and sort your into the correct order. Now the real problem is what happens when you have thousands of values and your program starts lagging. Here are three methods of sorting data to optimise your code

### Method 1: Selection sort

This is the most basic and easiest way to sort data, unfortunately it is also the most inefficient. Let's assume we have a hand of cards that we want to sort. We pull out the first card and check if it is greater or smaller than the next card if it is bigger we put it ahead one space and check the next card. If it is smaller we leave it where it is and stop the check. We then take the next card and, starting from the first card we compare it till we find a bigger card. We must also put in check that if there are no more cards in the hand, then this card is the biggest. We continue to draw out every card and compare them to every other card. When all the cards have been checked we know the sorting is complete and the data is in the correct order.

### Method 2: Bubble sort

This method is very similar to the first method except each time we draw two cards from the hand. If the first card drawn from the hand is greater than the second we swap them. We then move one card along. Now we are checking the second and third cards. We continue to check all the cards in pairs. When we have checked all the cards we start again, except we know now that the last card is in the correct position so we can check all the other cards. After two checks we know the last two cards are in place. We continue checking the hand till we know all the cards are in the correct place. The advantage of this method over selection sort is we lose one card to check each time. Making it slightly faster.

### Method 3: Quick sort

The third and fastest method is also similar to bubble sort. This time we start comparing pairs from both sides at the same time. Now we know that the card in the middle is correct. We now have divided the data. Now for each half we sort again from the start and end of each half. This will divide it again (We now have four divisions). We repeat the process. This method is faster than the previous two because it divides into continuously smaller sections of data to sort.

Here is some code of how to implement quick sort in Java

```
public void sort(int start, int end) {
    if (end - start <= 1) {
        return;
    }

    int front = start, back = end;
    for (int i = start + 1; i <= end; i++) {
        if (data[start].value > data[i].value) {
            temp[front] = data[i];
            front++;
        } else {
            temp[back] = data[i];
            back--;
        }
    }
    temp[front] = data[start];
    for (int i = start; i <= end; i++) {
        data[i] = temp[i];
    }
    sort(start, front - 1);
    sort(front + 1, end);
}
```

Although these are by no means all of the sorts available, they're amongst the most popular and are definitely some of the most useful ones for any programmer to learn. **SQLID**





# SWARM OF GAMES!

Each year in April, [www.ludumdare.com](http://www.ludumdare.com) holds its annual 48 hour game programming contest. This year was no exception. Although the announcement was late, they carried through with the contest, gathering themes and getting a new contest website up and running

**N**one of the themes for this year were startling in their originality but a few interesting ideas were put forward. Some favourites were clear with both Magnetism and Swarms being punted heavily in the forums.

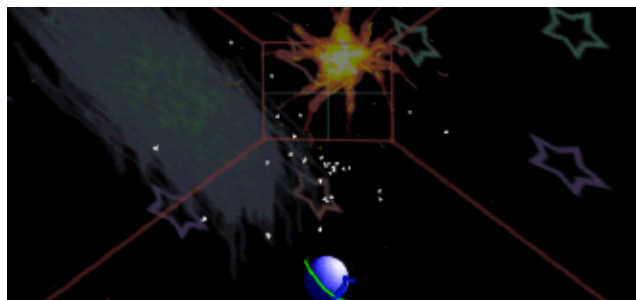
Another interesting theme suggestion was "Match Three", a contest theme that hasn't been seen elsewhere.

LudumDare has an interesting contest theme voting mechanism, with each contestant voting for a favourite, second favourite and third favourite. This allows the contestants to influence the vote even if their favourite doesn't win. The big day finally arrived the start of the contest.

The final theme was announced to be Swarms. This led to some interesting comments on the use of various flocking algorithms and various interesting uses of boids, however it seems that most of the contestants interpreted the theme to just represent a large mass of items, some organic and some of more interesting composition.

planned, I knew I had only a small amount of time available to me for the actual contest development work. Early on Saturday morning, I woke up and logged on to check what the theme would be. I then logged off and set the designer juices working. Some early ideas were around swarms of asteroids, honey bees and even

*(below) One of the better games that appeared in the contest.*

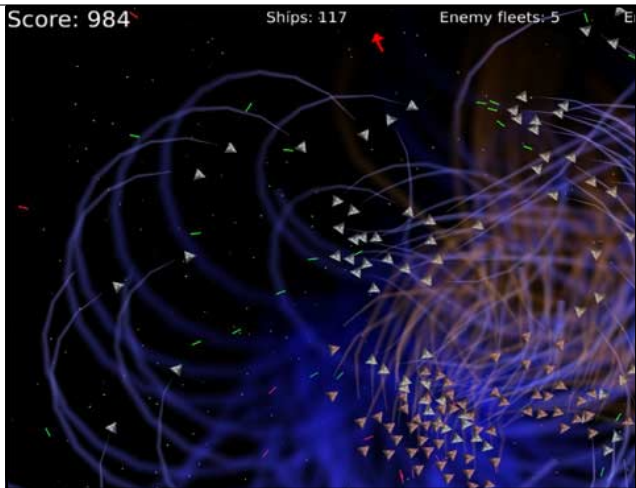


TALPIECE

swarms of cockroaches in a kitchen. After doing some other activities that were planned and having some LAN gaming fun with my children, I sat down to create my contest masterpiece. Initially I spent time working on the various game states and their transitions. When it finally came to making the game I still had not decided on an idea for my game. I most liked the Honey Bee idea and started working toward a bee hive management game.

I spent a reasonable amount of time working on some graphics for bees and more time on getting the bees to swarm together nicely. Late on Saturday afternoon I realised that there was not really enough time to get a full hive management game working and I needed to change the game concept to something similar. I loved the way the bees swarmed around together on the screen and wanted to keep it if possible. Looking at the swarming of the bees I decided that the swarming system would work just as well for flies. A nice simple idea would be to change the game I was making into a fly swatting game.

Pretty soon I had the game complete. Fly swatting fun for everyone. Quickly adding a high-scores table and some interesting backgrounds made with the gimp, I managed to round off the game nicely. After uploading the game I had an early night in. Overall, I was pleased with my final product. While nothing absolutely fantastic, it definitely was a complete game. On the LudumDare contest each of the entrants that completes a game becomes a judge. Each contestant



logs in onto the contest web site and downloads the other games. After playing the various entries each judge logs into the web site and enters scores on five categories "Innovation", "Graphics", "Sound", "Production" and "Fun". In addition space is provided for comments from the judges about each entry. and other entries of Each person that completed an entry should however be very proud of their achievement as of the 150 entrants only 66 managed a final completed game. If you are interested in seeing what can be

done in such a limited time, take a look at some of the following games:

Wee Ninja: [http://www.ludumdare.com/files/304\\_weeninja.zip](http://www.ludumdare.com/files/304_weeninja.zip)

Ultra Fleet: [http://www.ludumdare.com/files/305\\_jolle-ultrafleet-win32-final.zip](http://www.ludumdare.com/files/305_jolle-ultrafleet-win32-final.zip)

Files: [http://www.ludumdare.com/files/215\\_Files.zip](http://www.ludumdare.com/files/215_Files.zip)

These games are wonderful examples of what is possible with tight deadlines and a willingness to get a game completed and released. **CAIRNSWM**

# DIGITAL MAYHEM



Hmm... Oh, there's the link to the GM forums...



Tra La la la la...  
(Type.Click.Type)



LOADED

... !



PAUL

COMIC

*"So ... in case you hadn't heard, the Game Maker forums recently got hacked and some virus ruined everyone's day. No explosions, of course -- that's just Higushi's twisted imagination at work. We're concerned about him."*



O N L I N E

[devmag.googlepages.com](http://devmag.googlepages.com)