Chapter 00. Why Java for Games Programming?

This is where I revisit many discussions (i.e. arguments) about why Java is not a crazy choice for games programming. Possibly this chapter isn't necessary since you're already convinced of Java's qualities. But maybe you're not quite sure.

1. First the Advantages, but briefly...

One of my assumptions is that the reader (that's you) already has an introductory knowledge of Java, the sort of stuff gleaned from a semester's course at college. Near the start of that course, you'll have been regaled with Java's many advantages: object orientation, cross-platform support, code reuse, ease of development, tool availability, reliability and stability, good documentation, support from Sun Microsystems, low development costs, the ability to use legacy code (e.g. C, C++), and increased programmer productivity.

Rather than explain each of them again, I'll take a different approach. I'll discuss Java's suitability for games programming in terms of the typical misconceptions/complaints wheeled out by people who think that games **must** be implemented in C, or C++, or assembler, or whatever (so long as its not Java).

Here's the list, briefly:

- Java is too slow for games programming;
- Java has memory leaks;
- Java is too high-level;
- Java isn't supported on games consoles, so why bother using it;
- No one uses Java to write real games;
- Sun Microsystems isn't interested in supporting Java gaming.

2. Java is Too Slow for Games Programming

They mean that Java is slow compared to C or C++, the dominant languages for games programming at the moment.

This argument was valid when Java first appeared (around 1996), but has become increasingly ridiculous with each new release. Some figures put JDK 1.0 at *20 to 40 times slower* than C++. J2SE 1.4.2 (the current release) is typically **1.1-1.3 times slower**.

These numbers depend greatly on the coding style used. Java programmers must be good programmers in order to utilise Java efficiently, but that's true of any language. Jack Shirazi's *Java Performance Tuning* site (http://www.javaperformancetuning.com/) is a good source for performance tips, and links to tools and other resources.

A recent benchmarking of Java vs C++ by Keith Lea caused quite a stir (http://www.theserverside.com/news/thread.tss?thread_id=26634). He found that Java

may sometimes be faster than C++. The response from the C++ crowd was typically vitriolic.

The speed-up in Java is mostly due to improvements in compiler design. The Hotspot technology introduced in J2SE 1.3 enables the run-time system to identify crucial areas of code that are utilised many times, and these are aggressively compiled. Hotspot technology is relatively new, and it's quite likely that future versions of Java will find further speed-ups. For example, the forthcoming J2SE 1.5 is meant to be 1.2 to 1.5 times faster than its predecessor.

Hotspot technology has the unfortunate side-effect that program execution is often slow at the beginning until the code has been analyzed and compiled.

2.1. Swing is Slow

Swing often comes under attack for being slow. Swing GUI components are created and controlled from Java, with little OS support: this increases their portability and makes them more controllable from within a Java program. Speed is supposedly compromised because Java imposes an extra layer of processing above the OS. This is one reason why some games applications still utilise the original Abstract Windowing Toolkit (AWT) – it's mostly just simple wrapper methods around OS calls.

Even if Swing is slow (and I'm not convinced), most games don't require complex GUIs: full-screen game play with mouse and keyboard controls are the norm, so GUI speed is less of a factor.

2.2. My Program is Slow Because of Java

A crucial point about speed is *knowing what to blame* when a program runs slowly. Typically, a large part of the graphics rendering of a game is handled by hardware or software outside of Java. For example, Java 3D passes its rendering tasks down to OpenGL or Direct3X, which may emulate hardware capabilities such as bump mapping. Often the performance bottleneck in network games is the network.

3. Java has Memory Leaks

When C/C++ programmers refer to memory leaks in Java, it may mean that they don't understand how Java works. Java doesn't offer pointer arithmetic, and typical C-style memory leaks such as out-of-bounds array accesses are caught by the Java compiler.

However, they may mean that objects which are no longer needed by the program are not being garbage collected. This becomes an issue if the program keeps creating new objects, requiring more memory, and eventually crashes when the maximum allocation is exceeded.

This kind of problem is a consequence of bad programming style, since the garbage collector can only do its job when an object is completely dereferenced (i.e. the program no longer refers to it).

A good profiling tool, such as JProfiler (http://www.ej-technologies.com/products/jprofiler/overview.html), can be a great help in identifying

code using excessive amounts of memory. JProfiler is a commercial product; many open source profilers are listed at http://java-source.net/.

Another memory related complaint is that the garbage collector is executing at poorly timed intervals, causing the application to halt for seconds while the collector sweeps and cleans.

The JVM comes with several different garbage collectors, which collect in various ways, and can be selected and fine-tuned from the command line. Information on the performance of the chosen collector can be gathered and analysed. A good hands-on explanation of this topic, centered around the JTune visualization tool, can be found at http://www-106.ibm.com/developerworks/java/library/j-perf06304/.

4. Java is Too High-level

This complaint is the age old one of abstraction versus speed and control. The details of the argument often include the following statements:

1. Java's use of classes, objects and, inheritance add too much overhead without enough coding benefit;
2. Java's machine independence means that low-level, fast operations, such as direct Video RAM I/O, are impossible.

Statement (1) ignores the obvious benefits of reusing and extending Java's very large class library, which includes high-speed I/O, advanced 2D and 3D graphics, and an enormous range of networking techniques, from lowly sockets to distributed agents. Also forgotten are the advantages of object oriented design, typified by UML, which makes complex, large real-world systems more manageable during development, implementation, and maintenance.

Statement (2) impacts gaming when we consider high-speed graphics, but it's been addressed in recent versions of Java. J2SE 1.4 introduced a full-screen exclusive mode (FSEM), which suspends the normal windowing environment, and allows an application to more directly access the underlying graphics hardware. It permits techniques such as page flipping, and provides control over the screen's resolution and image depth. The principal aim of FSEM is to speed up graphics-intensive applications, such as games.

Statement (2) also comes into play for game perpherals, such as joysticks and gamepads; machine independence seems to suggest that 'non-standard' I/O devices won't be useable. Java games requiring these types of devices can utilize JNI, the Java Native Interface, to link to C or C++ and so to the hardware. There's also JInput, a new game controller API, due to be finalised early in 2005.

An interesting historical observation is that the gaming community use to think that C and C++ were too high-level for fast, efficient games programming, when compared to assembly language. Opinions started to change only after the obvious success of games written in C, such as Doom and Dungeon Master, in the mid 1980s. Also important was the appearance of cross-platform development tools that supported C, such as Renderware.

5. Java isn't Supported on Games Consoles so Why Bother?

Unfortunately, this criticism has some justification.

Video gaming is a multi-billion dollar industry, with estimates placing revenues at $US 29 billion by 2007. The market will cater to over 235 million gamers.

PCs and game consoles account for almost all the income, but only about 10-20% of it is from PCs, the majority coming from three consoles: Sony's PlayStation 2 (PS2), Microsoft's Xbox, and Nintendo's GameCube. Sony is the dominant console maker, having nearly twice as many units in homes compared to Microsoft and Nintendo combined. Microsoft accounts for about 95% of the desktop PC market.

Arguably, there are only two important games platforms: the PS2 and Windows, and Java isn't available on the PlayStation.

This problem has long been recognized by Sun: back at the JavaOne conference in 2001, Sony and Sun announced their intention to port the JVM to the PS2. Nothing has been released, but there are persistent rumours about a JVM on the PlayStation 3, earmarked to appear in 2006.

In the future, Java may have a better chance of acceptance into the closed-world of console makers because of two trends: consoles are mutating into home media devices, and the meteroic rise of online gaming. Both require consoles to offer complex networking and server support, strong areas for Java and Sun.

The Phantom console from Infinium Labs was announced at JavaOne in 2004 (http://www.phantom.net/index.php). It's essentially a PC running an embedded Windows XP, with a nVidia graphics card, a hard drive, and a broadband connection. Most importantly for Java gaming, it will come with a complete JRE. It was also demoed during E3 (Electronic Entertainment Exposition) in 2004, where it was shown running *Law and Order: Dead on the Money* (which uses Java 3D).

Diehard programmers may point out that it's already possible to get Java running on a PS2. One approach is to install Kaffe, an open source, non-Sun JVM, on top of PlayStation Linux. Kaffe can be obtained from http://www.kaffe.org/; details on Linux for the PlayStation are at http://playstation2-linux.com/. The gallant programmer will also need a Java-to-bytecode translator, such as Jikes (http://www-124.ibm.com/developerworks/oss/jikes/).

The Linux kit adds a hard disk to the PS2, so this development strategy will not work for ordinary PlayStations. Configuring the software looks to be far beyond the capabilities (or desires) of ordinary console owners, and I couldn't find any documentation about using Jikes/Kaffe on a PS2. The PlayStation only comes with 32Mb of RAM, while a typical JVM and its libraries requires 5-10Mb, so how much would be left for a game once Linux was up and running?

The difficulties with this approach should be compared to the availability of feature rich C/C++ tools and engines for consoles, such as RenderWare (http://www.renderware.com/) and Gamebryo (http://www.ndl.com/). They have a track record of best-selling games, and can port games across the PS2, Xbox, GameCube, and PCs.

The lack of Java on consoles is a serious issue, but the remaining PC market is far from miniscule. Microsoft estimates that there are 600 million Windows PCs at present, growing to more than 1 billion by 2010. Games on PCs benefit from superior hardware, such as video cards, RAM, and internet connection, to offer more exciting game play. There are many more PC games, particularly in the area of multiplayer online games. It is throught that the 40% of all gamers will start playing online by 2005. Revenues may reach US$ 1.1.billion by 2008.

Another rapidly expanding market is the one for mobile games, with sales of US$ 530 million in 2003, potentially rising to US$ 1.93 billion in 2006. There are perhaps 200 million Java-enabled phones at the moment.

6. No One Uses Java to Write Real Games

The word 'real' probably means commercial games. The number of commercial Java games is small compared to ones coded in C++ or C, but the number is growing, and many have garnered awards and become bestsellers:

Puzzle Pirates by Three Rings (http://www.puzzlepirates.com/), a multiplayer pirate game that includes Tetris- or Columns-like puzzles at various points. Both the client and server are written in Java. It won several awards during 2004, including the Technical Excellence and Audience Choice prizes at the Game Developers Conference.

Chrome by Techland (http://www.chromethegame.com/en/show.php). A futuristic multiplayer FPS made up of 14 different missions, in an amazing variety of landscapes. It received a Duke's Choice Award from Sun Microsystems in 2004 for the most innovative product using Java technology.

Law and Order II, by Legacy Interactive. (http://www.lawandordergame.com/index2.htm) A detective game written in Java, Java 3D, and Quicktime for Java. The first *Law and Order* sold over 100,000 units.

Kingdom of Wars, set in the fantasy world of Jairon, by Abandoned Castle Studios (http://www.abandonedcastle.com/).

Alien Flux, an exciting arcade shoot-em-up from Puppy Games (http://www.puppygames.net/info.php?game=Alien_Flux).

War! Age of Imperialism (http://www.eaglegames.net/products/WAR_AOI/wai.shtml), a computer version of the award-winning board game from Eagle Games.

Runescape by Jagex (http://www.runescape.com) is a massive 3D multiplayer fantasy adventure game. Clients can use a Java applet to play, or download a Windows-based client application.

Star Wars Galaxies from LucasArts (http://www.lucasarts.com/products/galaxies/) has its game logic coded in Java.

*IL-2 Sturmovik*and the new version *IL2-Forgotten Battles* by Ubi-Soft (http://www.il2sturmovik.com/). Award winning WW I aerial combat using Java and C++.

Pernica by Starfire Research (http://www.starfireresearch.com/pernica/pernica.html)
An online fantasy role-playing game first implemented in Java 3D, recently ported to Xith3D.

Cosm from Navtools, Inc. (http://www.cosm-game.com/)
Another fun online fantasy-based role-playing game.

C&C Attack Copter. A free online action game based on the Command & Conquer series from Electronic Arts (http://www.eagames.com/free/home.jsp).

Roboforge by Liquid Edge Games (http://www.roboforge.com). Train a 3D robot to fight in online tounaments. It was given an "Excellent 87%" by *PC Gamer Magazine*.

Galactic Village by Galactic Village Games (http://www.galactic-village.com), a massively multiplayer strategy game, written entirely in Java. Not yet finished, although alpha versions have been appearing.

Wurm Online by Mojang Specifications (http://www.wurmonline.com/). Another massively multiplayer fantasy game, written in Java. Still at the alpha stages of development, but the screenshots look great.

Jellyvision (http://www.jellyvision.com/) used a mix of Java and C++ in their popular *Who wants to be a Millionaire* (2000) and *You don't know Jack* (1995) games. They employed Java for the game logic, an approach also used in *Majestic* (2001) by Electronic Arts.

Java was utilized as a scripting language in the acclaimed *Vampire - the Masquerade: Redemption* (2000) from Nihilistic software (http://www.nihilistic.com/).

Tom Clancy's Politika (1997) from Red Storm Entertainment (http://www.redstorm.com/) was written in almost pure Java. Both *Shadow Watch* (2000) and *Tom Clancy's ruthless.com* (1998) mixed Java and C/C++.

A good source for non-technical lists of Java games, both commercial and freeware/shareware, can be found on the Java games pages at java.com (http://www.java.com/en/lifestyle/games/). It divides games into several categories: action, adventure, strategy, puzzle, cards, sports, and so on.

6.1. Freeware/Shareware Games

There are many, many Java games out on the Web, but finding a game that's well written requires a careful search. Many applets date from the late 1990's, and were designed using the outdated JDK 1.0 and 1.1 with their feeble media APIs (graphics, sounds, etc). The initial Java euphoria produced some less than exciting games, more concerned with technical trickery. This large pool of useless applets got Java labelled as a toy language.

Recent versions of Java are quite different: speed is vastly improved, and APIs crucial to gaming, such as graphics and audio, are of a high quality. There's been a move away from applets towards the downloading of client-side applications using Java Web Start.

Java's backwards compatibility allows the applets from 1996-8 to be executed, and they'll often run quicker than originally. However, it's probably best to steer clear of these Java dinosaurs, and look for more modern code.

There are numerous Web sites with Java games. The emphasis of the following list is on applications/applets for playing:

- Java Games Factory (JGF), http://grexengine.com/sections/externalgames/. There aren't many games at this site (about 50), but they're all high quality. The aim is to show off the variety of modern Java game technologies.
- ArcadePod.com, http://www.arcadepod.com/java/
  Over 750 Java games, nicely categorized.
- Java 4 Fun, http://www.java4fun.com/java.html
  Similar in style to ArcadePod, and a good set of links to other sites.
- jars.com, http://www.jars.com
  A general Java site with a ratings scheme. There are many games, but a lot of them are old applets.
- Java Shareware, http://www.javashareware.com/
  Another general site: look under the categories: applications/games/ and applets/games.
- Java Games Central, http://www.mnsi.net/~rkerr/
  A personal Web site which lists games with ratings and links. It was last updated in 2001.

Some of my favourite freeware/shareware games at the moment:

- *Super Elvis* (also known as *Hallucinogenesis*) by puppygames.net (http://puppygames.net/), which won the Sun Microsystems 2004 Technology Game Development Contest. Super Elvis can be downloaded using Java Web Start from
  http://www.puppygames.net/downloads/hallucinogenesis/hallucinogenesis.jnlp
- *FlyingGuns* (http://www.flyingguns.com/), a 3D multiplayer WW1 fighter plane game/simulator. This came second in the contest, but is my favourite.
- *Cosmic Trip* (http://www.mycgiserver.com/~movegaga/cosmictrip.html), an arcade style 3D game with striking graphics.
- *Squareheads* (http://home.halden.net/tombr/squareheads/squareheads.html) a multiplayer FPS (it came third in the developer contest).
- *Escape* (http://javaisdoomed.sourceforge.net/), a Doom-like FPS.
- *CazaPool3D* (http://membres.lycos.fr/franckcalzada/Billard3D/Pool.html), a pool game that allows online (single/multiplayer) play in an applet or as a standalone application.

Programmers looking for source code should start at one of the following sites:

- SourceForge, http://sourceforge.net/search/
  SourceForge acts as a repository, and management tool, for software projects, many with source code. A recent search for (java + game) returned over 70 projects that had 40% or greater activity. One of the drawbacks of SourceForge is that it can be quite difficult to decide whether a project is vaporware or not. Good

projects, which have been completed, will show low activity after a time, dropping down the list of search results.

- FreshMeat.com, http://freshmeat.net/
  Freshmeat maintains thousands of applications, most released under open source licenses. The search facilities are excellent, and can be guided by game category terms. The results include rating, vitality, and popularity figures for each piece of software. A recent search for Java in the Games/Entertainment category returned nearly 70 hits. Many applications turn up at both SourceForge and FreshMeat.
- The "Your Games Here" Java Games Forum, http://www.javagaming.org/cgi-bin/JGNetForums/YaBB.cgi?board=Announcements
  Implementors can post links to their games, and (perhaps more importantly) users can post their opinions as follow-ups.
- Code Beach, http://www.codebeach.com
  CodeBeach has a searchable subsection for Java games that currently contains nearly 90 example.
- Programmers Heaven, http://www.programmersheaven.com/zone13/
  It has a 'Java zone' containing some games.

7. Sun Microsystems isn't Interested in Supporting Java Gaming

The games market isn't a traditional one for Sun, and it'll probably never have the depth of knowledge of a Sony or Nintendo. However, the last few years have shown its increasing commitment to gaming.

J2SE has strengthened its games support: version 1.5 (due out at the end of 2004) has a decent nanosecond timer at last, version 1.4 introduced a full screen mode and page flipping in hardware. Faster I/O, memory mapping, and support for non-block sockets, which is especially useful in client/server multiplayer games, also appeared first in 1.4. Version 1.3 improved graphics and audio support.

Java extension libraries, such as Java 3D, the Java Media Framework (JMF), the Java Communications API, Jini, and JAXP (Java's peer-to-peer API) all offer something to games programmers.

Sun started showing an interest in gaming back in 2001, with its announcement of the Java Game Profile, a collaboration with several other companies, including Sega and Sony, to develop a Java gaming API. The profile was perhaps too ambitious, and was wound up at the end of 2003. However, it did produce three game-focussed technologies, a Java binding for OpenGL called JOGL, a binding for OpenAL (a 3D audio library) called JOAL, and JInput.

Part of the 2001 initiative was the creation of the JavaGaming.org website (http://www.javagaming.org), initially manned by volunteers. In 2003, the Game Technology Group was formed, and JavaGaming.org received a substantial makeover as part of the creation of the new java.net portal (http://www.java.net) aimed at the technical promotion of Java.

java.net hosts many discussion forums, user groups, projects, communities, and news. The communities include: Java Desktop, Java Education and Learning, Java Enterprise, and Java Games.

The Java Games community pages can be accessed through http://www.javagaming.org or http://community.java.net/games/. The site includes Java games forums, projects, news, weblogs, a wiki (http://wiki.java.net/bin/view/Games/WebHome), and links to games affiliates.

Numerous Java game forums can be accessed from http://www.javagaming.org/cgi-bin/JGNetForums/YaBB.cgi. These are probably the best sources of technical advice on Java gaming on the Web, with over 3400 highly opinionated registered users. Discussion topics include Java 3D, Java 2D, Java Sound, J2ME, networking, online games development, performance tuning, JOGL, JOAL, and JInput. There are also sections on projects and code examples.

The project sections (https://games.dev.java.net/) mostly concentrate on JOGL, JOAL, and JInput, but the games-middleware and games-forge sections are wider ranging. The games-forge projects include chinese chess, jbantumi (a strategic game from Africa), and an online fantasy football management system.

The most relevant Java user group for gaming is GameJUG (https://gamejug.dev.java.net/). Its sections include online and downloadable Java games, presentations and articles, lists of Java game programming web sites, and a collaborative Web page and mailing list for teachers of Java game programming. I'm currently the GameJUG president, a role which sounds grander than it really is. The real work is done by David Wallace Croft and James Richards.

Sun's substantial presence at http://community.java.net/games/ is mostly as a host for community forums and open source projects (or projects with licenses very close to open source). The projects include JOGL, JOAL, JInput, and Java 3D. Sun is relying on community involvement to move these projects forward, since the Game Technology Group is quite small.

One in-house product is a server architecture for massively multiplayer online games, the Sun Sim Server, first demoed at the Game Developers Conference in 2004. This focus isn't surprising since Sun makes its money from selling server hardware. Online multiplayer gaming is a potential growth area for its servers.

