Jason Brownlee

Mobile Game Engines

Interviews with Mobile Game Engine Developers

Mobile Game Engines
Interviews with Mobile Game Engine Developers

Chapter 1

Steffen Itterheim

Kobold2D

1.0.1 Background

Brownlee: Where in the world are you located?
Itterheim: Ingelheim, Germany.

Brownlee: Who do you work for and what is your current role?
Itterheim: I'm self employed. I wrote a popular book about cocos2d-iphone called "Learn Cocos2D". I blog on the website with the same name. I manage the Kobold2D game engine that wraps cocos2d in an easier to use package. I write a lot in general.

From time to time I do freelance work but only if it fits my bill (and pays my bills) and it's an interesting project.

Brownlee: Could you please introduce yourself?
Itterheim: I've been a gamer since Atari VCS 2600, aging through C-64 and Amiga to PC. Since 2009 I've been exclusively on Mac OS and iOS development, and most of my gaming time I spend on the XBOX360, and the occasional Steam game.

I've been working in the game industry since 1999, at first Gameboy (Color/Advance) games and later at Phenomic on PC strategy and role-playing series Spellforce and under EA on the real-time card trading game Battleforge.

My professional roles were very diverse, I worked as game designer, programmer and manager. I programmed game logic, small and large tools, script engines, dialog parsers, database stuff in C++, C# and Lua. I designed game levels, quests, user interfaces. The only job I was never really happy with was management. Mostly because I didn't know what to do, what was needed, and I still had my previous responsibilities. So like many before and after me in the same situation, I kept doing what I knew best.

Brownlee: How long have you been programming?
Itterheim: Well actually this is a bit difficult. Until around 2006 I hadn't been a full-time programmer by title or tasks. On the other hand, if you count scripting languages it goes way, way back. BASIC on the C-64, some Amiga Basic, MS-DOS batch scripts, a bit of Turbo Pascal. But nothing serious, just enough to get stuck way too early. Later came all the Quake and other game scripts, which landed me a job working on Gameboy games where the game logic was implemented in a proprietary, statemachine scripting language with a visual dropdown interface. That was awesome!

My first hardcore, low-level programming session wasn't until around 1993 when Doom was fresh and hot. I used C to poke around in the WAD files, displaying the textures on screen and saving them to disk as bitmap files. It was a hack but it worked. Since then I've been spending time educating myself on C, C++ and various scripting languages, mainly Python and Lua.

Brownlee: What are some of the game engines you have worked on?
Itterheim: Almost all of them were proprietary internal developments. If you count Game Maker as a game engine, then I spent a lot of time with that. I'd still wholeheartedly recommend this tool to anyone who wants to understand how to make games, not just program them.

1.0.2 Your Game Engine

Brownlee: What is the full name of your game engine?
Itterheim: It's Kobold2D.

Brownlee: Could you please describe your game engine?
Itterheim: First I like to say that I feel a bit uneasy calling it "my game engine" because it is powered by cocos2d-iphone, and Kobold2D is more like a Linux distribution that takes the kernel and puts a nice GUI on top of it, bundles other software and adds some extra features.

On the other hand cocos2d has done a fairly good job giving birth to Kobold2D because it kept annoying me so much that I felt I needed to apply a level of polishing, not just for my own sake. For example, it's been over a year that ARC[1] is available, and cocos2d still doesn't offer ARC support out of the box. Users are told to work through someone's tutorial

[1]Automatic Reference Counting

on the Internet which by now is pretty much outdated. Whereas Kobold2D has ARC enabled in all projects since the early days.

What you also get with Kobold2D are a lot more example projects. Several popular libraries are ready to use. Entire API references for all libraries, as HTML and XCode help is available. Lua support is there. Mac and iOS development in a single target. An installer. Tools to create new projects and an easy way to make template projects. A tool to upgrade your project to newer Kobold2D versions. And additional code that I've developed over time, such as pixel-perfect collision detection and taking screen shots.

I take care of the much needed maintenance when something out of the ordinary happens, such as a new iOS or XCode version. If there's a compatibility issue, it's the first thing I check and fix. With cocos2d it usually takes a while until those fixes are in an official or even beta release. Even if there's a solution, users first have to find it, then apply it manually. That's why Kobold2D exists. To me a game engine is much more than just the code.

Brownlee: What platforms does it support?
Itterheim: iOS and Mac OS X.

Brownlee: What programming language(s) is it written in?
Itterheim: Objective-C and Lua.

Brownlee: What is the current price and license structure of your game engine?
Itterheim: It's free. MIT License. Although, I have a store where you can buy my own products and affiliate products.

Brownlee: What are the engines top 10 core features?
Itterheim: Besides what cocos2d offers that would be:

- Gesture Recognition.
- Game Center support.
- iAd and Admob support.
- Accelerometer and Gyroscope input with smoothing.
- Mac keyboard and mouse input handling.
- Pixel-perfect collision detection.
- Taking screen shots.
- UIView hit test support.

- Sprite animation helpers.
- Integrated Lua.
- Extended cocos2d classes with convenience methods.
- Plus all the other libraries ready for use, including cocos2d-iphone-extensions, SneakyInput, Chipmunk Spacemanager, ObjectAL and iSimulate.

Brownlee: When was it first released?
Itterheim: I think August 2011.

Brownlee: Could you estimate how many games have been built and released using your engine?
Itterheim: Probably in the hundreds from the tweets I see. Not many developers are too keen to mention the engine they used, simply because it's not really relevant to them. So the numbers are in the dark.

I wish I knew how cocos2d comes up with their usage figures, and whether that includes Kobold2D projects as well.

Brownlee: What are some well known or notable games created with the engine?
Itterheim: I can't recall. There were a couple games that were quite nicely done, but I see so many games and apps I couldn't even remember whether that was a Kobold2D game or not if I played it a second time.

I'm working on a game that started late 2010 and I ported it to Kobold2D earlier this year. It's a really cool game, it just takes very long to develop. It's an on-off thing. I think that'll work as a show-off project for Kobold2D.

1.0.3 Building Your Game Engine

Brownlee: Why did you start working on this game engine?
Itterheim: Mainly because cocos2d frustrated the hell out of me, and because I've seen others struggle with it too. So I wrote a book to document cocos2d. Then I felt I knew enough about cocos2d that I could improve on it.

Take for example the installer script. As cocos2d became more popular more and more developers stopped by who had no idea how to run that thing. And even if they did it would fail frequently. Stuffing a copy of the cocos2d sources into every new project you create is just a maintenance nightmare. Especially because you can't easily upgrade an existing project either. Many have failed to do so correctly. And keep failing. Not to mention trying to create a cocos2d static library, which you have to do if you want to use ARC.

That's some of the reason why I did Kobold2D. I also did this as a spin-off project because I didn't want to have to compromise. Some of the structural changes (the XCode project itself, how templates work, and so on) were pretty radical but very necessary to move forward.

Brownlee: What was the first element you had working in the game engine?
Itterheim: The XCode project. Because it's the first thing I needed to get right so the project template, the static libraries, the upgrade process, the build configurations and many other things would work.

Brownlee: Did you initially develop alone or in a team?
Itterheim: Alone. There were several contributions from users though, including a Japanese translation of the documentation.

Brownlee: Why create a new game engine rather than extend or license an existing engine?
Itterheim: Well actually I did use an existing engine. Seriously, don't re-invent the wheel.

Brownlee: What are some challenges in supporting multiple mobile platforms?
Itterheim: Actually it's the project itself. This is one thing I fixed in Kobold2D. Cocos2d allows you to target both iOS and Mac OS X. However you have to create two separate projects to do so, and there's no way to merge them. This is specifically a problem with assets (including source code files) because you have to add, move or remove them twice.

In Kobold2D you have a single project with two targets, one for iOS and one for Mac OS X. If you write your code carefully you can just switch targets and build and run on the other platform. Even if you're not so careful there's very little code you need to fix.

As for the engine itself, the most recent iOS and XCode versions introduces breaking changes. All of them were fixed easily within an hour, half a day if you include testing. Even if the errors were in other libraries like cocos2d or Chipmunk I take care of them.

Brownlee: What are your thoughts on the trade-off between engine performance and flexibility?
Itterheim: This is one of my pet peeves. For a long time cocos2d was obsessed with performance. A percent here, a percent there. Nothing that really mattered though while at the same time an influx of new devs were struggling just to get started.

I certainly favor flexibility over performance. Ease of use over performance. Features over performance. Documentation over performance. Anything that gets the user's work done faster over performance.

Performance is really the last thing a game engine developer should consider. All of the other issues are more important. What good use is a game engine that's fast but users have trouble working with? You can always optimize it later on. A base level of performance is of course mandatory but not really hard to achieve.

I need a working engine that supports me in my daily work. One that I can extend. One whose source code is easy to read and understand and debug. One that's documented well.

I also think game developers specifically have a weird attitude towards performance. The way they ask about how an engine performs, and that often being one of the first questions, it's ridiculous. Just to put this in perspective, let's say I'm interested in buying a car and go to a car dealer. For every car they show me I would ask pretty much right away "How fast can it go?". Then they might be so annoyed they'd actually show me a car that's really, really, really fast, and then I'd protest "No, that's too expensive, I want a free (and open source) car!".

Asking about performance is something you may do before you close the deal, but really the manufacturer's speed limit doesn't tell you anything anyway. I drove a car that was specified as 170 km/h max speed and I drove it past 210 km/h. Another car, same max speed, and it barely managed to reach its manufacturer top speed even under ideal conditions. And that's in Germany, where you are legally allowed to go that fast on some roads. Finally, you have to consider the driver. How often do you see a high-powered car driving even below the speed limit? It's twice as annoying if you know they could go faster easily if they just cared enough not to impede the traffic that's behind them. But enough about cars.

The analogy for game engines being that the code you're going to write has a far greater impact on performance than the game engine itself. Anyone can make a super fast game engine slow quickly by writing crappy code. The real art of writing high-performance code lies in using any game engine, doing tons of stuff with it to create your game and still have it run smoothly. Writing high performance code is the job of the game developer, the engine developer can only strive to make the rendering fast under ideal circumstances and hope that game developers follow the engine's guidelines and best practices. Which quite frankly most of the game developers do not. Because doing so requires not just excellent documentation but also reading it, and understanding the game engine well enough to make the right decisions.

In my opinion cocos2d made a couple of exemplary bad decisions in favor of performance. For example, they included a hashed set implemented in C and a C array (CCArray). Now they have a hard time updating this code base to ARC because of all the embedded C code. Besides that, it made reading the source code so much harder for Objective-C developers, exactly why they attracted so many ObjC developers in the first place. And by replacing NSMutableArray with CCArray, several nasty bugs and even performance issues occurred.

In my opinion a couple percent better performance doesn't justify using an ever so slightly broken alternative piece of code. Stick with what's working and reasonably fast, and optimize only the absolute worst performing parts of your engine. Yes, it's a problem if the performance is bad. In that case it should be treated like a bug. Fix it where it's broken. If the performance is adequate, leave it alone. The code the user writes will have a greater (negative) impact on the resulting app's performance. You can rarely improve that noticeably by improving your engine's performance.

Instead of optimizing performance, spend that time writing better support code or tools for your users. I know that for programmers it's tempting to work on performance, because it can be measured so beautifully and accurately. The results then give you immediate feedback that your work was a success. It feels great! The problem is, you just wasted time working on something your users won't even notice. Performance is not a feature, it's a byproduct.

Brownlee: Which of graphics, physics and input handling were more of a challenge?
Itterheim: The biggest addition to Kobold2D is KKInput, an input wrapper that allows you to poll input states. This makes it easier to to use than the event-driven approach.

At its core, KKInput gets the input events and remembers them for the time being, which usually means until the next frame. So the user can at any time from any class or function check if a certain input event has occurred in this frame.

KKInput wraps iOS' gesture recognizers, gives you access to the gyroscope and also provides smoothing (high-pass and low-pass filtering) of acceleration values. On the Mac side I had to wrap all keyboard and mouse keys into a simple to use interface.

Brownlee: What is your approach for staying abreast of changes to the underlying technology and device APIs?
Itterheim: Wait and see. When the final release of a new iOS, Mac OS or XCode version comes out, I test all the projects for issues. They're usually easily fixed.

If you're working on projects that might get released in the next 3 months or so, it's a really, really, really stupid idea to install the iOS beta versions. Because you can't downgrade, and you can't release until the beta version is released by Apple, and you can't get support on the Internet for beta version issues either. It's worse if you might get a lucrative, quick 2-week job to make an app and you just installed the first beta version of the new iOS. Tough luck. Now you got to find a way to uninstall the beta version, or just hope for the best (and hope, it's not a strategy).

I wish Apple would allow for a way to switch between stable and beta versions of iOS, XCode and so on. But it's really up to you. Get another device and another Mac or at least a separate boot drive to be able to make the switch.

Brownlee: What is one big element of the game engine you wish you could go back and do differently?
Itterheim: Well maybe if I could go back and I had more time to spend on it.

I initially included Wax (Lua runtime scripting for iOS) but I really only needed the Lua init stuff. I should have just left it out. Same for cocos3d perhaps, it's not really catching on as I thought it might. Makes sense if you consider that even Unity is free, and you get all the important tools you need when you make 3D games.

1.0.4 Maintaining Your Game Engine

Brownlee: Do you currently think of the game engine as a project or a product?
Itterheim: It's definitely an ongoing effort, a service. At the very least one update for every major new release, be it iOS, Mac OS, XCode, Cocos2D or something related.

Brownlee: What development tools do you use in a typical week?
Itterheim: Well, there's XCode obviously. I also use the text editors Smultron and Sublime Text 2 frequently. Perforce and github (Tower) for source control. And the Hudson build server. I also use all the Cocos2D tools like Texture Packer and Glyph Designer frequently.

All pretty standard if you ask me.

Brownlee: How many people are currently working on the game engine and how is the effort structured?
Itterheim: Just me. Whenever I got some spare time.

Brownlee: What are some common maintenance tasks in a given week?
Itterheim: Not much day-to-day really. I'm developing a game on the side built with Kobold2D. So I'm using Kobold2D in my day-to-day work and whenever I come across something that needs to be fixed or added, I usually end up doing that right away. I recently wrote the 3rd edition of my book, and I'm deep into writing Essential Cocos2D, a online reference

documentation for Cocos2D and Kobold2D focusing on the tech details and best practices, but also user support. I'm sure out of that will come a more lively development cycle for Kobold2D as well.

Brownlee: Who wrote the documentation for the game engine?
Itterheim: That was me, except for the API references which was made by the respective source code library developers. A Kobold2D user also translated the Kobold2D documentation to Japanese.

Brownlee: What are the core benefits in having a development community around your game engine?
Itterheim: You get feedback, you learn what people want and need. This is interesting because I've come to use a few things they use too. And learned different ways to approach an issue. It keeps making me look up things nobody really knows, which I find intriguing to figure this stuff out and share it.

Brownlee: How do you generally engage the community?
Itterheim: Infrequently I make a poll or survey, but most of the time it's blog posts. Every two weeks I write a #iDevBlogADay post, and typically it's about iOS coding topics.

When you read this, Essential Cocos2D should be available. It's not just reference documentation but also a membership program where I help developers become better coders. I've always liked answering technical or design questions, I like to translate between the two sides.

Technology is useless without design, and many cool designs wouldn't exist if it weren't for technology.

Brownlee: What are some ways that your interaction with the development community around your engine have influenced features or the direction of your engine?
Itterheim: Game Center was added because users wanted it to be in the engine. Same with iAd and specifically AdMob. I hadn't considered AdMob until I learned from others how popular it was.

But mostly it was me watching beginners struggle with Cocos2D. That was also a great help to improve my book over time, getting all those questions from readers who needed more insights on this or that topic. Sometimes the result of that is a piece of handy code that I add as a convenience method to Kobold2D.

I would say I'm mostly an observer, not so much interacting. For example if there's something debated on the cocos2d forums but the community members don't get it (quite) right. This is often accompanied with a lot of code sharing, repeated tweaking, fixing (or breaking) other developer's code. You then end up with a thread that's several pages long and there's so many code fragments and in-between or specialized version, it's become practically unusable. Have two coders grab the code and implement it in their own project, and they end up with two different implementations with subtle differences and all that. Maybe one's not even working. Or both.

This is always good material to write a blog post about, because I know there's interest and I usually see room for optimization both in performance, memory usage or simply readability (which I find is most important). So I like to retrace the steps, build my own implementation with a demo, and share it so it can be applied with copy and paste or just runs after downloading. The result is a blog post and often the code ends up in Kobold2D as well.

Brownlee: How do you manage the problems around remaining backward compatible?
Itterheim: Not at all. There have been very little breaking changes in Kobold2D. Cocos2d is a different matter, but they're very conservative except for the recently released v2.0 which changed a number of API calls from v1.0. But again, not nearly enough to make it a real problem to upgrade from v1.0 to v2.0 if you know what to do, where to look.

Brownlee: How do you structure the release process?
Itterheim: At the moment I'm not following a plan. There'll be a new release whenever there's a new XCode, iOS, Mac OS or cocos2d version or there's a serious bug. I like to change that though, make more frequent updates.

Brownlee: What processes do you use to manage code quality in your releases?
Itterheim: The most important part is continuous integration and a build server. Whenever I check in a change, the build server (my old 2009 Mac mini) starts building every example project. A full rebuild of all projects takes nearly 2 hours.

The Hudson build server builds 54 targets almost evenly split between Mac OS and iOS. Each target is built twice in debug and release configuration. In addition the iOS targets are built for both iOS Simulator and iOS Device. The process is repeated with the second to last version of XCode, and sometimes even the third to last XCode version. Overall there's anywhere between 360 to 540 targets being built in a complete test cycle. So that's where the 2 hours come from.

Fortunately most builds are not full builds, so it takes around 15 minutes for a complete cycle if a recent build was (mostly) successful.

Brownlee: What features would you like to add to your game engine over the next twelve months?

Itterheim: KoboldScript, Lua game logic scripting. I really want to make it a simpler interface for designers to help develop the gameplay logic with state machines, modifying the scene layout, experimenting with object behaviors.

I'm also thinking about ways to extend Cocos2D with an MVC design without having to change cocos2d's source code. There's got to be a better way to write code for cocos2d. Currently too many developers rely too heavily on subclassing. I think I can design something where it's even encouraging to use for beginning developers, most MVC implementations for cocos2d are neither well explained nor straightforward to implement nor can you immediately see the beauty and benefits of the MVC approach.

Brownlee: On which other game engines do you keep a close eye?
Itterheim: I watch cocos2d closely because I need to know when there's a new release coming. I don't care about any other engine at the moment, not even any of the cocos2d variants for different platforms, like cocos2d-x. I listen up if something happens in the Corona, Unity or Unreal space, but those are very different ballparks so to speak.

There's really no competition for cocos2d, in the sense that there's no iOS game engine that's 2D, code-centric, free and open source. Sparrow Framework is the only one that comes close. But I think it's too close to cocos2d, so there's no real reason why cocos2d developers should switch over. If developers quit using cocos2d, it's for Unity. I'm excited to see what Matt Rix can do with his cocos2d-esque, code-centric 2D game engine for Unity. It's called Futile. I like the thought of going back to C# programming once again, maybe someday.

Brownlee: What are some features of your marketing strategy for the game engine?
Itterheim: No marketing, just the usual social media connections but I'm not actively pushing them or anything. I used to run Google Adwords for Kobold2D but that's just such a huge money sink for something that's free, it's not worth doing.

1.0.5 Getting Started with a Game

Brownlee: What would you suggest to a developer looking to start making a game with your engine?
Itterheim: There's my Learn Cocos2D 2 book obviously. The third edition is fully compatible with Kobold2D. And there's also Essential Cocos2D where I answer all the questions I get.

Definitely check out the many Kobold2D example projects because there's plenty of code examples to learn from.

Brownlee: What suggestions do you have for developers working with your engine daily?
Itterheim: Woah. Hmmm. One thing, perhaps. Don't be afraid to read the Apple docs. They're really, really good! I know people complain about them too. But compared to other vendor documentation I've seen, Apple's docs are fantastic, hands down. It's just a bit hard to find what you're looking for, but that's partly because there's so much documentation on the site.

Brownlee: What are some game ideas you have had but do not have the time to work on?
Itterheim: So many of them. I still have this "One Game Idea A Day" idea on the backburner. My idea was to post a game idea every day, and it should be about something no one has created yet. Personally I think we game developers are looking too much for the fantastic. It's all about orcs and rocket ships and princesses and blobs and cartoon characters. But where are the cool and fun games with a minor educational component?

I'd love to do a game about surfing the Pororoca. Don't know what it is? Why it happens? When and how often? Play the game!

Or a game where you try to save energy in your house year by year to counter the increasing energy prices. Learn what uses more energy: a 60 Watt light bulb that's on for 6 hours or the hair dryer that runs once a day for 5 minutes.

What about sea exploration, has no one ever done a fun game about that? Where you hunt for sunken treasures, for example. Or discover and research lifeforms in the deep sea.

I like watching documentaries. They give me all kinds of cool ideas what to make into a game. And sometimes I wonder why no one hasn't before? Why is everyone so obsessed making games with overly cute characters, or brutal military shooters, or boringly stiff simulators? Where are the real "real world" games? You know, the Ice Road Truckers or Storage Wars of gaming.

Brownlee: What do you think are the required skills for building a great game?
Itterheim: If you're a single developer, a little bit of everything helps. But I think the most important skill is to figure out what you're good at, and how to make up for the parts where you're not so good at. For example instead of creating coder's art, I've seen awesome games that were entirely vector art and effects. No drawing required. Other games like Doodle Drop simply declared badly sketched art as their style.

If you need or want a partner in crime, the most important skill is finding the right partner. Team up with someone who's not like you, who has different skills. Two programmers thinking alike is most likely going to end up in a technically well done, but ultimately boring game. Two opposing but cooperative minds, well the result is more likely going to be one of the two extremes: a disaster or pure genius.

1.0.6 With a Successful Game Engine

Brownlee: How much revenue have you generated from your game engine?
Itterheim: Nothing from the engine itself. It may have helped sell my products and affiliate product, but I don't know how much, if any. I can say that affiliate products alone are between $100 to $400 per month. That's easy money, I love that!

My own products, they wary between $200 to $800 from month to month. Especially the summer months, and the first really sunny spring month are bad for selling source code. And the book, most recently it must have been about $400 per month. Unfortunately, with all the money I'm spending on hardware and software, that means I'm not breaking even. I'm almost done with freelance work, I get paid a bigger sum maybe once more and then that's it. That's why I have high hopes for Essential Cocos2D. With about 50 members I can relax somewhat, with 100 I should be good for the time being and set something aside or re-invest.

Brownlee: What are some opportunities that you have received because of your game engine?
Itterheim: Many, many, many offers to work for others. Not really interesting for me to be honest.

There's one book publisher which shall remain nameless that writes me every 2 months or so, asking if I wouldn't like to write a book about Cocos2D for them. I keep telling them that I already wrote a Cocos2D book, and can't. They just don't seem to take no for an answer. Sometimes it's even the same person asking the same thing, that tells you how much they're interested.

Speaking, no. Perhaps if I went more out there and offered, there'll be some who would consider or like having me. But I'm in Germany, there are very few events here, and the only really interesting one that's about games, it's just ridiculously expensive to attend. I don't like to spend $1,000 overall just to attend a speaking event, either overseas or within Europe. I'm not much of a marketer, networker or drinker, so there's little benefit in it for me.

I also feel quite comfortable doing all of this from home. Don't get me wrong, I enjoy being at the events that I do attend, speaking or not. But they're over so quickly, and when I'm working, I just don't think of all the events around me. That's always been that way for me, I'm a little out of touch with the world and what's happening and when that is happening. I'm often surprised by, you know, how late in the week or month it is already. It's usually others who ask me whether I'll be attending event X that I realize "Oh, that's tomorrow?!?".

Brownlee: What elements make for a successful mobile game engine?
Itterheim: Be there first and provide something users need. I think cocos2d got successful because it was the first free, open source game engine that focused on 2D development. A major plus was that it was written in Objective-C, at a time where most engine developers were like "Noooo, Objective-C, that's way too slow to build an engine with, it has to be all C!". Fools!

So because of that, and alternatives were either C/C++ and/or 3D and/or commercial it became successful despite the lack of features, despite barely any documentation, mainly on the promise to write iPhone games in Objective-C and without having to know OpenGL. That promise, that's what's important. You have to have that one thing everyone wants or needs. Game developers don't want to write an OpenGL rendering engine, they want to make games. But they also don't want to learn a new language to do so.

If you look at the other Cocos2D engines, their drop-off in popularity is immense. Even though a whole team works on Cocos2D-X, and they too fill a niche for an open source, cross-platform 2D game engine, they're just a fraction of the popularity of cocos2d-iphone. The reason is that there's so many other competing engines and frameworks that provide cross-platform support. And of course C++ which speaks only to more seasoned developers. But most importantly, 90% of game developers simply don't care about cross-platform. Or maybe some more do but they still start targeting one specific platform first.

Personally I strongly believe that small game engines should focus on supporting their primary platform as well as they can, with the platform's native tools. The biggest mistake you can make is to tackle cross-platform development. It binds so many resources and energy, and you end up having to work with tools that are built on compromises, and you have to write code that has to make compromises too.

That's also what worries me a bit about Cocos2D's future, with the big plan to make a cross-platform JavaScript API that's supported by several major Cocos2D engines, very likely because Zynga wants it that way. I wager that 80% of Cocos2D developers couldn't care less about JavaScript, they'd much rather see other things being developed.

For Kobold2D I will continue to focus on iOS and Mac development, and make no compromises regardless of where cocos2d is heading.

Brownlee: What tactics do you think got you over the line when so many

other software projects fail?
Itterheim: Most software projects fail because too many people work on software whose users don't participate in the development. Or simply because too many people work on it, period.

For me it was a matter of pride, and to make a statement. This is how it's done! If you set your mind aside for a moment from writing code, instead listening to users and thinking about the infrastructure and how your project is being used and where the problems are in that, then what I did should come naturally. The result is my interpretation of the flaws of cocos2d that needed fixing; my statement is Kobold2D.

It's also my way of thinking. Many programmers think only in code, classes, methods, properties, types, structs, algorithms, and so on. But what's really more important in a project that's going to be used by users is the end user experience. At some point, that's what matters most. I still get this icky feeling when I think how cocos2d encourages bad code design just from the way it's built. A game engine isn't just a piece of source code, it's an object of study, something even beginners use and learn from. But the sad truth is, every time a cocos2d developer subclasses CCSprite to add game logic, a cute little kitten dies a slow and painful death.

To be fair, I'm not much of a game engine programmer, I'm more interested in the systems engineering part. Designing the public interface. The end user's workflow. And promoting good design principles. So I always have an eye on those things. It's what my jobs taught me is most important, help others get their work done quickly and efficiently. I like the challenge in that, it requires thoughtfully designed code and systems.

Brownlee: What is the most difficult module or sub-system when developing a game engine?
Itterheim: Designing the public interface. How the engine gets used. How straightforward, logical it is. How to make it simple yet powerful. How to make the simple things simple and the difficult things reasonably simple but hiding their complexity from those who don't need it.

The second most difficult thing is documenting it. It's actually not that hard, but you forget about it while writing code. But you shouldn't because documenting something gives you a lot of insights about your own work. How you could improve it. If you fail to explain a method in a short and concise manner, perhaps it needs refactoring.

Brownlee: Would you do it all again?
Itterheim: Sure, absolutely!

Brownlee: What would you do differently if you were to start the project from scratch today?
Itterheim: Perhaps slim it down a little bit more. Initially I wanted to

cram too much into it, that was counterproductive.

Brownlee: How does it feel to have a game engine that developers are using to build and release games?
Itterheim: Uhm, I really don't know. I don't think about it this way. I just hope it works and doesn't break for anybody or does any harm. I work on Kobold2D because I want to work out the things that make it better. I like making it better, and every new release with additional fixes where I learned how they work, why they appeared and how I could fix them, that's success for me. The learning part.

Getting it out there so others can benefit from it, it's hard to describe, I think it's relief in a way. Basically what that means is I don't have to worry anymore that I have a product out there many developers are using, and it's currently not working right or not having some features I think it should. That's my biggest worry, it's painful, but it's also motivating. I couldn't sleep knowing there's a big issue with Kobold2D and I didn't spend the necessary time to make it right.

I hate to be responsible, that's why I fix stuff.

But that also made me careful, because it's so easy to create something that might end up being a huuuuge responsibility. And you've got to really love what you do, otherwise the responsibility will kill your drive and dedication. The more you neglect something you're responsible for, the more it's going to hurt your self esteem and your will to carry on with it. If you just keep moving forward frequently, if only a little bit, you can go a long distance.

Oh my, maybe I should teach philosophy?

Brownlee: What is one thing about your game engine with which you are not happy?
Itterheim: The split between v1.x and v2.x. It just doubled the workload. So I'm going to stop updating the v1.x branch soon.

Brownlee: What advice do you have for a developer thinking of creating a mobile game engine?
Itterheim: Only one question: what the hell are you doing?

If you are developing an engine for the engine's sake, and to learn from the process, go ahead. If you think your game needs a special engine, think again. Write a prototype in an existing game engine. You can either use that engine or write your own and double if not quadruple development time. If you have time and money to waste, do the latter.

Brownlee: What are some must-read books and resources for a developer interested in creating a mobile game engine?
Itterheim: Hmmm nothing comes to mind right now.

1.0.7 On Games

Brownlee: What are some games on which you have worked?
Itterheim: Before mobile, that was Spellforce and Battleforge for PC. And Dave Mirra Freestyle BMX for Gameboy, among several others.

For mobile there'll be a game out soon from famous German cartoonist Joscha Sauer[2]. I programmed that game. A lot of time went into the weirdest thing: smoothly animating a stream of pee. You know, just the most normal thing in the world to put into a game.

Brownlee: Where do you think mobile gaming will be a few years from now?
Itterheim: Mobile devices and hand-held gaming devices will slowly but surely become more and more similar to one another. To a point where I think Sony's PSP will eventually be running on Android. Android will also move into TV boxes similar to Apple TV, and end up being built-in to some TVs not just for media streaming but as the TV's entire operating system.

Many more specialized and cheaper mobile devices will appear. Apple's new iPod nano is just a start. There will be mobile devices for less than $100 that run Android, perhaps even iOS, and have a touch screen and can download apps. I think this will bring a new market for "gadget" apps, very simple apps and games. They do one thing, and one thing only. Show me the weather. Entertain me for 5 minutes. Play fart sounds.

Brownlee: What is your favorite game and why?
Itterheim: I spent well over 200 hours in Skyrim. But is it my favorite game? It feels like I've already played my favorite games 10 to 20 years ago. I wouldn't be where I am today if it weren't for Doom, that's definitely an all-time favorite. I love what the mod community is doing to this day. Skulltag, try that out, just pure awesome! If we would have had that nearly 20 years ago, I think we would have overdosed!

Brownlee: What are some trends you are seeing in mobile gaming?
Itterheim: A very obvious trend is doubling the computing power with every new device generation, every year. At least that's the case with Apple devices. If you look at the quality of 3D games, the first devices had about Playstation 1 visuals. With the third, or the fourth generation at the latest we entered the Playstation 2 era on mobile devices. Now the iPhone 7 we should be much closer to what we see on a Playstation 3 these days. Of course that's all with a much smaller screen, but it's still impressive.

We'll definitely see the big major game engines making it big time on all platforms. Unreal and Unity are the pioneers, they're the Apples and

[2] http://nichtlustig.de

Microsofts of game engine development. Other game engines can only hope to find and serve a niche. New game engines, no way. No chance there's a new game engine that's not out yet and will play any role whatsoever, in particular if it's cross platform. The game engine market is saturated.

The hopes of a single indie developer hitting it big time on any App Store will be zero. You'll have to invest a significant amount of time and money to make a splash, and have a significant development experience to build on. That means the risk will go up for indies, as it did on other platforms before. Indies will start to compete against each other much more than they will compete with the big names though, that's the sad part. Eventually we'll come to consider the years 2010-2015 as the heydays of the indie developer.

Then the startup costs and risks will be so high that you need to have some startup funding in order to have a decent chance of making it, and the market consolidates again. Indie studios close or get bought out. It's a cycle that keeps repeating.

Until a new Indie-friendly market appears. Perhaps TV Apps? One thing is for certain: Indies will continue to play a vital role in game business, and there will always be lucrative niches only Indies can tap into.

1.0.8 Final Questions

Brownlee: How can readers best get into contact with you?
Itterheim: I prefer twitter. Short and to the point. Otherwise either join Essential Cocos2D if you want frequent Q&A and my help, or just send me an email at steffen@learncocos2d.zendesk.com if it needs to be personal. But expect a delay, it can take me a couple days to answer (non-member) emails. I definitely prefer more public communication for the simple reason that things I say may be of interest to others too. If you have a question that others might be able to answer as well, you should post on http://stackoverflow.com. I'm spending way too much time there already.

1.1 More Information

This was just a sample from the book "Mobile Game Engines: Interviews with Mobile Game Engine Developers". For more information please visit http://mobilegameengines.com