Phillip B. Shoemaker

March 13, 2009

My iPhone Developer Experience

Lessons Learned

I believe that Apple needs to work on the following in order to vastly improve the application submission process:

- Make the process of signing up for the iPhone Developer Program simple, understandable, and *reversible*. Make sure that everybody understands what it means to sign up to be a business developer, including the price, the business paperwork required, and the time it is going to take (estimated).
- Give the developer a starting point, and some tips on going into the iPhone application business: sign up for your license before you start your application (otherwise you will be waiting), here are good developer resources, here are the application requirements for submission, etc.
- Make it clear what the app review team is looking for. There's no call for absolutes and specifics; rather make the existing rules less vague, add new ones (business models, limited use, etc.), and clarify the items in the T&C.
- Lighten the restrictions around the HIG. The HIG is a set of guidelines, not requirements. Otherwise the document would be called the HIR. Additionally, we should be enforcing the core HI concepts (a button works like a button, a table works like a table, etc.), rather than enforcing silly concepts like tables view not maintaining a selection.

- The app reviewer should continue to review an application, despite running into a 'show stopper'. Unless the show stopper is one that does not allow the reviewer to continue using the application, the entire app should be reviewed prior to sending a rejection to the developer.
- When Apple makes icons or button graphics mean specific things, the graphic used should be specific. (!!! CITE and show pictures of issue)
- Seeing that many developers submitting their applications are relatively new to the ISV world, remind them the amount of testing required prior to application sub - mission.
- Remove the Limited Use restriction. Let the market decide. Obviously if there's a price issue in conjunction with Limited Use, then that's less of a market decision. Just LU? Let it through.
- Make rejections and responses timely. If something goes into a hold bucket, don't let it sit for 8 weeks. Rather, let them know immediately that there is a problem.
- Make rejections and responses clear. If there's nothing an application can do to pass the review process, try to make that clear. If there's an issue with a key area, don't just point to the T&C or HIG, but also to the part of their application where this error occurs.
- Let the developer open communication with the reviewers. When someone re - sponds to a rejection notice, it is imperative that Apple responds in a timely manner.
- Provide utilities to help the developer test their applications. At Palm we had an interesting tool called Gremlins that helped developers 'bang' on their product in

semi-random manner. Apple should create such a tool that output some hash that gets uploaded as part of the app submission process.

Please only continue reading if you are interested in hearing the whole, gory story I spilled.

Before the Beginning

I started seriously considering developing an application for the iPhone around the first week of December 2009. To me, the whole ecosystem seemed to make a lot of sense: gone was the need to find a sales channel, hire a sales team, or implement payment mechanisms and complicated websites. Rather, I had to worry about writing the code. On this front however, I had three major concerns.

1. **Objective-C**

My big barrier was learning Cocoa; I hadn't even thought about Objective-C in years, but it was a major concern. Having written various applications and tools for the Palm OS Platform, I wasn't that concerned with embedded development. I knew the basics of working in an embedded environment (remote debugging, cross compiling, memory management, etc.), but hadn't a clue about Objective-C. I had previously owned a NeXT box, and had dabbled with Cocoa on the Mac, but found the whole integration with Inter - face Builder (IB) a bit confusing. Learning a new language and IB made me nervous. I should point out that learning new programming languages don't often scare me: I had just wrapped up learning Ruby and Python in the past two years, and those were a pleasure to learn.

2. **iPhone Platform**

My next major concern was learning the platform. Certainly I had significant hours into *using* my two iPhones, but didn't know anything about the services provided to the developer. This is the disadvantage of going to *any* new platform. You always have new stuff to learn, but that is usually a fun process, and is not one that is done immediately. I

am more of an ad-hoc developer, and get to know the basics quickly, but spend my time writing code and getting my knowledge in a Just-in-Time (JIT) fashion. I have to say that doing this on the iPhone was a pleasure, although finding the information was some - times difficult.

3. XCode (and other development tools)

In my previous life, I designed the user interfaces of development environments: Borland C++, Delphi and the Palm OS Development Suite were all products that I had a significant hand in designing. Thus, learning a new development environment is some - thing that I have a lot of experience in doing, and found that the XCode environment was fairly trivial to learn for iPhone development. Granted, the majority of my work was simply using the editor and debugger, whereas the rest of the functionality did not aid in my development at all. I think this is pretty typical for the standard iPhone developer.

The other piece here is Interface Builder itself. While a great tool, it does take some getting used to. Other developers on my team would routinely give me a call to walk them through the process of specifying delegates, linking functionality, etc., even when following along with an example like the ones found on icodeblog.com. I think Apple does a great job of demonstrating code on the website and through examples, however I think we also help to confuse the developer by not using more Interface Builder. In my experience, I found that I was encouraged to use IB, yet most of the code snippets cre - ated the objects manually (read: in code), and it was left to the novice developer to fig - ure this out. It makes a lot of sense *now,* but at the time I was left wondering how to connect the control in IB to the dynamically created object. This was not obvious to me, and as I've found, to other developers.

The last piece I'll discuss here is the simulator. Simulators are a great way to debug your application because 1) they are native, 2) they are fast (due to the fact that they're native), 3) they don't require cables, and 4) they cannot damage your own iPhone. #4 may seem silly to some, but as a "Joe Schmoe" developer working out of my garage, I didn't have a cache of devices I could refer to at any one time. One of the downsides of using simulators is that they're not running the true application, and therefore can miss some significant bugs. Another issue with the simulator is the lack of plurality of that work: there's only one simulator. How then can I be sure that my application runs on the iPod touch? The answer is clear, and that is to go buy one. I'll throw out a plug to one of Palm's major accomplishments: they not only had a simulator, but an emulator as well, running the actual end application, and running the true ROM of the actual selected de - vice. This [1] helps to eliminate the need of owning every device that you're targeting.

The Beginning

Now perhaps I went about this process differently than others, but I just dove right in to develop my application. After all, I was just starting out and had no idea 1) how long the development of my application would take, 2) if I'd be able to hack together anything at all, and 3) if iPhone app developers made any money at all.

I spent the good part of two weeks learning the tools, the development process, the platform, the various kits, and Cocoa Touch. I certainly found some good reference documentation and examples on the Apple web site, but I found the best resources out - side of Apple's walls. icodeblog.com was one of my favorites. While I started at the Ap - ple site to get the SDK, get a brief introduction to the OS and SDK as a whole, I found

---

[1] It certainly helps to eliminate the need, but it doesn't negate the need entirely. An emu - lator can *never* do that.

that information targeted to the advanced-beginner was better found elsewhere [2]. (Example: finding code or even pseudo-code that shows one how to generically open and close a view. Granted, many examples do this, but they do them in different ways: one pops up a view using the modal view controller, one shows a view using the navigation controller, etc. While the true code to open and close a view is trivial, finding an example to do just this was near impossible for me).

I was pleasantly surprised how long it took me to start being functional with Cocoa Touch. Without books or classes (only the resources on the net), it took me about 2 weeks until I became comfortable. But while I was learning, I was still writing a lot of code, and rewriting it, until I got comfortable with it. By no means am I an expert, as there are little things that I never bothered to research (why do some methods have a minus sign and others have a plus sign in front of them?). But in two weeks, I was well on my way to writing my first application.

Testing my application

Once I had wrapped up the majority of developing my application, I wanted to move on from Simulator-based testing and on to device-based testing. I figured once I had finished my application, that doing the device specific testing would be trivial. It was not.

The issue behind device-based testing is that I needed to provision my device. In order to provision my device, I had to join the iPhone Developer Program (IPD). This was not as simple as I had hoped.

The iPhone Developer Program

---

[2] Apple has great introductory documents and videos, and complete reference documentation, but once you get beyond a certain skillset (not quite advanced), the site is somewhat lacking.

I had wrongly assumed that joining the program was a simple as paying the fee. After all, all I really wanted to do was to test my application on my devices. What I found out was that I had to go through some sort of process that I didn't understand, for an amount of time that wasn't communicated to me, and no clear contact that I could communicate with in order to clear this up.

When I joined the program, I did so under my regular email address, but since I had an LLC brewing, I figured I'd use that and go under as a company. Once I did that I expected some feedback: how much it would cost, how long it would take, what paperwork *precisely* it would require, etc. Instead I got a message telling me something along the lines of "your application has been submitted and once it gets reviewed...". At that point I realized that the business path was not appropriate in order for me to quickly get up to speed on testing my applications, but I let it go for a few days.

Three days later, still without word on anything in my business getting approved, I moved on to trying to use that email address to go in as a personal developer. Didn't work. I then chose another email address and was able to successfully join the program as an individual developer, knowing, unfortunately, that I probably wouldn't be able to submit under the company name in the future. I was just out of luck.

After a few days I was able to get my iTunes Connect login information, and was able to get everything else available on the portal, including provisioning, certificates, etc.

Once I got this going, the testing of the application was amazingly simple, and I uncovered NO bugs that only happened on the device. This is an amazing statement, as this was a very common problem on Palm devices: applications that ran on the simula -

tor often failed once they got on the device. This is the key reason we ended up writing an emulator, in order to help ferret out all hardware specific issues.

Application Submission

The process is fairly straightforward and self explanatory. The labels are clear, the fields are plenty and if you're really stuck there are help buttons everywhere. No prob - lem.

When you get to the last page and are asked to Review the content, the obvious next step is to press the Continue button. The step after that is unknown to most devel - opers, myself included

Application Review

All that most of us know is that there is an application review process and it involves a few things like:

1. *Terms and Conditions*. Clearly in the iPhone SDK Agreement are some program requirements for applications. But many of us developers just assume we know what is kosher and not, and stumble through this. The document is written in a vague enough manner for us third party developers to gloss over and presume.
2. *Human Interface Guidelines.* The HIG is written as a *guideline* (which is a rec- ommendation), but the T&C says that applications "must comply with the Human In - terface Guidelines". The whole 'recommendation' versus 'must comply' issue is con - fusing by itself, but the HIG is also written in a way that leads to misunderstandings and disagreements.

3. *Quality Assurance.* Although not expected, it is clear that the Apple team does some amount of testing of the product, to ensure that it meets the expectations that users have with the applications on the store.

Beyond these three items, most third party developers have no idea what else Apple does in order to approve an application for the app store. To them, it is a complete black box, as evidenced by some of the recent postings (!!! cite URLs here !!!).

My history with applications review was relatively fast, but inconsistent:

1. *Money Timer:* Went through the process within 7 days, but was rejected due to a table view issue. This was the first time I had received a user interface rejection, and they cited the fact that I had to be compliant with the HIG in order to submit an application. I disagreed with the table view issue, felt it was a trivial issue, and responded to the email. The next day I figured it would be much easier to just make the fix and resubmit. Later that day I did.

   However, about 7 days later I got another rejection for a bug fix, and another user interface 'tweak'. This was frustrating, because I was under the impression that the reviewer had done a thorough exam in the first place, and my fix would have allowed my application to sail through the process. Once again, I disagreed with the button issue, but made the change. I also was able to reproduce the bug and made a fix. I then resubmitted again. Roughly seven days after that, the application was approved and Ready for Sale.

   While I appreciated the attentiveness, I believe that the user interface issues were unfounded by people not trained in human interface-related issues. Addition -

ally, I was discouraged by the process: I thought a full review would happen the first time regardless, rather than the reviewer stopping at the first 'show stopper'.

2. *iWiz.* This was my second major application to be submitted to iTunes Connect, and seemed a no brainer. The user interface was simple, easy to use, and didn't abuse the built-in controls. However, it sat in the 'In Review' process for weeks. Seven weeks, to be precise, and then the response I received was because the content was considered obscene. The content, in this case, was two things: the sound of urination (found in other apps on the iTunes Store), and the cartoon depiction of someone urinating. I responded to the reviewer, asking a simple question: Content is too vague; is it the urination image or the urination sound? This time the reviewer responded relatively quickly. The response? "It was the content." Frustrating. Since I had heard these sounds in other applications I determined it had to be the image. I fixed it and resubmitted. It was rejected again. Clearly it was the audio samples. This was a very frustrating experience, and Apple blew it in three major places:

- *No clear statement in 3.3.12*. What is obscene, pornographic, offensive or defamatory? Clearly the American tastes are different from other supported countries, so what is allowable here versus Japan versus the UK?
- *No communication after the first seven days*. Once Apple put me into a 'controversial' bucket, it should have notified me, rather than letting me wonder for seven weeks.
- *Communication was not clear*. Once the communication was received from Apple, it was a vague statement, that made it sound like I could alter my application in

order to be compliant. I later learned that this was not the case, and the rejection, in my opinion, should have been extremely clear about that.

3. Stinky Meat. This was my third major application to be submitted to iTunes Con - nect, and seemed pretty clear. This was the first application where I hired a graphic designer to design all graphics and animations. Additionally, I spent over a week learning about core animation / graphics, and implemented various animations to enhance the product. However, despite these challenges the application was denied due to limited utility. I responded immediately to the reviewer about the time respon - sive nature of this application, as you had to wait for the images to deteriorate. No response. So I resubmitted with those notes, and got the Limited Use rejection again.

This is a case where I think Apple had it wrong. First of all, there were over eight images and fourteen animations built into the product. Definitely over the two images/sounds requirement. Personally, I think this is a case where we should let the market decide. If nobody purchase it, then it was not worth the writing and the developer learns a lesson. Lastly, this is a case where having more information up front could have limited the investment I made in the project. If I had known that it would never get into the store, I would never have invested all of the money and time in the project.

4. Business Poetry, Medical Poetry, 101 Cocktails, and others. These absolutely *sailed* through the app review process, in three to seven days.