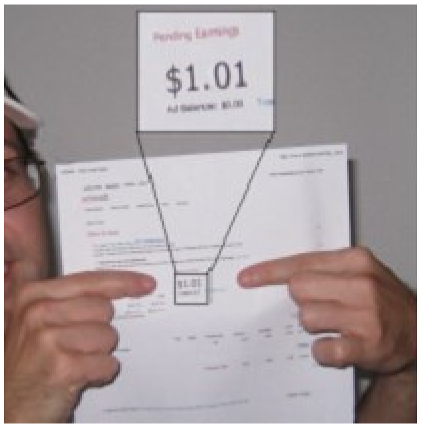
To Hell And Mac: *A PC Guys’ Brief Misadventure In The Kingdom Of Apple*

*Note: This was written **long ago**, before I wrote the novel Red Hope. This article has not been edited as thoroughly as a book. You have been warned.*

JJDreese@yahoo.com



Every Mom & Pop store has a picture-frame showcasing the first grubby dollar they ever made. I just earned my first dollar from the iPhone app I made and it only took one year. To quote Rodney Dangerfield, I made that dollar *the hard way*.

Two years ago a coworker stunned me with all the amazing things his iPhone 3G could do. So when my contract was up, I wandered into the AT&T store and wandered out $200 poorer with an iPhone 3G. Phone, internet, GPS and MP3 player all wrapped up into one. It was useful right from the beginning and I never had buyers remorse.

And man, the whole concept of the app is so cool. These little software tools turn this cellphone into a magic machine; apps make it the Swiss Army knife of cellphones. There’s an app for everything, including dozens just for the purpose of making fart sounds. As a PC guy, I see computers as tools and the iPhone was quickly becoming my favorite tool; so much so that I stopped carrying a laptop while travelling.

One night the idea for an app popped into my head; a program that converted the weight of the coins in your piggy bank to a total value. I’m no programming guru or computer science major, but I did start a computer software company 12 years ago out of my shoebox apartment in grad school. It’d made me a small fortune; so small in fact that I still have a fulltime job to pay the bills. Up until I got the iPhone fever, all of my applications had been written in Visual Basic and my biggest hit (relatively speaking of course) was a program used by aerospace designers for airplane wing design. I know, it’s a real snoozer to anybody outside of engineering, but the point is that I have a decade of experience programming commercial applications in Visual Basic. As I would soon learn though, that experience didn’t help much in the world of Objective-C, the language that iPhone apps are written in.

To help any would-be developers out there, here is my guide to go from nothing to iPhone App Developer in just six painful steps.

Step one: Buy a Mac.

You can use a PC to move mountains and cure diseases, but you can’t write an iPhone app with it. The software development kit (SDK) that Apple gives to prospective app writers only works on newish Intel-based Macs. It’s either perfectly clever or perfectly evil; you can be sure that part of the resurgence in Mac sales involves PC users like myself trying to strike it rich in the iPhone app gold rush. I bought a refurbished Mac Mini from Apple; didn't want anything too fancy and the Mac Mini allowed me to use my existing monitor, keyboard and mouse.

As it turns out, learning the Mac way of life ended up consuming about 25% of my efforts. The last time I worked with an Apple was the same year Return Of The Jedi hit theaters *the first time around*. A short time later, my Dad bought an IBM compatible and I've been working on PC's ever since. Bye bye Apple.

So there I was in 2009, like a monkey with a slide rule, trying to figure out the basics of how a Mac worked; like where the hell all the basic software utility tools were hidden that I usually get to through my Start menu on Windows. What and where is the Paint equivalent? If you are thoroughly PC, then many aspects of the Mac feel very foreign. Here are just two examples.

Regardless of what application you're using or where the window is located on the screen, the menus are glued to the tippy top of your monitor. The window you're working in could be floating off toward Cuba, but your menu tree will still be stuck to the top of the screen. The fact that main menu systems are disembodied from the "floating" working windows causes all kinds of frustration. I still find myself accidently mis-clicking the top menu, resulting in an accidental click on the *empty desktop* which deactivates the working application and then (without realizing it) re-clicking on the File or View menu of some spuriously chosen application and then FUBARing everything up. Here I am nearly a year later and I still do this.

The second example has to do with copying a file. When you copy a file on a Mac, sometimes you don't really *copy the file*. Instead it creates a shortcut called an alias; an invention borrowed from the user-hostile world of UNIX. Unfortunately, by the time you learn about this Byzantine labyrinth of copied files, you've dug an irreparable web of lost files and misery. It's not better than a sharp stick in the eye.

Step two: Sign up as a developer and download the software kit.

To write an iPhone app, you need to fire up your new Mac and download the iPhone Software Development Kit (SDK). This is simple enough. You can sign up and download the software for free if all you want to do is play with the "virtual iPhone" on the screen. It acts as a test device to run your apps on. If you want to actually put your test app on a real iPod or iPhone, then you have to pay the $99 fee to Apple so the CEO can add a little more height to his million mile high tower of cash.

At this point, you may be at the one and only fun part of the iPhone app process: designing the interface. Even that can be a bit tedious if you are a Visual Basic kindof guy mainly because it is done with a separate program outside of the Xcode programming environment called Inteface Builder. Figure 1 shows what my initial thoughts were on the interface for PiggyBank By Weight. Figure 2 shows the final screen.

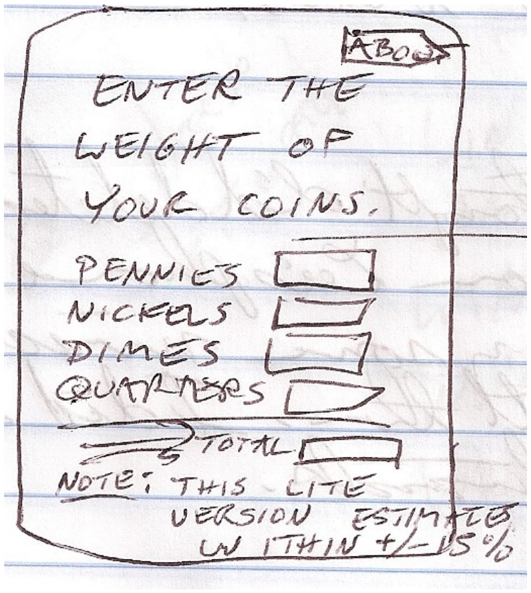

ABOUT
ENTER THE WEIGHT OF YOUR COINS.
PENNIES
NICKELS
DIMES
QUARTERS
TOTAL
NOTE: THIS LITE VERSION ESTIMATES WITHIN +/- 15%

Figure 1: Initial design idea

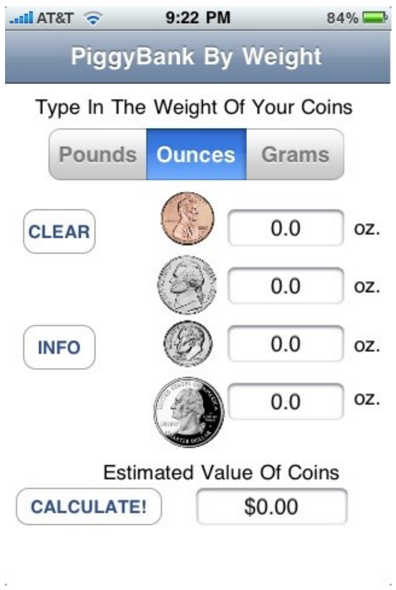


Figure 2: Final interface

Step three: Get data for your app if necessary.

My app required finding out the statistical averages of coin weights. Because the US Mint has been changing the metallic formulation of coins (especially pennies) over the years, some finagling of the data was required to get good approximations. Fortunately, I had a huge pile of coins to use, but they first needed to be separated into piles of pennies, nickels, dimes and quarters. This gave me a great opportunity to have a father-son bonding moment with my toddler. And it gave me a chance to prove to my Grandpa that he was wrong back in 1978.

He'd invited my siblings over to help count his mountain of wheat pennies. I was summarily sent away from the table because he said my incessant talking would make him lose count. So when it came time to separate my own coins for this project, I had my three year old toddler help me do something even simpler; just to separate the red coins from the silver coins. In the end, my son poured a bucket of separated coins over his head. Kids will be kids. I guess my Grandpa was right.

Step four: Use tutorials to build your own app.

You can go through the trouble of learning Objective-C from scratch, but it is far quicker to go through a few tutorials and then jump whole hog into your app; starting from a sample project and then twisting it into your own application. Even this won't help completely because of all the nuances of the quirky Objective-C programming language. For that, I relied heavily on the online app help pages and discussion forums. In a way, it's good that so many people have so many problems with the simplest of tasks regarding the iPhone SDK. Their agonizing search for answers saves you time.

Step five: Test and submit.

Writing the app was a struggle every step of the way. I finished it at the end of January and sighed relief. Little did I know that the frustration level I'd seen before was just an appetizer. Submitting the app to the iTunes store proved to be laughably complicated. As all app developers find out, they fill out all the forms and then submit the app only to have it rejected immediately due to some spurious code signing error. If your eyes just glazed over at the term "code signing error", you're not alone. I had no idea what it meant either. There's a special level of apoplectic rage that happens when you complete a long journey only to have everything derailed by something that is totally incomprehensible. It's like running a marathon only to trip and fall just a few feet from the finish line.

In addition to learning the Mac and learning Objective-C, you also have to juggle a new set of items called Certificates and Provisioning files. To submit the app, you have create and download something called a Submission Certificate. And of course, you have to download a Submission Provisioning File to verify the certificate (rimshot please). This is a process that you cannot stumble through. I know; I tried for days. About a week after I first tried to submit my creation, I finally got the code signing to work and figured out that an app compiled with the Submission signing cannot be put on your iPhone for testing; needless to say it is a very tedious process. So at 12:49am on a Saturday morning I received confirmation that my app, PiggyBank By Weight, was accepted into the review queue.

Step six: Wait for rejection.

After all that trouble, many developers have their apps rejected by Apple. Sometimes for good reasons (bugs, crashing apps, etc…) and sometimes for style reasons; how very Jobs'ish. To prove how simplistic my app was, it was actually approved in less than a week. During his Apple WWDC 2010 keynote speech, Steve Jobs claimed that 95% of all apps submitted were approved quickly. I can include my piggybank app among those statistics.

So those are the six steps you need to create an iPhone app. Tremendous patience and endurance are a requirement. With tens of millions of iOS devices out there, I figured that my app would sell hundreds in the first week. Over the course of the first four months, I sold ten copies for 99 cents, bringing my royalty earnings to roughly $7. However, since Apple doesn't cut you a check until earnings are much higher, I will never see that money, sending my effective profit back to zero dollars and zero freaking cents.

A change was in order. After four months of craptastic sales, I decided to make the app free and put an AdMob banner ad at the bottom. Again I went through the whole submission process and within a week, it showed up on iTunes; there is a giddy feeling you get when your app is approved. Within 24 hours it was downloaded 50 times. That's an eye-popping difference between free and not-free. The beautiful thing about

AdMob is that you can actually see how often your app is being used even after users grab it from iTunes; each time they run it, the AdMob account increments the usage numbers. AdMob only pays you a few cents for each time somebody clicks on the ad, but after a month or so I'd earned my first dollar. That's right, the hardest dollar I've ever earned.

As was the case during the Gold Rush, very few actually have made it big in this digital gold rush. It is the rare superstar app success story that emboldens us to continue creating apps and failing time and time again. I have to accept that I spent many months of my life working for essentially nothing. Although proud that I did what I set out to do, I must admit it wasn't fun. I got no enjoyment from using Objective-C; very unlike the fun I have whipping up useful applications with Visual Basic.

Had I known how big a challenge this would become, I would've simply spent my time on more enjoyable things like hammering my nuts flat. The true lesson for me was how much I respect anybody who makes an iPhone app with any complexity. I am humbled by their ability to decipher the programming eccentricities and, even more amazing, their ability to navigate through the paranoid Apple bureaucracy of certificating and provisioning files. As for me, it's been a long nine months. To complete the journey, I whipped up a "clone" of my iPhone app in Visual Basic to run on my PC. I had it up and running and debugged in about 30 minutes. There's something to be said for simplicity.

Epilogue.

As any good product developer knows, you can increase the value of a product by aligning your product goals with those of another company. It's a situation where one plus one makes three and everybody wins. To make PiggyBank By Weight as useful as possible, it needed a zip-code searchable map window that would point the user to the closest coin-counting service or kiosk. So I did some digging and contacted the two companies in America that offer coin-counting services: I'll call them Company A and Company B.

I wrote several times to Company A's "biz development" department, each time refining the email so as to let them know that a) I didn't want any money and b) this would only help point customers towards their locations. No response. Ever. Not even a bounced "could not deliver" email. Their biz development email address was the black hole of all entrepreneurial hope.

So I did some snooping and took a wild-assed guess at their CEO's email address. By sheer luck, he quickly responded noting that he'd forwarded my email to their coin leadership folks. This is the point in any fictional story where the lowly app developer joins forces with a big company and his project takes off. Of course the complete opposite happened. The coin leadership guy sent me a very friendly email saying that they didn't want to collaborate and that, ominously, "I'll definitely keep your app in mind as we continue to evolve *our marketing plans*." Strike one.

So then I wrote to Company B which has counting machines in their bank branch offices. They replied right away that they would have their folks look at it. Nothing. I reaffirmed that I didn't want any money, just the chance to collaborate and have a zip-code searchable map that led customers to their banks. The sound of silence was deafening. Strike two.

In all honesty, I am stumped. I've been in business for over a decade and never has an outside developer dropped such an opportunity into my lap. "You say you've already written the app? You don't want any money from me? You just want to add a feature that creates more customers for both of us? Nah, screw you and your little app too."

So I'm done. I'll take my one dollar and one cent all the way to the bank. One more adventure screeches to a halt.

Three months later, a final epilogue:

I discovered one thing where Apple excels over all other tech companies: resale value. After this app adventure ended, I put my Mac Mini up for sale on eBay. I bought it eighteen months ago for $479 and just sold it on eBay for $450. That's like selling a used car for "new" sticker price. And to continue with this theme, used iPhone 3G's are going on eBay for around $190; you can thank the wildly inflated "off-contract" pricing the carriers charge for replacement smartphones (let's say your toddler drops it in the toilet and you have to buy a replacement at full price).

I know the Ebay value because I sold my iPhone 3G for $190. Recently my contract was up and I was reluctant to try the iPhone 4 due to the bad experience my brother is having with it. After another friend demo'd his Droid X to me, I switched carriers and bought into the Android operating system. The advantages over the iPhone 3G were too big to ignore, especially the free Google Nav GPS app (I literally laughed out loud when it understood my voice instructions).

So my excitement about smartphone app development has been renewed; only this time for the Android OS. I hear it's easier, more fun and doesn't have the byzantine Apple bureaucracy.

There's an old line that history doesn't repeat itself; instead it often rhymes. As far as my computing history goes, I'm starting to rhyme with the 1980's all over again. Back then I started on the Apple II, but my family quickly jumped onto the more ubiquitous PC bandwagon due to both pricing and features. Once again Apple may have invented another new industry, but also once again the fast-paced competition is steamrolling them. The difference this time around is that Android is the new PC.

Epilogue to the Epilogue (Summer 2015):
Google bought up the advertising service that my old app used. They cleared out their accounts recently and I received a check for about $10 which represents all of the advertising revenue from Piggybank By Weight.