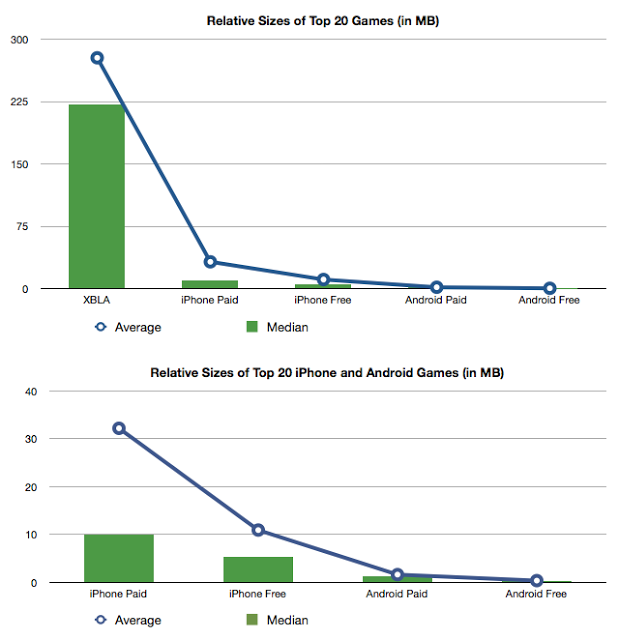
## WEDNESDAY, NOVEMBER 11, 2009

### [It's Not the Size That Counts](http://replicaisland.blogspot.de/2009/11/one-of-concerns-that-i-occasionally-run.html)

One of the concerns that I occasionally run across when working with Android game developers is that Android handsets don't have enough storage space for applications.  Some folks, such as John Biggs over at MobileCrunch, see this as an obstacle to "real" game development.  In his [article on the subject](http://www.mobilecrunch.com/2009/10/29/here-are-all-the-great-android-games-the-answer-is-simpler-than-we-think/" \o "article on the subject" \t "_blank), Biggs notes that *Myst* for iPhone takes up 727 mb, making it too large for any of the existing Android handsets. His conclusion is that the lack of space on Android devices is preventing game developers from bringing "real" games to Android.

"The real culprit behind the lack of Android apps isn't lack of developer adoption or a difficult SDK – it's the ludicrous 256MB limit on app storage for most current Android phones and Android 2.0 itself. The OS also does not support the installation of apps on removable storage like SD cards, further ruining chances for more effusive and expansive titles. "

Actually, Android imposes no size limit on application space; each device maker may choose what kind of storage and how much they want to put in their devices, and there is already a bit of variation across the handsets currently on the market. The current partition space is the result of manufacturer preference, not some characteristic intrinsic to the Android OS.   
  
While it's true that apps can't be installed on the SD card, and that the amount of available application space on existing Android handsets is smaller than some other devices, I think that this argument is based on the assumption that "better" games must take up more space.  I'm here to tell you that there's no strong correlation between game quality, or even game length, and application size.  
  
**Level of Detail**  
If we take a look back at video games over the last thirty years, we can see that, with each new generation, games have required more space than before. Take the *Super Mario* series.*Super Mario Bros.* (1985) for the Nintendo Entertainment System shipped on cartridges that could only hold around 40k of data. *Super Mario World* (1990) for the Super Nintendo shipped on a 512k cart. *Super Mario 64* (1996) for the Nintendo 64 ran in 8 mb, and *Super Mario Sunshine* (2002) for the GameCube came on a single 1.5 gb mini disc.  
  
The reason that this increase in space is necessary is that the target resolution of the host device has increased over time. The original *Super Mario Bros.* was designed for a 256 x 240 pixel display, while *Super Mario Sunshine* was designed for 640 x 480, the standard SDTV resolution. This increase in resolution required that art assets also grow in size; extra pixel density isn't useful if you don't actually have content to fill it with.   
  
This is true whether you're talking about 2D or 3D. In a modern 3D game, resolution is a function of texture detail and model complexity. Texture data includes normal maps, height maps, bump maps, reflection maps, and all kinds of other image data that has increased in resolution (or only recently become viable due to increases in storage space) since previous generations. If you are a gamer with a good memory, you might remember Hideo Kojima talking about how Solid Snake's mustache in *Metal Gear Solid 4* [contains the same number of polygons](http://www.1up.com/do/minisite?cId=3146394" \t "_blank) as an entire enemy character in *Metal Gear Solid 3*; presumably, the increase in complexity was necessitated by the move to HD televisions.   
  
So there *is* a correlation between application size and target screen size. Just not one between application size and game quality.   
  
**Running the Numbers**  
To drive this point home, I took a look at the top games on three different digital distribution networks: Microsoft's Xbox Live Arcade, Apple's iTunes Store, and Android Market. Xbox Live Arcade games are written to be run on HD television sets with 1080p resolution. That's 1920 x 1080, or 2,073,600 pixels. iPhone apps and most existing Android apps are written to run at HVGA resolution, which is 320 x 480, or 153,600 individual pixels. So the XBLA games are running at a resolution roughly 13 times greater than that of the iPhone or Android devices like the T-Mobile G1 or myTouch 3G. So, if we assume that all things are equal and that cost of detail increases at a linear rate (neither of which are very true, but bear with me here), we can reasonably expect about a 13x difference in the physical size of the same content on an HD device compared to HVGA device.  
  
Below is a graph comparing the average application size for these three networks. To calculate this average I used the sizes of the top 20 games (as of Nov. 3, 2009) from each network.

[](http://2.bp.blogspot.com/_EIf-QLT1Gfk/SvrSBVmAm8I/AAAAAAAAACI/FHSIkBmJeFQ/s1600-h/game_relative_sizes.png)

The average top Xbox Live Arcade game is 277 mb. Interestingly, paid games on both iTunes Store and Android Market tend to be a little larger than free games, perhaps because the cost of generating art assets requires the developer to sell the app rather than give it away. In any case, the average iPhone game is 21.5 mb, which is about 12.8x times smaller than the XBLA average--pretty dead on with our resolution-based expectation. The average Android game is even smaller--around 1 mb--though as on the iPhone, paid games tend to be larger than free games.  
  
Though the Android games are smaller in aggregate, there certainly are a number of titles at or above the 10 mb mark on Android Market. This is within the same range as the top 20 iPhone games; *Myst* at 727 mb is clearly an outlier (and, as an aside, it's larger than all but two of the top XBLA games). You can fit a bunch of top-tier iPhone games in a G1, and even more in a myTouch or Droid.  
  
It's interesting that the Android games are so much smaller. The most popular free apps are tiny--the average of the top 20 free games is only around 300k. This might have to do with how popular games are ranked, or the fact that paid apps have been available on Market for a much shorter amount of time than on Apple's platform. Or it might be something else;[Labyrinth Lite](http://labyrinth.codify.se/" \t "_blank), which is available on both platforms, is less than half the size on Android compared to its iPhone version. At any rate, even if the Android games were larger, the average user would still be able to fit a bunch of them on their phone.  
  
**So, Size Doesn't Matter?**  
I don't mean to suggest that size has absolutely nothing to do with game content. There are some types of games, like *Myst*, that are unusually data-heavy. But content quality isn't a function of size; *Super Mario World's* tiny 512k footprint is evidence of that. And Android*should* support installing apps to the SD card (though I suspect that some game developers would be uncomfortable with that option). In the mean time, apps are free to stream data over the network or cache resources to the SD card (as the [Android Quake port](http://grammerjack.blogspot.com/2009/10/gles-quake-port-of-quake-to-android.html" \t "_blank) does), though admittedly this isn't as elegant a solution as true SD card support.  Still, it's an easy solution for big games like *Myst.*  
  
While we're on the topic, 727 mb for *Myst* is a fairly surprising number. The original version of *Myst* targeted 640 x 480 displays at 256 colors and filled a single CD-ROM; even assuming that the iPhone version is using 24-bit color versions of the original graphics, the reduced screen resolution of the iPhone combined with advances in CPU speed and image compression technology should make it fairly easy to cram the game into less space than it originally required. After all, they fit *Resident Evil 2*, a 2-disc CD-ROM game for the Sony Playstation, [into a single 64 mb N64 cartridge](http://www.gamasutra.com/view/feature/3148/postmortem_angel_studios_.php" \t "_blank)!  
  
Plus, a game like *Myst* should be trivial to stream over the network; the spatial layout of each room is known and so room images close to the player's current location can be streamed in the background. Even if the developer elected not to cache those images to the SD card, 16 mb of application heap is a lot of room for [80k images](http://en.wikipedia.org/wiki/Myst" \l "Development" \t "_blank), so it should be easy to stream far enough ahead into runtime memory that the player never experiences a loading pause.  
  
The point I am trying to make is not that the *Myst* iPhone developers did something wrong, just that there are many ways to implement this sort of game and not all of them come with huge space requirements. I suspect that *Myst* on iPhone is as large as it is because that was the simplest way to port the game, not because the game itself cannot be made any smaller. And there's value in that--giving developers as many options as possible to make applications is the number #1 reason that Android should support installation of apps to the SD card. But the claim that this sort of game is impossible on Android platforms is, I think, wrong.  
  
**Wrap Up**  
Game quality has almost nothing to do with application size. For most games, physical size is most directly linked to the size of the screen and the resolution that the host device is running at. By looking at a selection of popular titles, we can see that on HVGA devices (such as the iPhone, or myTouch 3G) there is a fairly small range within which most games of significant scope reside. Looking at the top iPhone games, that range is well within the capabilities of existing Android devices. And while there are exceptions, they are few and far between; the existence of outliers, especially those that are large because of implementation decisions rather than necessity, is not sufficient to damn an entire platform.   
  
Should Android support installing applications to the SD card? Yeah, sure, that'd be great. Is the lack of that functionality blocking games from working on existing Android devices? Not in the slightest.   
  
And for the record, Replica Island is about 4.8 mb, which is a little heavier than I'd like, but not heavy enough that I'm going to spend time doing something about it. It has over 40 levels.

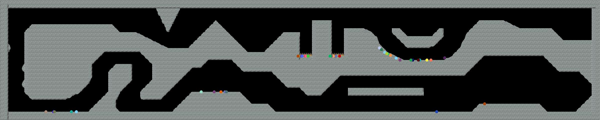
POSTED BY [CHRIS](http://www.blogger.com/profile/16079744366139636224" \o "author profile)AT [7:11 AM](http://replicaisland.blogspot.de/2009/11/one-of-concerns-that-i-occasionally-run.html" \o "permanent link) [6 COMMENTS:](http://replicaisland.blogspot.de/2009/11/one-of-concerns-that-i-occasionally-run.html" \l "comment-form)

LABELS: [TARGETING ANDROID](http://replicaisland.blogspot.de/search/label/targeting%20Android)

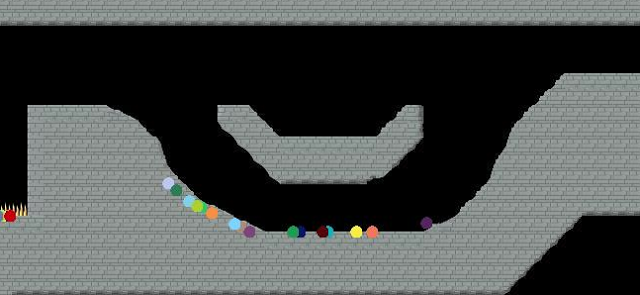
## SUNDAY, NOVEMBER 8, 2009

### [Tuning Play With Automatic and Anonymous Player Metrics](http://replicaisland.blogspot.de/2009/11/tuning-play-with-automatic-and.html)

As the developer of an "indy" game (meaning I don't have the backing of a big publisher, though Google is nice enough to give me 1 day per week of work time to spend on this), I have a problem that a lot of us indy developers share: play testing is hard. Play testing is the tried-and-true method of determining the quality of a given game design by watching users who are not experts play. When I worked in the game industry we'd bring kids in to the office, plunk them down in front of a TV that had a game machine and VCR hooked up, hand them a controller, and record their entire play session. This is hugely valuable information because problematic areas in the design are almost immediately identifiable. Nothing settles a design question faster than watching a real user fail to understand what you are trying to get them to do over and over again.  
  
But as an indy developer, I don't have an office, I don't have a pool of play testers to draw from, and I can't easily connect my Android device to a VCR. My experience has been that play testing can vastly improve the quality of a game, so I really don't want to skimp on it. And the common refrain repeated by developers of universally loved, amazingly high-quality games is that they tested and iterated and tested and iterated and tested and iterated again until their game felt right.  
  
For example, Naughty Dog, the rockstar developers behind the *Crash Bandicoot* series, the*Jak and Daxter* series, and most recently, *Drake's Fortune*, has been talking about the value of play testing for years. Back in the Playstation 1 days they pioneered methods for recording metrics about player style so that their levels could be tuned to increase in difficulty gradually. On top of that, their Crash Bandicoot series is a poster child for Dynamic Difficulty Adjustment (DDA), an automated system for dialing a game's difficulty up or down as the player plays to keep the level of challenge constant (full disclosure: I worked on a couple of Crash games for GBA at Vicarious Visions). Their goal is for the user to never see the game over screen; if the player dies repeatedly in the same spot, that indicates a failure of the design. With DDA and play testing, Naughty Dog has been extremely successful in smoothing out these "difficulty cliffs" and, as a result, their games get a lot of love.  
  
So with these lessons in mind I released a build of Replica Island to Google employees about two weeks ago. Since I can't stand over every player's shoulder and watch them play, I asked for people to tell me about bugs that they encountered and provided a very short survey form at the end. Though a lot of people downloaded the game and tried it out, very few bothered to fill out the survey and even fewer wrote me directly about their thoughts.  I learned a few things about my game but I did not collect not nearly the amount of information I need to tune my level design and game play decisions for a mass audience.  
  
Fortunately, I also built some automated metric recording functionality into the game. Taking a page from Naughty Dog's book, I decided to focus on death locations: every time a player dies, I want to know about it. Looking at the aggregate results over a number of players should reveal hotspot areas where lots of people died; these areas are the first candidates for polish and tuning. So I added a very simple function to ping a web server every time the player dies and restarts the level. The ping carries no information about the user himself: just which level he is playing, where he died, and a session identifier so I can see if people died multiple times in the same place. After I released the game to Google employees the data began to roll in, and after a while I started to have enough information to be statistically relevant.  
  
Once I had the data, I needed to decide how to use it. This is still a work in progress, but here's how I'm using it so far.

[](http://1.bp.blogspot.com/_EIf-QLT1Gfk/SvemLg_FcjI/AAAAAAAAAB4/rzHJfqWD72k/s1600-h/player_metrics.png)

This is an image of an early level in Replica Island. The little colored dots represent death events from various users (each user is a unique color). As you can see, there's a couple of obvious spots where a lot of users died. (Actually, this is only part of the data I have received so far--the quality of the results increases as more players respond).

[](http://1.bp.blogspot.com/_EIf-QLT1Gfk/SvemO484fyI/AAAAAAAAACA/FPgg5cxZZ5Y/s1600-h/player_metrics_zoom.png)

This spot is where the player encounters an enemy robot, one of the first in the game. This is a significant example because it takes place so early in the level; deaths in this location mean that the user got hit here at least three times, as there's nothing earlier in this level that can reduce the player's life. Also, now that the data has singled this location out, I can see why it's problematic; the low ceiling makes the Android's butt-stomp move (which is the only attack he has in the game) hard to pull off for novice players, and the slope likely causes people who are not used to the physics to roll down right into the enemy. This is a failure of the level design: I really don't want first-time players to die after playing a few seconds into the first or second level.  Fortunately, the solution is simple: in this case I can just remove the enemy from that location.  
  
Though this type of metric reporting and visualization is pretty rudimentary, it has already proven extremely valuable in identifying problematic level segments. It's also confirmed one of the conclusions that I drew from the survey feedback: nobody reads on-screen text, so in order to teach the player how to play I need to use big, visual indicators and design levels to require specific moves.   
  
I am so happy with these results that I'm thinking of expanding the level and frequency of the events that I report; it would be great to know where players take the most damage, or which pits they fall down the most, or how long player spent traversing specific level segments. Now that the framework for event reporting and visualization is in place, it is pretty easy to add new types of events to it.  
  
In fact, I could even release the game with this system turned on. It's probably a good idea to let the user opt-out if they want (even though I don't transmit anything personal), and I'd rather do my test and polish before the game is released, but leaving these metric reporting systems in place for the full release is very tempting. It would give me real, concrete data to tune and improve Replica Island even after it has been released.

POSTED BY [CHRIS](http://www.blogger.com/profile/16079744366139636224" \o "author profile)AT [9:28 PM](http://replicaisland.blogspot.de/2009/11/tuning-play-with-automatic-and.html" \o "permanent link) [3 COMMENTS:](http://replicaisland.blogspot.de/2009/11/tuning-play-with-automatic-and.html" \l "comment-form)

LABELS: [GAME DESIGN](http://replicaisland.blogspot.de/search/label/game%20design)

### [Android Game Development Resources](http://replicaisland.blogspot.de/2009/11/android-game-development-resources.html)

I get a lot of questions about when the Replica Island source will be released. Believe me, I'd like to see it finished more than anybody else in the world. If I had a date I could guarantee I'd give it, but I don't. It's close, though; I'm in final bug and optimization phase now. The laws of physics are simply not in my favor.  
  
But in the mean time, there's a bunch of other code and resources that you can check out if you are looking for a way to get started on Android game development. Here's a few that I found:  
  
- The Rokon game engine ([blog](http://stickycoding.com/) and [source](http://code.google.com/p/rokon/)).  
- The [Cloak open source game framework](http://code.google.com/p/cloak/).   
- An [Android Quake port](http://grammerjack.blogspot.com/2009/10/gles-quake-port-of-quake-to-android.html).  
- Source to [Alien Blood Bath](http://code.google.com/p/alienbloodbath/), a side-scrolling action game.   
- AGE engine [source](http://code.google.com/p/age/).   
- [SpriteMethodTest source](http://code.google.com/p/apps-for-android/source/browse/trunk/SpriteMethodTest), a sample which shows how to render sprites using Canvas and various OpenGL ES methods and profiles them.  
  
With the exception of SpriteMethodTest (which I wrote), I don't have any connection to any of these projects, so your milage with them may vary. However, they all look promising and I think that there's plenty of information contained within them that is useful to the budding Android game developer.  
  
If you have other links or resources that you've found useful, please post them here!

POSTED BY [CHRIS](http://www.blogger.com/profile/16079744366139636224)AT [7:51 PM](http://replicaisland.blogspot.de/2009/11/android-game-development-resources.html) [15 COMMENTS:](http://replicaisland.blogspot.de/2009/11/android-game-development-resources.html#comment-form)