eludamos
Journal for Computer Game Culture
Vol. 5, No. 1 (2011)

http://www.eludamos.org

Moving Targets: The Constant Change of Mobile Game Development

Judd Ethan Ruggill, Ken S. McAllister

Eludamos. Journal for Computer Game Culture. 2011; 5 (1), pp. 93-98

Moving Targets: The Constant Change of Mobile Game Development

JUDD ETHAN RUGGILL AND KEN S. MCALLISTER

This interview is excerpted from a series we conducted in early 2011 with James Johnson, an independent mobile game developer in the US. Prior to creating his own games, Johnson worked as a programmer for Octopi, F.U.N. Technologies, World Winner, and Sony Online Entertainment on titles such as Snood Deluxe*,* Makeover Madness*,* Pox Nora*, and* The Agency: Covert Ops*. In this interview, he talks about the dynamism of mobile game development, the interplay between programmers and designers, the differences between large and small companies, and the future of the game industry.*

How did you get into game development?

Games weren't my first choice believe it or not. After graduating from the University of Arizona with degrees in Systems Engineering and Computer Engineering, I just wanted to do some sort of programming in general. So, I applied to many different places: Raytheon, IBM, Honeywell. It wasn't that I wasn't interested in games; I just needed a first job as a programmer. I found this company in Tucson, Arizona—Octopi—that needed a Java programmer. Folks there were building cell phone games. It was a small startup that had maybe ten people including me.

Clearly, I was lucky. A friend from Sony told me that he'd looked for a job in the game industry for almost two years before he found one. I just happened to be in the right place at the right time—Octopi had just landed the Snap mobile deal. Snap was a project from Nokia, and this was back when cell phones had different resolutions and very limited memory. The deal was to create ten cell phone games in a short amount of time. So, for Octopi, that meant hiring developers. The games themselves were pretty simple—shuffleboard, chess, spades, etc.—but the whole project was kind of the beginning of creating multiplayer games between cell phones. When you would download chess, for example, you would get paired up with a human opponent to play against. Anyway, there wasn't a big launch. Nokia just released the games when they were ready.

Pox Nora was already in development then, correct?

Yes and no. *Pox* actually started before the Snap deal, but was put on hold once the deal came through. I wasn't yet at Octopi when the project first started, but initially *Pox* was conceived as a smaller multiplayer version of what you see today. The idea came from Octopi's owner and another programmer—this was when the company was two or three people working out of a garage. As I said, the Snap project came and *Pox* was put on hold. Snap was a paying project, *Pox* wasn't. Paying projects make all the difference in the world, especially to a small company. We started

Eludamos Vol. 5, No. 1 (2011)
pp. 93-98

working on *Pox* again I believe in January 2006, and actually rewrote the entire game from scratch before we launched in August that same year.

With *Pox*, the game itself is free to play. It's the runes—the collectible cards associated with the game—that are the source of revenue. You play with a sample deck that can't be upgraded. When you're interested in upgrading your champions and getting better spells, you buy more runes. The catch is that the packs are randomized, so when you buy a pack you can get common, uncommon, rare, and exotic runes, with exotic being the rarest. You can get duplicates, of course, but part of the fun of the game is not knowing what you're going to get.

Pox was always envisioned as a multiplayer experience. After launch, though, we added a single player component. Usually single player is something that comes first in game development, but the way single player works with *Pox* is that you can play a few campaigns for free or buy additional campaigns that tell stories.

Can you talk a little bit about your role as a programmer?

My job is to make sure the games I work on are done as quickly and cleanly as possible. As a programmer, you generally don't come up with your own projects. You can, but it takes a lot of time to develop them. When you're part of a small team, though, there are times when you do get to play designer. On *Snood Deluxe*, for example, a game my team did as a work-for-hire project with Word of Mouse (WOM), we were just two programmers, an artist, and an audio guy. The goal was to give the game a fresh coat of paint. Part of this update was a multiplayer mode but without network connectivity. WOM wanted to have one person use the keyboard and the other use the mouse. So, as a team we got together, did some brainstorming, and came up with a cool spider idea. The person on the keyboard would earn points by dropping and swapping snoods. The person on the mouse would fire the cannon like in the original *Snood*.

Over the years, a number of the companies you have worked for have been restructured in one way or another. What is it about you and your skills that kept you employed during times of change?

A few things have probably kept me at the table. For one, I've been told I'm a good team player. When it comes to code, we put our heart and soul into it. Believe it or not, we all have egos about code. But when you see a bad piece a code from a teammate, or someone who needs help, it's how you approach them that's usually the key. You can't go around bad-mouthing someone else's code all the time.

Another advantage for me has been my diversity in languages. I'm good with Java, C++, ActionScript, Flex, etc. Each time a new project came up for which I was asked to take on a different language, I never turned it down.

Is it your sense that this is indicative of contemporary game development, that is, that today's programmers need to be highly adaptable?

That depends. If you're with a big company, probably not. A lot of the guys I've worked with have been on the same project for years in the same language. If you're in a small company and doing work for hire, it's maybe more common. When I was

with small companies, we got projects in C++, ActionScript, and Java, mainly because we were doing work for hire. With a big company and big projects, a lot more time and energy is needed from programmers per project. *The Agency: Covert Ops*, even though it is a Flash game, is a *big* flash game. There's a lot in it. *Pox Nora* was also a huge game to develop. However, maintaining the game requires fewer people than are needed to build it. Anyway, I think maybe I'm more the exception to the rule when it comes to needing to be adaptable.

In the material that amateurs are routinely exposed to about how to break into the industry, there is a good deal of emphasis on how much programmers have to collaborate with other members of the development team: artists, musicians, designers, and so on. Does this match your experience?

I've had to collaborate with designers, artists, and musicians a lot. More so with artists and designers, actually. You kind of need to as a programmer to see what the artists need—how they envision what they're working on (clothing, landscapes, etc.)—and then you need to tell them what you need, what the engine supports, and so on. There is this kind of symbiotic relationship that happens.

It usually works a little differently between programmers and designers. We—programmers—try to figure out what the designers need, and we let them know what the engine that's powering the game supports. What's difficult though, as a programmer, is sometimes designers can have big imaginations, which can lead to feature creep. As a programmer, you kind of have to know what to push back on based on what you need to do to change the engine. Keeping track of time is also important, of course, because time is money.

It sounds as if knowing your own limits as a programmer is important. A programmer needs to be able push back and at the same time not give the impression of doing so for reasons other than the limits of skill. Is that true?

Right, knowing your limits as a programmer and the limits of your team is crucial for planning. People don't like it when deadlines slip, but everyone wants to create the best product possible. Finding that happy medium is not always easy, and something usually has to give, which may not always be a feature (but often is).

As an independent developer, you have chosen to focus on mobile gaming. Do you have a preference for that platform, or is your choice indicative of something else?

One of the reasons I'm focusing on mobile, and iOS [Apple's mobile operating system] in particular, is that the market is still decent for developers who don't have a lot of resources. So, building a good game for the iPhone doesn't necessarily mean creating a ten hour game. Just look at *Doodle Jump*—the graphics aren't deep or spectacular, the game's not long, but it's a million dollar seller.

Another reason for mobile is that Apple as a publisher has good terms. I believe that the developer's cut for a product in the iPhone store is 70%. If you can get your game in the top 10, it's going to be visible, and you'll have a very good revenue stream.

But Android is also good. With my current project, the goal is to develop for the iPhone and then port it to Android. Sense Android has been picking up a lot of people lately.

How has mobile development changed since your days with Octopi?

It's changed a lot. For one thing, I'm only building for one, well, three devices right now: the iPhone, iPod touch, and iPad. But, I can focus on one and get all three—a person with the iPad can still play the game if it's built for the iPhone. The devices are essentially the same. Back when I was developing games for Nokia handsets, though, things were more complicated. I had to make sure games worked for at least three different screen resolutions and memory requirements. We had an engine at Octopi that did some of the leg work to keep a handle on this, but still, porting to different platforms is a laborious process. And it wasn't like porting to a different market like it is between Android and iOS.

Memory requirements today are also nicer. You still have to manage memory, but having a game that's over 100k is no longer a big issue.

The language I'm using right now is Objective C. I started learning it initially when the iPhone first came out, and went back to it again in October this past year. It's a very different language from what I'm used to. If I end up porting the current project to Android, I'm going to have to rewrite it in Java. I'm not too worried about it, though, because I'm building it in such a way that it will be easy to port. The assets might need some rescaling, but the code has a layer of programming such that as long as I'm using the engine I built for it on Android, it should make the conversion a lot easier. Sure the syntax is a bit wonky between the two, but there is a layer of object-oriented design that's similar between Java and Objective C.

Having worked for both small game companies and one of the largest in the world, what are some of the differences you have seen in terms of being an employee (specifically a programmer)?

When you're an employee of a small company, the goal is to be lightweight, keep costs down, and sometimes wear multiple hats in terms of the jobs you do. You're generally a bit more involved with how things are being designed, and a little bit more input is given and taken. You're not a captain, necessarily, but definitely the kind of sailor who keeps the sails hoisted. You're not steering the ship, or changing the direction, but you are making sure the ship is going where it needs to go.

Sticking with the nautical theme here, working as a programmer in a big company is more like being a sailor in a fleet of ships. There's an admiral who tells all the captains what needs to be done, who then gives orders to someone else, who then tells someone else, and so on until it eventually gets to the programmers. The big thing is communication. At a small company you can pretty much talk to the president any time. You hang out, have lunch, and exchange funny emails. You don't do that at a big company, even though the actual programming work is pretty much the same.

That said, with a big company there are generally a lot more development tools available (for version control, bug reporting, etc.), or at least more pay-for tools. It's

common for small development houses to use open source tools, but a big studio will typically prefer commercial tools.

What are some of the game development trends you see going forward?

I see a couple of things happening as far as game development goes. In fact, they've pretty much already happened. One, developers are going to be looking at shorter development cycles, especially for new IP. It's a risky investment to go with new IP, whether or not the game turns out well. If public opinion isn't there, or marketing is poor, you're sunk. For example, *Pox Nora* has an 86 on Metacritic, but there are a lot of people who've never heard of the game. So, definitely smaller life cycles for unknown IP. And this on both the front end and the back end.

Another trend is sequels. They're safe, and require less work on the engine. With a sequel, you're either polishing or adding features to the engine, but you're not writing the game from scratch. As an artist, if you're doing 3D models, you can reuse the same models to start from, maybe making a few changes here or there. Then all you have to do is add a few more baddies. Sure, the levels could be new, but players like to revisit old locations to see what's changed.

In the end, as developers, we're always trying to bring down development time, which is huge for a lot of games today. I don't think it's something that companies want to sustain. It's the same for a lot of players too, actually. I was in Blockbuster this morning debating what game to rent with my two game coupons. I looked up and down the racks and the thing that kept nagging at me was that I can't pick any of these games. They'll all take forever to finish. I should just buy them and it'll cost me less money. In the 1990's, games were shorter, and Blockbuster was making a killing off rentals. Now the trend is toward phone-based games that take less time to develop and to play. Console games aren't rentable in the state they're in right now. Lots of companies want games that last as long as *Final Fantasy VII* (though they don't necessarily want to build them), but as a player how many times do I have to rent *FF VII* to complete it?

What does this mean for MMORPGs, multi-player shooters, and other sorts of games that typically try to deliver a lengthy play experience?

There will always be a place for them, but the market doesn't need a whole lot of them. MMOs are fun, and they're good money—the subscription plus the up front. The bad thing is that they take years to develop, and a lot of the development staff have to continue working on the game after release. So, they're costly, there are a lot of eggs in one basket, and the competition can be really stiff (think of going up against *World of Warcraft*).

Multiplayer shooters are a different story. Think about *Left 4 Dead*. Yes, it's a shooter, but it's pretty unique. The game has four campaigns (five for the sequel), each with four levels. It seems limited, but each time you go through the campaign it's a different experience. You can also increase or decrease the variation of this experience with random or non-random players. If you were to finish all of the campaigns on normal difficulty without dying it would probably take about six hours. It's a short game that had a short development cycle (especially the sequel) and is very popular. I think this is where companies are headed. MMOs will still be around,

but only for companies willing to take a big risk. The same for big *Final Fantasy*-style games, but even Square Enix has been taking a step back and asking whether it makes sense to produce another fifty hour *Final Fantasy* title, or to look at making more mobile games like *The 3rd Birthday* instead.

Thank you for taking some time to share your experiences with us, James.

My pleasure. Keep on gaming!