Chapter 11

How We Make Mobile Work

An Indie Perspective

James Barnard

After 15 years of corporate game development I went indie, built a studio and made many games. I made many mistakes and had many successes. How did I balance the idea of making art versus making a business? What would I do differently and how can people not make the same mistakes? This chapter will be an overview of the differences between indie games in mobile and corporate game development, and how we can still find ways to make money despite not having the tools that the bigger studios have.

OK – welcome to my chapter! Like any good game pitch or steam product page, I am going to try and ‘sell you’ on the benefits of my chapter, and if it is not for you then skip ahead… no hard feelings. I have to first accept that what I write isn’t for everyone, just like a hardcore JRPG fan isn’t going to buy my Arcade-Prison-Sim, no matter how pretty I make the screenshots, or how many months I spend refining that trailer. Everything is not for everybody, which is probably partly why I now run an indie studio – but more on that later.

The synopsis: Well, this chapter is about how triple A works and why that contrasts with indie. The processes used by big studios draw parallels with a lot of creative industries, and even some uncreative ones. How
do these processes help in the creation of games, and why do they exist? I have seen people apply AAA methodologies to indie often with disastrous results because they did not understand what certain processes were designed for, they just see it as a part of ‘how you make games’. Processes are there as tools to help us, but we need to understand them, just as a brain surgeon uses a drill to open your skull, he probably should not use the same drill when trying to cure you of the flu.

Understanding the way big and small studios work is an opportunity to give yourself a better chance of success in the deadly world of indie game development. In this chapter you might realise some of the reasons AAA companies are amazing places to work, and also some of the reasons many people decide to strike out on their own, just like I did around seven years ago.

Before we get going, a little bit about me – I have been making games for over 20 years, and throughout that time I have been lucky enough to work as a producer, artist, musician and programmer. However, my main job has always been design and creative direction. I worked on several franchises you may have heard of, like *The Sims*, *Star Wars* and *Pac-Man*. I also have shipped just over 50 games. Entertainingly, while that all sounds great as a LinkedIn profile, when we pull back the curtain a bit, the sheen quickly falls away.

Most games are not glamorous: I worked on *SimCity 2000*! But, no, not ‘that’ *SimCity 2000*, I was writing music and testing the Gameboy Advance (GBA) version. Porting a game to an inferior platform is never going to be the best job in the world. For one, we were porting the game to a console with barely any processing power so people hoping to get ‘a portable version of that game they love on PC’ were only ever going to be disappointed. No matter how hard we tried, it wasn’t going to be possible to perfectly recreate *SimCity 2000* on a Gameboy. As a game designer, even if you have an idea that could make the game better, you can’t actually do anything with it. Because, of course, the game is a port, and all the design has already been done. So making a change that isn’t required due to technical limitations would be a deviation and, therefore, not what players would be expecting. This feeling of general uselessness is not as terrible as it might sound, the experience of building something with a team, overcoming technical challenges and eventually seeing the game on a shelf in a store (this was a long time ago!) is still something that can be relished. But ultimately no one gets into the games industry to do ports of existing games, you get into it to create new worlds and new experiences
that delight players. What may surprise you is that the further I got in my
career, the more and more I realised that the experience of working on a
port like SimCity was not that dissimilar from being the creative lead on
the next Star Wars game.

That may sound like a stretch, but bear with me. I am going to try to
explain the structure of corporate games (and probably any large com-
pany with shareholders), and what it means to you as a creative individual
working within a large company.

First, why did SimCity on the GBA exist? Well, clearly EA made the
game to make money, but why that? Well I am sure it’s obvious, SimCity
is a well-known game, no doubt Toys ‘R’ Us alone would have pre-ordered
enough copies to ensure that the COGS (cost of goods – i.e. manufac-
turing the cartridges with Nintendo) and development costs would be
covered. Therefore, by starting development with zero financial risk, EA
could guarantee a small profit for them and their shareholders. As a busi-
ness you would be stupid not to do this wouldn’t you? Who cares about
the game designer in his mid-20s feeling like he is wasting some of his life?
As a game developer he is just a tiny component in this whole machine, in
which the only measure of success is profit.

It is important that we stop and think about this, a company is not usu-
ally altruistic by nature. Even if perhaps they appear to be, it is probably
just good PR that in some way is expected to pay off as profit in the long
term; 99% of the time, the people at the top of a big company are no longer
involved in the creative process, their responsibility is to their sharehold-
ers and, to a lesser extent, keeping their employees gainfully employed.
Therefore, when you walk into an executive’s office and tell them about
your amazing game idea, they do not really care about the idea at all. They
mostly care about two things. One – will this game make money? And
two – do we have the ability to actually make it, whether that means peo-
ple on hand in the studio, or a production plan that factors in working
with a third-party vendor such as Virtuous or any number of work for hire
companies that have sprung up over the last couple of decades.

So point one, ‘Will the game make money?’ How can this be answered
in a meaningful way that eliminates as much uncertainty as possible. I
think as game designers we often just believe in our ideas so much that
we just know people will be falling over themselves to play it. ‘Look – you
play as a sentient spaceship with giant tanks for arms! What’s not to like!? ’
Unfortunately, that is not going to work. Your passion will really help with
point number two (do we have the people to make this), but no one is
going to drop potentially millions of dollars on your idea, just because you think it is great. Imagine if you were a president of a large company, and you invest in a game just because you think it sounds ‘cool’. Now imagine that game fails at launch and endangers the entire company’s future. The first thing someone will ask you is ‘why did you approve this game?’ If you managed to reach the level of company-president, you probably need to know more about business than you do game design, and answering your shareholders ‘The designer seemed to really like the idea, so I thought why not!’ is probably going to get you fired.

So in order to actually put a game into production you would probably go through several steps. First, you would have done a P+L (profit and loss summary). In this, your most overpaid experts will do everything within their power to predict the future. This starts with some accuracy when the team thinks about what might happen if the game was going to be released tomorrow, but then proceeds to become less and less accurate as your team tries to imagine what the market will be like in two or three years’ time when this game finally launches. So they look at trends, are games of this genre becoming more popular year over year? Do we see multiplayer as a safer bet because it has a longer tail (the period of time that it continues to sell after launch) as opposed to single-player story games that tend to crash in price quickly after launch due to sales of pre-owned copies and other factors (because once someone has experienced the story, they don’t need to play anymore). If we are making a single-player game, how can we counter the negative sales impact inherent with that type of game? Perhaps release some DLC, so people need to hang onto their games for longer to get the whole story, or sell a season pass that not only makes people hold on to their games, but also gives the opportunity for boosting the bottom line revenue. Based on today’s knowledge, how are sci-fi games generally selling against other genres, perhaps we should suggest that the game be a fantasy game instead, as these tend to sell 15% more on average. And if the main character is a white male, we will sell 32% more than if the character is a woman from a non-Western country, however, there seems to be a trend in breaking that norm, which may be bringing about a new norm. So yes, perhaps a white female with slightly revealing attire would sell better, let’s focus test it and see how potential players react, and map their reactions on top of predictions made by market analysts and so on. You get the idea. Almost every decision has to be validated against the current market, which is used as a barometer to test potential sales and profit, which then eliminates risk. When the president is answering those
hard questions, they will have a big pile of data to back them up ‘it should have done well because of all these predictions, we didn’t know at the time that 100 player battle royale games were going to become so popular and destroy the market, we just went on the information we had to go on at the time’.

So, if you understood my waterfall of words, you may have realised a problem with this approach? Yes, innovation becomes the victim on the side of the road to sustainability, predictability and profit. Because our designer wanted to make a game about spaceships with tanks for arms, the marketing department couldn’t find enough similar products to benchmark against. Which then resulted in the game being too much of an unknown entity to green light into production.

There are ways to lessen this risk (for example, building prototypes and market testing them), but generally this is how things work. Why does EA make \textit{FIFA} games year after year? Because they know almost exactly how many copies they will sell before they even start making the game! How many totally new IPs do we really see from big companies? Sticking with EA and their one new IP \textit{Anthem}, they want to do their best to guarantee success, ‘Let’s make a game a bit like \textit{Destiny}, even if it costs 150 million dollars, at least we have an example in the market that we can look at and measure ourselves against’. The fact the game failed spectacularly is not really the point. It was a sound plan, backed up by predictions and calculations that eventually failed due to unforeseen circumstances. But don’t worry, they spent so much money on the thing, that I am sure there will be a sequel that does ever so slightly better than the first game.

It’s a real shame that things are this way, it perpetuates stagnation in the types of games we play and reinforces stereotypes and a whole host of other terrible things (why is almost every Ubisoft game set in an open world full of fetch quests?), but it makes sense. Indie companies die every day, they release game after game that doesn’t sell. When you are Namco, and your monthly burn rate is probably 30 million dollars, it doesn’t take too many failures to empty your cash reserves and sink your company. Of the many indie games that fail, a small percentage go on to become huge hits, rewarding their creators with unimaginable fortunes. It is fair to say that if all indie games were released by one company, their ‘\textit{Minecraft}’ would not make up the shortfall of all the other games that fail. It’s very fair to compare making indie games to gambling, you know that the odds are stacked heavily against you, but you still want to do it anyway. Just like people that play poker who say it is a game of skill, no doubt indie
developers (like me), think we have some magic edge that puts us ahead of everyone else and increases our chance of ‘winning’, but really, we probably sound like those scrawny sleep deprived gambling addicts in Vegas who will tell you they have a ‘system’ and that it’s all going to work out eventually.

In 2018, the average return on an indie game was less than US$30,000. Seeing as most games are made by multiple people, and take more than a year to build, the return on these games is generally worse than working a minimum wage job a few hours a week. And this is only part of the picture, the way sales are distributed means that the top games make the majority of the money, so the games at the bottom aren’t making US$30,000, they are probably making double digits in sales. A report on Steam stated that in 2017 the top 0.5% of games on the store accounted for 50% of all revenue spent. And historically this is a trend that is getting more and more pronounced, meaning that it’s harder than ever to make a living as a game developer. This pushes big companies to take fewer risks than ever and rely on existing known IPs to generate sales, so the aforementioned spaceship-tank-arms game is looking even more unlikely than ever.

In their search for profit, larger companies have begun to follow the successes of smaller indie companies very closely. In that one in a billion chance that an indie game does become successful, large companies are increasingly taking inspiration from those titles and building their own similar product. When Minecraft became a huge hit, it wasn’t too long before people started following them with games like Lego Worlds, or the somewhat unlikely Dragon Quest Builders, eventually it felt that the building things your way was permeating into almost everything from Fallout to indie ultra-hit No Man’s Sky. The time it took for Minecraft’s influence to go through the industry could be measured in years. The next game to have a similar impact was PUBG, which spawned Fortnite and Apex Legends not long after. Fortnite was a very different game originally, but it was reshaped in the image of PUBG and released into the world, where it generated a staggering US$2.4 billion in 2018, while PUBG still made a respectable US$1.08 billion that same year. It wasn’t too long until Activision got in on the action, and launched their very own take on the genre, with Apex Legends. The time frame in which these big budget clones appeared was substantially shorter than the time it took for Minecraft to end up rebuilt as a Dragon Quest game under the Square Enix umbrella. Fast forward to the indie game/mod sensation known as Auto-Chess and it took only six
months for both Valve and Activision to launch their own fairly shameless takes on the genre (they both released within the same week!).

So what does all of the above mean? It means that when working in a large videogame company, the chances of you getting to ‘Make your dream game’ are technically zero. Even if your concept gets through to production, the marketing team may well shift your design pillars so violently that it doesn’t feel anything like the game you wanted to make anymore. You perhaps have read some of the stories surrounding production of large games like *Anthem* and *Destiny*, and how they went wildly off the rails. To me this is endemic in AAA games, because no one really has a final say, even when you think the creative director is the final person in the decision chain, that person can be entirely undermined if a marketing test does not go to plan and the game needs to be changed to increase the probability of success.

Earlier on, I mentioned that the company-president was looking at two main factors when agreeing to start developing a game. The first was profit potential, the second was being sure that the company could make it. In many ways this ties into the profit, because even if you have the greatest product idea, the truth is that if you do not have the people to make it, you will not get very far. So how does a big company ensure it has the team to actually build a game? There are a couple of factors, one is having the right people leading the project, these directors need the passion and drive to push the team through the tough times and deliver the game. This group of people need to solve day-to-day problems, while keeping everyone on track to deliver the game that matches the original risk assessed product description. Usually, once this leadership group is in place, their job is to break down the next few years of work in as realistic way as possible. This is about as scientific as you can imagine and is just a big list of random guesses and gut instincts. The more games you make the better you get at guessing what it will take to make one, but really there is no way to accurately predict how long anything will take in game development, which is partially why most games are late. So with a schedule in place, the team will start making things. Despite the pretence of ‘Agile’ development, there is almost always a target end date for the project. This means that while you can explore different ideas and concepts, you still need to do them within a certain time box. And better yet, those explorations still need to line up with the product the marketing department thinks they can sell.
In simple terms, a project has an estimated sales potential, which in turn means there is maximum budget that the company should spend, which finally means you need some kind of schedule and timeline in order to try to ensure that the game will make money.

As you can imagine, predictability is a very important part of this process. If a game is going to go over budget or over time, you would want to know as soon as possible. Either can clearly have such drastic impacts on the game that it might need to be cancelled. If the game is going to be late, perhaps a new console will have been released which might affect your sales negatively (compounding the fact that the longer timeline probably pushed your budget up too). So how does a company deal with predictability in the unpredictable world of game development? The answer is a combination of micromanaged schedules, milestones deliveries and expected redundancy.

A milestone is your big delivery. Often a milestone is viewed as a possible cancellation point for a game. For example, delivery of a vertical slice (a section of gameplay that is expected to be representative of the final product) might represent the first time external people get to play the game. If it does not meet expectations in focus tests, the game could easily be canned. When milestones are late, or do not achieve all their objectives, it is usually a sign that things are not going to plan. If milestones start to slip, it is usually impossible for a team to make back up the time. From there, it is possible to get a prediction of how late the game might be and decide the future accordingly.

The idea of redundancy is a harder one to explain, but it hit me several times in my career. I would be working on something and get to the end of my task list. I would then want to use that extra time to add something extra to make the game more awesome, only to be told not to. To me that did not make any sense at all! Why wouldn’t we want more stuff in the game!? Well, the issue is that if I add something else, I might make more bugs, and that would have a knock-on effect for the rest of the project, this extra content would have to be tested and maintained until we shipped. Equally, I might overload myself and shift from being ahead of schedule, to dropping behind, and as we know, the most important thing is that we stay on schedule, and ship the thing that we set out to make in the first place. Perhaps my awesome new idea is slightly out of line with the original vision for the project? I can see the logic behind a producer thinking: it’s probably best to not take the risk and just find something else to do to fill my time. Finally, everything in games is always late so, when we make
a schedule, it is a good idea to add a whole heap of contingency time in. If a programmer says something will take him a day, write down two. Anyone who has made any number of games will agree, nothing ever goes to plan, so this kind of practice is fairly standard.

In general, from my personal experience, everything really starts to go wrong when the game starts coming together. That design idea where the robots tank arms would detach and be controlled by other players on the internet is too complicated to make work well, or maybe it is just not as fun as we imagined. Whatever the issue is, it usually results in something getting changed, and that redesign means suddenly there is a lot more work, and there isn’t space in the schedule for it. So either some of that redundancy gets taken up, or if there isn’t enough left, you have to cut something from the game. This regimented drive towards delivering games on time and on budget almost always affects the quality of the product you are working on. And it is something almost all teams have to deal with at some point or another.

Task micromanagement exists in many teams but not all. The concept is that you have a set of tasks to complete, you may have a set order in which they are to be done, or you can just pick whatever you fancy doing from a bucket of tasks. When you take on a task you set it as in progress, so everyone knows you are doing it, and then set to complete or ‘ready for review’ as soon as you are done. In general, this is considered to be a great way for the team to see what other people are working on or easily see the history of a task so they know how to find out more information. The task itself should have all the required information written down to make achieving it possible. This is all very efficient. The reality is that this inter-team efficiency is really only a very small part of the picture. Task tracking software like Jira or Hansoft is very good at generating graphs and reports, that in turn make it very easy to see whether a team is working to their schedule. If you drop behind by just one hour every day, the compounded nature of this constant slippage can be brought to light quickly within the project management reports. This information can then be used to correct the path of development by adding more people or cutting content.

As far as I know, a variant of this kind of tracking is used at every major games company. I personally despise this system. When a valuable programmer is typing in task information, forwarding tasks to other people, reviewing someone else’s tasks or a host of other project management related activities, they are not programming. In my career, I have seen companies that assign as much as 25% of a day’s working hours to...
administrative tasks such as Scrum meetings, task planning, progress reviews and so on. When we are making games, the worst thing we can do is do something that does not directly contribute to putting something on the screen. These admin tasks exist mostly to alleviate concerns in management by building a predictable record of the team’s progress. So by having such a detailed history we can attempt to predict the future! Even if 25% of the games budget (at least) is spent making sure that the company knows whether or not to stop development it’s worth it. In a game that costs US$100 million, if US$30 million in you can see that it might actually cost double that amount, the corporation can cut their losses easily. Equally, if you start realising that the game will be late, perhaps pull the plug on that US$10 million-dollar marketing campaign you had lined up for this year’s E3 convention.

When I started my company, I did some very high-level milestone style scheduling, and that was about it. I quickly realised that by keeping the team small, we did not need to have complicated communication channels. I also know that realising a game will be late is not going to change anything! Unlike a big company, most Indies don’t have the money to cancel a project and move on to something else, we simply have to do our best to finish it. If we do it ahead of time, then great! But if it takes longer, then every second we spent pushing tasks around on virtual whiteboards is a second earlier we could have finished. For people exiting AAA and going to indie, this is usually the biggest mistake I see happen time and time again. I often think when people are in AAA development, they do not question the systems around them. They just accept that this is how things are done in game development. For me, the most important thing to remember is, ‘If I am doing something that isn’t directly contributing to the game, or selling the game, then maybe I shouldn’t be doing it’.

Another thing that happens when AAA developers become indie, is that their confidence can undermine them. In a big company there is a huge support structure all around you, you take it for granted that when the network goes down, someone from IT will fix it. Or if your PS4 dev kit needs updating, someone else will do it for you. When you go indie, you realise that no one is going to do things for you, you have to do it all yourself! This sudden influx of additional responsibilities can result in people being able to do a lot less work than they anticipated. Going back to the rule that everything takes longer in game development than you anticipate, sometimes the switch to being a solo developer can make you feel so unproductive that the pressure mounts and you become even less...
productive. It's a kind of psychological stacking: where you hit a roadblock and there is no one there to help you over it, you just throw your hands up in despair. To succeed, you have to forget the life of AAA, you are now the same as anyone else and have to fight to make it by pushing yourself to be productive each and every day. The fact that you were a lead engineer on \textit{GTA VI} means nothing now, and you will have to do all those boring uninteresting tasks that you used to just pass off to junior programmers.

On the flip side of this, there are people who become indies without ever having worked in a games company. The kind of mistakes these developers make are often similar, but many new developers lack the sense of urgency that the endless grind of AAA teaches you. If you are in this situation, it makes sense to step back and envision what success means for you. To do this, you have to know a bit about the industry, watch YouTube videos from developers who cover their own experiences and have discussions within your local games dev community. It is easy to trawl the internet and read about the success stories of Toby Fox with \textit{Undertale} or Eric Barone with \textit{Stardew Valley}. Believing you can replicate the success of these people is like buying a lottery ticket and putting a down payment on a house because you are convinced you will win. If you study a broader range of developers you will realise fairly quickly that success is a rarity, not the norm. Once you understand that, try to set out a plan that aligns with a more informed version of reality.

Don't assume you will do any better than the average, in fact many of the people who make up that average will have things over you such as experience and industry connections.

So ‘what does success look like for you’ is an important question, and when I was first asked it by my publisher Kongregate, I didn’t fully grasp just why it was so important. My first answer was probably something like ‘To make crazy money and make whatever games I want to for all eternity’. That was clearly unrealistic. Knowing the things I know now about the industry, I think obtaining ‘crazy money’ in mobile requires ‘crazy investment’, large teams of people making sense of the data collected on your players, lots of money to spend on UA (user acquisition), along with teams of people who can actually optimise your UA strategy so it can return a positive ROI (return on investment). On top of that, you need people managing customer support and ensuring that your ad networks are optimised for the biggest profits. If I wanted to make as much money as possible, I should probably get some investment, and move away from actually writing code on a daily basis, and create products for the
demographics with the highest spending power and so on, essentially running my business like a AAA studio. The other part ‘make whatever games I want to’ is just a contradiction to the first part.

If I break down my answer to this question today, it would be something like ‘Have a sustainable income, creative freedom and have fun every day’. Looking at my business, I can break down the ways that this can be done. We do not earn a lot from our existing games, and I see the trend of new games not getting the big sales spikes that they once did. So in order to make enough money to survive, we just need more games. Looking forwards and thinking that the next game will be a huge hit, is just wishful thinking. Therefore, each game I make, I try to think about whether the idea is simple enough to make in a reasonable amount of time and what monetisation potential it has. This sounds cold, but it is not. It’s just sensible. I find it easy to come up with ideas, so if an idea does not fit my plan, then I just scrap it and think of another one.

The point is, when you have a more realistic vision of what success means for you, you can work backwards and determine a plan that makes that possible. If your plan does not result in that success, then you need to try a new plan! I have met too many developers who think vast success is easy to obtain, and so far, only one developer I know has managed to become successful in true fairy tale fashion.

The other thing that almost all new indie developers do is procrastinate and over scope. If you think your game will take a year, double it, Hell, maybe even triple it. Your first game will fail, and probably your second one too, so it is probably best to make something very small as your first game and save the ‘big dream project’ for later. I still have not started on the game I dream of making, and I probably never will. The experience of launching something will teach you so much, and if you have a community in place (like Reddit/Discord) you can start building a small following. My company really started using Discord about a year ago, and when we added links to our most recent game, the community started to grow. If you are selling premium games, having a community or a mailing list is really important because with premium games it doesn’t take too many sales for it to be meaningful. But the most important thing is just to release things, don’t make it perfect, don’t rebuild an entire system in the game unless you really know it’s worth it. A question we ask ourselves is ‘what is the value of this feature?’ By ‘value’ we mean ‘will this translate into more sales? Will this get the game to generate more traffic on social media?’ We then compare that value with how long it takes to make the feature
(which really means how much it will cost). Even though sometimes it is hard to estimate what small details will do for the overall experience, it’s important to be critical and look at your game as a business otherwise your company will fail pretty quickly.

Also, never stop learning. Chances are your team are going to let you down at some point (or you will let them down) so unlike AAA development, it always pays to know enough about every discipline so you can get by. Perhaps you will need to fix a small bug and upload a new build to the App Store or open a file in Photoshop and move the layers around so you can create that new image that Sony requested. Successful indie developers are usually doers, they work hard at making the actual content of the game, rather than being managers or ‘ideas-guys’. This isn’t totally isolated to indie games, Naughty Dog (makers of Crash Bandicoot and Uncharted) employ the same approach, and have publicly stated that they don’t have producers on their teams (producers do schedules and management tasks, not actual development of the game).

Reading back over this chapter, I see I painted a fairly bleak picture, but I think it’s important to understand the realities behind the ‘fun creative world of videogame development’. I personally would never do anything else, making games brings me fulfilment like no other creative endeavour. Whichever path you decide to take, the road is a hard one. To me, the safest bet is to work for a big company and learn enough to take the leap into your own games when you have an honest and realistic plan. Or make games on your weekends for fun while you hold a game development job during the week. My very first indie release was actually made in my evenings while I was still working at LucasArts. While that went against the contract I had signed with them, it was an opportunity for me to prove to myself that I had what it takes to actually build and launch a game on my own. Without that experience, I am pretty sure I would still be working for someone else.

LUDOGRAPHY

7. Fortnite, Epic Games/Epic Games, 2017.