Pipeline Management: Best Practices

Jennifer Tidwell

There is no set process for managing a video game pipeline. It is something that a producer must develop alongside their team each time they begin a new project. Coming up with a solid plan usually consists of trial and error as producers build experience. So, I talked to five project managers in the hopes of finding a common thread that could help pinpoint how to start a project off right. This report is divided into three parts. Part One will cover five steps to consider when managing a new project, while Part Two will outline various management methods used in software development. Finally, Part Three covers additional advice for pursuing the path of production management.

Part One: Five Steps to Stay on Track

Step One: Do Not Hover. Pipeline management comes down to more than just technical skill. It also requires interpersonal skills that come from experience and practice. It is important not only to keep a team focused, but also to know when to step back and allow the people on the team to deal with issues on their own. A producer should know when to step in and when to stand back. According to Bryan Haskell, when he started as an associate producer, he tended to do what he called “shoulder-surfing” – checking and double-checking on his team unnecessarily, to the point of annoyance. Now he makes “a point to learn [his] teammates' strengths and where they can improve, and coach them over time to get really good at raising any concerns they may have”. This encourages the team to be active in seeking his help, rather than having them be reactive to his prodding.

The difference between a team that reacts to a producer and one that actively seeks their help is trust. No one on a team wants to believe that they are being unnecessarily scrutinized. It is important that the producer is seen as trustworthy, just like it is important that the producer trusts their team.

Stepping back does not mean stepping down from confrontation. Disagreements between teammates will occur, and mediating those confrontations can be another aspect of a producer's job. When asked about the best advice he had received from other project managers, Haskell said "stick to your guns. I equated disagreements with failure, and tried not to make waves… You should never be afraid to raise your voice if you feel strongly about something, or feel a game will be made better with your ideas". This is not only true of a producer or manager, but of every member of the team. Passion can create disagreements, but it can also create a better product, and it is important that a team is passionate about their job. A producer can be essential in making sure that passion is directed in a way that is good for the project.

Step Two: Socialize. When asked about the worst advice he had received as a young producer, Ian Shepherd said that he was told to accept that he needed to always be available, and that he could "not separate [his] personal and work lives because this is video games". Ian believes that a producer should not sacrifice their personal life for the professional one, nor should they need to. He says that it is possible to "have it all". A simple way to do this, which also contributes to a better work environment, is to make friends with colleagues.

My friend and former colleague Ari Patrick, who began his career at ID Software, is currently a project manager at Reel FX Creative Studios. I have witnessed first-hand Ari's dedication to cultivating his team's morale. While he is dedicated to keeping the team on track with daily meetings, he also plans social outings for the team to take together. These events

ranged from group lunches to an entire half-day off from work to ride bumper-cars. Team-building can be incredibly simple when done at a gradual, organic pace.

"I spend a lot of time working to understand people's motivations, what problems they see, and how they would resolve them," Ari said. It does not mean that he can instantly fix every problem, and I have also seen how people can negatively react to the number of meetings he plans for his team. However, Ari is dedicated to making sure that everyone knows he is available to his coworkers if they need him, both on and off the clock.

Step Three: Look for New Information. Personally, I consider this step to be one of the most important. So much of a producer's job is mediating not only between people, but between their disciplines. When speaking to the five different managers, each one mentioned the idea of remaining up-to-date on new information in one way or another. Bryan Haskell said, "Actively seek out new information to grow your skills…. Keeping abreast of the latest technologies, trends, and development methodologies… will make you a much better producer than one who sits on their laurels and grows their abilities and knowledge via osmosis."

Producers and project managers are often assigned to individual departments that together make up the entire video game. For example, Ari is the project manager for the technical directors at Reel FX, not the entire crew of a film. This means that he needs to understand the work that goes into being a technical director. Jeff Mills told me, "If you don't have background in the disciplines you're managing, do your best to learn the day-to-day processes required…. When programming fell into my purview, I spent a great deal of time interacting with the programming staff to learn the pace and range of their work…."

Having knowledge of how to do the job a producer is managing makes them a greater asset to the team. It gives team members another person to bounce ideas and problems off. That

is a producer's major responsibility: to be a problem solver. Brandon Johnson of Game Circus told me, "Never should the words 'well, that's not my job...' come out of your mouth, or even run through your mind. Your primary focus is to remove any impediments that the team is facing." Scheduling, task management and communication are all tools a producer creates to help their team solve problems and operate on a clear path.

Step Four: Make Cuts Where Needed. A sometimes difficult but necessary choice that producers must make is when to make cuts to a project. This does not always mean cuts to the scope of the product, it can also mean making cuts to the team. There can be multiple reasons for letting a teammate go. The game design pipeline can often be envisioned as a staircase, which each step being a different discipline or piece of the team. Certain disciplines rely on others, so scheduling artists, or animators to begin and end at different times is an important consideration.

As a project grows, however, the schedule might change. Team members may need to be kept for longer periods, or for shorter ones. That is a decision that can be difficult to make for a producer. It is a balancing act: balancing the needs of the project with the needs of the team. A careful thing for a producer to consider is not taking too much of a project onto themselves, because no matter their intent, their work will trickle down to the rest of the team. "It's easy to agree to things, especially when you're fresh in the position and eager to prove yourself. A noble 'go get it' attitude for sure, but if the goal is unrealistic… you're setting the team up for failure," says Brandon Johnson. Knowing what to scope, as well as where and when to scope it, can be a difficult question to answer, but knowing how to approach the situation comes from team communication and experience.

Another reason to make cuts to a team is if someone on the team is too difficult to work with. It is part of a producer's job to work with people they might not like, but there comes a point when difficult coworkers become a detriment to the project. A producer cannot be afraid to let someone go if their attitude is unprofessional or unacceptable, no matter how talented they are. It is the same with the project itself: sometimes a great idea is simply not working and needs to be cut. Often there is just not enough time in the schedule and the project must be scoped. These are the decisions that a producer must be capable of making.

Step Five: Know the Project's Needs. It is important for a producer to understand the people on their team, and what they need. It is equally important, when beginning to plan for the project, to know what the *project* will need to succeed. There are multiple methodologies that producers can use for managing a project, as well as different software programs that cater to these different methods. A popular method is Scrum, which is a framework that focuses on iterations, which makes it ideal for large software development projects and teams.

When asked if he preferred Scrum or another method, Haskell told me, "I've found… that the type of project management applied to a project should fit the requirements of the project itself". Haskell states Scrum is appropriate for projects where there is technical uncertainty, but it is not the only choice, and it is not always the best method for the job. "If we're developing something… which doesn't have a lot of unknowns, you can get away with something like Waterfall, where the tech design is pretty well concrete from the outset." Waterfall is a model that focuses on sequential design, where the process trickles downward through distinct phases, unlike Scrum which is a continuous loop. Both methods have their advantages, but neither is perfect for every project. It is up to the producer to know what the project needs, and to find and adapt a method that suits those needs.

These methods are not tied to any piece of software. That is another decision that a producer must make: what software will work best with the chosen management method. "I start with *Microsoft Project* to build the framework for a schedule for the entire project, but in day-to-day task tracking, I use checklist apps like *Todoist* that allow sharing among team members, assigning and nesting tasks," said Jeff Mills.

Many software packages, like *Todoist*, offer free and premium (paid) versions of their software. A premium version of *Todoist* allows for more tasks to be active and tracked at one time. Other software, like *Fogbugz*, which is utilized in the ATEC Game Production Lab, has prices based on the number of users on a team. It is good to become familiar with the practical applications of many types of task-tracking software, as they are likely to share many similar traits. Many companies also use proprietary software, so having a host of experience with multiple packages can make learning a new software easier each time. An experienced user can make the software a company chooses work to their advantage.

Not every project will use the same management method, which is why it is important to take the time to truly understand the project before settling on the method and software the team will use moving forward. Placing this step below *Socializing* is also deliberate, as it should come after a manager learns more about their team. "When I started my career, process was king - I believed a well designed (sic) process could eliminate/mitigate nearly any issue," said Patrick. "I have since learned that the best processes are the ones built around the team." This means that a producer's process is always changing to suit the dynamics of the team and the project itself. It is important to familiarize oneself with multiple methods, to be aware of which one will best suit the team and the project.

Part Two: Management Methods to Consider

All the management methods listed below fall into the category of Agile software development. Agile is not a single method, but a series of different methods that all promote customer satisfaction through early delivery, welcome changing requirements, and encourage flexibility, even late in the development cycle. The larger a team gets, the more important it becomes to have a solid plan to work with. According to Jeff Mills, "Agile techniques mitigate this loss of productivity to a certain extent by intermingling the disciplines and maintaining some focus across the entire team, albeit in fragments."

Scrum: When I first began learning more about video game production and pipeline methodologies, Scrum was the first method I learned about. It was also the most common. Scrum is a cyclical framework that requires a significant amount of planning upfront to be successful during the main phases of the project. The Scrum process is filled with specific terminology which can confuse those that have never used it before.

The first step in the process is to create the product backlog. This is a prioritized list of everything that needs to be accomplished in a project. After the backlog is created, it passes to the sprint planning phase. The team picks a task from the top of the list and proceeds into a "sprint" – a period to work on the task, with a hard deadline. Scrum focuses heavily on both flexible deadlines and daily communication (called a "daily scrum") between team members (Why Scrum?, 2015).

However, the larger a team becomes, the more difficult it can be to implement the daily meetings outlined by the Scrum method. If everyone must be present for a meeting every single day, this can slow the process down. Some team members may prefer to be working on their tasks instead of discussing their current status. A way to lessen this frustration can be to divide

the daily scrum into smaller groups of people, or only require attendance from those that have something important to share and discuss.

Waterfall: As mentioned above, the Waterfall model operates on a sequence of events. Actions are done in phases, which move from one to the other in a downward fashion. There are seven phases in total, beginning with Definition Study/Analysis, and ending with Management and Maintenance. Each Phase must be completed before moving on to the next phase, ensuring completion and polishing of necessary tasks before moving on to the next step (The Waterfall Model, 2015).

This is also a method that relies heavily on documentation. One advantage of this is that it allows team members that are brought into the project after it begins to have a clear point of reference for their work. It is a good method to use when the project is well planned from its inception, and creates a framework of steady, solid design. However, it is not as flexible when it comes to external factors or changes late in the development cycle, due to its linear nature.

Extreme Programming (XP): Much like Scrum, XP is a methodology that focuses on communication and feedback. It utilizes a series of core practices called the Twelve XpXtudes. These twelve practices are further divided into four categories: fine scale feedback, continuous process, shared understanding, and programmer welfare. A team that functions on the XP method works around a business representative called “the Customer”, who lists the goals and priorities of the project, steering the team towards the goal.

Ron Jeffries, who writes extensively on XP on his website, said, “The best teams have no specialists, only general contributors with special skills” (Jeffries, 2011). XP seems highly focused on making sure everyone feels as though they are on equal footing with everyone else on the team. In my experience, this philosophy of equality is only sustainable in small teams, and

only for a short period. The larger a team becomes, the more necessary it is for other members of the team to step into leadership roles.

Lean Software Development (LSD): The Lean system is an adapted methodology that originated from the Toyota Production System. Lean manufacturing operates on the idea of eliminating the "seven wastes", or "muda" – which is Japanese for futility or wastefulness. LSD operates on a similar set of seven principles that were adapted to the software development space by Mary Poppendieck and her husband, Tom Poppendieck, in their 2003 book *Lean Software Development: An Agile Toolkit.* These seven principles are:

1. Eliminate Waste
2. Build Quality In
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole

When asked which methodology he prefers, Ian Shepherd expounded on his appreciation for Lean, which he also relates to a similar method called Kanban. Lean, he says, allows the producer to "just feed work into the woodchipper (sic) and it gets done. Long term plans are usually worthless and [in my opinion] the best way to get something done is to begin. It's also light on meetings, syncs, and other [bullshit] that vampires away time from my engineers."

Part Three: Additional Advice

How Team Size Changes Every Aspect. The size of a team will vary with every new project. Video game teams can range from two people to over two hundred. "My best experiences have been teams of two or three programmers, one to four scripters and five or six artists," said Jeff Mills. "At that size, I was able to maintain complete awareness of all aspects of the game while still overseeing the final scripting that combined all the assets into a final game." The larger a project becomes, the more project managers are needed to handle the day-to-day processes. Unsurprisingly, the other four managers I spoke to all seemed to share Mills' view of preferring to work with smaller teams.

A larger team can also diminish the feeling of comradery that is important to the social step listed in Part One. However, while making small, tight and polished games is easier with a limited number of people, there are certain benefits to larger teams. "Working on a big team affords you the opportunity to make some VERY cool games," says Bryan Haskell. However, he quickly adds, "When you're working with a sizable development team - often numbering in the hundreds or more, and spread out across the globe - an effective, flexible communication plan becomes absolutely essential to success."

Prioritize Experience over Certifications. Of the five people I spoke to, only one of them began their career in the video game industry as a producer. Two of them began in the video game industry as a customer service representative (CSR), and another began as a quality assurance (QA) tester. These are entry-level positions that an aspiring producer should not shy away from. Brandon Johnson gave his thoughts on the importance of beginning in QA: "Starting in QA provides you with a deep understanding of the production pipeline and really gives you critical insight on process flow." He adds that beginning at the bottom, in the QA department,

will also tell someone a lot about a company. “A company that doesn't view all pieces of the puzzle as valuable is an organization you don't want to be a part of.”

Ari told me that his first professional job was as an associate producer. When I asked him about some of the worse advice he had received, his answer surprised me: “To prioritize getting certifications over working… because employers care more about certifications than some small game you made. Looking back, I wholeheartedly disagree with this advice.” This is not to say that proof of education is not valuable, but that proof should supplement practical experience. This need for experience is one reason many producers begin in entry-level positions.

As mentioned above, certain methods like Scrum offer certifications courses. There are several certifications project managers can pursue, they are not a substitution for practical experience. While both Ari Patrick and Jeff Haskell are Certified Scrum Masters, and Ari has said that he plans to receive his Project Management Professional as well, he insists “certifications mean nothing without practical application…. Certifications should supplement experience, not be a prerequisite for it.”

Keep Everything. Taking notes seems like an incredibly obvious tip to provide new producers, but sometimes the most obvious advice is often the most overlooked. When I first began managing projects in Game Lab, my only notes were in emails that I sent to my teammates. These emails were overly long and difficult for my team to hone in on the feedback specific to them. They also were easily lost among other emails in my inbox, and so it was difficult to reference them later. I also stubbornly refused to make backups, which caused me no shortage of grief. There are many times when I almost lost entire project files due to a mixture of server, computer and human errors.

Ari seems to have learned a similar lesson during his time as a project manager, saying he now takes “notes, pictures, and backups of everything, because the number of times I've needed to reference something from a couple years ago or repurpose an old spreadsheet has been astounding.” This is especially true of code samples and schedule templates. It is much easier to rework a template to suit a new project than to rebuild it from scratch. Research and record keeping can save precious time.

In Conclusion. A producer’s responsibilities are numerous, and difficult to teach. It is something best learned through experience. This includes having both successes and failures, and learning to recognize what works and what does not. There is no set path for becoming a producer or project manager, just as there is no set management system for every project. However, I do hope that the steps, methods and advice I have outlined in this paper will help aspiring producers understand what to expect and provide them with ideas for how to tackle their next project.

Appendix – Interview Transcripts

1. Did you begin your career as a project manager or in another department?

I began my career as a tester at Nintendo of America. Then the progression was up the ladder of QA, then moved over to Design, specifically Level Design. After that I switched to Production where I am currently. – *Ian Shepherd*

Like most people who get into Production in games, I didn't start out there. My first "games" job out of college was as a temp customer support representative at a portal games site - making minimum wage! It definitely wasn't what I was hoping for, but since I knew breaking into games was (and is) pretty hard, I stuck to my guns and did the best work I could. As luck would have it, I discovered and applied for a production internship at THQ, and once that concluded, I got lucky and scored my first Associate Producer role at a studio down in Houston... which helped define my career from there.

So, it was a lot of lucky circumstances that got me to where I am today. I enjoyed every job and every new opportunity between then and now, and I consider myself very fortunate for the chances I've gotten to prove myself. – *Bryan Haskell*

I began my career as an artist, but the studio I joined fresh out of college lacked management and direction. I took on the added responsibilities of a manager to facilitate the delivery of art into the game engine developed by the studio. – *Jeff Mills*

Professionally, yes, my first job was as Associate Producer at id Software. Before that job though, I worked on a ton of mods and small game projects - some in a project management capacity, some in marketing, testing, design, etc. – *Ari Patrick*

I actually began my career in the game industry as a Quality Assurance tester. I couldn't be happier that I decided to start there and highly recommend that for anyone looking to break into this industry, regardless of their "end game" goals. Starting in QA provides you with a deep understanding of the production pipeline and really gives you critical insight on

process flow. It's worth noting that our industry has shown to not always show appreciation or give deserved respect towards the QA team. It's a sad reality, but often you'll see QA segregated from the rest of development and actually discouraged from interacting with teams outside of QA. I strongly advise anyone looking to join a team to do their research. Invest the time to really find out how QA is viewed in any organization you're looking to join. A huge red flag is a company that's not inclusive. A company that doesn't view all pieces of the puzzle as valuable is an organization you don't want to be a part of. – *Brandon Johnson*

2. What is the first project you managed?

My first project in a Production capacity was an online slot machine game for a small studio. We were bought out by Zynga and I was on their Casino division from then on, working on Slots, Bingo, and other titles. – *Ian Shepherd*

My very first project was at THQ, Inc., where I helped manage the recording and cataloguing of motion capture, voice-overs and audio for Smackdown vs. Raw 2008. It was only a small piece of a much larger machine with many moving parts, but I sincerely enjoyed my role. I also got to work with the game's story and some great wrestling personalities (they're actually very tame and normal in person!). Most producers' first runs will be tasked with managing a smaller part of a bigger game - and my advice for small roles is the same as for big roles: learn as much as you can, and take those learned lessons to heart as you go on to bigger, better stuff. – *Bryan Haskell*

Return to Krondor, a PC RPG. I managed the schedule for the art department. For my next project, Nocturne, I took full reigns as producer over all departments. – *Jeff Mills*

I was Associate Producer on a total-conversion mod for the Unreal Engine title "Warm Gun", which was a "Best FPS Mod" finalist in Epic's Make Something Unreal Competition (2009). – *Ari Patrick*

4 Pics 1 Song! It was interesting jumping into a team as their producer / scrum master for a game where development had already begun. Since then I've had the opportunity to be involved with games from inception to release and can say it's more rewarding. You develop an attachment to the IP and a sense of ownership. - *Brandon Johnson*

3. What advice would you give new producers?

While you need a set of production morals, established good practices/habits, and overall experience/street cred in your arsenal: trust your gut and ask the question. More often than not people are just scared to challenge standards or ask a question for fear of looking dumb. What's the worst that is going to happen? You're incorrect? If you are correct, pitfall averted. That's why you're here, to navigate the project around pitfalls. Play dumb, be a devil's advocate, and don't do it because it's "always been done that way". You can be questioning and not come off negatively. – *Ian Shepherd*

Be a sponge. Constantly absorb new information about your industry - from new technical advancements and ways of doing things, to gaming trends and new markets. Actively seek out new information to grow your skills. While nobody is going to expect you to be able to code or render 3D models, keeping abreast of the latest technologies, trends, and development methodologies - and having a working knowledge of them - will make you a much better producer than one who sits on their laurels and grows their abilities and knowledge via osmosis. – *Bryan Haskell*

If you don't have background in the disciplines you're managing, do your best to learn the day-to-day processes required for each team member to accomplish their goals. It was easier for me to manage the art team because I came from that discipline. When programming fell into my purview, I spent a great deal of time interacting with the programming staff to learn the pace and range of their work so I could best anticipate how new tasks would affect the schedule. I picked up enough programming skills from watching over their shoulder that I eventually shifted into programming roles on future projects. – *Jeff Mills*

In the role of production - everything is your responsibility. Never should the words "well, that's not my job..." come out of your mouth, or even run through your mind. Your primary focus is to remove any impediments that the team is facing. Your goal should be keeping the programmers coding, artists drawing, designers designing, etc. Whatever that obstacle may be, you get it out of the way or at least do your best in identifying a clear path towards resolution. - *Brandon Johnson*

4. What is some of the best advice you have received?

If there is some type of issue or problem within the team, take care of it now. This goes for personal issues, business, design, anything. Drag it into a room and put a bullet in it's (sic) head in that same room. – *Ian Shepherd*

"Stick to your guns." When I first started into video games, I was kind of a shy, non-confrontational personality - I equated disagreements with failure, and tried not to make waves with whoever was the most convincing or aggressive person in a room or meeting. But the game industry is one fueled largely by passion, and in a passionate industry, you get passionate people... so impassioned arguments are kind of common! You should never be afraid to raise your voice if you feel strongly about something, or feel a game will be made better with your ideas. But be civil, of course, and be ready to compromise - compromise is essential to successful collaboration, and when everyone feels like they have buy-in, it keeps that critical passion running strong throughout the project. – *Bryan Haskell*

I came up on my own with very little mentorship, so I received very little advice. A mistake I made once was trying to undercut bad news (overtime during crunch) with bitter wit in an email. It didn't translate well, and people felt insulted that I didn't feel more compassion. Don't rely on written communication to convey sensitive topics. – *Jeff Mills*

The best advice was to work on anything and everything I could. Following that advice I worked on a lot of projects, some great, some not so great, but every project provided me

with experience to better myself personally and professionally. Looking back, that's the advice I'd pass on to others - when you're starting out, find projects to work on, even if they're not paid, and gain as much experience as you can. – *Ari Patrick*

"It's ok to say no, just be sure to have a clear explanation as to why." To give some context to this quote, this is in relation to properly scoping. In the earlier days of producing, I like many others took on the delusional approach of "I can do everything and will make it fit, period!" While that may seem like an admirable stance, it's really more of a recipe for disaster. This may come off as contradictory to my response for Number 3 above, but let me explain the key differences. It's easy to agree to things, especially when you're fresh in the position and eager to prove yourself. "You want to see this brand new system implemented into all three of these games, all platforms, by Friday? We'll have it delivered by Thursday!" A noble "go get it" attitude for sure, but if the goal is unrealistic because you're also juggling the introduction of 10 new items, a new sound system and a sea of new art assets for a separate game, you're setting the team up for failure.

One thing you need to understand is PO's (Product Owners) want everything implemented in their game, asap. They'll ask for the sky every dev cycle. It's part of your job to make sure the asks are realistic and most importantly that they fit. Proper scope is huge. Don't make blanket promises without first meeting with the team, retrieving estimates and applying them towards the Sprint plan. You're doing the team a huge disservice by signing up for more than can realistically be delivered. Estimates are approximate at best, but it's better to flesh out a plan as much as possible before claiming the team can deliver on it. - *Brandon Johnson*

5. What is some of the worst advice you have received?

Acceptance that you need to be on call or not separate your personal and work lives because this is video games. You'll have kids and husbands and holidays and bad days. Life comes first. I know you're thirsty but ease in, don't jump in if you don't know how deep the water is. If you're good, you can have it all. – *Ian Shepherd*

"CYA (Cover Your A**)." While it's important to document decisions and get verifiable confirmation on all requests, doing so out of a desire to cover yourself in case things go wrong is a bad reason for doing so. You should, by all means, take rightful steps to protect yourself properly in a difficult situation, particularly to protect your job - but doing this in order to deflect responsibility (particularly things you should be responsible for) is the wrong way to go about things. – *Bryan Haskell*

From the external producer assigned by our publisher: "You should make the game more like Call of Duty." This represents a flaw almost all third-party management has shown during my career. They find a game they like and make that their new gold standard for what the game should be, regardless of its dissimilarity from our game. Coming from outside the team (and often outside the state), these liaisons from the publisher rarely have a strong grasp of the goals we have as a studio working on the project from start to finish, day-by-day. – *Jeff Mills*

The worst advice I received was given to me around the time I finishing school - it was essentially to prioritize getting certifications over working on (small/indie) projects, because employers care more about certifications than some small game you made. Looking back, I wholeheartedly disagree with this advice. Certifications are great - I have my Certified Scrum Master certification, and will be pursuing my Project Management Professional certification - but certifications mean nothing without practical application. Also, in interviews, being able to answer questions with experience is way more valuable. Certifications should supplement experience, not be a prerequisite for it. – *Ari Patrick*

This will sound like such a cliche (sic) response, but I do my best to forget advice that's proven to be ineffective. I highly suggest a similar approach for anyone looking to support fast paced teams. I saved this question for last and have not been able to recall a real example. As a generic response, unless the advice given is clearly immoral or undisputedly bad, you should entertain the idea of at least attempting it. Be open to suggestions, from anyone. You never know when someone will point you in a more efficient direction. - *Brandon Johnson*

6. Do you typically deal with small or large teams? If you have dealt with both, which do you prefer and how are they different?

Both. Be ready for both and be versatile enough for both. They are going to ask about that in a job interview. Be able to say both. Me personally I prefer small and I would suspect most would also just because it's less complicated/more agile. Large teams bring in complications and generally are seen from higher above as being capable of more, so the stakes are usually higher. Also bigger means you're probably at a larger company, so that bring's it's (sic) own concerns/hassles. – *Ian Shepherd*

I've worked with both big and small teams in my career, and while I can say that I prefer being on small teams - where everybody works closely with one another, and there's some good camaraderie - working on a big team affords you the opportunity to make some VERY cool games. But the downside of having a larger team is that communication - and generally keeping everyone looped in on things important to their work - becomes exponentially harder. When you're working with a sizable development team - often numbering in the hundreds or more, and spread out across the globe - an effective, flexible communication plan becomes absolutely essential to success. – *Brian Haskell*

I stepped out of the management roles when the teams became too large to manage single-handedly. My best experiences have been teams of two or three programmers, one to four scripters and five or six artists. At that size, I was able to maintain complete awareness of all aspects of the game while still overseeing the final scripting that combined all the assets into a final game. In Nocturne, I was the sole scripter, and as subsequent games grew in scope to require more scripters, I spent more time overseeing the work of others during this final stage of development than actually producing the work myself. By the time it was necessary to bring in producers whose sole responsibility was tracking schedules and maintaining relationships with publishers, I gladly handed the reigns over to these others so I could focus on the act of making the game itself, which has always been my real passion.

It's a real shame when a team is so large that it requires multiple levels of management to hold separate meetings to discuss the development of the game. The game loses cohesion and

focus, and a smaller percentage of the work day involves actual work. Scrum and Agile techniques mitigate this loss of productivity to a certain extent by intermingling the disciplines and maintaining some focus across the entire team, albeit in fragments. – *Jeff Mills*

At Reel FX, I work with a team of approximately 250 people (~15 in my department). Outside of Reel FX, I typically work with smaller teams (<20 people) on various projects, but mostly indie games. It's tough to choose between the two environments because they are so different (large teams require more well defined processes and structure to keep everyone on the same page, while smaller teams can be more organic and intimate), but if I had to choose, I'd go with small teams, because of their intimacy and sense of comradery. – *Ari Patrick*

Typically I deal with small specialized teams which function on 3 week Sprints (development cycles). I've had much larger teams in the past, but this was in a customer service servile leadership role in an entirely different industry. From a team management perspective, I can't say that I have strong feelings one way or another in terms of preference. If I had to choose I would give a slight edge to smaller teams specifically for how much more you can invest in each team member. With larger teams you're going to be spread thin. Larger teams are typically able to get more done in shorter time frames. Smaller teams are usually easier to manage due to less pieces in the machine to take care of. - *Brandon Johnson*

7. How has your approach evolved over your career?

I am now much more of a peacemaker/arbitrator than I was before. I know now (because I've seen it happen) that many kinds of issues can sink a team and therefore the project. I take less sides and I am more open. Being tolerant of people you don't want to work with, taking all opinions in, avoiding the forming of little groups/cliques etc. This doesn't mean you don't take a stance on issues. You simply become flexible. Be water. – *Ian Shepherd*

When I first started out as an associate producer, I thought I had to keep constant tabs on everything and everyone - checking, double-checking that things were going smoothly and

would come in on time, to the point of annoyance. Thankfully, I got away from that habit, and now instead of shoulder-surfing, and sitting on their shoulder to make things are done, I'm more of a hands-off manager who has learned to trust the people on his team. It's also a product of my taking the time to get familiar with various technologies, best practices, and better management techniques. Now, I make it a point to learn my teammates' strengths and where they can improve, and coach them over time to get really good at raising any concerns they may have. In a perfect world, a guy like me isn't necessary; impediments would be readily solvable to any developer. But since we live in an imperfect world, I can settle for building up very independent developers and teams, which improves flow and efficiency over time. – *Bryan Haskell*

As might now be apparent, I'm not a typical producer / manager. I filled in those roles as needed throughout my career while still maintaining my productive output. In smaller teams, I was able to focus the output from my fellow developers to provide the exact resources I needed to craft the game into its final state, something a non-producing producer might not have had the perspective to accomplish. The studios I've worked for have always been growing, as have their team sizes, so I've moved further from management and more toward smaller, encapsulated disciplines like user interface and game system / tool design. – *Jeff Mills*

When I started my career, process was king - I believed a well designed (sic) process could eliminate/mitigate nearly any issue. While I still believe well defined processes are important, I have since learned that the best processes are ones that are built around the team. People aren't robots, so even with the best processes things can fall apart because people. As a result, I spend a lot of time working to understand people's motivations, what problems they see, and how they would resolve them - all of which are factors I take into account when creating/updating processes and rolling them out. Also, I've learned to take notes, pictures, and backups of everything, because the number of times I've needed to reference something from a couple years ago or repurpose an old spreadsheet has been astounding. – *Ari Patrick*

That's an interesting question, and it's one that I'm not sure I'm the best suited to answer. I would imagine those that I've worked with over the years would have much better insight here, but I'll give my biased opinion. I would say that my approach itself hasn't evolved too much really. Since day one in the industry (in both QA and Production) I've taken the approach of "What else can I do to help?". This approach naturally gravitated me towards the Production department I believe, and it's an approach I've stuck with. The neat aspect about this is it has allowed me to wear several hats within the development process. Now this could be partially due to the smaller size of our studio, but I like to believe that my approach has opened up otherwise nonexistent opportunities for me throughout my time in the industry. - *Brandon Johnson*

8. Do you have a preferred management method such as scrum or another type?

Generally, there is Waterfall, Sprint based Agile, and Lean (kanban) methodology. I prefer Lean. I am a fast and loose style of production and Kanban works with that. Take it off the line, get it done to completion, pick up another one. You just feed work into the woodchipper (sic) and it gets done. Long term plans are usually worthless and [in my opinion] the best way to get something done is to begin. It's also light on meetings, syncs, and other [bullshit] that vampires away time from my engineers. Also I am a high C/S DiSC[1] type (the majority of Production types are high D) so I tend to like to build a machine and then let it run, tune it, let it run, etc. I will say that I do miss my sprints sometimes, but I am not a fan of arbitrary delivery dates or pressure for no reason. – *Ian Shepherd*

I've found, over the course of working on many games, that the type of project management applied to a project should fit the requirements of the project itself. For instance, if there's a lot of technical uncertainty on a game, and we're going to have to do a lot of investigation, Scrum and XP are useful both for working from a fungible backlog and

[1] The DiSC method refers to a series of behavior profiles. The letters stand for Dominance, Influence, Steadiness and Conscientiousness. Ian’s statement indicates that he is falls into the Steadiness and Consientiousness half of the DiSC chart.

keeping tabs on everyone's progress and impediments. But if we're developing something like a Words with Friends app or a Bluetooth garage door controller, which doesn't have a lot of unknowns, you can get away with something like Waterfall, where the tech design is pretty well concrete from the outset. – *Bryan Haskell*

I start with Microsoft Project to build the framework for a schedule for the entire project, but in day-to-day task tracking, I use checklist apps like Todoist that allow sharing among team members, assigning and nesting tasks. Back when I started, there weren't as many task listing apps, so I usually managed such lists manually, something that was only feasible in small teams. – *Jeff Mills*

9. Have you ever had to deal with unprofessional team members? If so, how did you approach the situation?

Of course, as will anyone in any industry for that matter. The best way to approach a situation like this is to communicate with them in a 1 on 1 environment. You don't want to put someone on blast in front of their peers. That's just asking for them to immediately go into defensive mode as they'll feel attacked. It's important that you first explain to them the behavior you're observing. This allows them to see your perspective, it shows them that you care enough to bring it to their attention and it also allows you to ask them if this has been their intention. What you might view as obvious unprofessional behavior, someone else may perceive as standard. Perceptions will vary.

From here you can discuss ways to work towards improving on whatever areas are deemed unprofessional. In most cases, you'll see huge success from this approach. Communication is so crucial, in any environment really. In the rare event where you've discussed someones behavior (from a completely constructive standpoint, that's important), they disregard your advice and continue to perform poorly, you owe it to them and the rest of the team to give a heads up to their direct supervisor. Remember, your'e in a servile leadership role as a teams Producer. Your'e not the guy/girl to reprimand anyone. You're a proponent towards overall team cohesiveness and ensuring everyone has everything they need to get the job done.

One exception where I would advise against this approach is if the person is exuding any sort of violent behavior. In that scenario you need to report it to your direct supervisor or someone in upper management. It's best to not get involved at that point, for both parties. - *Brandon Johnson*

Bibliography

Jeffries, R. (2011, March 16). *What is Extreme Programming?* Retrieved from RonJeffries.com: http://ronjeffries.com/xprog/what-is-extreme-programming/

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit.* Addison-Wesley Professional.

The Waterfall Model. (2015). Retrieved from Waterfall Model: Advantages, Examples, Phases and More About Software Development: http://www.waterfall-model.com/

Why Scrum? (2015). Retrieved from Scrum Alliance: https://www.scrumalliance.org/why-scrum