

So You Want to Make a Game

by Philip Chu

Table of contents

1 Publication Information.....	2
2 Getting Started.....	2
3 Think Inside the Box.....	5
4 Less is More.....	7
5 First Things First.....	9
6 Going International.....	10
7 It's Software Development.....	12
8 The Peanut Gallery.....	13
9 Don't Be Evil.....	15

1. Publication Information

Copyright ©2004-2008 by Philip Chu All rights reserved.



"Once you can remove no more, you are done." - Japanese saying

2. Getting Started

My first piece of advice to anyone contemplating a game development project - don't. (Well, at least not until you know what you're getting into.) But if you go ahead, anyway, here are some tips beginning with staffing:

2.1. Dispassionate Gamers

Game job postings typically list "a passion for gaming" as a prerequisite (along with its

partner, "willing to put in extra hours"). Other industries don't do this - you don't see ads requiring "a passion for factory automation software" or "a passion for working on banking systems". Anyway, a devotion to games doesn't necessarily translate to productivity and work ethic.

- Most of the game developers I've worked with were avid gamers but many, including one of the best gamers I've ever met, didn't apply the same concentration and motivation to their work. I've seen plenty of work with obvious defects, even an FMV with a two-second blank gap, either due to inattention or laziness (one artist even told me his video package could reexport his FMV in AVI format, until I stated that was the only format I could process for our target console - then he promptly reexported it).

I'll take professionalism over passion any day. If you need to hire janitorial staff, you're not going to look for some crazy nut who has a passion for toilets - you want responsible people who take pride in doing their job.

- Most of the best game artists I've worked with were only moderately interested in gaming - they were artists first and their results far surpassed those of mediocre artists with high gaming skills. They were conscientious enough to play-test their work during development but once the games were released, typically never touched them again.

Now, game design is one area where presumably an absorption in games would be really useful. But even there, you'll find the best game designers are who have the intellectual breadth and analytical ability to figure out what makes a game fun and to move beyond knockoffs of existing games.

- One designer who was an extremely good skateboarder and skateboard game player decided to differentiate his game from all the Tony Hawk clones by removing the fun parts, resulting in an obvious knockoff that was unplayable.

A common entry route to game development is via QA groups, and I think that is a good one - testing is a good introduction to the game development process without getting in the critical path, and anyone who can't develop the patience and analytical abilities for that role is probably going to do a worse job in a production position.

To put a final point on this, if a passion for gaming is so important for game development, why are so many non-gamers in high-level decision-making positions on game projects? Many, if not most, of the high-level producers I've met are mediocre gamers at best, and it gets worse as you go up the corporate ladder.

- I know of one game prototype that had all kinds of pizzazz thrown in at the producer's request, and then the whole thing had to be scaled down at the last minute so the demo wouldn't confuse the executives who had final approval. (Of course, the producers should

have known this)

2.2. Programmers are from Mars, Artists are from Venus

The production pipeline is always the bottleneck, so you need people who not only work well, but work well in the context of a production.

Even some experienced and talented game artists do not work well in game development teams. Just throwing assets over the wall and assuming the programmers will fix it wastes time and engenders hostility from the programming team. The more difficult artists I've met were stubborn to the point of belligerence and considered the rest of the game team as staff supporting (or limiting) their artistic accomplishments.

- Some phrases that have lost their novelty - "I didn't change anything", "It's broken - fix it!", "That's how it works in 3D Studio Max".

And while programmers often grouse with justification that artists aren't feeding suitable data into the production pipeline, it is incumbent upon them to provide the proper guidance, in the form of clear documentation and explanation.

- Programmers often don't like to commit to numbers early, but artists and game designers need reliable asset budgets to do their job within the proper constraints, just as programmers need to know the hardware and delivery constraints. If artists and designers actually ask for budgets and guidance on how to optimize assets, then by all means accomodate them.

2.3. The Producers

Producers can be useful, if they're not puffed up by the term "producer". They're the only the part of the production pipeline that doesn't actually produce anything (except those producers who produce funding - they get to be called "executive producers"). A better term would be "facilitator".

In the worst case, a producer will just cause more work for everyone else. The last thing a game project needs is a high-maintenance producer.

- I was once plagued by a throng of producers who would hold daily status meetings, carry around lists of tasks and ask everyone to update the number of hours remaining per task on the lists (none of which matched), and stroll around asking, "So...what are you working on?" After may of the producers were laid off due to tightened funding, the project ran more smoothly.

But a knowledgeable producer who is focused on keeping things running smoothly is indispensable.

- I was vastly relieved that a survivor of the aforementioned layoff was an assistant producer who did a fantastic job of keeping on top of things. Every time I needed clarification on a game feature, he had the answer, knew where to get it, or could make a reasonable call on the spot. In any case, I had an answer within minutes. And he was the guy who ordered all the late-night meals.

At least producers in development shops have an idea what it actually takes to make a game. Producers who've only worked for publishers are less knowledgeable and more self-important.

- One stupid publisher trick: planting a producer at a developer's office during crunch time to make sure they're all working late into the night. When I showed one of these producers, who was camped out in my office during one of these crunches, the set of game design, production and business books on my shelf, he commented his boss had recommended he read some of them. And then he kicked me out of my office so he could have it to himself.

3. Think Inside the Box

"Think outside the box" is an oft-misused mantra. Questioning your underlying assumptions brings about innovation. Just making stuff up is a waste of time.

- In high school, I joined other students who wanted to pad their college applications by competing in a statewide brainstorming competition, where we attempted to outdo each other in constructing the most fantastic scenarios possible. Odds favored those who most lost touch with reality.

What separates game artists and programmers from their brethren in other fields is the ability to create for resource-constrained platforms.

3.1. Screen Size

Take into account the screen size. For PC games, you have to decide how many standard PC monitor resolutions, refresh rates, color depths, and full-screen vs. windowed modes you're going to support, and be sure to test the game with those settings.

Even with console games, you may have more than one setting to worry about. If your game will be in both the US and Europe, then you need to handle both NTSC and PAL, which have different screen resolutions, with corresponding aspect ratios and memory requirements, and different refresh rates, which may affect any game behavior dependent on per-frame

computation. And there are other modes like EURGB60, M-PAL, 480p (progressive scan) and multiple levels of HDTV.

3.2. Preprocess Everything

Data created by game artists and designers eventually gets converted into formats usable on the target platforms. PC game engines often defer this conversion until runtime for convenience, but for consoles, where memory is comparatively limited, loading from disc is slower, and the main CPU may be relatively underpowered, it is important to have as much data preparation and optimization done offline as possible. Even for PC games, while developers may be lulled by the latest and greatest in PC hardware, there is still a customer base with configurations a few years old, and if they were budget PC's then, imagine how limited they are now.

3.3. Get Lots of Hardware

Game development schedules are always tight, and even expensive hardware is cheap compared to personnel cost and the cost of missing a milestone and having your project cancelled, or missing the holiday retail season and losing out on those sales. Cutting-edge hardware, especially console development kits, is notoriously fragile, so you want to have extra units on hand if and when the hardware fails.

- Almost every time I've worked on a console game, the game development hardware malfunctioned at some point and had to be sent back to the console maker for repair. In each case, the turnaround time was no more than two weeks, but two weeks on a crunch time project with monthly milestone deliverables is crucial. Fortunately, the developers always had on hand an extra kit that could be repurposed from less critical tasks.

Another reason to have redundant development hardware is to identify bugs that are due to glitchy hardware versus those present in your game.

- Near the end of one console project, I ran into crashes of our game that occurred after several hours of the game running idle. Since we had multiple test kits, I could run several soak tests in parallel and isolate the crashes to one unit, and thereby conclude that it was a test hardware problem (and it was verified later that some units were known to have overheating issues)

Console games have the advantage that you only need to verify proper operation on a very limited set of configurations. For PC games you should have a variety of different hardware, operating systems, and various configurations for testing your game. This is true for cell phone games, too - phone models vary in screen size and color resolution, refresh rate, memory, etc.

- I nearly missed one Windows compatibility issue with a PC game after Microsoft introduced a service pack that removed support for a video codec that we used for an FMV (in fact, the codec was used by one of our middleware vendors). None of the development or test machines used by the developer had this service pack, but fortunately the publisher's salespeople noticed the problem when they installed the game on their demo laptops. (when you're relying on your publisher's sales team for QA, you're just asking for it!)

4. Less is More

More than in most other software fields, game development is about efficient deployment of assets.

4.1. Think Small

Game designers and artists often believe it's easier to create more content than you need and pare it down as needed than to start small and add. This may be true in the "micro" sense but poses huge risks in the "macro" sense, particularly for console game development. One of the common "crunch time" factors in console games is the late attempt to get the game running in console memory. Usually this problem is hidden until an inconvenient time by the fact that content developers usually develop on PC's (and sometimes XBox's) with high-performance graphics hardware and several times more memory than a console. And programmers, too, will develop on console devkits that feature more memory than the retail consoles.

- Many designers and artists complain about the constraints of video game development, but I was gratified to encounter one exception. A junior game designer who was ordered to pare down his level commented to me that his level was actually improved by the streamlining - it forced him to make sure everything that he did retain was effectively used.

Constraints are a good thing - they keep us from wandering all over the place trying out everything possible and instead focus on validating the cliché - quality over quantity.

4.2. Every Polygon Has a Price

Each polygon, texture, and frame of animation should be justifiable. Serving as "eye candy" isn't enough reason to put something in.

- On one front end I spent quite a bit of time debugging some animated hieroglyphic textures, only to find out later that those icons had no connection to the game at all - they were just there for artistic, but meaningless, effect!

HUD's in particular tend to echo the worst of web design, ranging from the early blinking text to the modern Flash-filled pages. Games should be no exception to the principle that the interface should not get in the way - the best interfaces are interfaces that you don't even notice.

4.3. Sound Advice

What goes for graphics, goes for sound.

- Laboring under the misconception that more options are better, one game developer president stated that our extreme sports game should allow ambient sounds and the soundtrack to be played concurrently. It turned out that loud rock songs easily obscured twittering birds in the background and hardly warranted the extra development complexity (the console had hardware support for just one stream). Another game had a list of sound effects for every element on the HUD, potentially resulting in a Las Vegas slot machine effect - the sound designer threw out most of them.

More is not necessarily better, and often it's worse. Imagine all the sounds that could go off at once doing so, and scale it back if the result is cacaphony.

4.4. Dialogue

Same goes for dialog - every line should have a purpose. We don't want voice-over (or text-over) dialog distracting from the interactive flow of the game, and each piece of dialog requires space, scripting, services of a voice actor (if voice-overs are used), rework if the dialog has to be rewritten or different voice actor is selected, and translations and re-recordings if the game is localized for different regions.

- On one game project where I script-doctored the dialog, the publisher looked over the results briefly and asked for some crowd NPC dialog in one of the "cinematic" fight scenes. I added it just to keep them happy, but sure enough, once the level designers worked in the voice-overs, it was just a big muddle.

One practice I have used in writing game dialog is to include notes at the beginning of each section explaining what the dialogue is supposed to accomplish. Some of these objectives are the same as in any story, e.g. increasing empathy for the player character, establishing the badness of the bad guy. But dialogue can also highlight aspects of your game - if you hear the NPC's talk about what they see or hear and how they coordinate amongs themselves, then you know the capabilities of the game AI.

Another thing to keep in mind is that scriptwriting for games is really more like scriptwriting

for animated features rather than film. As with the former, you can't rely on the range and subtlety of visual expression conveyed by human actors, so you must make sure it's clear in the words.

5. First Things First

As a general rule, anything that can be completed early in the project should be completed early. Get it out of the way, let it get tested thoroughly, and leave crunch time for the hard stuff, of which there will be plenty.

5.1. Front End

The front end is typically an afterthought, fleshed out in the final months of a the game development, but it really should be one of the first things implemented. Even if the front-end requires some assets that will not be finalized for a while, placeholders can be substituted.

1. Completing a front end early provides a real functioning component of the game that can be demoed immediately and shows you've got more than just some storyboards and mockups with Flash.
2. Designing the front end early forces the game designers to complete the game design to that extent, so critical decisions like game modes (single-player, multiplayer, story, arcade, etc.), scores, game-save interface, are resolved early.
3. Implementing the front end early gives the developers (and anyone else who sees the game, including the publisher) a specific idea of what the game is about. A common complaint among developers on game projects is that they don't understand what the game is supposed to be about - starting up the game in the same way as the eventual customers can alleviate that problem.
4. Implementing the front end early will expose design flaws, logical inconsistencies, and potential incompatibilities with the console makers' requirements, e.g. memory card usage. Developers usually implement shortcuts that start up whatever level or feature they're working on or testing, but then they get in the habit of relying on these shortcuts and the front end is not well-tested.

5.2. Media

The final distribution on media may seem like the last thing to take care of, but again, it's something that can and should be done early. Preparing the game assets for the target media is typically an elaborate process, and it's best to get a handle on that before crunch time. And ideally, you want to test the game running on the media as soon as possible, so you'll know if

load times are acceptable, sound and FMV streaming works, and even if the game fits on disc. Many games now depend on "world" streaming, so it is crucial to verify that disc performance can keep up. Console development systems often provide disc emulators, but the performance characteristics often do not properly match those of the real hardware. I've seen games work perfectly on the emulator and then, a rude disappointment, fail when run from disc, sometimes just before a required milestone delivery.

5.3. First Playable, Last Shippable

Programmers complain the design is late and the designers complain they can't finish the design until the programmers have the game up and running so they can tweak it. They're both right. So while it's probably not a bad idea to do as much in preproduction as you can, you can hedge your bets by working toward a "first playable", consisting of at least one, but no more than three levels of the game.

1. This allows you to start off your project with a smaller team while you implement your core technologies and work out the basics of your game, and then you can staff up later to crank out the remaining levels.
2. The smaller target allows you to get to the playable point faster than if you tried to develop the full game at once. This allows you to reach a point much earlier at which you can evaluate the gameplay, asset budgets, target performance, and if the result is satisfactory, then you have a demo for the trade shows and game magazines.
3. If it turns out that the game is fundamentally flawed (or just not what the publisher wants), which is not uncommon, then you can change direction or even start over much more easily than if you had invested a full team and spent much longer in getting the game to a playable point.

Note the first playable is not the same as a prototype, which is basically a minimal demo that you shop around to get the deal. The first playable is really playable, which means it has all the user interface elements, game saves, functionality and polish that you would see in the final game.

6. Going International

Localization is another typical afterthought. But as with everything else, get things ready early, so you won't have to deal with it in the crunch.

6.1. Who's Your PAL?

PAL resolution is slightly different from NTSC - in particular, the aspect ratio is slightly different, so you'll want to verify that 2D elements in particular, such as the front end, text,

HUD and movies, still look OK. The higher vertical resolution of PAL also can mean greater main or video memory usage.

- I was astonished to find in one huge game project that none of the programmers had access to a PAL-capable PS2 devkit. They resorted to kludging in the code and then sending off a special build to QA for them to see if it really ran.

The PAL frame rate is also different, so anything in your game dependent that assumes a certain frame period, e.g. movies or code executed per frame that doesn't take in account real time elapsed, will behave differently.

6.2. Watch Your Language

I've seen more than one project where it was assumed that the publisher would deliver localized assets (text, audio) just once, and that the assets would be correct and final. That's a pretty unrealistic assumption - publishers will express the importance of getting a game finished on time in no uncertain terms, but when it comes to deliverables from their in-house departments, don't expect them to respond with the same urgency.

- At one developer working on a console title for the US and Europe, the president of the company assured me that the localized text from the publisher would be correct and final, so not to worry that the deliverable would happen just before the ship date. As it turned out, the holiday retail season came and went as we went through several iterations of the localized text, with 3-4 weeks between the updates. One of the near-final deliveries was incomplete because our producer contact took off for Christmas vacation without bothering to give us a complete set of corrected translations (and he didn't mark which ones were changed - that was considerate!)

When you do actually receive translations, chances are they're not going to be suitable.

- The translation may be inappropriate for the desired context. For example, "button" could be translated differently depending on whether it belongs to a controller or piece of clothing. Or the translation may be just plain wrong. I've received translations that left me wondering if the publisher had just hired some beginning language students from the local college. In many cases, I've had to rely on European coworkers and the internet to find the right translations.

Console makers typically require certain phrasing as part of their submission requirements. This can cause your submission to be returned several times until you get it right.

- On one localization project, we went through several iterations of translations for the various mandatory disc and memory card error messages, until the console maker published a set of recommended translations for all of them. Upon which I gladly tossed

out all of our publisher-provided translations.

Finally, translations may not fit in your existing on-screen buttons and other user interface elements, so you'll either have to find shorter translations or rearrange your screen. You'll probably need to load multiple new fonts, increasing your asset budget, and if you have a type-in screen, arrange to display all of those characters. All the more reason to set up your game for localization early and avoid rearranging a bunch of your screens later.

7. It's Software Development

Although game projects bear an increasing resemblance to Hollywood projects, game development is still fundamentally software development, yet despite being possibly the most challenging form of software development, the game industry lags behind other industries in software engineering and management practices.

7.1. Office Space

It is conventional wisdom by now, supported by studies, that programmers are more productive (and less irritable, I might add) in their own offices. The elite technology companies like DEC used to make a point of getting programmers their own offices (if you were senior enough, you got a window).

- One of my favorite employers, BBN, was one of these companies steeped in technology culture and provided individual offices for each of their engineers, until the fiscal pressures of the early eighties prompted them to move to cubicles and euphemisms ("come see our new open office environment!").

The game industry, on the other hand, is the only software business I know of where people, including engineers, talk about cubicles and open "bullpens" as productivity-enhancing. This is a case where the appearance of activity and interaction doesn't mean more productivity - more likely there's an inverse relationship.

- In one particularly abysmal game company, I was crammed into an office with two other programmers, with the finicky one sitting three feet from me complaining that my typing and mouse-clicking was too loud. (Apparently, he'd never heard of headphones) As a bonus, the men's toilet on that floor was not designed to handle twenty male game developers and clogged up constantly (now I know why the British call it a "bog") - combined with the lack of air conditioning, it made for swampy weekends.

Not the best environment for the regular workday, much less long work hours, which brings us to....

7.2. Avoid the Crunch

Crunch time happens everywhere, but outside the game industry it's recognized as a failure of management. In the game industry, however, even while paying lip service to the notion that crunch time is a bad thing, publishers and developers trot out the same trite spiel - it's an unavoidable consequence of industry pressures, management has to tell developers to stop working and go home because they are motivated to work so much!

That's quite a conceit. It's one thing for someone to get on a roll and work through the night to get something cool done, it's quite another for management to mandate 12-hour work days and weekend shifts for months at a time. It's ultimately a self-defeating proposition.

1. Everyone has a natural pace and level of productivity. The reliable key people on your team will put in the extra hours, anyway. Everyone else will just hang around the office longer, web surf more, and just get on the nerves of those who are really working. In either case, they will resent the disruption of their personal lives due to mismanagement of the project schedule. And they won't be completely wrong.
2. Contrary to popular belief, crunch time does not engender an increased level of production - you get more code and assets, but you get more mistakes due to rush jobs, weariness, and decreasing motivation. The risk-reward balance of crunch time is becoming more dangerous as game projects get bigger and more money is at stake, especially for console games that are unpatchable and the only recourse for flaws introduced at the last minute is to recall the product from the retail shelves.

8. The Peanut Gallery

The cool thing about games is that everyone's got an opinion. The bad thing about games is that everyone's got an opinion. Playtesting provides critical validation of your game, but sorting the wheat from the chaff is required.

8.1. Focus Groups

The problem with focus groups is a lack of. General comments often support the industry's reputation for catering to adolescents - "Bigger boobs! More visual effects!" As with formulaic big-budget Hollywood movies, these elements may be reliable ingredients for drawing mainstream interest, but do you really need a survey to give you this information? Used in this manner, focus groups are just a pseudo-scientific way of reinforcing current preconceptions.

- A game company president was peeved that most of the team showed little enthusiasm for making our game more risque, so the only comments she emphasized from a focus

group test were the juvenile ones asking for more skin (the sole female tester demurred, but her opinions were not considered useful). However, the game was based on a family-friendly license and was supposed to achieve an "E" rating, so we spent a lot of time at the end of the project toning things back down.

But focus group results can be useful, if you really focus on "finding the fun".

- I was particularly impressed with one console maker's rigorous in-house play-testing. Their testers would report initial impressions of the game, their impressions after one hour, then three hours, then eight hours. At each step, they gauged their level of empathy with the player character, their satisfaction with the visuals and audio, their frustration level if any with the difficulty level, and their motivation to continue.

8.2. Publishers, Who Needs Them

Unfortunately, you do. At least if you're working on a console game. As in Hollywood, publishers think that because they make their money off games, they know how to make games. And since they make money off one in ten games, they know what consumers want. And then there's reality.

Do not let a publisher design your game. If they were qualified to do that, they could develop the game themselves. Publishers should stick to their knitting - providing licensing, tools, testing, marketing, submission requirements, translations for localization, and most importantly, payment. Remarkably, I've seen publishers fail miserably in all these areas.

- On one project I worked on, the publisher was late with the console development kits, borrowed more equipment from us for their other projects, paid late on almost every milestone, lost the license and then complained the game was too boring to sell (and it was fun with the license?), got briefly excited enough about the marketing campaign to order a case of custom-labelled Jones soda, and then lost interest and drank all the bottles.

A commonly-stated excuse is that publishers are "just trying to make money". That in itself is enough reason for developers to make sure they protect their own interests, whether it's making a great game or just surviving. But as with any other business, such a statement is overly simplistic - any publisher you deal with is a mix of people who are out for personal glory, personal wealth, job security, and some may be actually trying to make money for the company and some may actually be trying to make a great game.

- I heard one publisher complain that they helped unknown developers get their start only to be left behind when those developers ended up on well-known franchises. Yet this publisher had a history of imposing tight schedules and low budgets on these hungry developers, and they would sell the resulting titles to other publishers at the first

opportunity to make a quick profit.

The most successful game companies control their own fate as much as possible.

- The most successful game I've worked on relied almost entirely on in-house resources. An impressive marketing effort, including a fan site, magazine interviews, developer blogs, and even a comic book. The in-house QA group was knowledgeable about console submission requirements.

9. Don't Be Evil

Publishers aren't just often bad at their business - they are notoriously bad about business.

- In just the past few years I've heard of publishers going on group outings to strip clubs, granting projects to developers in return for trips to strip clubs (sense a theme here?), producers propositioning female developers, and even rumor of a publisher bribed with a new Porsche to get a project. Aside from the sleazy, there's the adversarial. I've heard that a publisher lawyer at a game development conference stated the best negotiated deal for a publisher is one that bankrupts the developer. With this attitude, it's no wonder that a favorite publisher trick to enforce crunch time is to send a producer to camp out at a developer's office and make sure they're working into the evenings), And not only have I seen publishers make lame excuses (or no excuse) for late payment and non-payment to developers, I've encountered those tactics in my own contracts with them - contracts rewritten without informing me, micromanagement in order to penny-pinch ("spend ten minutes on this, fifteen minutes on that..."), and weasely attempts to get free work ("Oh, we thought you could just take a look at it...")

But as easy as it is to blame industry woes on publishers, developers who engage in the same practices have my utter lack of sympathy.

- The one time I heard management say they were opposed to mandatory crunch time work schedules, they shortly announced six months of required weekend work for the entire staff. And I've seen the same gamut of bad-client practices from developers, ranging from renegotiation of ongoing contracts (one client had a practice of this with her contractors, even telling me at the end of a contract "when I have time, I'll let you know what I think is reasonable"), to blatant non-payment (on a project that was dragging on, the client said they had no expectation of paying me for the extra time, yet sued their publisher for the same thing), to sneaking in as much as possible into a contract (I started out at one developer with four weekly paid milestones which turned into two payments for four milestones which turned into one payment for four completely unspecified milestones and an unpaid "transitional period" of work.)

At times it seems developers and publishers are engaged in an unholy alliance.

- I've heard of developers expressing their "appreciation" to publishers with tokens ranging from gift certificates, birthday gifts, hotel accommodations, strip club outings, and even rumors of a car given to a producer in order to seal a development deal. I saw one breach-of-contract lawsuit filed by a developer against a publisher settled by not only payment of cash to the developer, but also a hefty pile of stock and a "consulting" agreement granted to the developer's president. It's not uncommon to see the owners of a game developer shutter the company, leaving the employees unpaid, only to start up a new venture. Who suffers? The rank and file.

I'm the last one to defend publishers, but sometimes you guys deserve each other. Don't be part of the problem.