

Work Hard, Play Hard: Applying Game Design to Project Management

by Philip Chu

Table of contents

1 Publication Information.....	2
2 Game Design Your Project.....	3
3 Why Do Players Play?.....	3
4 What Do Players Expect?.....	6
5 A Never-Ending List.....	12

1. Publication Information

Copyright ©2003-2007 by Philip Chu All rights reserved.



*Software engineering is a messy sport in general, but game development has the rep for being the worst of the worst, probably due in combination to the insularity of the industry, youth of the workforce, and established crunch-time practices. After reading Richard Rouse's book *Game Design: Theory and Practice*, I applied the principles in that book to game production management as a thought exercise. The result may appeal to game developers, but I think it's applicable to software development projects in general.*

2. Game Design Your Project

Game development is the software engineering equivalent of an extreme sport - the combination of fast-moving technology targets, the complexity of integrating design, art, audio and software into a single production pipeline, the demands of publishers (sometimes reasonable, sometimes not) and the typical "death march" schedules should dissuade anyone who wants a normal life from joining a game project. And it doesn't pay startlingly well, at least compared to other forms of software and media production.

So why would anyone want to be a game developer? And how do you motivate and retain game developers once you've got them on your project?

Well, game companies typically list "a passion for games" as a prerequisite in their employment ads. Perhaps this is the key - if game developers are also gamers, we may be able to apply game design principles to attracting and motivating game developers. After all, with game projects, as with games, we want to attract participants, inspire them to work through and complete the endeavor, then come back for another round.

In this article, we see how far we can stretch this analogy by applying some game design principles to game project management. The principles are selected from the introductory chapter of Richard Rouse's book *Game Design: Theory and Practice* and are listed in their original order and with their original titles and categories, but the principles themselves are paraphrased for brevity.

3. Why Do Players Play?

Our initial query was: What draws game developers to this industry?

This is an appropriate first question to ask when designing a game. What brings game players to video games over other forms of entertainment? In other words, let's figure out what are the unique offerings of video games so we can capitalize on them.

Likewise, game developers could easily be working in other fields, other types of software development or media production. So once we realize the unique attractions of the game business, we can leverage them.

3.1. Players Want a Challenge

Game players enjoy facing challenges in games, especially single-player games. These challenges entertain, provide intellectual stimulation, and teach lessons that can be applied to other problem-solving situations.

Developers also seek challenges. Not challenges in the form of how-may-bugs-can-I-fix or how-many-hours-can-I-go-without-sleep, but, as in games, challenges that are intrinsically enjoyable and result in knowledge that can be applied in the future. Programmers want to implement cutting-edge algorithms that exploit the latest hardware yet operates within those hardware constraints. Artists want to create the best-looking models and animation within their given asset budgets. Game designers want to create entertaining and addictive gameplay experience through story, AI and level design.

Developers also have a pragmatic motive for intellectually and creatively extending themselves. Career longevity depends on staying current with the latest tools and technologies (ask all the developers who were slow to make the transition from 2D to 3D). Game companies are constantly looking for experience with the newest generation of consoles, and developers are always eager to work on the latest hardware.

3.2. Players Want to Socialize

Long before their appearance on computer hardware, games have mostly been social activities. Social computer games range from Quake-style fragfests with players typing insults to each other in the heat of battle, to massively multiplayer persistent worlds featuring chat-style interaction.

In my experience, even single-player games take place in a social environment, either in taking turns, watching others play, and kibitzing. Certainly in the heyday of arcade games, you could see crowds gather around superb players. And a visit to any of the countless web-based game forums will show that gamers argue vehemently about the merits of their favorite games as much as any sports fanatic.

Likewise, many employees want a socially engaging workplace, where they can learn from their coworkers, banter and sometimes compete with them. Some companies have company picnics and Friday afternoon beer socials. Game developers have after-hours fragfests (which blurs the distinction between game development and game play even further).

3.3. Players Want a Dynamic Solitaire Experience

On the other hand, sometimes players like to play games by themselves, either because no one else is around, or the player wants a dynamic interactive experience without the hassles of dealing with (real) people.

Even in a social workplace, most employees need some time to themselves to get their work done. This is particularly true of programmers, who need to do a lot of thinking, (paired-programming aside) but everyone needs to don the headphones during some stretch of the day to get work done.

In a larger sense, the workplace also offers the developers refuge from the people of their "regular" lives - spouses, children, friends, pets. (I had one manager who, at 5pm, would start moving from office to office, ostensibly to talk with people, while his wife called random numbers to track him down.)

Then there are those who don't want their work hours to encroach on their non-work life. These employees prefer minimal distraction and regular work hours so they can get home in time for dinner with the family, attend evening classes, play in a rock band, or engage in whatever other activities that constitute their personal lives.

3.4. Players Want Bragging Rights

Playing to win means getting respect from fellow players and bragging rights over them. This bragadaccio is evident in multiplayer fragfests, organized game competitions, and the high-score tables on game screens dating back to the early arcade games. This earned respect also includes self-respect - completing games and winning, even against the computer, can inject a shot of self-esteem into someone who might be athletically, socially or academically less accomplished.

Developers would like to brag that they worked on the latest smash-hit AAA title, worked for game companies that are household names, worked with game industry luminaries such as John Carmack or Will Wright. Even just mentioning you're in the game business is a good conversation starter just about anywhere. I've worked in half a dozen different industries, but only since I started developing video games (even lesser-known titles), have I gotten a reliably enthusiastic response from laymen, including the guy at Starbucks. When I was working on manufacturing software, on the other hand, talking about my work inevitably resulted in a glassy-eyed stare, even from fellow programmers.

Part of the thrill in working on a game is seeing your name in the credits. I don't really like the idea of displaying credits in a game - it certainly isn't a common practice in other types of software, and it can easily be seen as an unfair and arbitrary process (in the film industry, it has political to the point where there are union rules about film credits). But it does seem to provide motivation and is important for career advancement in the game industry (many game job ads list credits on shipping titles as a requirement).

3.5. Players Want an Emotional Experience

Compared to other media such as film and literature, games usually exhibit limited emotional range, but at the very least all games attempt to produce excitement and a sense of accomplishment in game players.

Game developers want to experience those emotions in game development, too. Ideally,

developers should feel an increasing level of adrenalin rush and anticipation as the project nears completion, and the gold submission should be accompanied by a profound sense of achievement. Pace is important - the end of the project should be the most gratifying, not anticlimactic.

Good emotions: interest, confidence, anticipation, excitement, satisfaction. Bad emotions: frustration, nervousness, disappointment, anger, ennui.

3.6. Players Want to Fantasize

Games take players away from their normal lives by immersing them in fictional environments and circumstances. As in the movies, the game world is idealized - with the exception of *The Sims*, you're not bothered by household chores, taxes, and addressing bodily functions.

In the workplace, of course, you still have to deal with bodily functions and taxes, but a job still offers a world separate from the home life and roles and missions that are different and possibly more exciting than available in the personal life. To different degrees, companies are like role-playing games in assigning employees to different official categories - game designer, producer, programmer, artist. Startup companies with more loosely defined responsibilities allow employees to assume a "be all you can be" attitude.

4. What Do Players Expect?

Now that we've established what motivates people to become game developers, it's time to address the second part of our query. Once we've got a developer on our project, how do we keep him present, happy and motivated?

In game design, we have the same issue - once a player has decided to play a game, he has expectations that must be fulfilled for him to enjoy and complete the game, so it is important to identify those expectations, conscious or not.

4.1. Players Expect a Consistent World

Players expect their actions in a game to have predictable results. Not right in the beginning - the player has to experiment a bit, in the way that infants do learn about their environment, to understand how things work in the game world. Seemingly arbitrary cause-and-effect will discourage the player and give the impression that the game is rigged.

In a game development setting, motivational pats on the back and even concrete rewards like raises and bonuses will not have the intended effect if the developer cannot understand how to get those rewards. Even worse are unpredictable punishments - if working harder or taking

extra responsibility results in censure, then the developer will adopt a fatalistic and passive attitude. In some extreme cases, I've seen employees refuse to do anything until explicitly given an order or perform deliberately bad work.

4.2. Players Expect to Understand the World's Bounds

In games, players expect to recognize boundaries on actions and movement. Visual cues such as walls and precipices indicate the world's physical boundaries. The available controller actions constrain physical actions, e.g. some games have no jump button.

In the workplace, the employee wants to know the boundaries, too (they may think they don't want to have boundaries, but at the very least, they need to know the ones that do exist). For legal reasons, corporations often communicate boundaries on acceptable workplace behavior through orientations for new employees, employment handbooks explaining company policy, and, when they're particularly nervous about it, various forms of "sensitivity" training.

But all the legalese is ineffective without visible enforcement. Termination or other corrective actions will signal to everyone that the offending behavior, whether it be sexual harrassment or just talking back to the boss, is out of bounds. Letting the behavior go will either lead to ambiguity or signal tacit approval.

Boundaries also include organizational boundaries. Game projects typically provide the publisher with a project document listing key roles and responsibilities of the development staff. This type of documentation is also useful to the team, so they have an idea who's in charge of what and who reports to whom. Management often finds it convenient to keep these roles vague, which has the advantage of providing some flexibility, but doing this for political reasons will just result in confusion and recriminations.

4.3. Players Expect Reasonable Solutions to Work

After gaining some experience with a game, the player has idea how to solve problems in the game. The player will be frustrated and irritated if any reasonable solutions based on the gameplay so far turn out to be ineffective. So designers should take care to accomodate such solutions even if they are not the primary solutions intended by the designer.

Game developers also expect reasonable solutions to work (how many times have you heard, why doesn't this work?) Game development is often at the cutting edge in terms of technology and scale, so oftentimes techniques that "should" work, don't. Even, tools and middleware and equipment for game development are notoriously flakey.

So it is important to ensure that everything within control works like it should. The devkits provided by the console makers may arrive late and crash frequently, but the commodity

hardware and software (desktop and server computers, email, backup software, for example) should be rock solid. Compilers, debugging tools, and game engine middleware are often inadequately documented and in a beta state, so the production pipeline and internal tools should be well documented. The smoothest-running game projects I've seen still had plenty of mystery code, halting production pipelines, and IT glitches like servers crashing, data irretrievably lost, and ill-timed upgrades during crunch times.

4.4. Players Expect Direction

A game should give some indication of the the player's objectives. Otherwise the player may roam the game aimlessly wondering what to do, randomly attacking objects, NPC's and other players just to see if something will happen.

Game developers can also roam around a game project aimlessly. In the worst case, they will break other developer's code and art, and bitch, moan and complain.

Everyone wants to know what the game is about. What's the story? What type of gameplay are we trying to achieve? How does it fit in and compete with other titles on the market? This is what a high-concept documents is supposed to communicate, and this document should be readily available in-house as well as distributed to publishers. If you can't convince your own team of the viability of the game vision, then it's much less likely you can persuade publishers, their marketing staff, and retailers to buy into that vision.

Direction is also provided by the schedule. The final release date and interim milestones, including specification of the critical features required at each point, should be clear, reliable and changed only in drastic situations that warrant changing the entire schedule. In other words, the milestones and release dates should not be moving targets.

Scheduling is often performed down to a fine-grained level, in some cases to tasking day by day by day or even by hours. But while some developers may require micromanagement, it is important to make a distinction between this kind of supervision and a global schedule that the entire organization needs to be working toward. A day-to-day or even weekly schedule is volatile - vacations, sick days, emergency bugs and demos, server crashes and other natural disasters happen. Some tasks take longer than expected and some turn out to be easier than anticipated, and sometimes it makes sense to reorder them, but overall they should average out to meet the scheduled milestones.

4.5. Players Expect to Accomplish a Task Incrementally

Players usually know the overall objective in the game but expect to achieve this objective via a succession of subgoals. This provides awareness of incremental progress and reassurance that the player is on track. Without this feedback, a player could go off course

and not realize it.

Not only do game developers find completing subgoals more tractable than large monolithic tasks, subdividing large tasks is vital to risk management and project scheduling. Then progress can be measured and validated by monitoring the completion of these subtasks. When developers jump into general assignments such as implementing renderers, physics engines, AI without defining components and tests that can be completed in sequence, then such a project can drag on for months without visible progress, until it becomes apparent that it's going in the wrong direction or that no progress is actually being made.

4.6. Players Expect to Be Immersed

Obtrusive user interfaces and game glitches, particularly crashes, distract from the player experience. And a character that is difficult to control or unappealing will also prevent the player from comfortably playing that role and feeling part of the game world.

In a game development environment, you also want each team member to feel immersed in the project and concentrate on getting the job done without distraction. Bureaucratic and corporate artifacts should not intrude on what should be a project that is rewarding unto itself. For example, timecards and sign-in sheets, thick employee handbooks, administrative paperwork, will remind employees they are on the clock and working for "the man". Instead, the necessary evils of running a business should be kept simple and to a minimum, and the environment should exude the exciting aspects of the game industry - the office should have plenty of games, industry magazines, posters, etc.

As far as providing a suitably appealing and easily assumable character to play, this does have a counterpart in the game project, too. Each developer plays a role (sometimes more than one) - an enjoyable role will be played with gusto, a distasteful role will be dreaded and performed without enthusiasm.

4.7. Players Expect to Fail

Players want challenge, so naturally they expect to fail at some points in the game. Moreover, those failures should stem from inadequate or incorrect play, rather than "tricks" or "cheap shots" utilized by the game. And the game should start out easy and ramp up later in difficulty to avoid discouraging players before the reward of the gameplay becomes apparent.

Game developers are also in the business for a challenge (or at least they should be), so they cannot be held back by fear of failure. On the contrary, developers should learn from failure. Attempts to implement new algorithms, use new tools will almost certainly result in some failures, all as part of the learning process and should be anticipated in the schedule. (This area where breaking schedules down too far will diverge from reality - you don't know how

many different implementations of say, a dynamic shadow algorithm, you might try, but you should know when it has to be completely done).

These setbacks are acceptable as long as they are natural byproducts of the learning process, but aggravating if they are imposed by outside factors. Unrealistic schedules and frequent crunch times will leave room for less error while simultaneously increasing the number of mistakes. Unreliable hardware and tools...

As with a game, a game project should start out easy, so everyone gets in the flow and understands the rules - how to work with the production pipeline, how to work with the rest of the team.

4.8. Players Expect a Fair Chance

Although players expect to fail, they also expect a fair chance. Ideally, a player should be able to make it all the way through the game on the first attempt if no mistakes are made. This means that progress shouldn't require trial and error - it should be possible to deduce a successful path through the game. If the player finds that the only way to progress in the game is through guessing from sets of random choices, then it will seem like a waste of time.

Game developers also will become frustrated if it seems they have no way to make decisions short of guessing.

4.9. Players Expect to Not Need to Repeat Themselves

Players get annoyed if they have to repeat any tedious or painful portions of the game. Hence the availability of game saves, and, in particular, checkpoint saves.

The most obvious analogy in game development is avoiding loss of work. Code, game assets, and even production documents should be frequently and regularly checked into a source control system. And everything should be backed up periodically, with the archive media stored off site and test restores performed to verify the backup integrity. This will seem obvious to some and extreme to others, but data loss due to accidental erasure, hardware failures, and absent or faulty backups is all too common.

Another interpretation of this game design principle is that team members shouldn't be duplicating work. For example, timely communication and visibility of the code base should allow programmers to avoid redundant work and encourage code sharing. The game design and requirements, production pipeline, and any project and corporate procedures should be documented and easily accessible to avoid inefficiency in explaining and learning.

4.10. Players Expect to Not Get Hopelessly Stuck

A game should not allow a player to get stuck in a position from which there is no chance to complete or win the game. For example, a player should not be able to jump into a while from which there is no escape, aside from quitting the game. Either provide a way out or put the player out of his misery.

Developers also resent ending up in situations where they can't succeed, and rather than hit the Quit button, they may just sit there, resentful and apathetic. If they feel they're faced with unfair expectations given unrealistic schedules, unbounded features, late or scarce tools and assets, then they won't even try. If a developer is just not capable enough to succeed, then there's no point in breeding resentment by letting him linger on.

4.11. Players Expect To Do, Not to Watch

Rouse opines that players want to play, not watch cut scenes. While cut scenes can be instrumental in communicating narrative and setting up new levels, the duration of cut scenes should be kept to a minimum. Games that rely on cut scenes rather than gameplay inevitably fail to keep the gamers' attention.

Developers don't want to sit around and watch a game being put together - they want to be part of the action. Everyone has opinions on games and would like to develop a game that they actually want to play. You can't have game design by committee, but soliciting ideas from everyone will make them feel like part of the creative process. Internal newsgroups or message boards can be used for exchanging ideas. Contents can be held for names. Songs voted on. Blog-style developer journals can be kept for historical and promotional purposes.

One way for developers to feel part of the game is to literally make them part of the game, by modelling characters to resemble staffers, incorporating inside jokes, recording project members' voices for voiceovers - all of these will personalize the game for them.

4.12. Players Do Not Know What They Want, But They Know It When They See It

Once a game has reached a playable state, it is important to test it with real players and gauge their reactions. Focus groups cannot be relied upon to make game design decisions - that would be too easy. Rather, it is incumbent upon the game designer to observe their reactions and use observation and experience to discern what is and is not working in the game.

Similarly, employees may not be able to articulate precisely what they want in a workplace or project. Putting up suggestion boxes and soliciting feedback in employee reviews may elicit some useful ideas, but you'll get a lot of advice on running the project that still doesn't necessarily fix problems that are disturbing them. There is no substitute for the scientific method - observe your team dynamics, hypothesize about what's fundamentally not working

(or working), make corresponding adjustments and verify your fixes work.

5. A Never-Ending List

Besides playtesting and following the rules described above, a game designer can come up with any number of game design principles based on experience and personal tastes.

In project management, also, the best rule of thumb is to use one's own preferences. What kind of project do you want to work on? Why are you in the game business, what attracts you to a project and what would be your expectations? As they say, your mileage may vary, but if you at least design a project that you would want to be part of, then you have something that appeals to at least one known type of person.