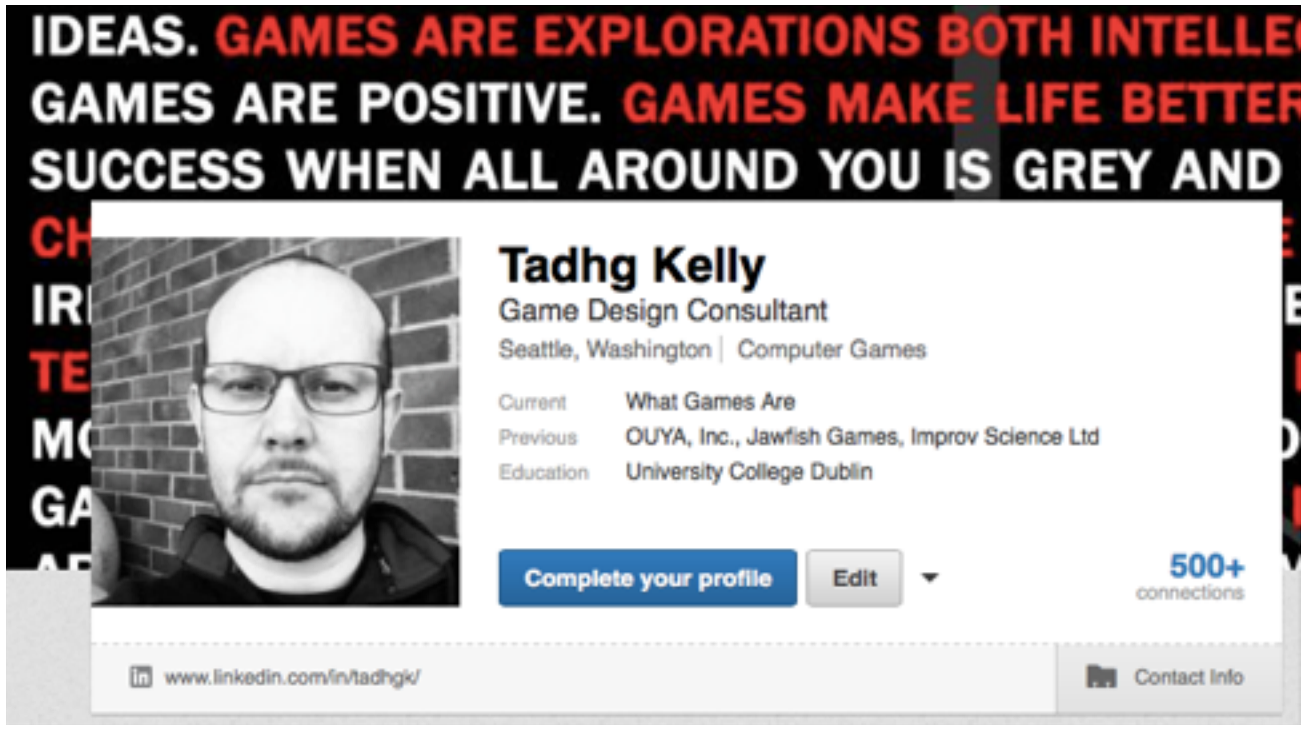
Raw Game Design

You Have A Game Idea. What Comes Next?

Tadhg Kelly
Game Designer

GAME DEVELOPERS CONFERENCE NEXT
FEATURING APP DEVELOPERS CONFERENCE
LOS ANGELES CONVENTION CENTER · LOS ANGELES, CA
NOVEMBER 3–4, 2014



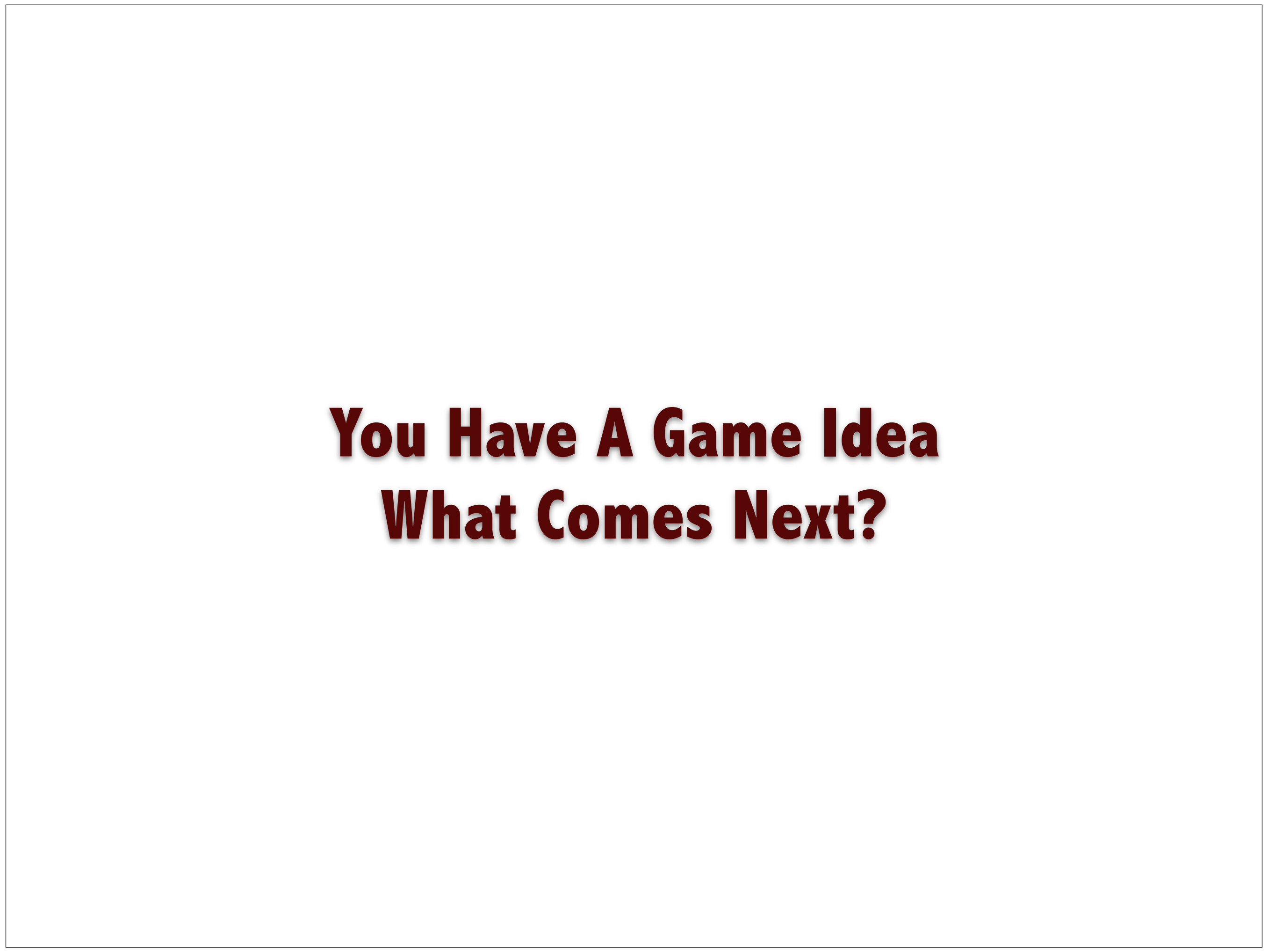
IDEAS. GAMES ARE EXPLORATIONS BOTH INTELLE
GAMES ARE POSITIVE. GAMES MAKE LIFE BETTER
SUCCESS WHEN ALL AROUND YOU IS GREY AND

Tadhg Kelly
Game Design Consultant
Seattle, Washington | Computer Games

Current	What Games Are
Previous	OUYA, Inc., Jawfish Games, Improv Science Ltd
Education	University College Dublin

Hi I'm Tadhg. I've been a game designer since 1993, and a video game designer since 2002. Though I'm Irish I live in Seattle, and from there I work as a designer and advisor for numerous companies across many sectors. I've worked with multinationals and tiny indies alike in the US, Canada, Australia, Europe and Asia. I also write a lot about game design and the industry, on my own site What Games Are, on TechCrunch and - until earlier this year - in the pages of Edge. And I give talks like this one at conferences like Casual Connect and the big GDC.

One of my favorite sci-fi quotes is Virginia Madsen in the opening scene of Dune: “A beginning is a very delicate time.” It’s a line I think of a lot in relation to games. Like many of you I’ve been involved in more failures than successes over the years. I’ve seen countless badly formed projects go awry, burning teams, budgets and companies in the process. Most of the time the root problem lies in how those projects are incepted in the first place. They set themselves up for failure.

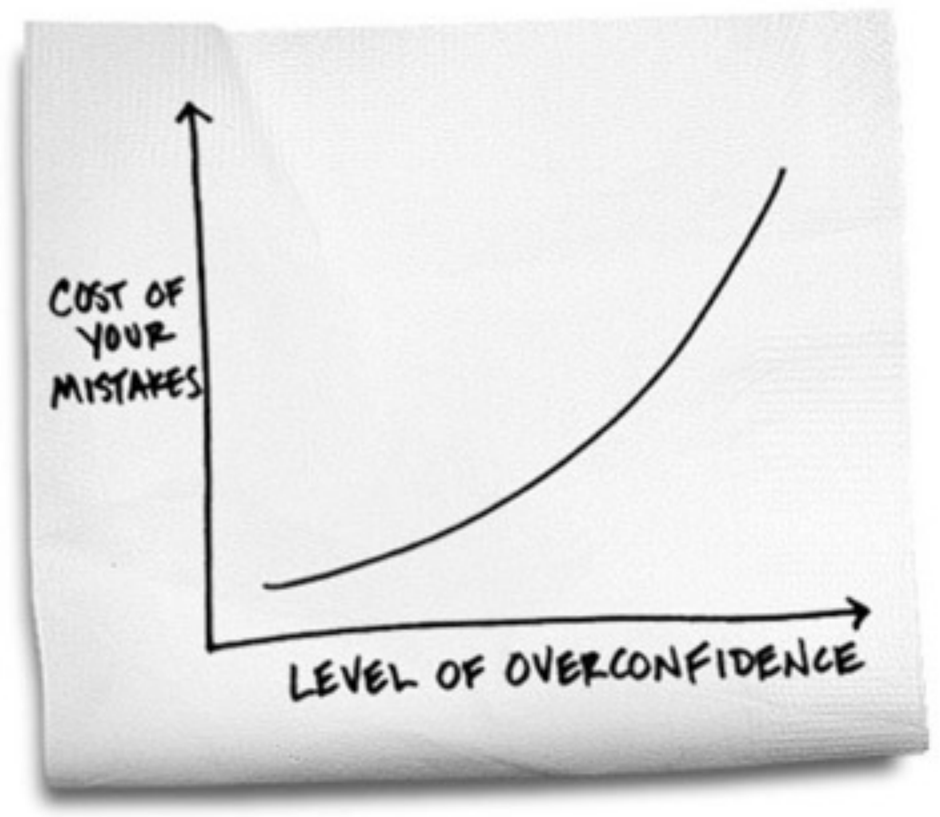


You Have A Game Idea
What Comes Next?

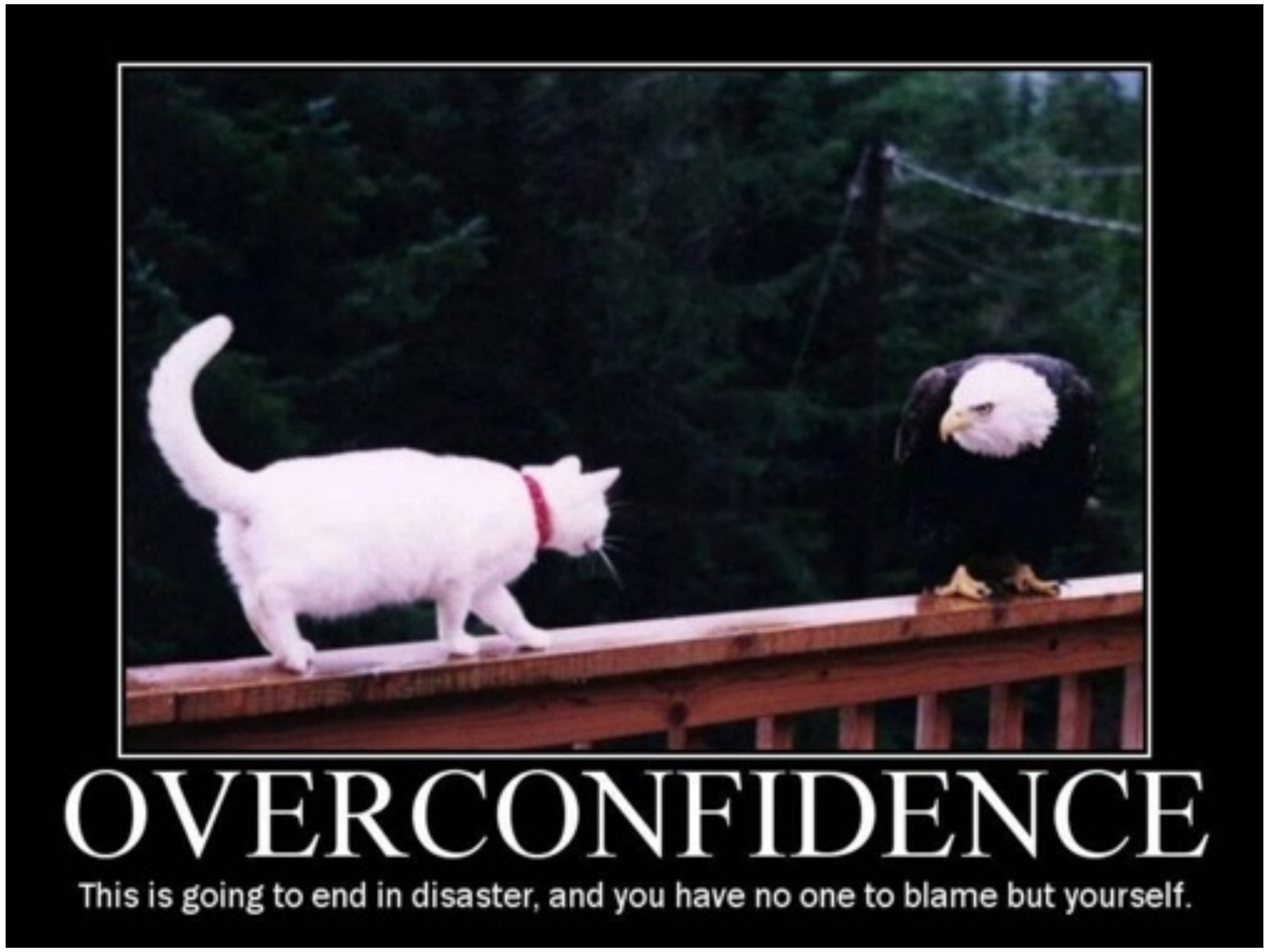
Today my topic is “beginnings”. How to start a game, what hazards to look out for, and how you can take advantage of prior learning to begin a project more effectively. I believe that beginning a project well is the single best way to ensure a successful outcome. Beginning badly, on the other hand, usually leads to disappointment.

Successful Games Start From Successful Beginnings

I believe that successful games start from successful beginnings. A beginning is when projects are conceived, ambitions are raised and excitement fills the air. A beginning is when the project has new money and new people and new ideas. A beginning is when the project has hope.



But a beginning is also where naïveté strikes. A beginning is when untested assumptions are accepted which will later cause a project to implode. A beginning is where team ego can seriously overestimate ability and bite off more than it can chew. A beginning is the point at which team members form very different ideas about what the game they're making actually is, and they start working on parallel projects.


OVERCONFIDENCE
This is going to end in disaster, and you have no one to blame but yourself.

A beginning is when people who don’t know what they’re doing make stupid decisions that come back to haunt them. A beginning is indeed a very delicate time.

The problem is that most of us don’t know how to begin well.

1: “Raw” Game Design

5: Wrap Up

2: Delicate Beginnings

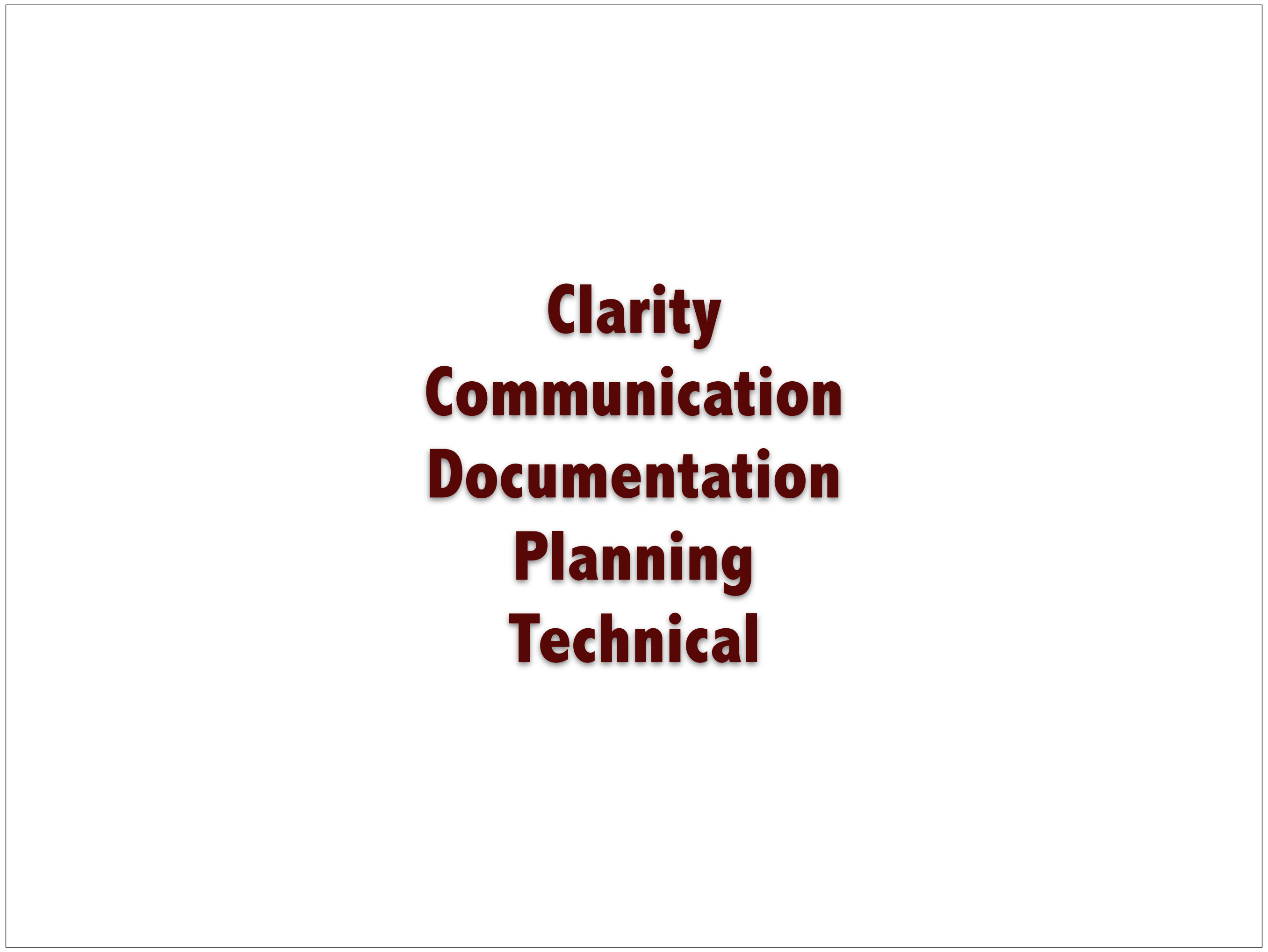
4: Seven Questions

3: The Awkward Part

My talk has five parts. The first part is an introduction to the book I’m writing, called Raw Game Design. The second part is about how developers usually begin a game. The third part is about challenging an awkward belief of game design theology in order to improve how we begin. The fourth part describes seven essential questions you should be answering before you start developing. And finally the fifth part is my conclusion.

1: “Raw” Game Design

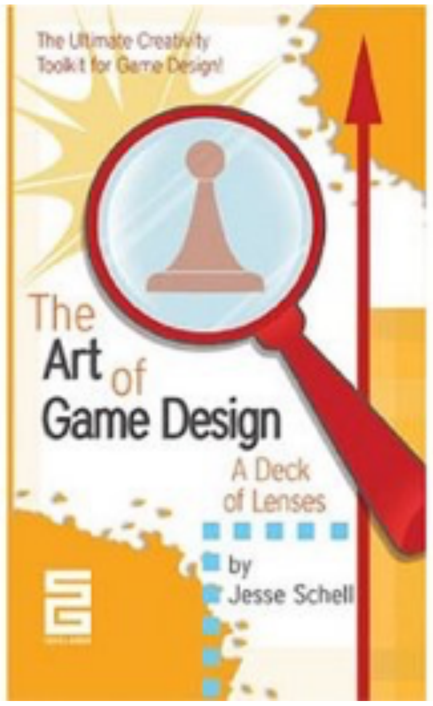
As I mentioned, I’m writing a book called Raw Game Design, which is being published next year by Focal Press. It’s all about a game design process, and a very important part of that is how games begin.



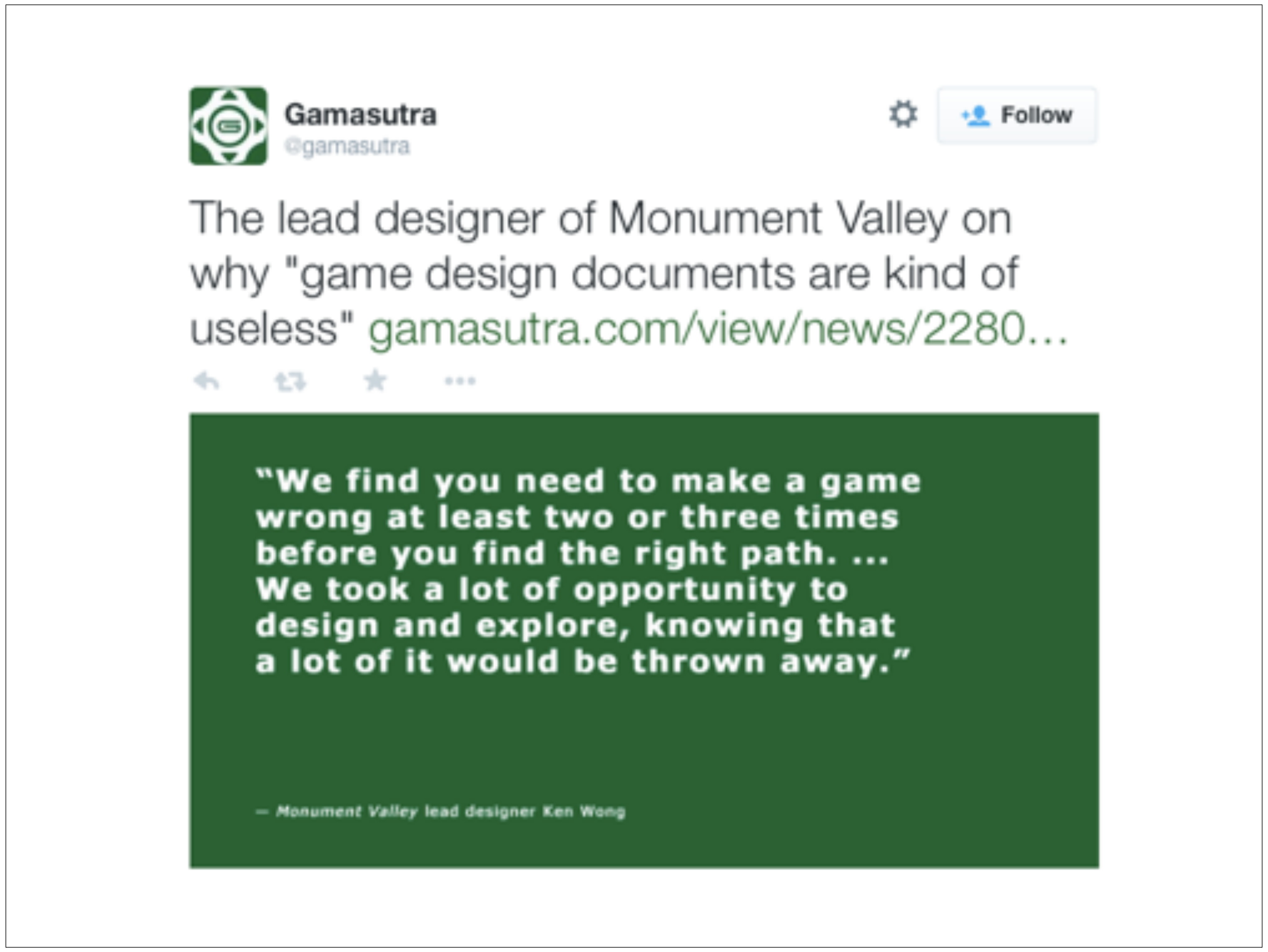
Clarity
Communication
Documentation
Planning
Technical

I asked myself in all my years what were the biggest problems that I had seen in game design.

Some problems are easy to identify like clarity, communication, documentation, too much or too little planning, focusing on the wrong problems like technical details. I've seen a lot of these kinds of problems, and I'm sure you have too.



Other problems are more challenging, like justifying what game design is and why studios should use it. For example a lot of people think that game design and game making are the same thing - which is not true. Game design is a skill distinct of programming, art, production, content creation, writing and more, but a skill that many people find hard to explain. This makes it hard to teach.


Gamasutra
@gamasutra

Follow

The lead designer of Monument Valley on why "game design documents are kind of useless" gamasutra.com/view/news/2280…

"We find you need to make a game wrong at least two or three times before you find the right path. ... We took a lot of opportunity to design and explore, knowing that a lot of it would be thrown away."

— *Monument Valley* lead designer Ken Wong

The opposite problem is people who understand game design is a distinct discipline, but consider it a waste of time, a domain of eggheads who write documents for no good reason and only get in the way. Critics tend to think that the only real game design is just prototyping, iterating and learning. But this method often doesn't scale well.

“Apple is like a ship with a hole in a bottom, leaking water and my job is to get this ship pointed in the right direction.”

— Gil Amelio - Apple CEO 1996

However perhaps the biggest problem is how modern design avoids decision making. It is in the nature of studio culture to gravitate toward consensus, to keep all options open and avoid conflict or blame. Part of why it takes so long to get any game made is that teams prefer to steer to the middle. Indecision is safety. But for the game designer indecision is a trap.

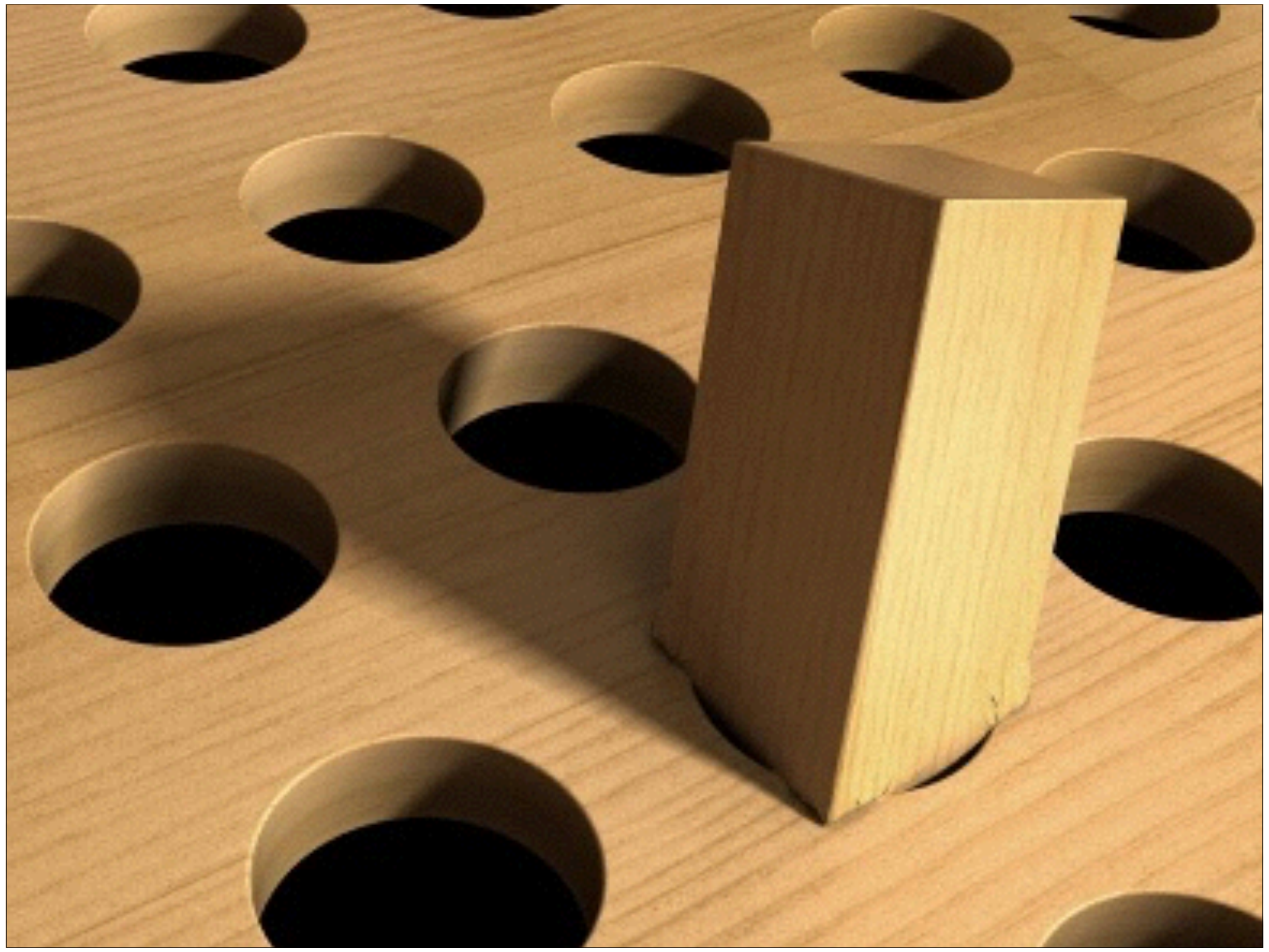
Indeed the game designer is often trapped. She understands the kind of game that she wants to make, knows the theory that underpins her ideas, is familiar with the themes that will make her game appealing to players and so on. She has a sense of how her game would push boundaries. She has a high level understanding of the topic derived from reading many books and articles and roughly knows what the shape of the game she wants to make looks like.

But her team has not read the same books. Her team does not share her perspective. Her team hates “waffly theory” and dense discussions. They want to know what to make. And that what they are making meaningfully adds to the project. They want the instruction set for what they’re supposed to build to be as clear as possible. They want decisions even though the studio culture tends toward consensus.

This is why designers often react with long game design documents. It may surprise some of you, but while it may appear that studios have moved on from game design documents, most actually haven't. They sometimes call them something else now, like design wikis, but the root reason for their existence is still the same. The game design document is alive and well and causing as much misery as ever.

For the designer a GDD is essentially a comfort blanket. It’s her way of proving that she works just as hard as everyone else, but when that document is handed to developers they won’t read it. They skim and cherry-pick the bits that seem important to them and then ignore the rest. Big documents die on the day that they are delivered, and most of us know we shouldn’t write them. Yet we sort of do anyway because usually that’s all we know. What could we replace them with?

My answer is process. There are numerous books and blogs out there to teach you the language of design and how to see like a designer. If you're not sure where to start here are some names: Chris Crawford, Raph Koster, Jane McGonigal, Steve Swink, Jesse Schell, Anna Anthropy or Daniel Cook. You'll find tons of great material from these people on how to think about designing a game.

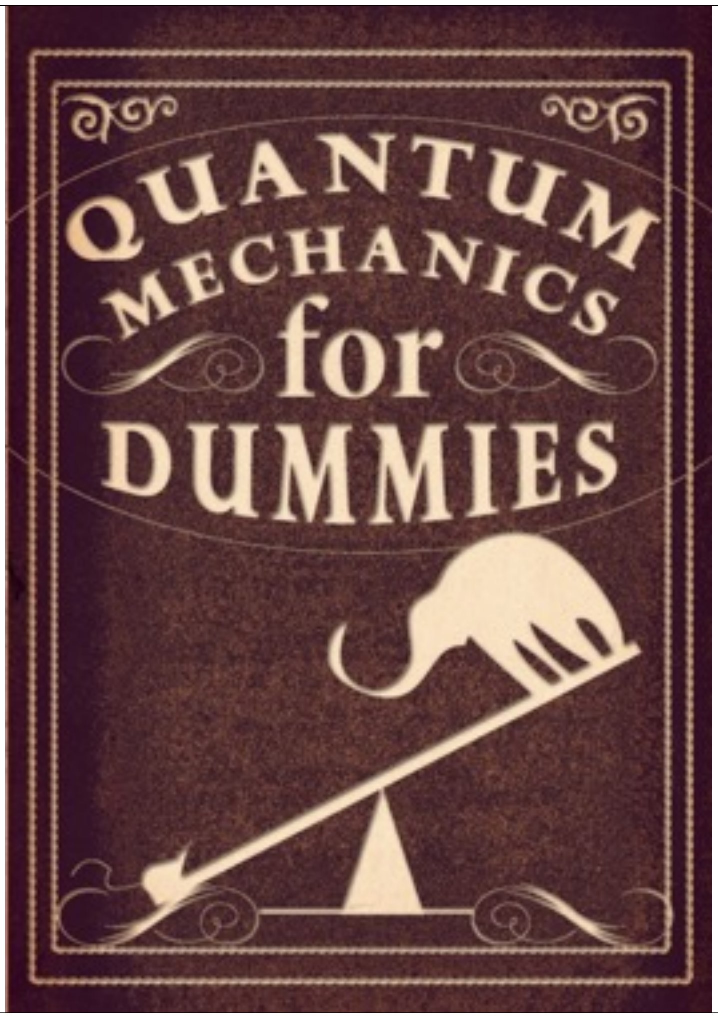


What you won’t find is standardized process. Game designers have never managed to settle on an set of effective standards. At best they have a formula for GDDs that’s passed around on sites like Gamasutra that advocates for writing Dungeons-And-Dragons style manuals.

With my book I’m trying to address that.

My argument is that the quality of paper design is often poor, not that the act of paper design itself is evil. Bad paper design is wasteful, even a risk to the project. It is usually too long, too contradictory, too set-in-stone like a manual of all time for the game. Good paper design is short, unambiguous and above all temporary. Good paper design prevents waste by forcing designers to be decisive. Getting from one to the other is a process problem. It can be fixed.

Raw game design is about understanding the design audience and what it needs. Game designers often think their audience is players, but this is not true. I recently attended a talk in Dublin by Brenda Romero in which she said it best: Your audience is actually an irritable programmer who's working on your game at 3am. He doesn't need to hear explanations. He just wants to know what your game does.



It’s not about “how to design a game” but rather a system for “how to communicate game design” and “how to make game design work in the studio environment”. Raw game design is a game design system. In a sense it games game design itself.

The way I look at game design is this: You might be a very creative person, but as a game designer your job is not to be the only creative person in the room. You might consider yourself technical, but your job is not to be chief technologist. You might have good visual skills, but your job is not to be lead artist. As a game designer your job is not to be the keeper of the flame.

Your job is to present a coherent functional vision of an end product. Your job is to set the target as clearly as possible so that other skilled professionals can deliver. Your job is to work iteratively, slightly ahead of your team to help them. Your job is to make decisions that will get the team to the next iteration and the next release, and back those decisions up. Your job is to prevent waste.

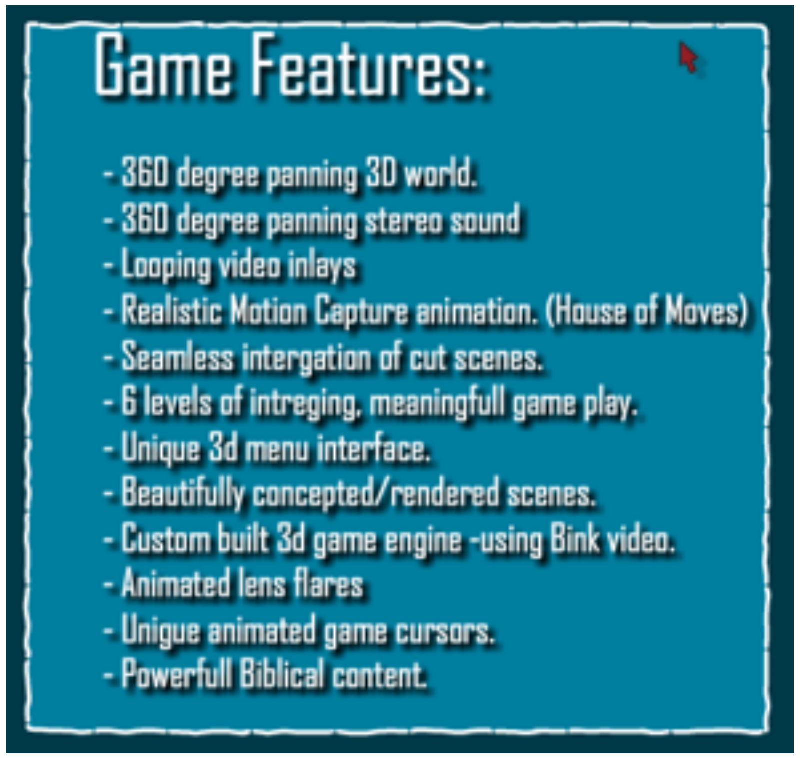
In the end of the day design only has value if it’s effective. In beginning a project design can be very valuable in setting up what my friend Amy Jo Kim calls “the right conditions for success” We can be smarter about how we take an idea and validate it, asking some hard questions early and demanding answers that will shape our conditions. If we are to call ourselves game designers it is imperative that we do this.

1: "Raw" Game Design

2: Delicate Beginnings

So you have an idea for a game, an idea that you believe in. You can see it in your mind's eye, how it works and what it does. You can imagine the faces of players when they play it, how they'll react, what emotions they'll experience. It's so real that you can practically taste it. Now what?

Well most of you traditionally go one of three ways. The first is to write a pitch or brief. You create a three or four page document that outlines the wider idea in order to sell the concept to your team, your executives, you publisher or financier.



Game Features:

- 360 degree panning 3D world.
- 360 degree panning stereo sound
- Looping video inlays
- Realistic Motion Capture animation. (House of Moves)
- Seamless intergation of cut scenes.
- 6 levels of intreging, meaningfull game play.
- Unique 3d menu interface.
- Beautifully concepted/rendered scenes.
- Custom built 3d game engine -using Bink video.
- Animated lens flares
- Unigue animated game cursors.
- Powerfull Biblical content.

You write a few paragraphs of aspirational text about the characters, include some concept art, enthuse about the target market and platform and present what you think will be unique selling points and the core mechanic. It will be 3D, you say. It will use tilting as a core interaction. It will feature great sound, great physics, cool customization possibilities and so on.

Maybe when you get everyone excited by the prospects of the game you assemble a more serious proposal to talk about likely risks, a go-to-market strategy and IP issues. You include more detail on the game itself such as worlds, numbers of levels, numbers of modes and so on. And from there, a game design document, wiki, project plan and more.

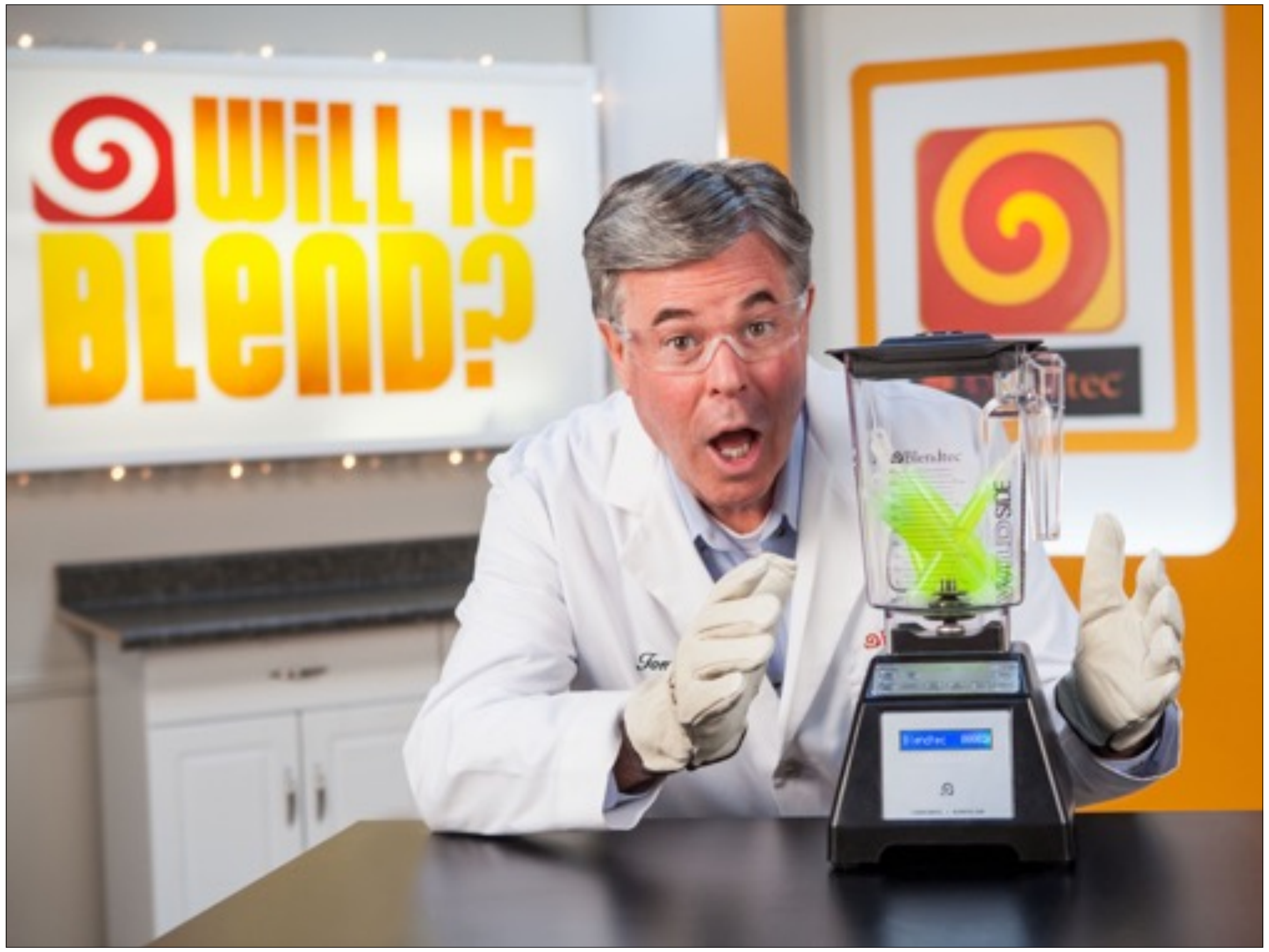
Alternatively you dive into prototyping. You say: “We have the idea, we’ll just starting putting stuff together in Unity and see what happens.” Sleeves are rolled up and some objects are assembled, and what follows is weeks, months or sometimes years of futzing about to try and make something. The belief is that the prototypes will eventually go somewhere given a long-enough timeline.

“METHOD”

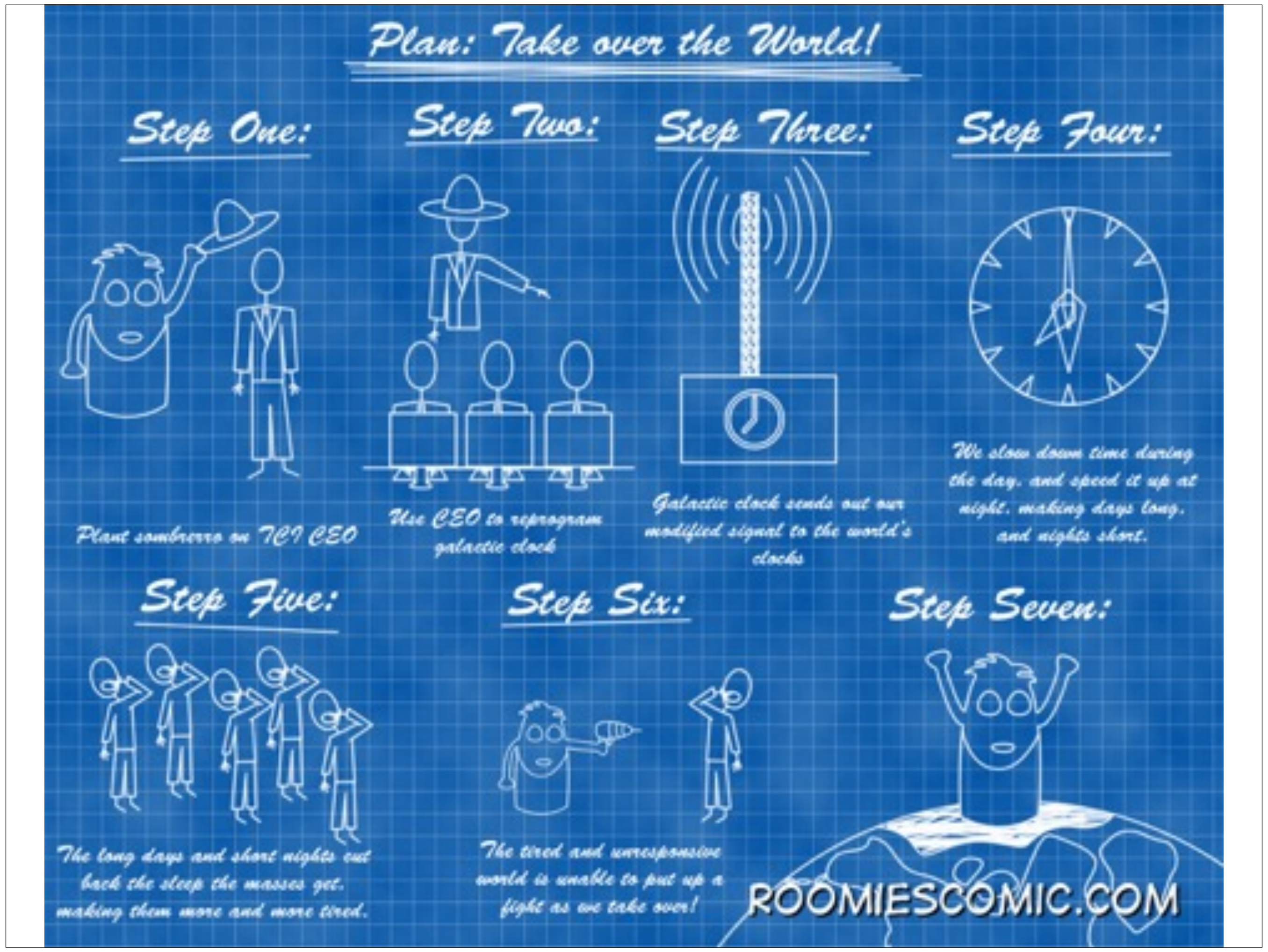
Mark Cerny
Cerny Games, Inc.

GameDevelopers
Conference Europe

The third option is to mix the first and second, perhaps taking inspiration from Mark Cerny’s “Method”. For action-adventure games Cerny advocates that games be split into two distinct phases: His first phase is “prototyping” in which the game is found and the end goal of which is to create a vertical slice of two playable and finished-quality levels. Then you can move into production.



None of these methods have a particularly strong success rate. The pitch-brief-design-doc method tends to go awry because it only makes contact with implementation quite late in the process, after the powers that be have been sold on the vision and signed checks and - in many cases – invented schedules and laid down milestones.


Plan: Take over the World!
Step One:
Plant sombrerro on TCI CEO
Step Two:
Use CEO to reprogram galactic clock
Step Three:
Galactic clock sends out our modified signal to the world's clocks
Step Four:
We slow down time during the day, and speed it up at night, making days long, and nights short.
Step Five:
The long days and short nights cut back the sleep the masses get, making them more and more tired.
Step Six:
The tired and unresponsive world is unable to put up a fight as we take over!
Step Seven:
ROOMIESCOMIC.COM

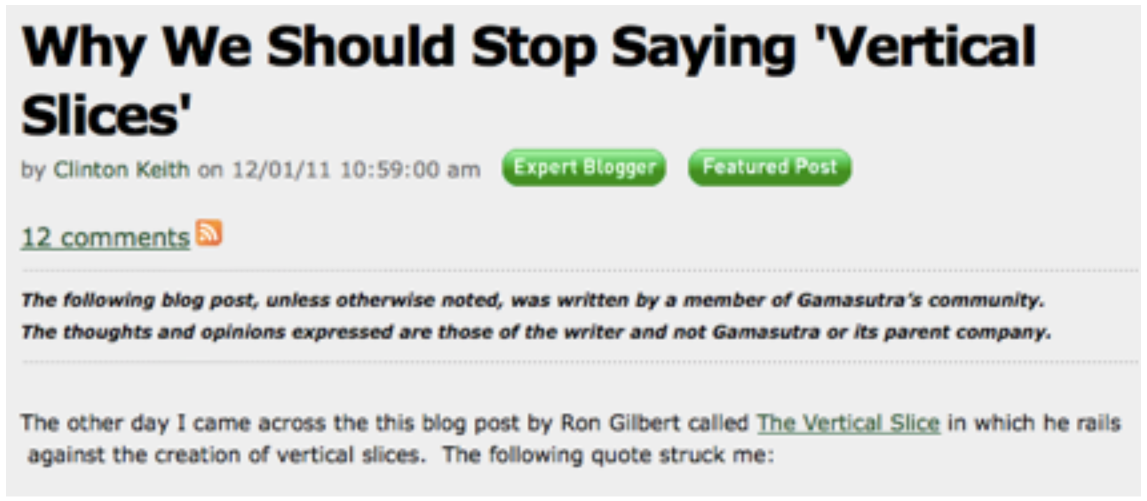
In that model it’s easy to add line items to the docs that turn into massive features, usually before anyone involved with the work (engineers, say) have had a chance to weigh in. Crunch, death marches, cancellation and major pivots are sure to follow.

I SAY A DAY WITHOUT DENIAL IS A DAY YOU'VE GOT TO FACE.
FROM NOW ON, I'M NOT GOING TO THINK ABOUT ANYTHING THAT'S UNPLEASANT.
ISN'T THAT A PRETTY SELF-DECEIVING WAY TO GO THROUGH LIFE?
I'M NOT GOING TO THINK ABOUT THAT.

On the other hand the just-prototype method tends to induce denial. When you're staring at code and level constructions all day long it's all too easy to lose sight of the end goal. Worse, you lose your objectivity. I've often seen prototyping purists descend into magical thinking and becoming immune to feedback. They may be making clear foundational mistakes, but have been staring at the game for so long that they can't see them. All they can do is argue and get defensive.

Meanwhile in larger studios that run on a pure prototyping philosophy the problem becomes that they descend into popularity contests. The studio may have a couple of evergreen games that essentially pay for an infinite amount of futzing, and so an infinite amount of futzing commences. Often the only way such a culture can ever get anything seriously made is to import projects from the outside.

Finally Cerny's method works IF you're a first party studio willing to pay a million dollars multiple times over in order to try and find success (Not that I'm saying Mark is Doctor Evil - I just like the picture). You have a considerable degree of protection from failure in that kind of environment.


Why We Should Stop Saying 'Vertical Slices'

by Clinton Keith on 12/01/11 10:59:00 am

Expert Blogger

Featured Post

12 comments

The following blog post, unless otherwise noted, was written by a member of Gamasutra's community.
The thoughts and opinions expressed are those of the writer and not Gamasutra or its parent company.

The other day I came across the this blog post by Ron Gilbert called The Vertical Slice in which he rails against the creation of vertical slices. The following quote struck me:

But most of us simply don't have pockets that deep, or we're tied into publishing contracts tied to vertical slices that don't tolerate failure. Vertical slices are also an illusion: What looks like 20% of the work to make two levels is often 60% of the work because you need to make a lot of the mechanics, technology and source models first. That can lead to immense pain if the publisher is only paying 20% to get there.

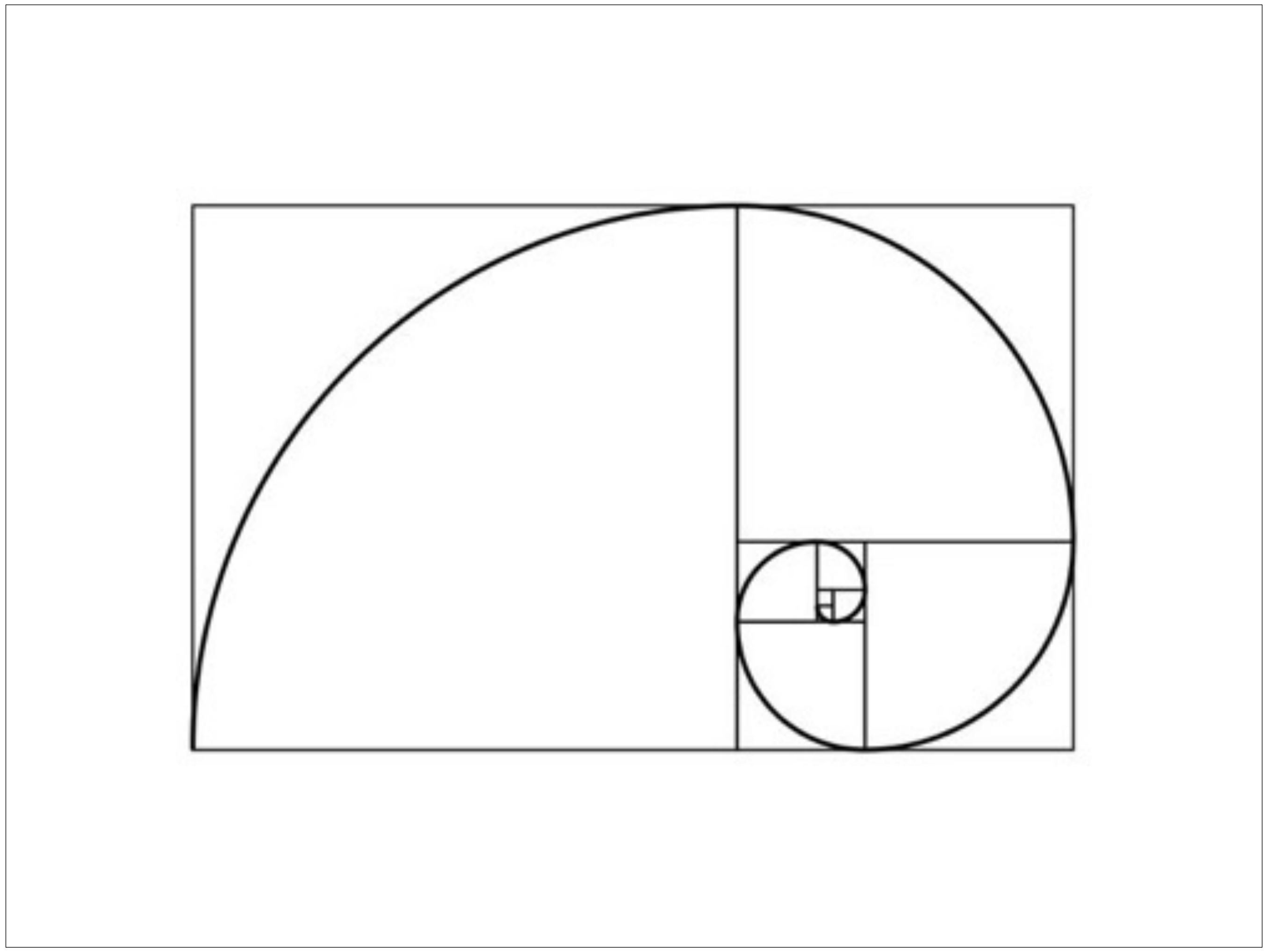
1: "Raw" Game Design

2: Delicate Beginnings

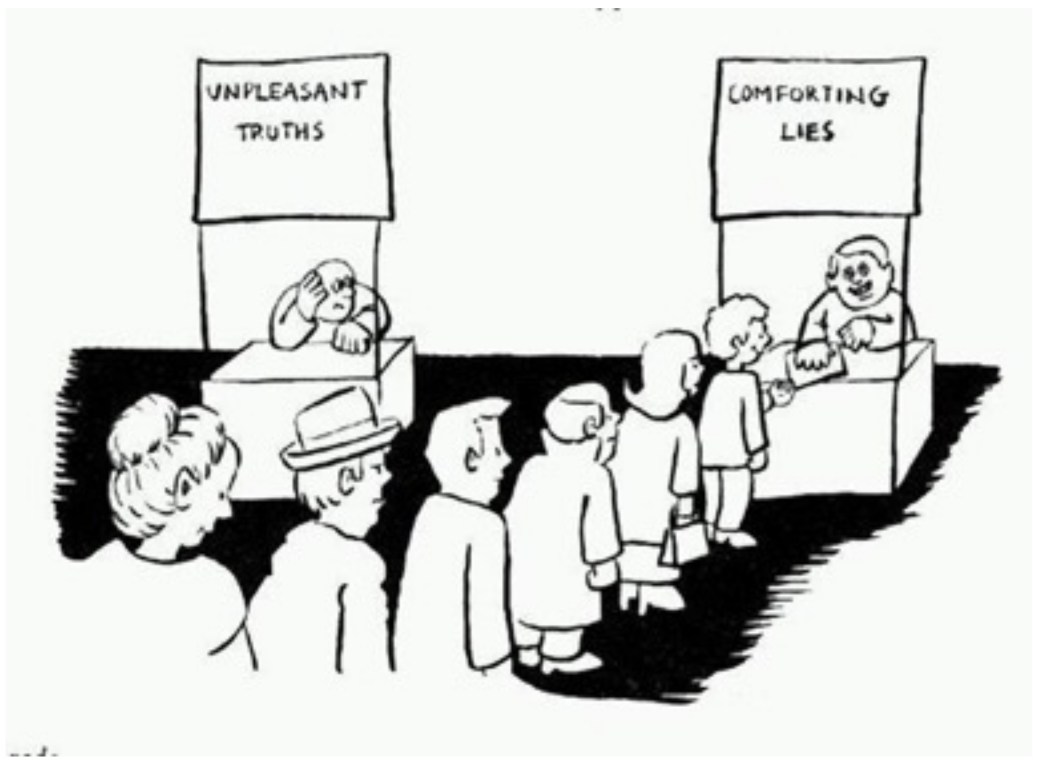
3: The Awkward Part

That leads me to talking about the awkward part.

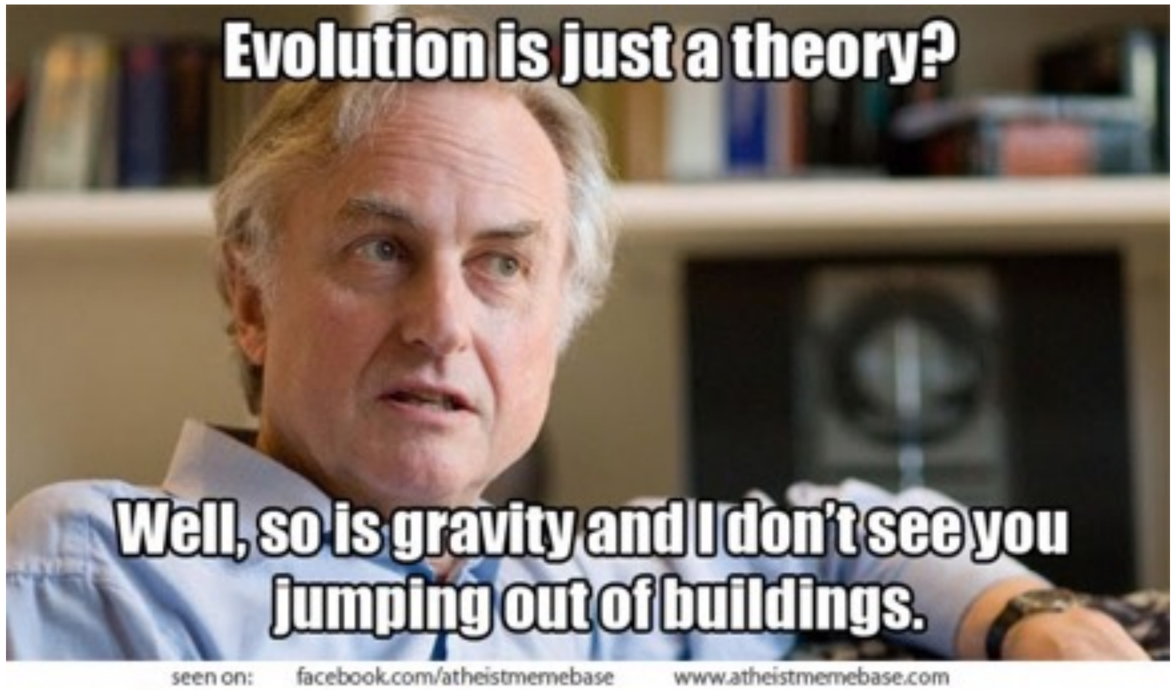
A curious thing about the usual methods for beginning is that they end up in very familiar places. Despite all of the heartache of document-led methods the projects that survive are often similar to other already-released games. Despite the noble questing of the prototypers, they seem to largely make platformers, graphic adventures or first person shooters etcetera. Despite the principles of split prototype-vs-production methods, their output is largely familiar too.



Don’t you ever wonder whether this means something? Is it possible that what seems to work in successful games is replicable? That successful games seem to adhere to certain shapes? That when it comes to assessing “what works” for games there are some partially objective answers? And that maybe we could apply that lesson using some sorts of tools?



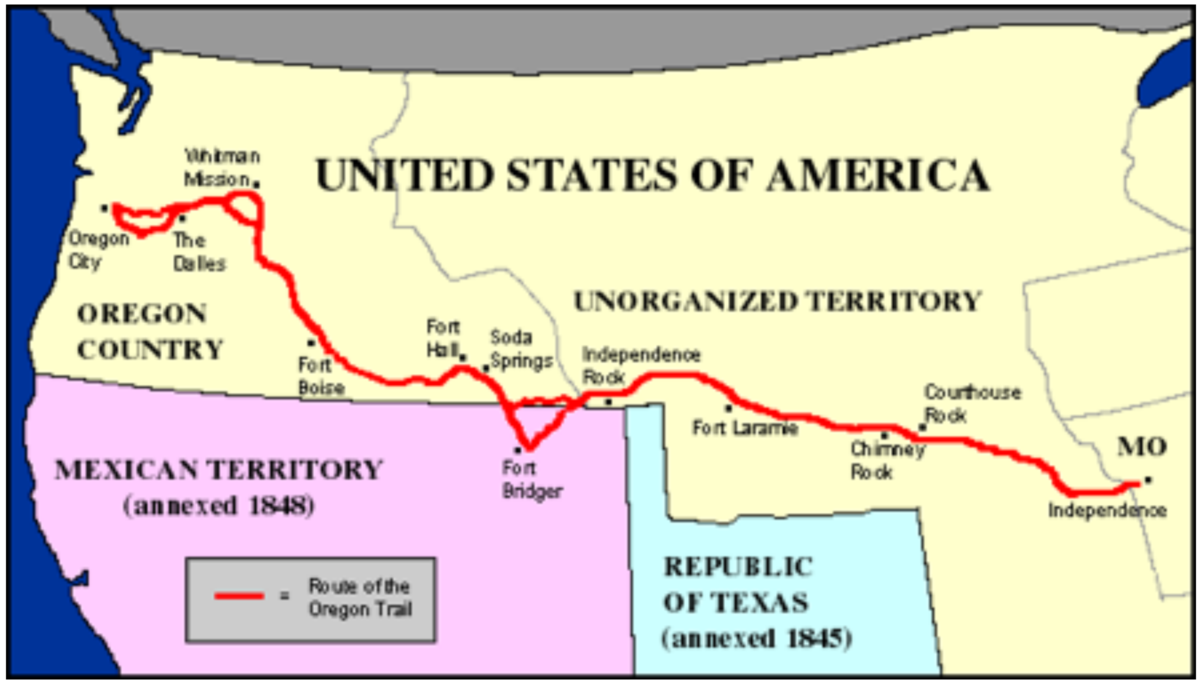
An interesting contradiction pervades game design. We tend to express a preference for scientism when talking about design but also harbor a deep terror of acknowledging progress within said science for fear of “ruining games”. We like to think in the ideal and the abstract, but when the time comes to say that maybe this means game design could be expressed as templates we could use, we run for the hills.


Evolution is just a theory?
Well, so is gravity and I don't see you jumping out of buildings.
seen on: facebook.com/atheistmemebase www.atheistmemebase.com

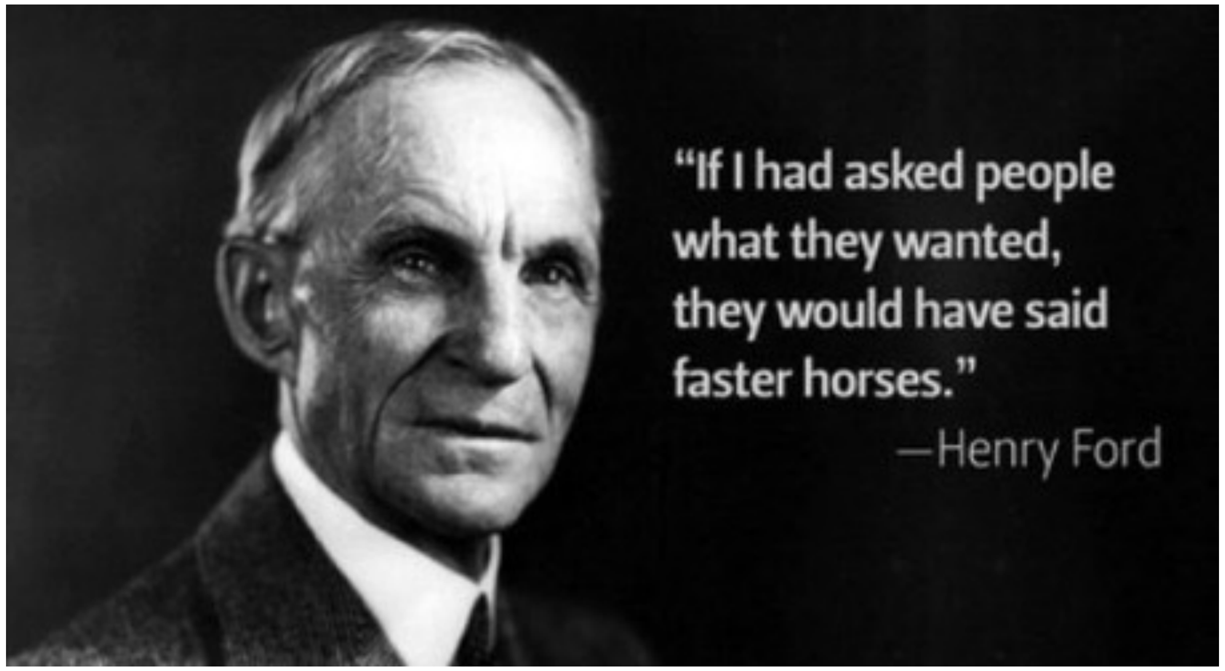
In other words most of us attend conferences like this is to find "the answers", but there are some answers that we don't like to hear. We don't like to hear that there are some design directions are just bad. We don't like to hear that – just like the novel, the movie, the automobile or the standing structure – there are describable ground rules for what works and doesn't in producing successful game entertainment.

"Actually It's About…"

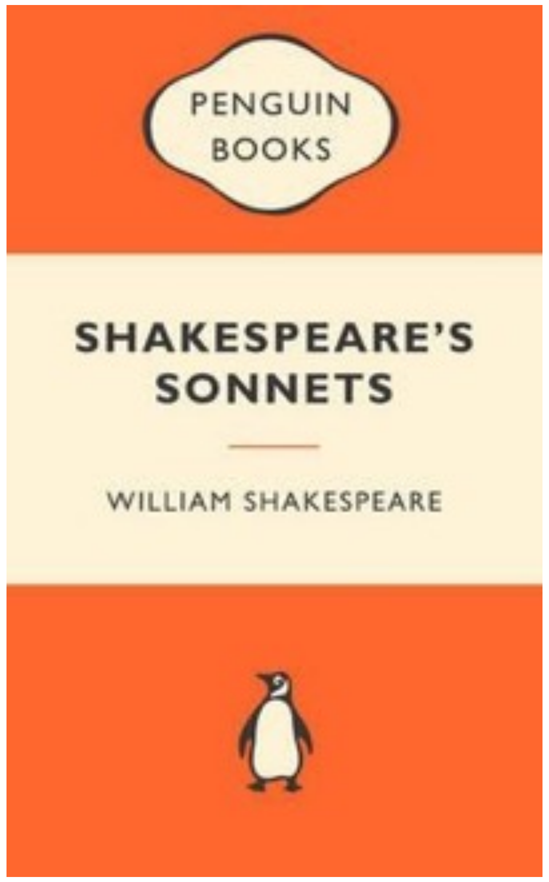
Most especially we don't like to hear that games are long past their infancy and actually coming through their tantrum teenage years (hello Gamergate). That we, in fact, missed out. That we don't live in the founderwork era of games.



What do I mean by founderworks? We like to think that it's still the 80s and early 90s when games like Mario, Tetris, Streetfighter, Day of the Tentacle, Zelda, Elite, Populous and others were released every week. We use all of these prior games as cultural references. But we don't like to face up to the idea that they mapped out the shape of video games.


"If I had asked people what they wanted, they would have said faster horses."
—Henry Ford

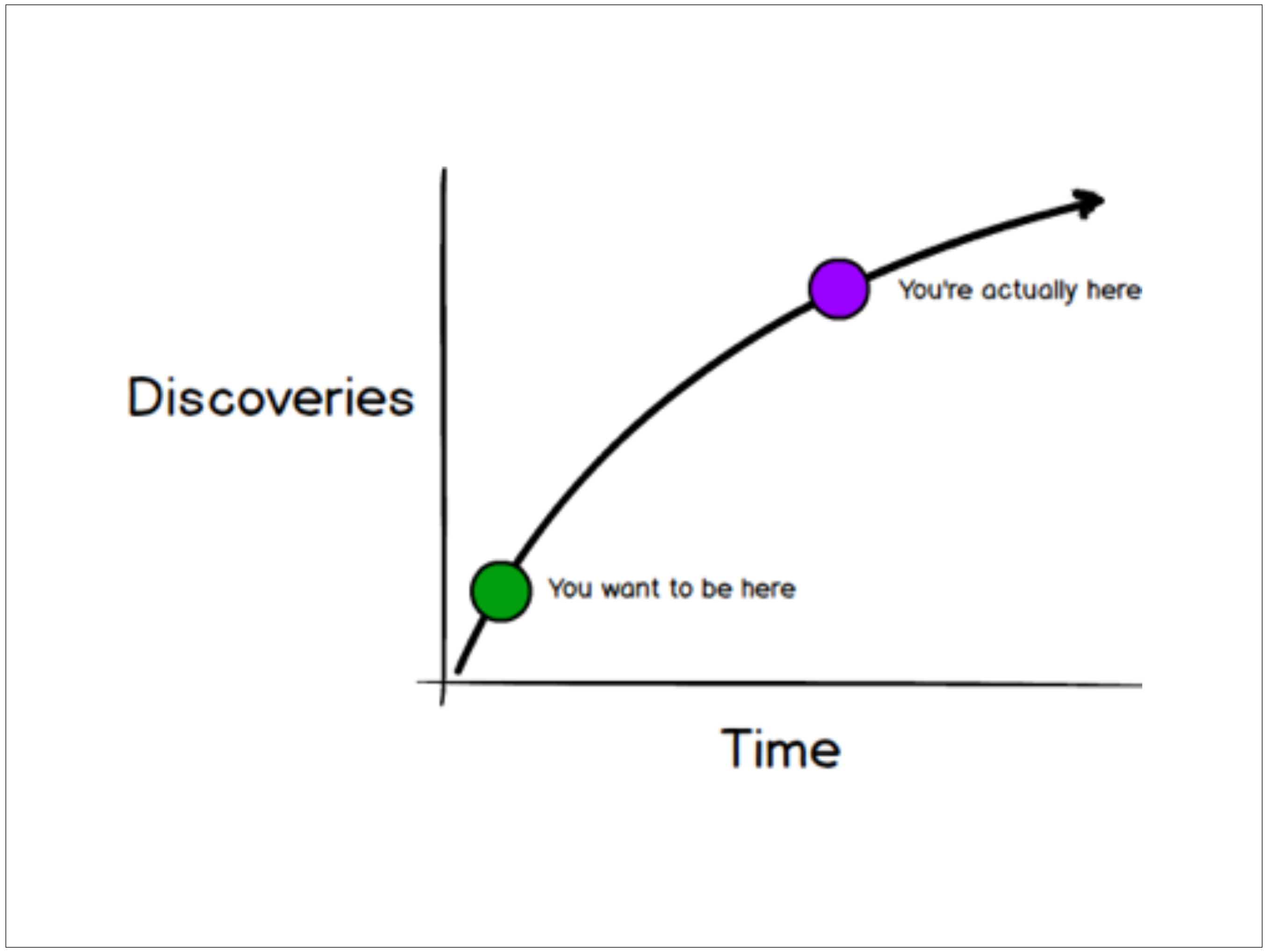
Some of my friends tend to think that in expressing this opinion that I'm arguing solely for faster horses in a new age of motor cars. That's not it at all. I'm not saying there's no room for innovation or creation in games. What I'm saying is that founderworks discovered shapes that were successful, and often the way that modern games succeed is by finding their way back to those shapes.



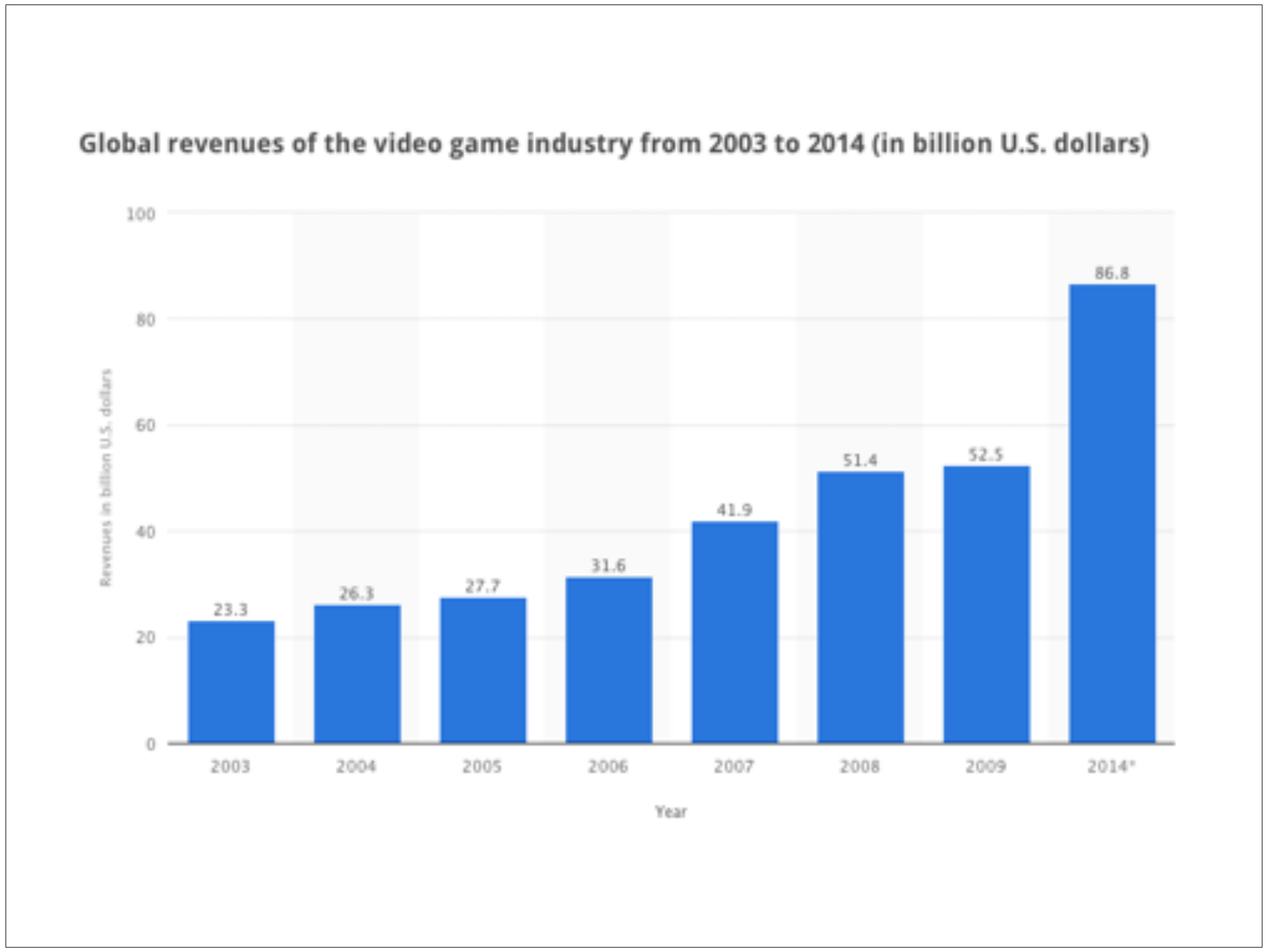
Over and over we see this happen. We see new interfaces and new technologies, but the same shapes re-emerge. We see new franchises and characters, but from a play-function perspective games maintain a remarkable consistency to prior works. We see a lot more extension rather than invention, a lot of cross-pollination rather than beginning from first principles. Like Shakespeare's formula for sonnets, some shapes just work better than others.

For example: We make use of player conventions in certain ways, handle progression and structure and camera in ways that are actually pretty historical if you take the time to look. Thirty years after their invention it turns out that players still like to jump in two-dimensional games, like to pilot first-person-shooter characters and death match, like to complete falling-sorting games. Different names, same Lego set.

If you can allow yourselves to accept that games have a history and use it rather than pretending to reinvent the wheel, you might be better able to begin projects well. The other option is to be depressed that you were born too late.



Does it mean there are no new shapes to discover? No. It means that the discovery of new shapes has slowed, and our creative opportunity often lies more in using and recombining existing shapes. I know many of you want to believe that you're still living in a time when the rate of new game invention is high, but you're not. You think you're here, but actually you're here.



After 40 years, a million games, more than a trillion dollars of revenue and more successes and failures than we can reasonably count, it's fair to say that games are not taking their baby steps.

And if that’s true then we can explore successful vs unsuccessful shapes. Rather than clone or copy them we can apply the larger lessons from previous successes to what we conceive and see if they stand up to scrutiny. We can retain the sense that our games are individual creations, but kick their tires early to see if they’re made of rubber or glass.

1: “Raw” Game Design

2: Delicate Beginnings

4: Seven Questions

3: The Awkward Part

How? By asking seven questions.

Q1: Is The Game’s Fiction Urgent?

Question One: Is the game’s fiction urgent?

The Seven Constants Of Game Design, Part One

Posted Jul 20, 2014 by Tadhg Kelly (@tiedtiger), Columnist

Editor's note: *Tadhg Kelly writes a regular column about all things video game for TechCrunch. He is a games industry consultant, freelance designer and the creator of leading design blog What Games Are. You can follow him on Twitter here.*

NEWSLETTERS

✓ **TechCrunch Daily** Top hea delivered daily

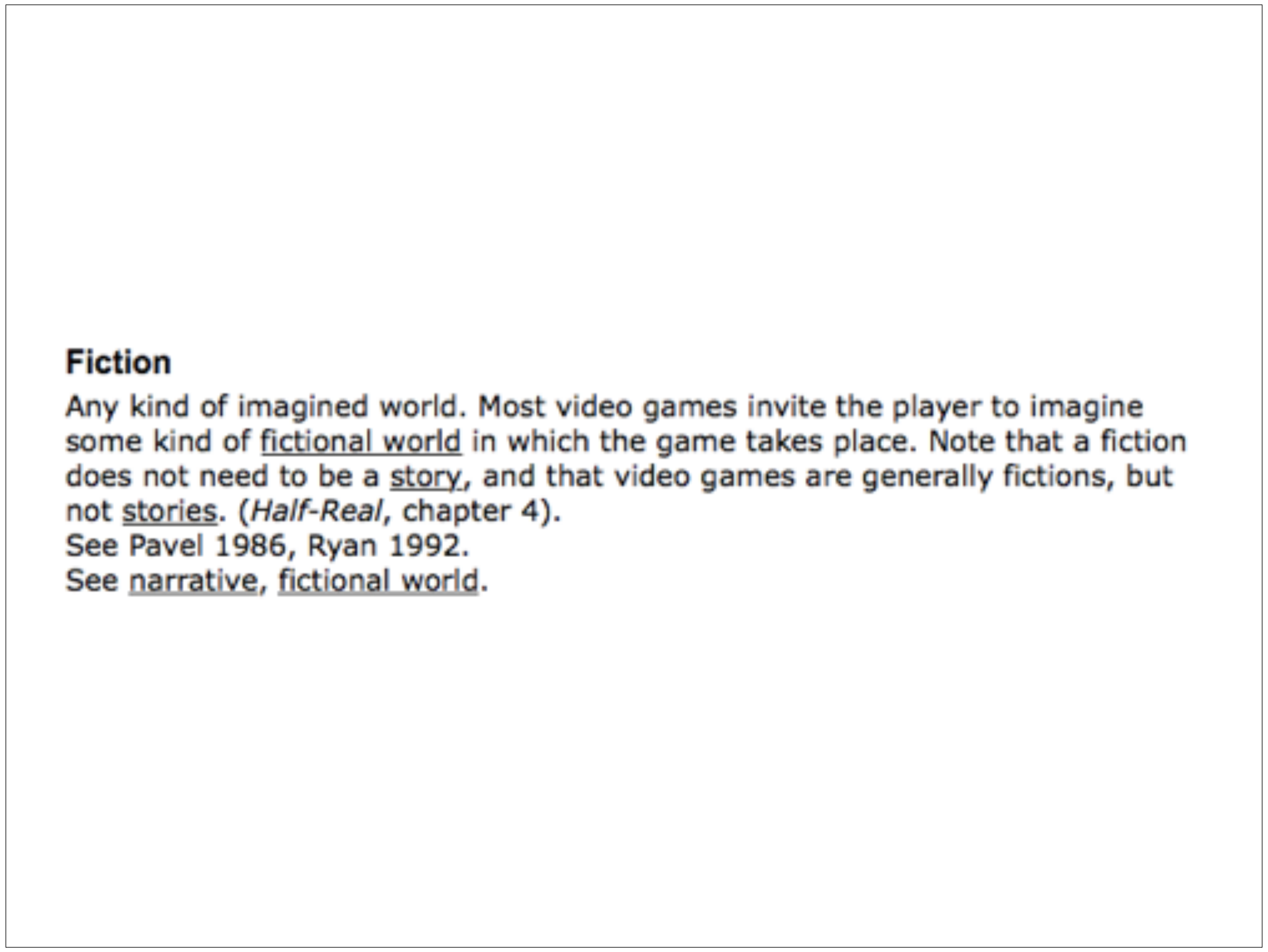
✓ **TC Week-in-Review** Most stories, delivered Sundays

✓ **CrunchBase Daily** Latest s fundings, delivered daily

Enter Addresss

SUBSC

In an article series on TechCrunch I talked about “Seven Constants of Game Design”, seven factors that determine games’ shape and which designers can use. They are Fascination, Urgency, Naturalism, Purpose, Imperfection, Time and Self. A Twitter user subsequently pointed out to me that they formed the acronym FUNPITS.



Fiction

Any kind of imagined world. Most video games invite the player to imagine some kind of fictional world in which the game takes place. Note that a fiction does not need to be a story, and that video games are generally fictions, but not stories. (*Half-Real*, chapter 4).
See Pavel 1986, Ryan 1992.
See narrative, fictional world.

Urgency is one of the most pressing constants to focus on early, in particular as it relates to the fiction of the game. By fiction here I don't mean "story", but rather in the sense that academics like Jesper Juul use it: Fiction is the broader canvas in which the game is set, its creative world if you like. Urgency is the quality of that canvas which creates pressure, and thus a need to solve, to do, to overcome.

Fiction may well mean a fantasy world in which demons are attacking humanity, or it may mean an arcade world in which blocks are falling from the sky. It may mean suburban life and the struggle to keep your head above water, or a text adventure in which the player is trying to escape from a maze.

Most fictions have some aspect to them that lends themselves to urgency, but the trick is finding it. Just as cop and lawyer shows tend to have an innate drama, some game fictions have an innate urgency like work scenarios, abstract scenarios or war scenarios. Others do not.

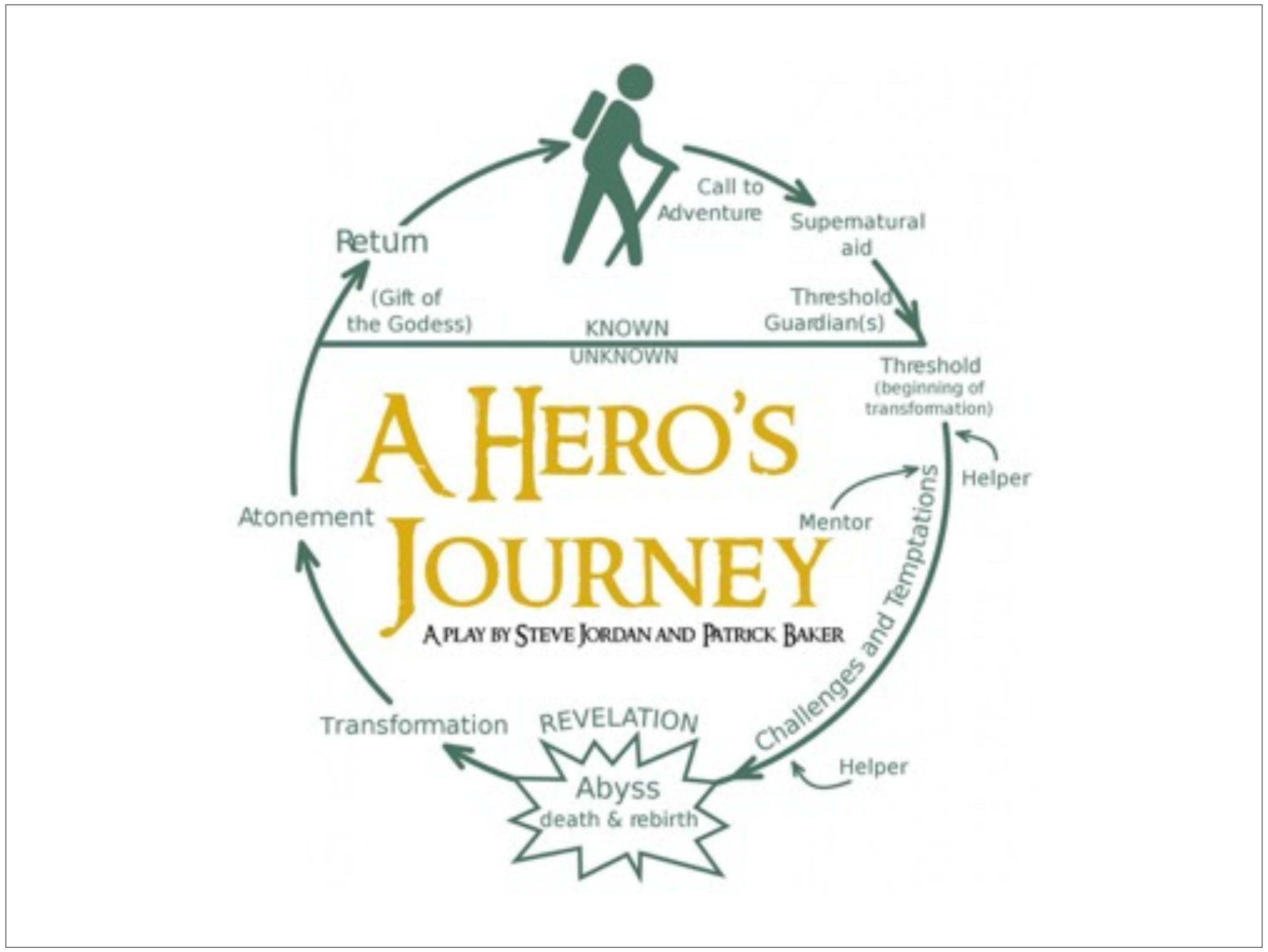
The Truth Inside Destiny's Loot Cave

by Ludwig Kietzmann @ludwigk (1 month ago)

Fictional urgency is usually related to quantification. Successful games tend to be fascinating logic boxes, and their urgency is associated with how resources move, how they are blocked and the various win or fail states that emerge. So if you are able to express the urgency of the game's fiction in terms of the movements of blocks of stuff, you are likely on a stronger path than if you can't.

Q2: What Is The Player's Job?

Question Two: What is the player's job?



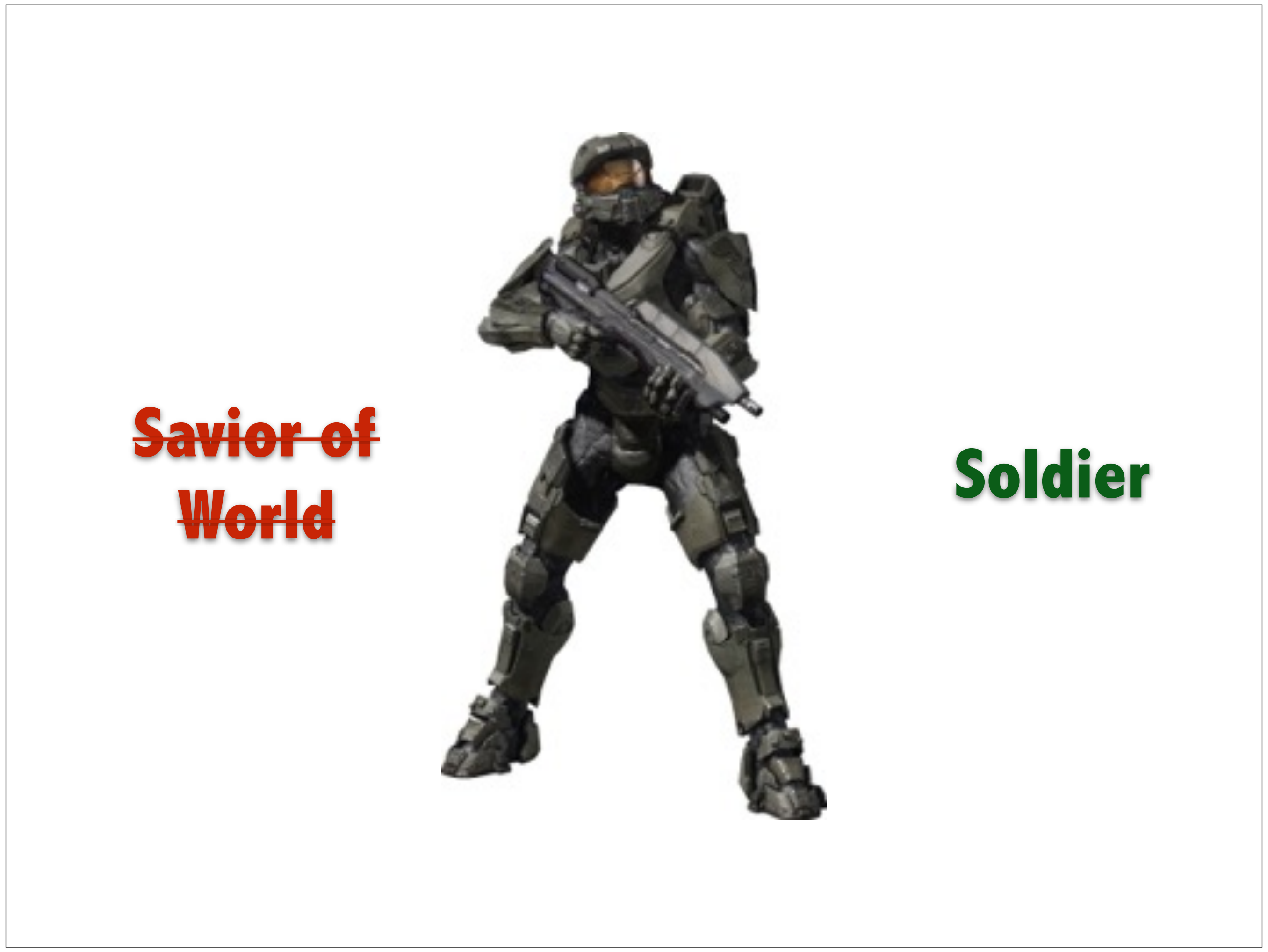
Most game pitches focus on expressing who the player's avatar is, what his origin story might be, what his motivation is (in the manner of an actor) or why he appeals to a certain demographic. While all of this information certainly helps sell an idea it rarely helps in determining whether it works.

Instead it’s important to know what the player will do. I don’t mean character here, nor role, nor dramatic motivation. Depending on your game those might be important, but what I’m referring to here is the player’s job.

My job in the game of Tetris is “sorter”, in Tomb Raider it’s “adventurer” and in Rome: Total War it’s “general”. Job means the linchpin activity that the player performs in the game and the context in which she performs it. As a designer I’m not interested in who the player’s avatar is as a person. It’s just a doll. I’m interested in what the doll does, what its affordances are and what, once the player picks up the controller, it can do.

Defining the player's job early has two benefits. The first is it establishes how the game world moves around the player. This is one of the key points of failure commonly experienced in projects developed through pure iteration. Often such projects become about creating a working simulation of a world and they forget about the player until later. It takes so long to make the simulation function at all that the player has no part to play.

The second benefit is that a job establishes a fictional context, which helps give the player a sense of the likely boundaries of the game. If the player's job is "criminal" in Grand Theft Auto then it's unlikely that the game will ask him to sort threes of objects. If it's "footballer" as in FIFA, then it's unlikely that he'll be waiting tables. Context helps center a game for the player and also gives the designer a sense of the balance between activity and economy throughout the rest of the game.



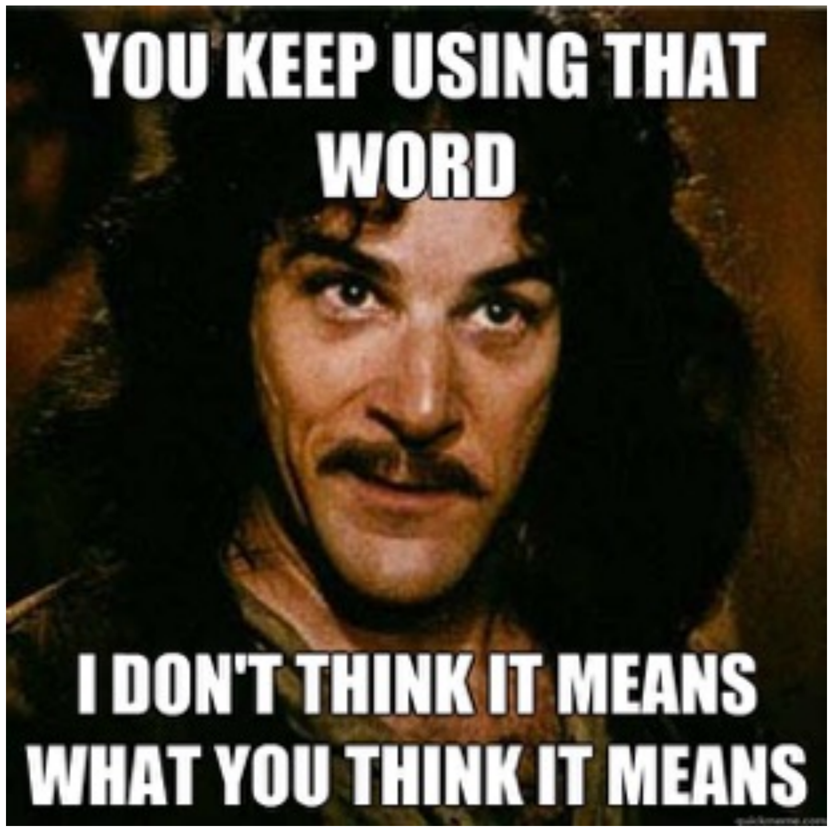
~~Savior of World~~

Soldier

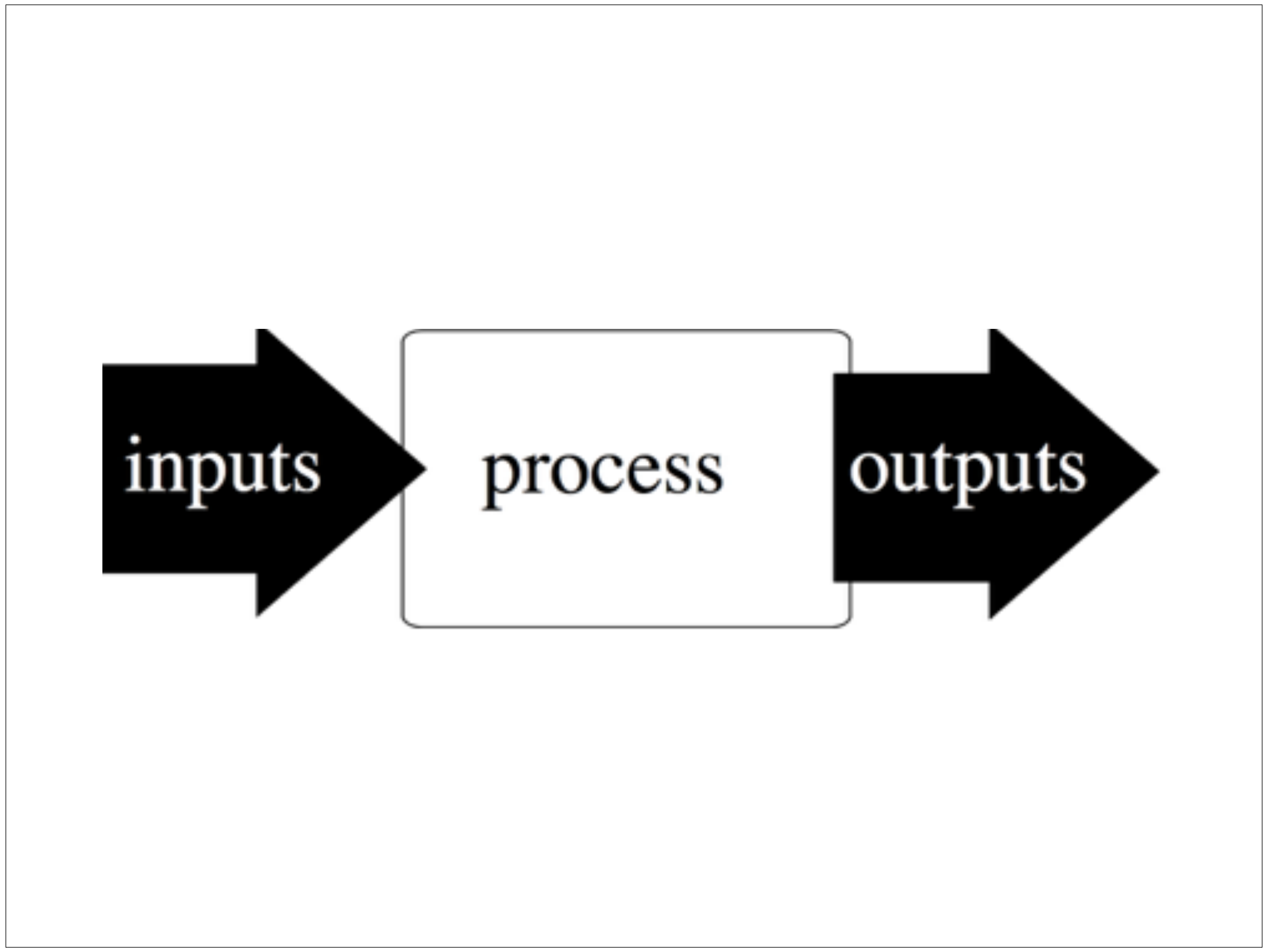
By the way the player’s job should always be a one or two word phrase at best, and unambiguous description of an activity. And furthermore you should avoid the temptation of assigning him multiple jobs. That just means you’re making multiple games, and it’s quite likely that your end product will lack cohesion.

Q3: What Are The Verbs?

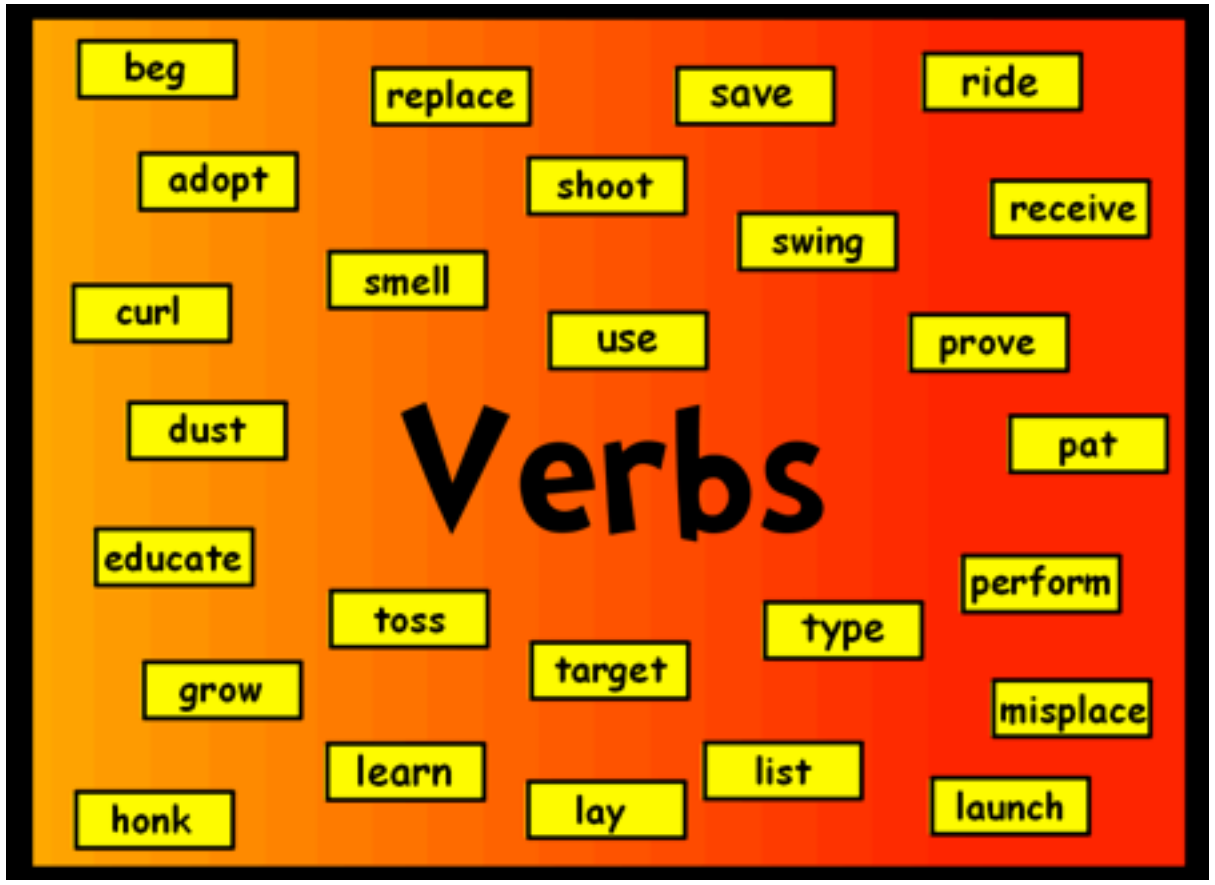
Question Three: What are the verbs?



In a number of game design books and blogs you'll often see mentions of "inputs", "verbs" and "actions" and a generalized understanding that game design works by starting with the user. However the problem you'll quick run into is that there's no one common usage of input, verb or action. So here's how I use them.



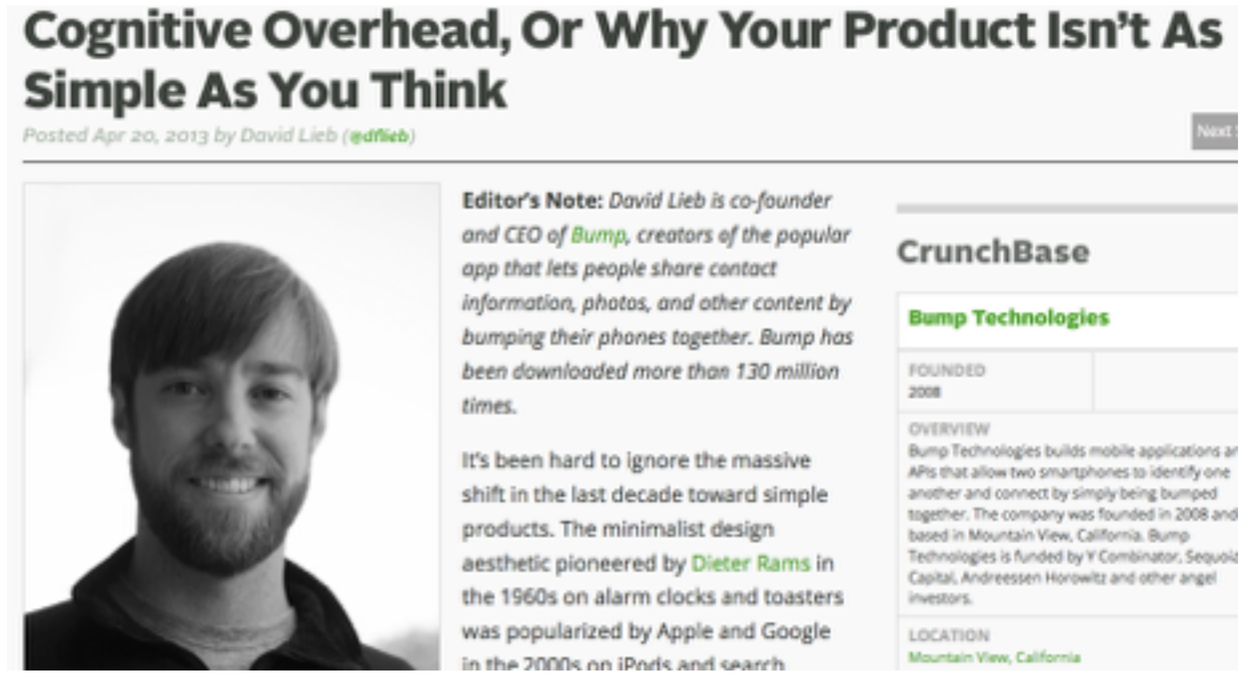
“Input” is the generalized term that means “stuff that players do”, just as “output” means “stuff that the computer screen and other devices show”. This is a general understanding that has existed as long as computers have, and is unlikely to change.



“Verb” is how I describe what happens on this side of the screen during input. It’s the thing that you do in the real world, such as moving the mouse, clicking the left button, pulling the right trigger or holding X. Verbs are how we describe how we use computers. We say “I clicked that” or “I swiped left” and when we do we mean things we did with our fingers, thumbs or voices.

What happens on the other side of the screen? I’ll get to that in a minute.

It's good to know what verbs you're likely to use as early as possible. To know on day one that you want to use drag-and-hold, for example, leads to a number of potential screen-side inputs that frame what the player is doing in the game. Similarly knowing that the player is likely to hold a trigger for long periods of time (for instance in accelerating the car) introduces a finger constraint for some other kinds of input.



Cognitive Overhead, Or Why Your Product Isn't As Simple As You Think

Posted Apr 20, 2013 by David Lieb (@dflieb)

Next

Editor's Note: *David Lieb is co-founder and CEO of Bump, creators of the popular app that lets people share contact information, photos, and other content by bumping their phones together. Bump has been downloaded more than 130 million times.*

It's been hard to ignore the massive shift in the last decade toward simple products. The minimalist design aesthetic pioneered by Dieter Rams in the 1960s on alarm clocks and toasters was popularized by Apple and Google in the 2000s on iPods and search

CrunchBase

Bump Technologies

FOUNDED
2008

OVERVIEW
Bump Technologies builds mobile applications ar APIs that allow two smartphones to identify one another and connect by simply being bumped together. The company was founded in 2008 and based in Mountain View, California. Bump Technologies is funded by Y Combinator, Sequoi Capital, Andreessen Horowitz and other angel investors.

LOCATION
Mountain View, California

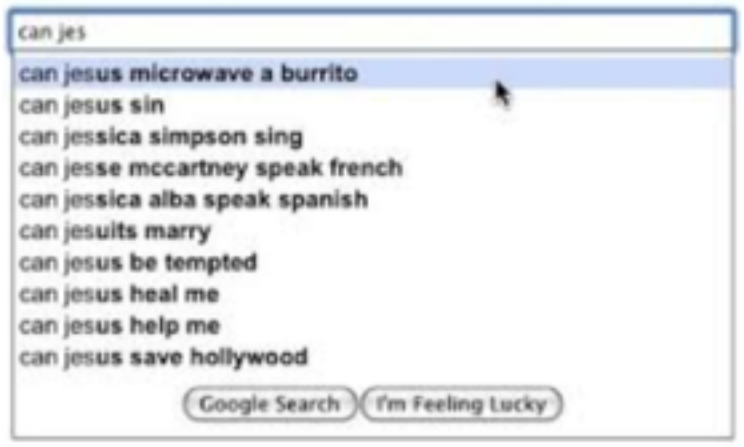
Knowing your verbs gives a good sense of whether the game is likely to have technical challenges, but also whether it's likely to create cognitive ones as well. A complicated game is not necessarily a bad or a good thing, but a verbally complicated game is usually bad. Verbal complexity means the player has to physically learn a lot of control, which in turn means staring at his hands or keyboard rather than the screen, which detracts from immersion.

Sometimes you can get away with verbal complexity if your verbs have an internal and guessable logic. Fighting games use a lot of button patterns for example, but these are essentially combinatorial verbs constructed from simpler verbs (three face buttons etcetera) that have a high degree of predictability. When the player gets that, the game is not as overwhelming as it initially appeared to be. The same is true of music games, dance games and other physically active games.

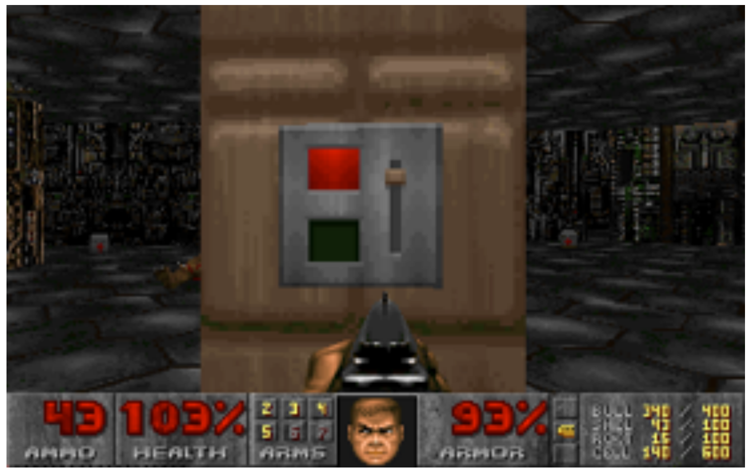
Ask yourself can you reduce the verbs intended for the game while retaining the same screen-side input. If the answer is no, can you figure out a strong guessable logic to hold all your verbs together? If the answer is still no, stop. Your verbs are a problem.

Q4: Are The Actions Extensible?

Question Four: Are the actions extensible?

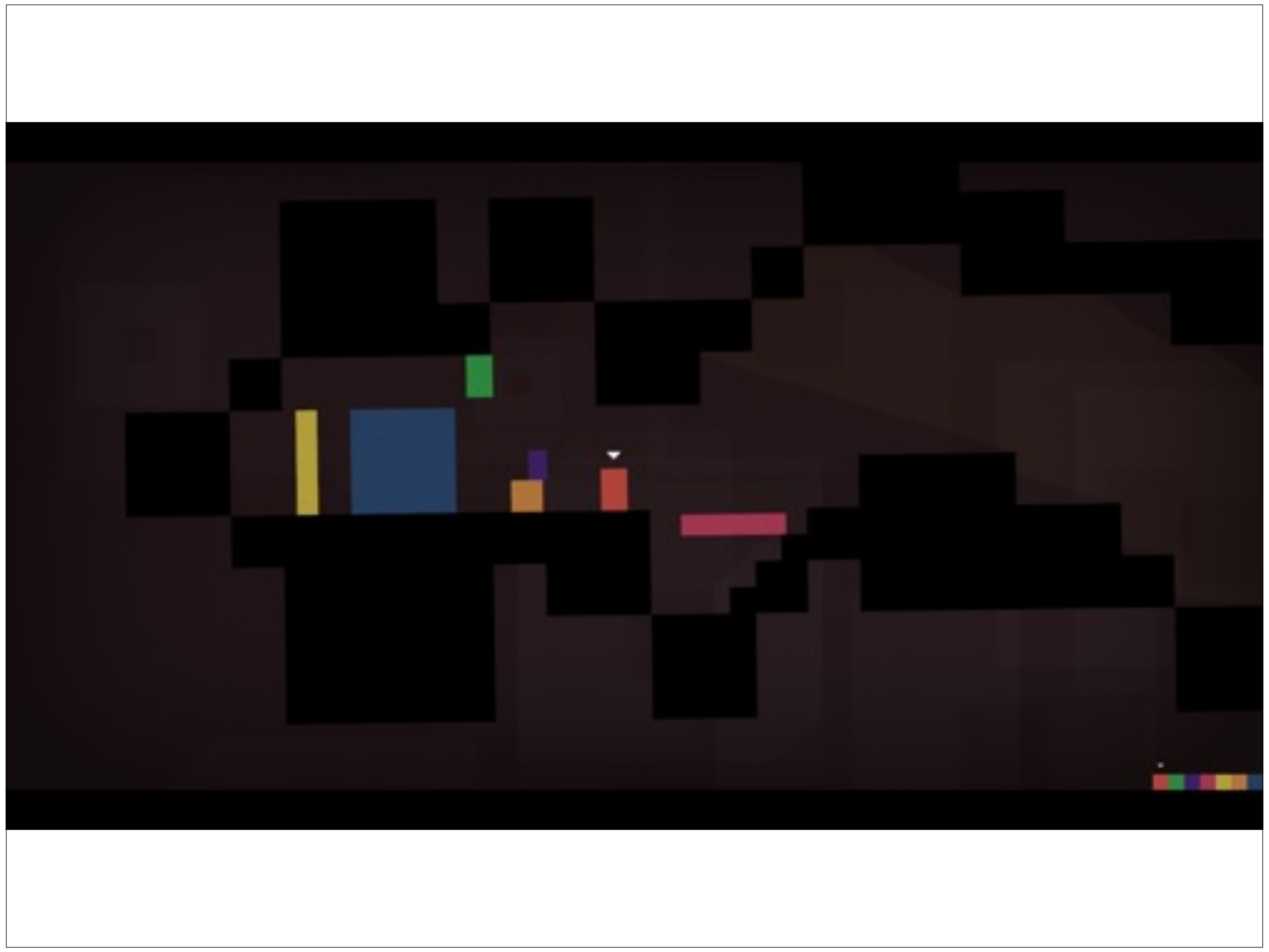

can jes

can jesus microwave a burrito
can jesus sin
can jessica simpson sing
can jesse mccartney speak french
can jessica alba speak spanish
can jesuits marry
can jesus be tempted
can jesus heal me
can jesus help me
can jesus save hollywood



While verbs happen on this side of the screen, they can cause three classes of input on the other side. The first is the query. The player taps and more information appears, giving him more knowledge of the state of the game but not affecting it. The second form of input is the nudge. The player trips the switch that ends the level, presses the “next” button to move on the story dialog, or that kind of thing. Nudges change the game’s state in a pre-scripted manner.

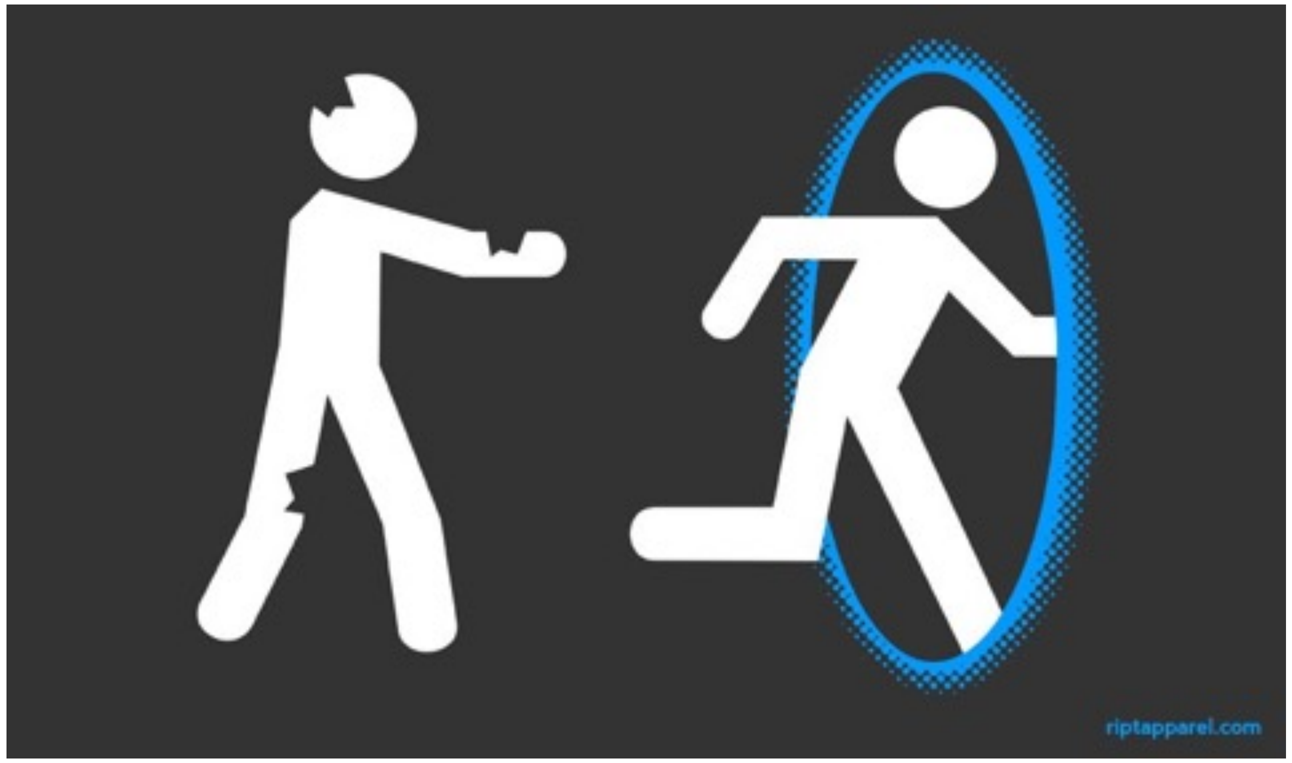
The third kind of input is the action. An action is a form of input that causes an unscripted state change. Whether you're moving an avatar from one location to another, shooting a gun or commanding troops into battle, actions matter. Actions are one half of what we generally label "mechanics" in a game context, with the other half being rules.



But what does “extensible” mean? Consider Mike Bithell’s Thomas Was Alone. In Thomas Was Alone there are essentially two kinds of action: to move and to jump. It seems basic enough, but Mike manages to extend these basic actions through the use of a query: to swap. In the play of the game you swap from one avatar to the next like Thomas to Claire to Chris to James, and they all have different properties of move and jump. Through swapping, Mike extends his basic actions and uses that capability to build ever more complex and interesting puzzles.

Actions that don’t extend are boring. It’s not necessary for actions themselves to change, but rather that the range of their utility increases. In a war game the actions are all about commanding troops, for example, but the behaviors of the different kinds of troop lends itself to complexity. In Grand Theft Auto your actions remain the same but the variation in vehicle and weapon type extends them. In Diner Dash you are always giving Flo orders, but the types of customer constitute extensions of orders, leading to a complex game.

Non-extensible actions may confer an instant thrill the first time the player uses them, but that eventually fades. Perhaps your goal is to build a short experience not intended to be replayed (Dear Esther, for example, or Gone Home) in which case the boredom factor doesn't really matter, but more often than not repeated actions with no extension run dry.



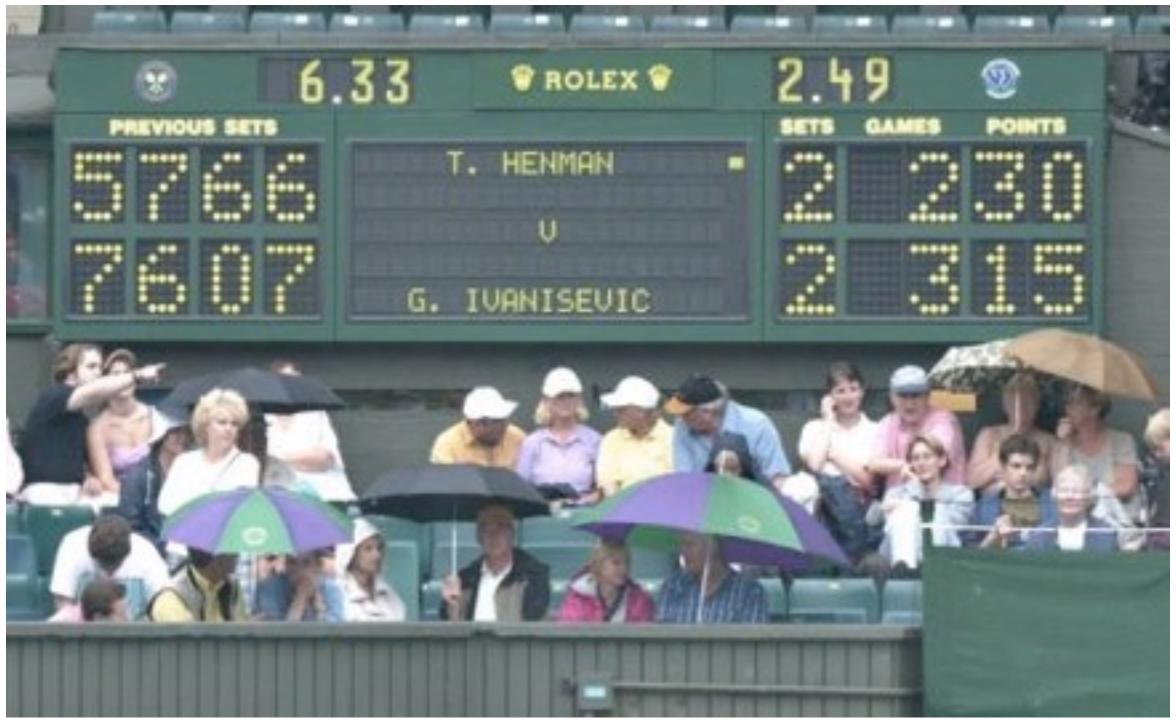
You’ll often find that this question more than any other is what leads the shape of your game to resemble the shapes of prior games. There are zillions of potential actions that we as designers can use, but a much smaller number of them are genuinely extensible. Most are one-shot, one-dimensional and fail the test. Pressing X to create portals can lead to interesting games for years to come. Not every action is so lucky.

Q5: Are There Three Resource Types?

Question Five: Are there three resource types?

In the game of tennis two players face off against each other in a battle of wits and skill over the course of hours. They serve, volley, return, play winners and try to overcome one another. Tennis is, as its best, an epic struggle and one of the finest games in the world.

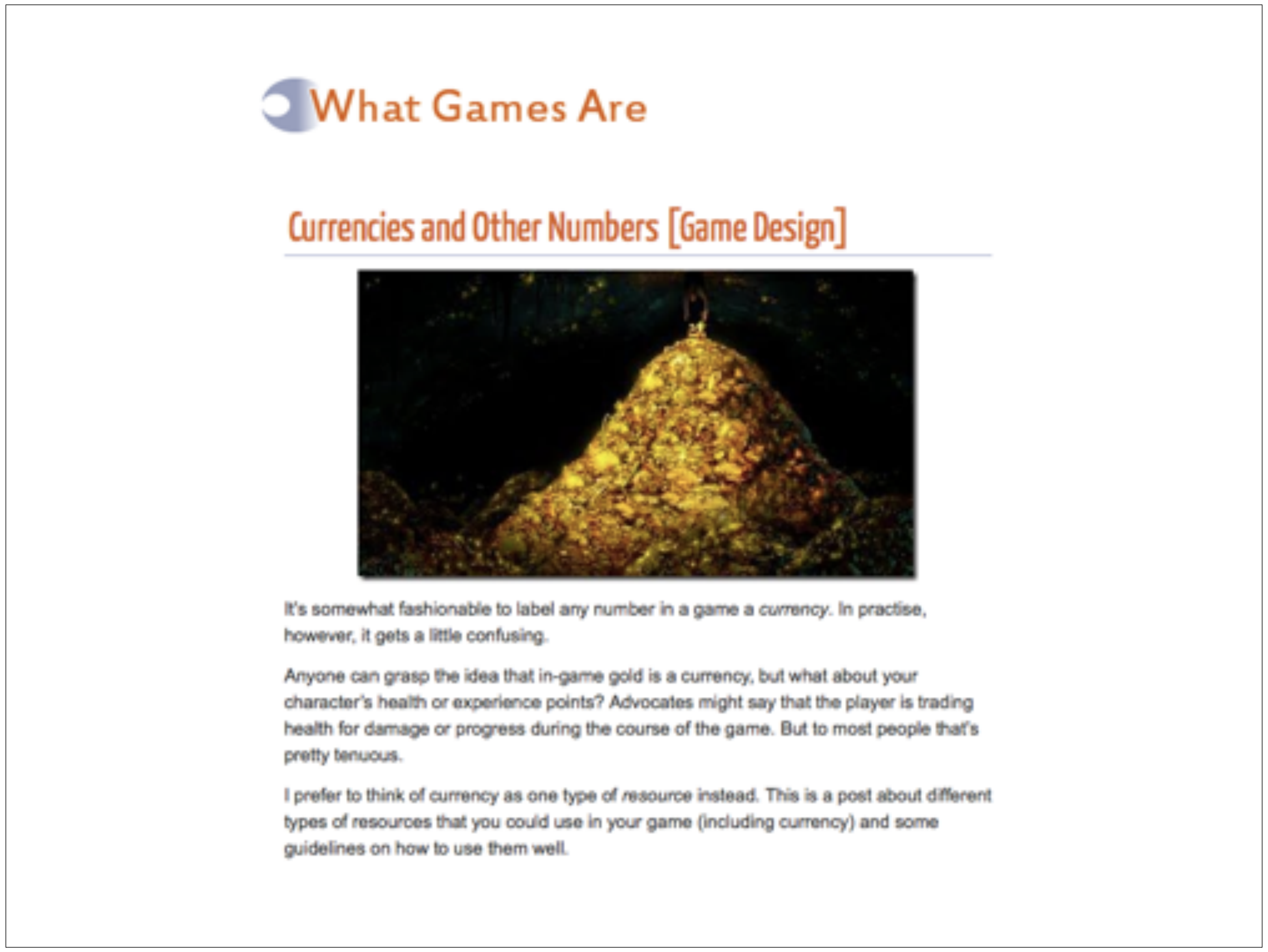
Why does it work? Well there are numerous reasons but one of them is to do with its resources. There are four key resources in tennis. The first is the ball. The ball is a currency token that the player attempts to trade by bouncing it through the opposing court and then out again, and the opponent tries to prevent that trade from being successful.



The second resource is points and their larger denominations of games, sets and matches. Points are a meter, a number that the player accumulates through successful ball trades. Unlike the ball currency, players generally never lose points, and accumulating them to a certain threshold triggers the win condition of the game.

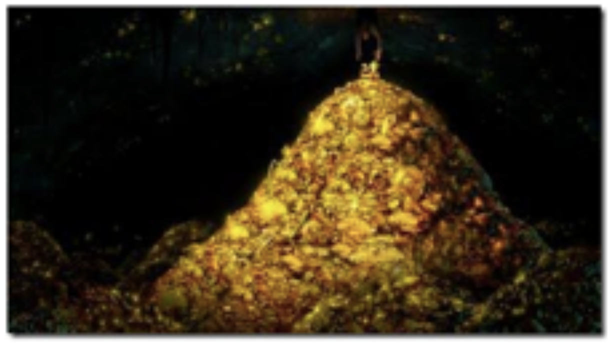
The third resource is the racquets. Racquets are tools, objects of play or weapons of war if you like. In tennis each player is permitted to use one racquet at any time during play and racquets all have to conform to certain standards (though they are not identical). The player uses her racquet to trade the ball with considerably more force than she could likely manage with her hand, so the tool empowers her play.

The final resource is court position. Most of the tactics of tennis revolve around trying to pin your opponent into a bad position, such as over on the left or the right, opening up more room to send the ball for a successful trade that he can’t catch. Ownership of territory resets after each successful trade.



What Games Are

Currencies and Other Numbers [Game Design]

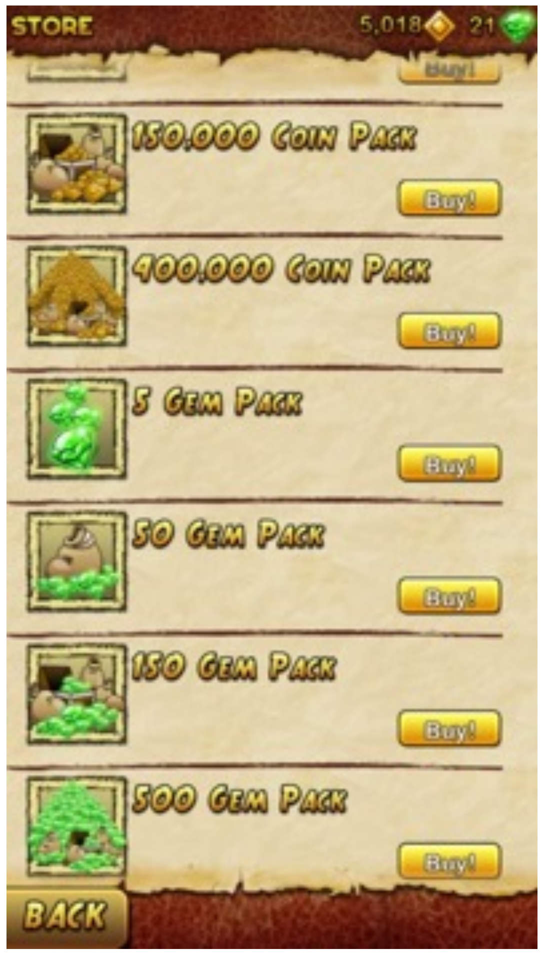


It's somewhat fashionable to label any number in a game a *currency*. In practise, however, it gets a little confusing.

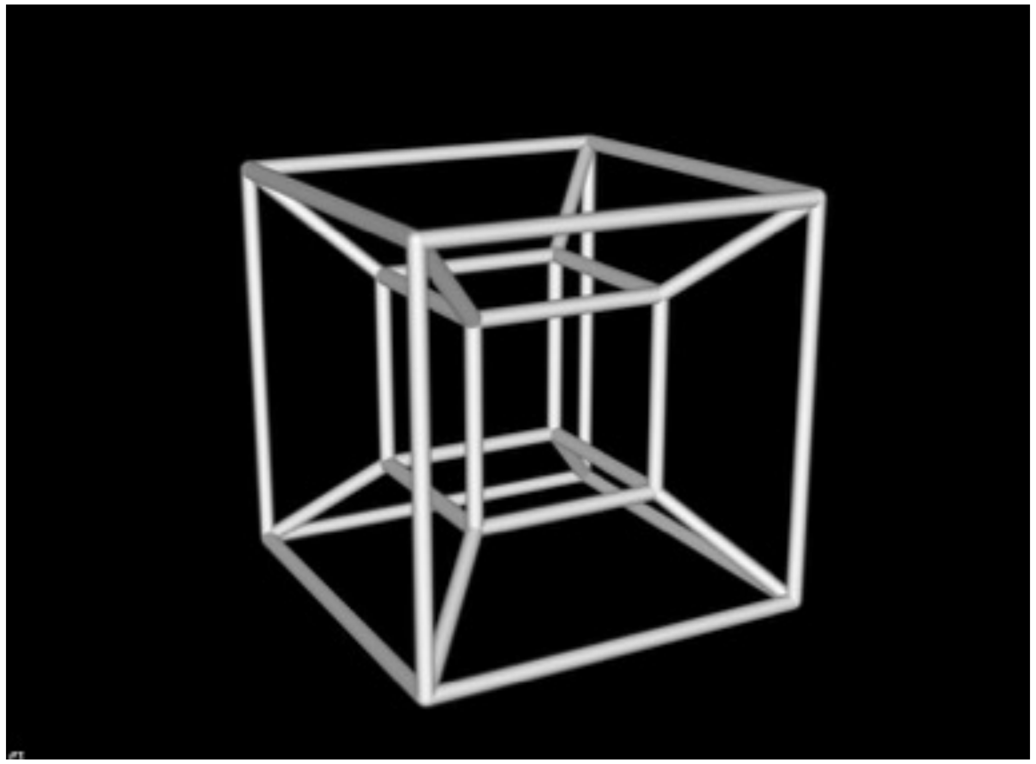
Anyone can grasp the idea that in-game gold is a currency, but what about your character's health or experience points? Advocates might say that the player is trading health for damage or progress during the course of the game. But to most people that's pretty tenuous.

I prefer to think of currency as one type of *resource* instead. This is a post about different types of resources that you could use in your game (including currency) and some guidelines on how to use them well.

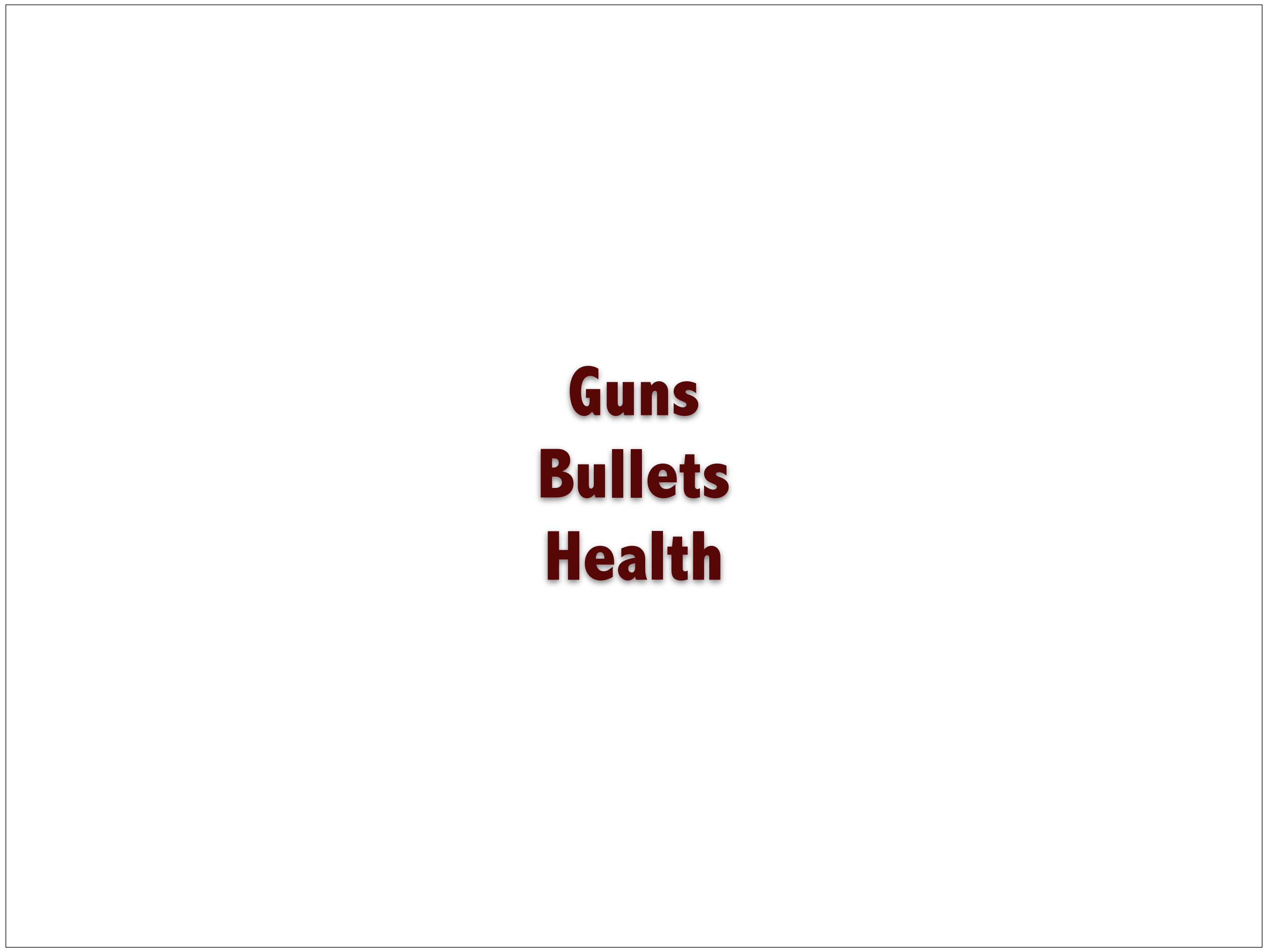
So we have currencies, meters, tools and territory. These, it turns out, are the four cardinal kinds of resource that feature in all games. But not all of them feature equally. Some games lack territory, for example, whereas others don't use currencies. However it's rare to find a successful game that doesn't have a variety of them.



I don’t mean having multiple resources of basically the same type. Having eight currencies that behave the same way might make your game economist excited about potential monetization opportunities, but players would quickly sense that they all do the same thing. That’s a boring kind of game.



Instead I mean having several types of resource. Using multiple types of resources makes play function in multiple dimensions, which is key to making the game fascinating. You don’t need to work out all your currency tables or progression charts ahead of time (indeed, doing so is usually bad practice) but you do need to have some sense of what resource types you’ll use.



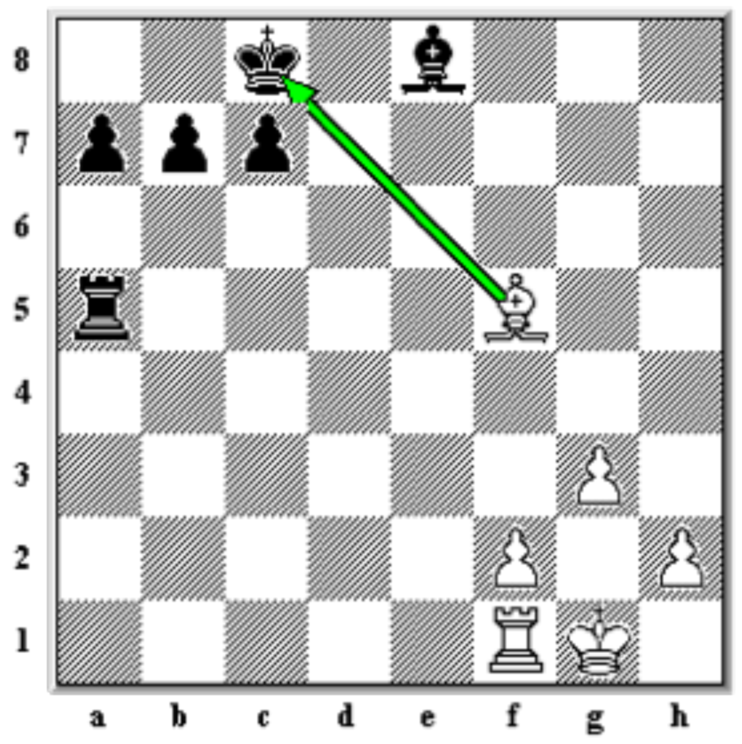
Guns
Bullets
Health

I often make it as simple as writing three words, such as “guns, bullets, health” or “land, spells, life points” or “frisbee, goals, position”. Try to keep it that simple, but think dimensionally.

Q6: What’s The Defining Rule?

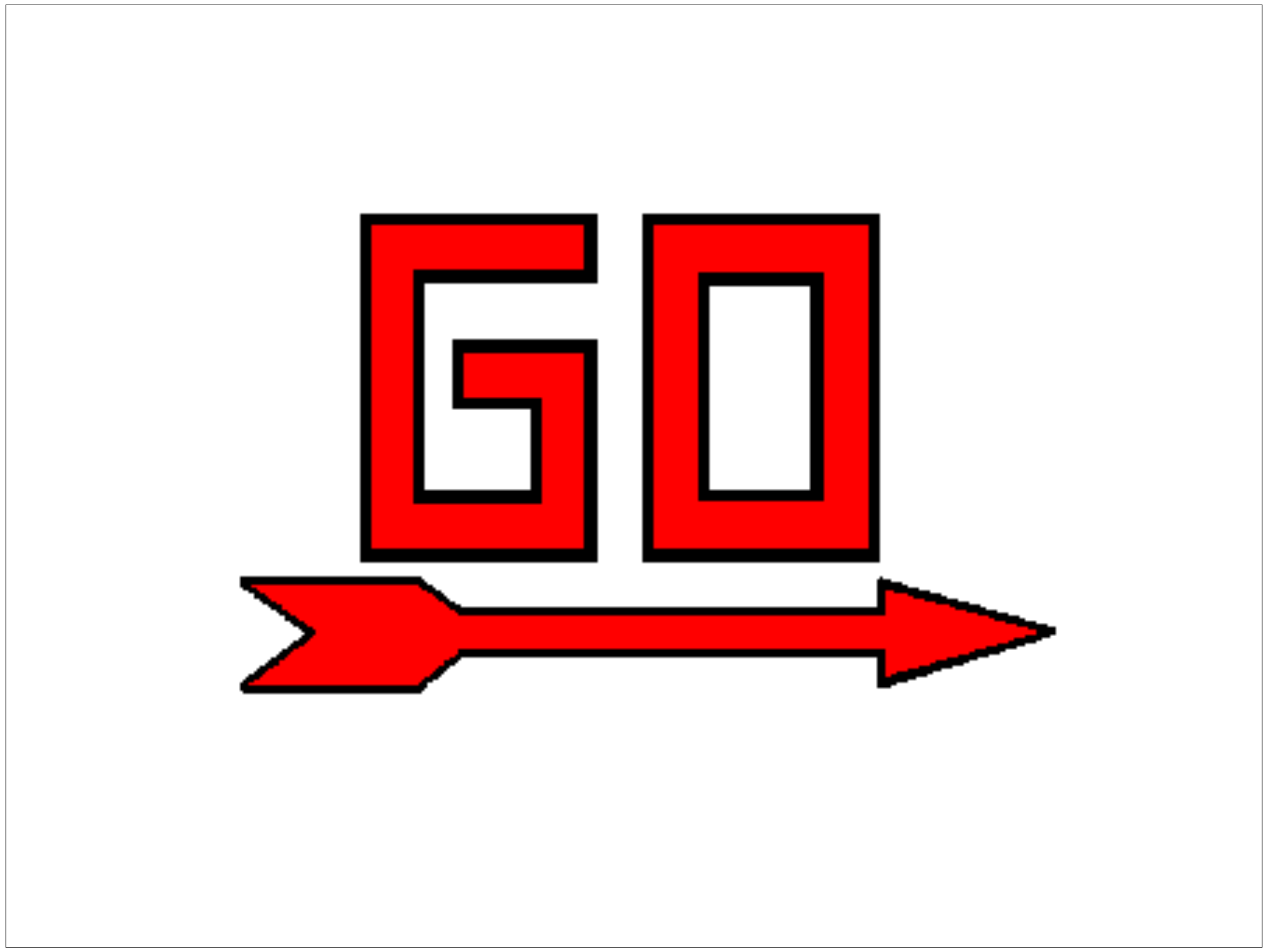
Question Six: What’s the defining rule?

Video games are different from other games in that most of their rules are not explicitly known by the player. They’re a black box. The entire rule set of a video game is rarely known to players but they can play it nonetheless. However one interesting aspect of many successful games is that they have a defining and knowable rule.



For example survival horror games use deliberately awkward controls in order to engender a feeling of powerlessness. That's a defining rule. So is the handball rule in soccer or the rule for no forward passing in rugby, or the rule for check in Chess. Animal Crossing uses time as a defining rule, making players wait. Ico uses partnership as a defining rule, limiting what the player can do by making him consider Yorda.

The temptation of the game designer is to give the player more power, to make better guns, bigger splashes, louder bangs, to include more stuff, more choice and so on. He's aiming to get to that "series of interesting choices" moment, but adding more choices doesn't make them interesting. Rather it's adding choices and a defining rule that leads to a great game. A defining rule creates trade-offs.



You can have more than one defining rule of course, and during the process of development you may well figure out many rules to add or subtract. My point, however, is that you should begin the game with a “defining rule” mentality and nail down one hard limit that will bind the game.

Q7: What Are You Testing?

Question Seven: What are you testing?

GRIM FANDANGO
the EPIC Tale of CRIME and CORRUPTION in the LAND OF THE DEAD

I often think of games as either activities or economies. The player is either trying to solve an obvious problem by performing good skill actions, or trying to solve complex problems by performing rudimentary actions. Grim Fandango, for example, is about working out puzzles by applying the right resources in the right order, but its interface is pretty basic. Resident Evil 4, on the other hand, is no test of strategy. But it’s tough on your reflexes.

Many games mix the activity and economy, however one tends to be dominant while the other is recessive. This is important because the dominant of the pair determines what the game is testing. All games test something, so you need to be able to say what that is.

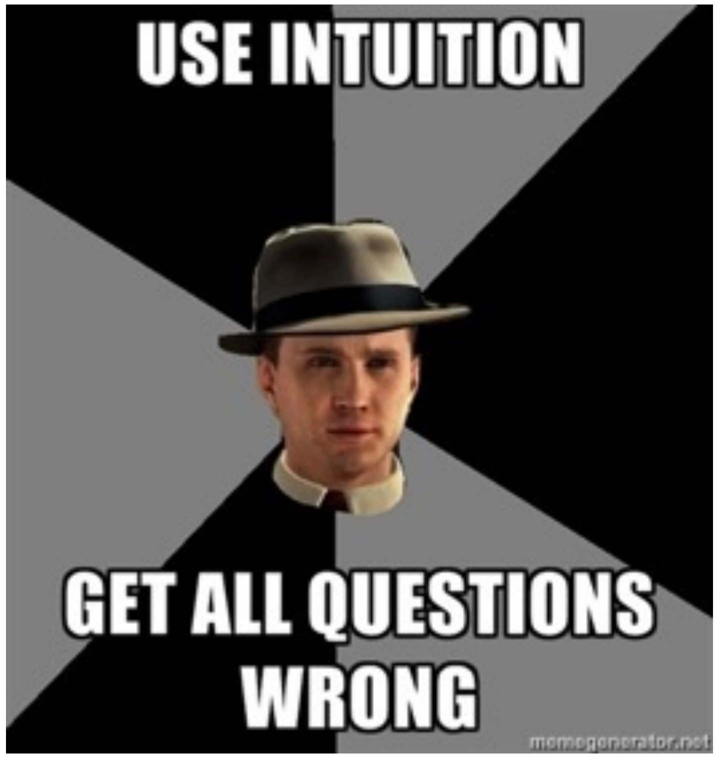
I Want To Test…

Game tests doesn’t need to be longwinded. It’s “I want to test the player’s ability to sort shapes” or “I want to test the player’s ability to make complex jumps” or “I want to test the player’s ability to make words” or “I want to test the player’s ability to organize armies” or “I want to test the player’s ability to manage resources” or “I want to test the player’s ability to solve math puzzles” or “I want to test the player’s ability to repeat sounds they hear accurately” or “I want to test the player’s ability to score goals” or “I want to test the player’s ability to shoot demons”.

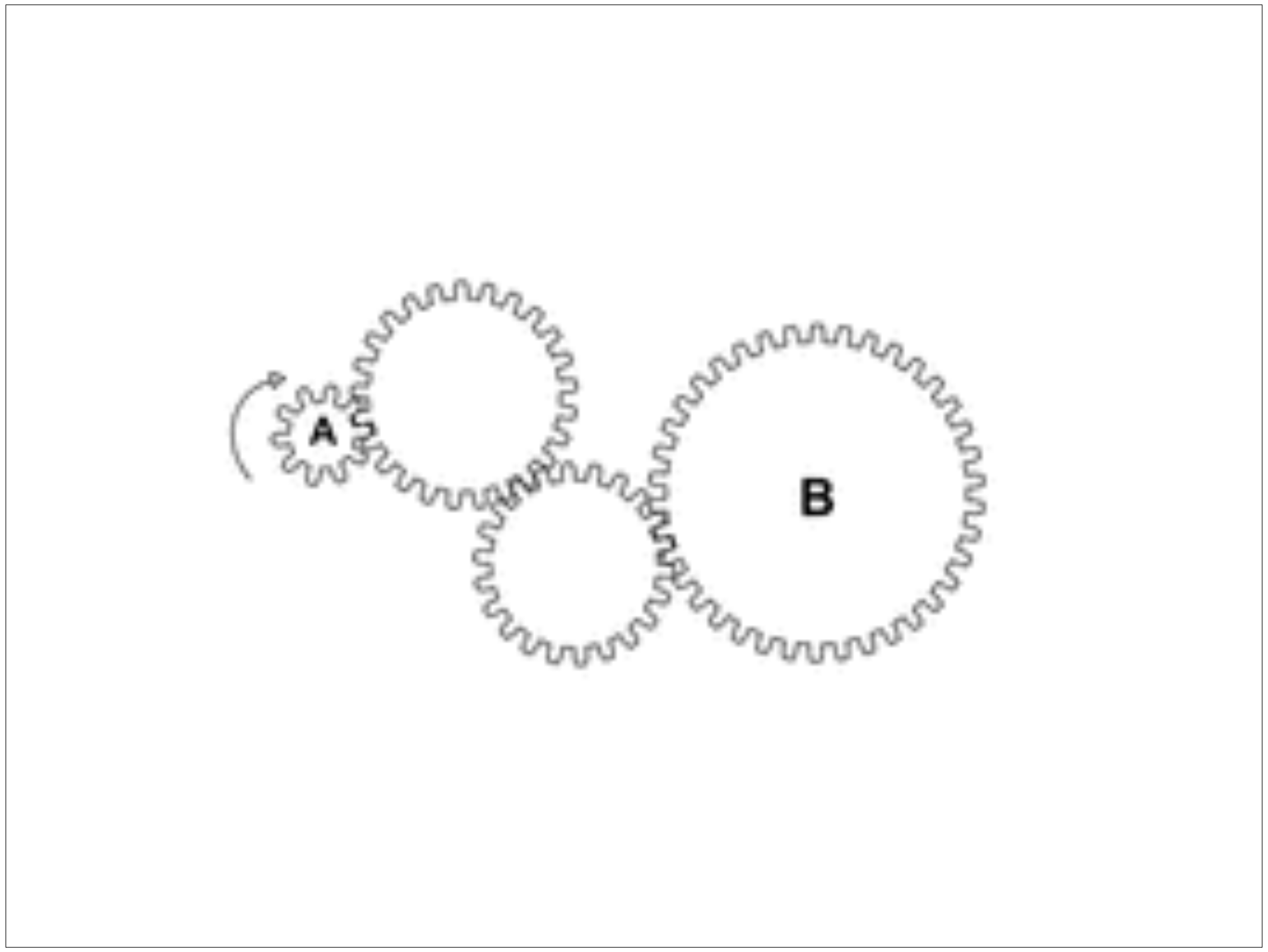
You want to, in other words, express what the primary challenge of the game is, the problem that the player will keep encountering over and over. And just as the actions of the game need to extend, so does the game's test. It also needs to make sense with all the other questions, like the verbs, the resources and the defining rule. So think of a good one.

REDUCTIONISM
THE ULTIMATE FORM OF SARCASM
PLEASE LET THESE PLANT GENITALS INDICATE MY REMORSE FOR FORGETTING THAT EARTH HAS GONE AROUND THE SUN FIVE TIMES SINCE WE AGREED TO LEGAL PAIR BONDING.

You should know that answering this question will feel very reductive. It does reduce the game to instrument and operation and tasks, but that’s unavoidable. Remember how I said that game design is about removing waste? Reducing a game to its bare bones is how you achieve that. That’s why my book is called “raw” game design.



You should also be aware that not all tests are made equal. There are plenty of bad tests. There are tests whose solutions are opaque and leave the player permanently guessing. There are also tests that rely on intuition, culture, tolerance or story that also often don't work.



The best tests tend to be expressible in mechanical terms. Focusing on making sure the core test of the game is mechanical is one of the best ways of avoiding falling into the trap of making a game opaque. Get it right early and you'll literally save yourself years of pain.

1: “Raw” Game Design

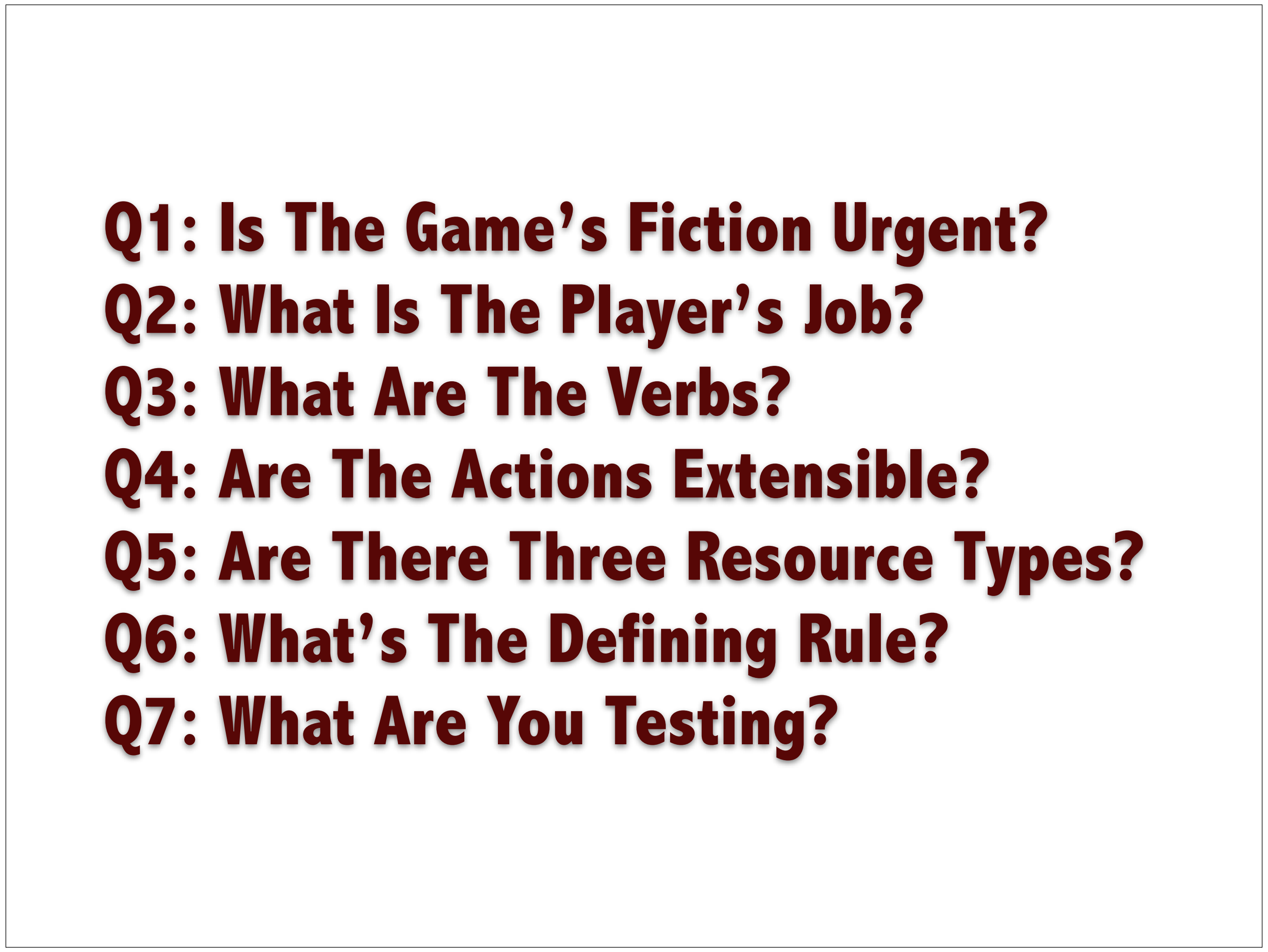
2: Delicate Beginnings

3: The Awkward Part

4: Seven Questions

5: Wrap Up

And so to conclude.



Q1: Is The Game's Fiction Urgent?
Q2: What Is The Player's Job?
Q3: What Are The Verbs?
Q4: Are The Actions Extensible?
Q5: Are There Three Resource Types?
Q6: What's The Defining Rule?
Q7: What Are You Testing?

So you have these seven questions: Is the fiction urgent? What's the player's role? What are the verbs? Are your actions extensible? Are there three resource types? What's the defining limit? What are you testing?

Great. What should you do with them?

I recommend that you create a one-page table that lists all of these questions and is included with any other materials that you typically generate. If yours is a pitch-doc sort of studio, include this page. If you're prototypers, place this page on your JIRA (whatever) and set a task that it be answered and reviewed. If you're into prototyping and production splits, set this page as a task on day one. Demand it be answered.

The point of doing such an exercise is not to get the answers 100% correct on first try. It's to make you think, to anticipate and to realistically assess what your grand ideas imply. Answering the seven questions faithfully makes you see where there are many gaps, and thus how you might fill them. Even if you don't stick to the answers over the course of development, making yourself think earlier rather than later will increase your ability to anticipate pitfalls.

Thank you!