Good morning and welcome to this year's Production Keynote, "Working Hard and Having Fun," which is a talk about the fast, loose and somewhat unusual way that Naughty Dog makes the commercially successful and critically well-received games that we make.

I'll be talking about *Uncharted: Drake's Fortune* along the way, which we shipped last autumn. It was our first game for the PlayStation 3, and we were very pleased by the reaction that it got from both the gaming public and the press.

My name's Richard Lemarchand and I was the Lead Game Designer on *Uncharted*. I got my games industry break in the early nineties, working for the British office of MicroProse, in South Gloucestershire. I've made character action games the main focus of my career, and these are the titles that I've worked on so far.

I feel very honored to be speaking here today - even though I've been working in California since the mid-90s, I've kept close ties with the British development industry, and it's great to see so many old and new friends in the audience.

A few months ago, the European Union became the largest market in the world for the PlayStation 3. As videogames mature as both a mass-entertainment and literary form, I think that Europe's vibrant cultural perspective, our technical and academic ability, and the quality of our media and entertainment industries remain tremendously important to the international development scene.

Grand Theft Auto IV, MotorStorm, GRID, SingStar, Fable 2, Buzz and *LittleBigPlanet* are all testament to the phenomenal creativity and production ability of Britsoft developers.

So who are we? Well, Naughty Dog is a small game studio of about 90 people, based in Santa Monica – which is a little seaside town bolted onto the side of Los Angeles.

The company was founded in the mid-1980s by Jason Rubin and Andy Gavin, who had their first smash hit with *Crash Bandicoot*.

We went on to make the *Jak and Daxter* series of games which were extremely successful on the PlayStation 2, and we were bought by Sony Computer Entertainment in 2001. We've sold over 35 million games worldwide, which adds up to over a billion dollars in retail sales.

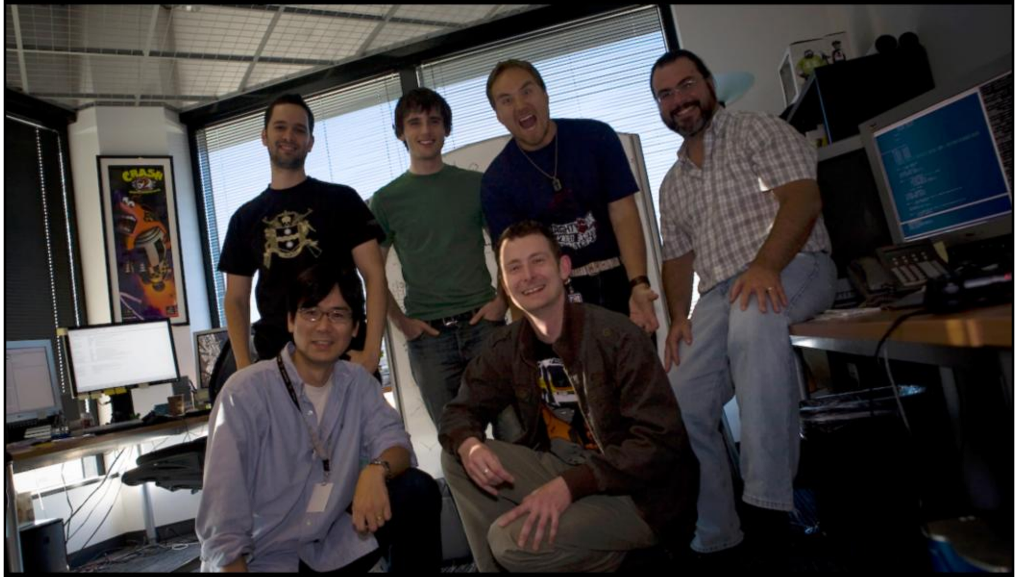
The studio's now headed up by long-time Naughty Dogs Evan Wells and Christophe Balestra who are our Co-Presidents. Evan is a game designer that I first worked with at Crystal Dynamics in the 'nineties, and Christophe is a programmer and a veteran of the French demo scene.

I work alongside quite a few Europeans, and in fact a full quarter of the Naughty Dogs are from somewhere other than the United States.

I must admit that I feel like a bit of a fraud, standing up here in front of you today to give the Production Keynote at Develop 2008. Like I said, I'm not actually a producer – I'm a game designer. And to make matters worse, we don't actually have anyone at Naughty Dog who has the job title of producer.

That's not to say that we don't have respect for production as a discipline, though, and in fact we do a lot of production work at Naughty Dog. We spend more than our fair share of time making lists of jobs that need doing, talking to people about how who's going to do them and how we're going to get them done, and following up to see if everything's going OK.

Produced by Artisans



But rather than having dedicated producers, each game we make at Naughty Dog is produced by the people working on it – by the game designers, team leads, programmers, Game Directors, and ultimately, by our Co-Presidents as well.

In fact, anyone who wants to step up to bat and take responsibility for an aspect of the game is encouraged to do so, and is empowered to simply start organizing. No-one at Naughty Dog will ever get told off for trying to get something done – we might gently mention that now isn't the right time to tackle something and redirect that person's energy elsewhere, but we never, ever slap down that will to get busy.

No-one Only Does Management

ND KENNEL-2007
CASE FILE_PRIOR CONV.
AMMIE.P AMY.H ANDREW.C BEHROOZ.R BENSON.R BOB.R JOHN.K JONATHAN.L JON.L JOSH.S JUDD.S
BORIS.B BRIAN.P B.STRALEY B.SWANSON CARLOS.G CEDRIC.L JUSTIN.M KION.P LUCAS.P MALCOLM.H MANNY.K MARSHALL.R VICTORIA.F
CHARLES.D CHARLIE.T CHARLOTTE CHRIS.C CHRISTIAN.G CHRISTIAN.N MATT.M MICHAEL.F MIKE.H MIKE.Y MOHAMMAD MORGAN TROY.A
CHRISTINE.H C.BALESTRA C.DESSE CONNER.M CORT.S CORY.H NEIL.D OMAR.G OSIRIS PAL.E PAUL.M PHILIPPE.M TROY.S
DAN.L DARCY.K DAVE.S DON.P ERIC.B ERIC.I POGO REUBEN.S RICARDO.A RICHARD.L RICH.D RICK.L TYLER.B
ERICK.P EVAN.W FENQUAN.W HIROKAZU.Y INKYO.L JANE.M ROB.A RYAN.T SANTIAGO.G SERGIY.M SHADDY.S SHAN.C VAN.P
JASON.G JASON.S JEREMY.L JOHN.B JOHN.G SWAMI.N TATE.M TEAGAN.M TOBY.T TRAVIS.M VICKI.L

Something that might surprise you even more is that we don’t have anyone at all at Naughty Dog who only manages, not even the company presidents. When I turned up for my job interview at Naughty Dog, I walked into Evan’s fancy presidential office to find him laying out the signal regions that control the level loading for Jak 3 - a job that some might consider menial, but that Evan saw as a great way for him to help build out the game.

This is a fundamental thing about our team culture: that everyone at all levels of the company works directly on building the game.

No-one Only Does Management

It’s fundamental because working like this helps us stay very focused on quality and fun, since the people with the responsibility are the creative people actually making the game. No-one ever signs off on an asset or a feature that isn’t up to scratch, just because a piece of paper says it’s time to do so.

It also means that the discipline leads – the people in charge of each department, of course - know how long things should take the people that they’re managing, because they’re doing the same work themselves. They know what kinds of dependencies exist in the work, and what kind of problems might arise.

No-one Only Does Management

Now it might sound difficult to be expected to both manage and contribute directly to a project, and I won't lie to you – one of our ongoing struggles is finding a good balance for our leads in the time they split between management and building the game themselves. But I bet it also sounds ideal to anyone in the audience who's been promoted up and away from doing the creative work they love.

Admittedly, it might only be possible for us to work this way because the majority of the developers at Naughty Dog are fairly senior, self-starting, and very passionate about what they do.

Give People Responsibility

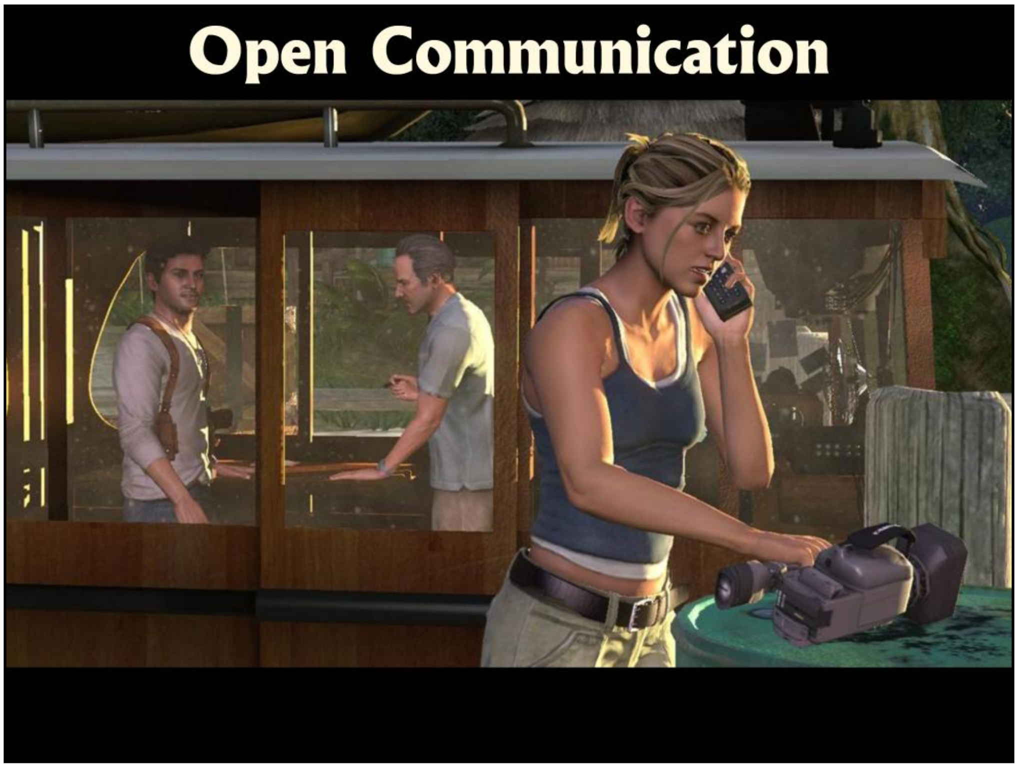
But it’s been my overwhelming experience in managing people over the years that by investing people with direct responsibility for getting their work done on time and to great quality, you bring out the very best in them, and they’re much more likely to outstrip even the highest expectations that way.

So even though we’re very management-light, we take pride in how healthy our production processes are at Naughty Dog, and in fact we’ve never missed a shipping deadline in the history of the company.

Open Communication

Another thing that we encourage everyone in the studio to do is to contribute ideas and constructive criticism to the design of what we're working on, and to our evolving production processes. Our company presidents lead in this regard by having an "office door always open" policy and people know that they can always approach Evan or Christophe to talk about anything that's on their mind.

It takes a surprising amount of energy to make this ideal work, and every day we try to keep a mindset of not assuming that the problems we see in the game are necessarily getting taken care of, and if we do see something that we think is broken, we get off our butts and go and find someone to talk to about it.



Open Communication

The people on the receiving end always take time to listen patiently to what their team-mates have to say, to talk through the pros and cons of their suggestions, and either figure out a plan to fix the problem, or table it for a later date.

So things that suck about the design, or things that have been implemented but that aren't working very well in the game, simply don't get a chance to survive in the version of the game that ships, because someone on the team will draw attention to them.

Face-to-Face Communication

As I've already hinted, we've found that the best primary way to communicate with each other is by talking face-to-face while we're looking at the game, rather than sending an email or IMing.

It's simply much more quick and efficient to be looking at a screen together and to be able to say "look at that, there – it's all messed up!" than it is to try and describe it in text. There's less room for misunderstanding in a face-to-face chat, and issues get worked through much more quickly.

Face-to-Face Communication

Ironically, “face time” seems to be less disruptive of creative people’s “flow state” than email is, too, perhaps because it’s easier to ask someone to come back later than it is to resist getting drawn into an email thread. Talking to someone face-to-face probably builds team spirit better than dry old email, too.

So, the majority of our game design takes place by people talking to one another informally at our desks – discussing the pros and cons of what we’re planning to do, and finding a plan that everyone likes.

Short Meetings

We try and keep our formal meetings nice and short, and only have them when we really need one.

To avoid wasting anyone's time, we make sure that only the people who really need to be there are there, and we set a time limit for the meeting, which helps keep people on-topic for the duration.

Collaboration

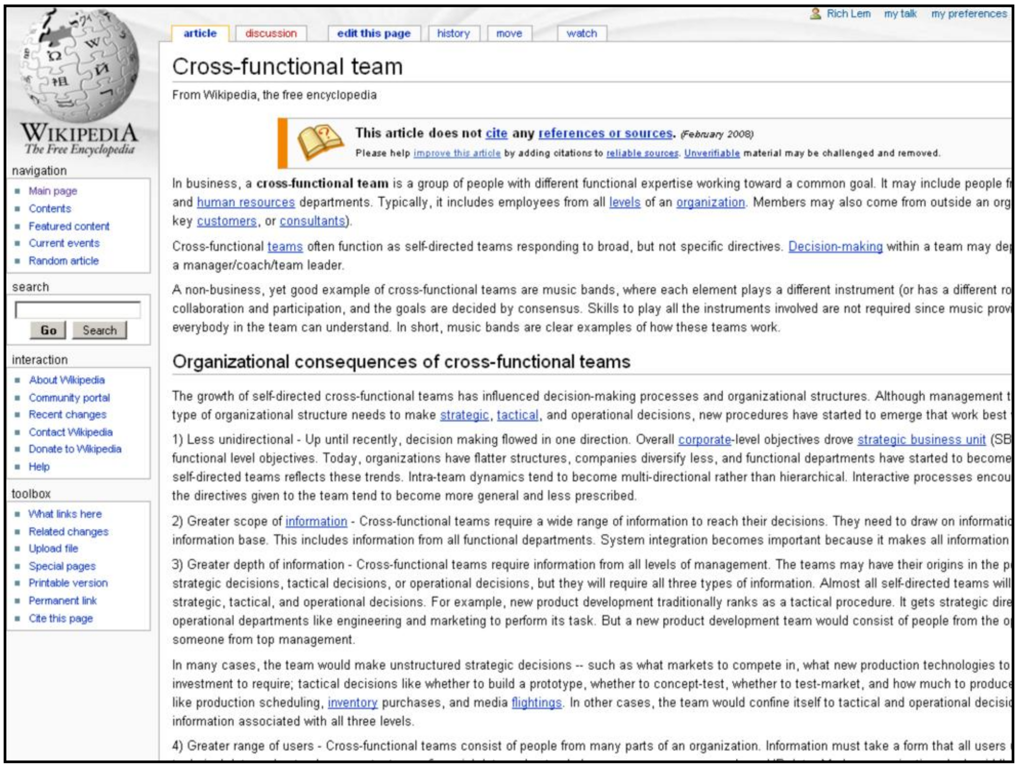
Collaboration is something that's very important to us. As you walk around Naughty Dog you'll very often see two or more people from different disciplines sitting together at a development station, working on some aspect of the game.

Working closely together like this in a sustained way over a long time was how our Lead Programmer and our Lead Animator created the fantastic control scheme that allows the hero of *Uncharted*, Nathan Drake, to move very fluidly with realistic, varied animation, while at the same time controlling in that "on the button", immediately responsive way that players love.

Collaboration

It’s a truism, of course, that a problem that might be hard to crack from one person’s point of view often becomes much, much easier when a fresh perspective is brought to bear on it.

I’ve always found that when I get out up of my chair, take a short break, and talk to people about the problems I’m facing with what I’m working on, I immediately get a much better handle on the scope of the problem , and on the possible solutions. I also usually end up finding someone else having a problem that I can help them with, too!



Cross-functional team

From Wikipedia, the free encyclopedia

This article does not cite any references or sources. *(February 2008)*
Please help improve this article by adding citations to reliable sources. Unverifiable material may be challenged and removed.

In business, a **cross-functional team** is a group of people with different functional expertise working toward a common goal. It may include people f
and human resources departments. Typically, it includes employees from all levels of an organization. Members may also come from outside an org
key customers, or consultants).

Cross-functional teams often function as self-directed teams responding to broad, but not specific directives. Decision-making within a team may de
a manager/coach/team leader.

A non-business, yet good example of cross-functional teams are music bands, where each element plays a different instrument (or has a different ro
collaboration and participation, and the goals are decided by consensus. Skills to play all the instruments involved are not required since music prov
everybody in the team can understand. In short, music bands are clear examples of how these teams work.

Organizational consequences of cross-functional teams

The growth of self-directed cross-functional teams has influenced decision-making processes and organizational structures. Although management t
type of organizational structure needs to make strategic, tactical, and operational decisions, new procedures have started to emerge that work best

1) Less unidirectional - Up until recently, decision making flowed in one direction. Overall corporate-level objectives drove strategic business unit (SB
functional level objectives. Today, organizations have flatter structures, companies diversify less, and functional departments have started to become
self-directed teams reflects these trends. Intra-team dynamics tend to become multi-directional rather than hierarchical. Interactive processes encou
the directives given to the team tend to become more general and less prescribed.

2) Greater scope of information - Cross-functional teams require a wide range of information to reach their decisions. They need to draw on informatio
information base. This includes information from all functional departments. System integration becomes important because it makes all information

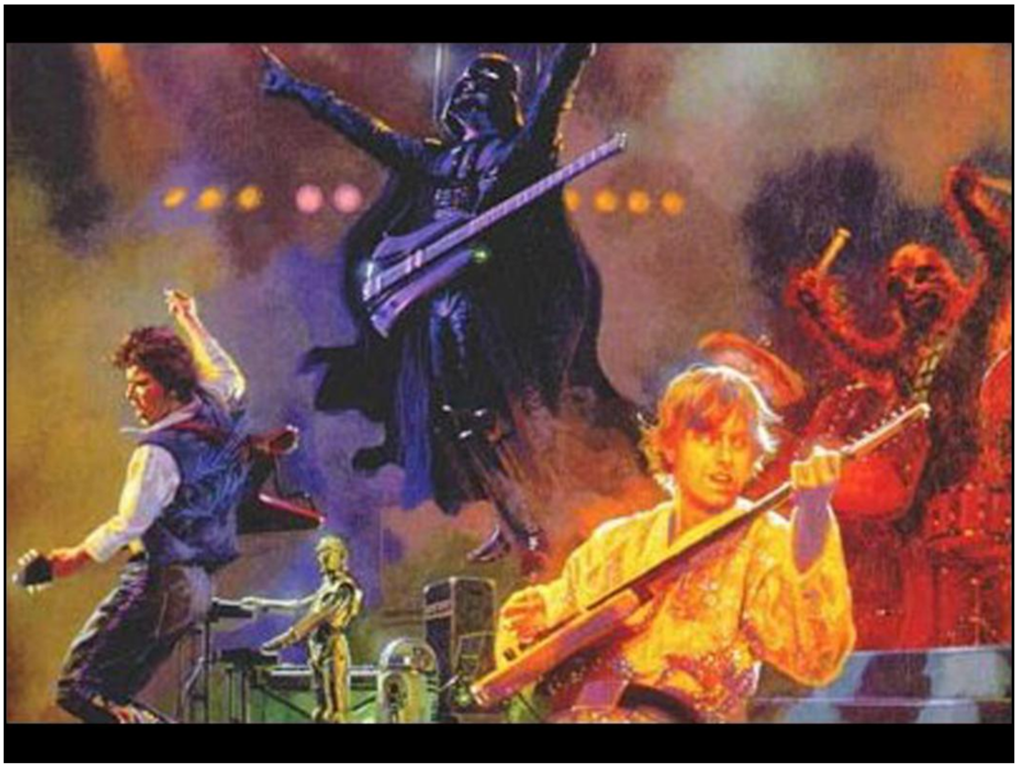
3) Greater depth of information - Cross-functional teams require information from all levels of management. The teams may have their origins in the p
strategic decisions, tactical decisions, or operational decisions, but they will require all three types of information. Almost all self-directed teams will
strategic, tactical, and operational decisions. For example, new product development traditionally ranks as a tactical procedure. It gets strategic dire
operational departments like engineering and marketing to perform its task. But a new product development team would consist of people from the o
someone from top management.

In many cases, the team would make unstructured strategic decisions -- such as what markets to compete in, what new production technologies to
investment to require; tactical decisions like whether to build a prototype, whether to concept-test, whether to test-market, and how much to produce
like production scheduling, inventory purchases, and media flightings. In other cases, the team would confine itself to tactical and operational decisio
information associated with all three levels.

4) Greater range of users - Cross-functional teams consist of people from many parts of an organization. Information must take a form that all users

I stumbled across the idea of Cross-functional Teams on Wikipedia while I was researching this talk, and I immediately recognized the concept as being relevant to our collaborative approach.

The article said that rock bands are great examples of cross-functional teams.



Each person in the band might only play their one instrument, but music provides a standard language that everyone in the band can understand, and the songs that they write are a result of collaboration, consensus and participation.

That seemed to sum up something that I really recognized about all of my good game development experiences. But then again, maybe I just like the idea of being in a rock-and-roll band… especially with Chewbacca on drums.

Allocate work to those who are passionate about it

Something we do that I think is pretty unusual for games production relates to the way that we allocate work. Wherever possible, we encourage people to take on any task that they care about – even if, strictly speaking, that task falls outside the discipline that someone usually works within.

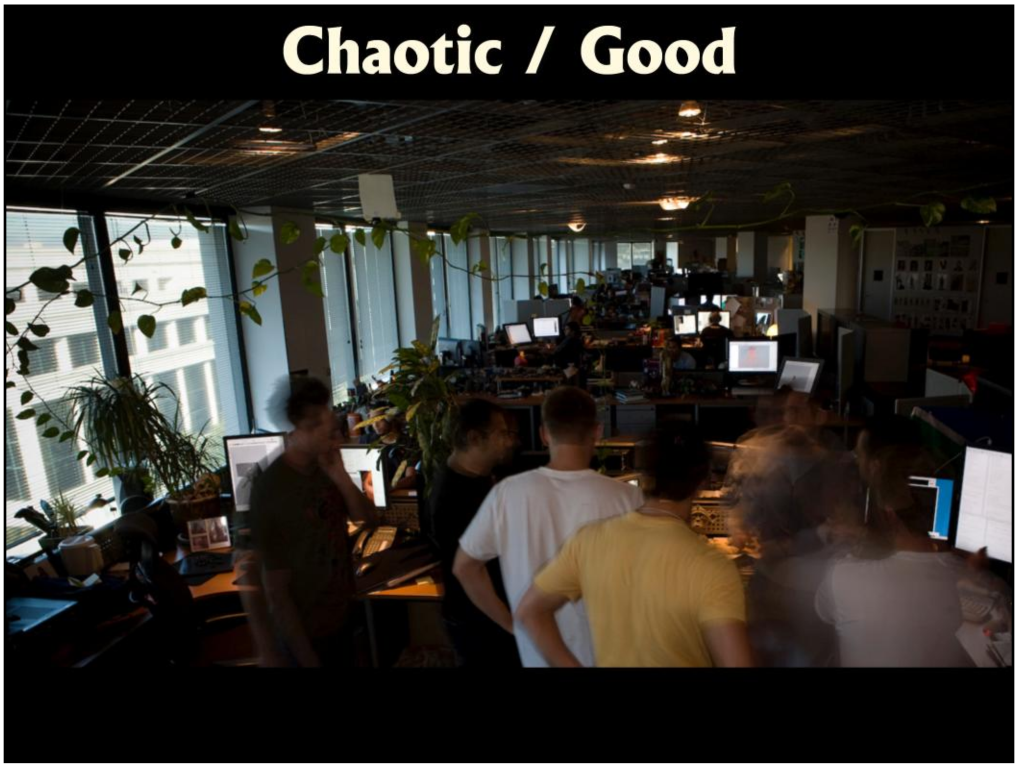
The alternative to this is assigning work to people who might not be passionate about doing a stellar job, and we've always found that greatness in a game comes directly from the passion of the people building it.

This was certainly the case with the guys who became the core of our enemy animation and AI effort.

One of our animators – who you can see here posing in the toilet for a concept art reference photo – got chatting with one of our programmers about some innovative ideas to reconcile the AI control of the enemies with a method for transitioning between their animation states, and before we knew it, the animator had packed up his desk and moved across the office to sit with the programmer, so that they could bat ideas back and forth as quickly as possible, and put together animations to try out with the emerging system.

In this sense, Naughty Dog is a "do-ocracy". A do-ocracy is an organizational structure that's emerged in places like volunteer work and the Burning Man community, in which individuals choose roles and tasks for themselves and execute them. Responsibilities attach to people who **do** the work, rather than to people appointed by some authority to have responsibility.

As you can imagine, this very naturally creates a team culture that is meritocratic. It doesn't matter if you're the most junior person on the team – if you start to work on something off your own back, and get good results, then your passion, ability and increased contribution naturally draws more and more responsibility to you. We often say at Naughty Dog that people only get formally promoted to a role if they've already been filling that role for quite some time.



Chaotic / Good

This might all sound rather messy and chaotic, and it does make for a challenging environment, sometimes. We communicate and often make important decisions on the fly; we don’t have a formalized sign-off process, and we don’t always stop to say “let’s make sure that we get everyone in the room who needs to be here”.

This can mean that people can get briefly out of the loop or get their toes stepped on and as a result everyone has to be a little more thick-skinned than might be usual. People who are precious about their work, or who respond aggressively to constructive criticism, generally don’t thrive at Naughty Dog.

Chaotic / Good

One way that we mitigate against the potential downside of straight-talking criticism is by making it a rule that we never, ever get personal.

We try to stay very focused on the facts about the game whenever we're giving feedback to each other, even in the heat of what might be an impassioned argument about some game mechanic, and we have a kind of informal contract between us all, that we won't get bent out of shape, and will keep in mind that we're all working together towards greatness.

Chaotic / Good

So at the end of the day, we believe that we’re messy in a good way. The upsides in terms of quality and, perhaps even more importantly, getting things done quickly, far outweigh the downsides in terms of the extra effort it takes to make it work, and our results hopefully speak for themselves.

Growing the Team

I promised that I'd mention how we grew our team – in fact, nearly doubling its size over the three years that we were developing *Uncharted*, because I figured that's it's something that's probably of interest to many studios right now, as projects grow and get more complex.

Of course, our hiring policies were an important part of growing the studio. We have a tendency to hire slowly and painstakingly carefully, and we only hire people that we think are great. Partway through *Uncharted*'s development, we brought on an in-house recruiter from the company Digital Artists Management to help ease the workload of filtering and screening applicants for our open positions, which definitely made the process run as quickly and as smoothly as possible.

Trust People To Do Great Work

We try and make sure that each person we hire is as much of a perfect fit for our team as possible; not only in terms of the match between their skills and the role that we need filling, but also in terms of a temperamental fit with our team. Either thing on its own isn't enough – we need people who can adapt to our often hectic environment.

However, once we *have* hired someone, we make it a point to trust them to do what we've hired them to do. Micromanagement is usually the enemy of excellence, of course, and our outlook is that if we've chosen someone because we believe that they can walk the same walk as us, we'd better trust them to exceed our expectations in the same way that we all trust each other.

Growing the Team

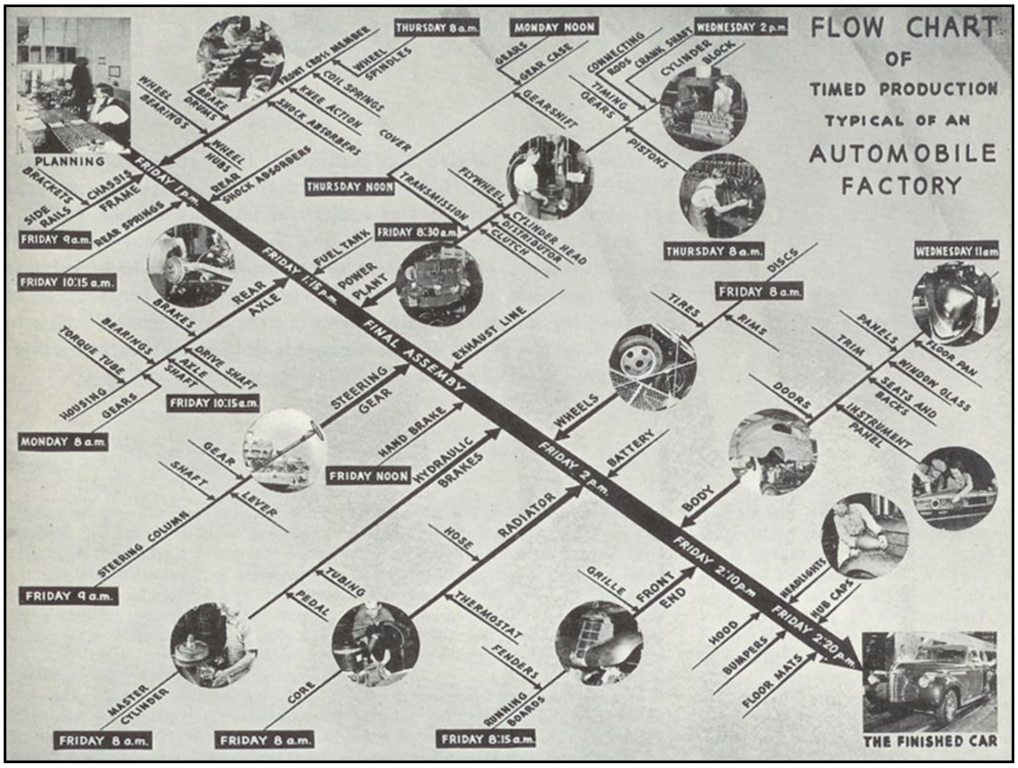
When I asked Evan what he thought was the single most important thing that we did to grow our team while keeping our studio culture, he told me that he thought it was because we'd kept intact that idea that everybody at all levels of the company works directly on the game.

If your managers are doing the same work as you in a good style that you can see is working well, it's natural that you would adopt their same good habits. All of the things that I've mentioned – the collaboration, the open communication - and the cultural things that I've yet to mention – filter down and propagate very naturally to every level of the team, including new hires.

The temptation to add layers of management as the complexity of our production grew was – and to be honest, sometimes still is – very strong, but by resisting it, we made sure that our studio culture stayed intact, even as we grew at a rapid rate.

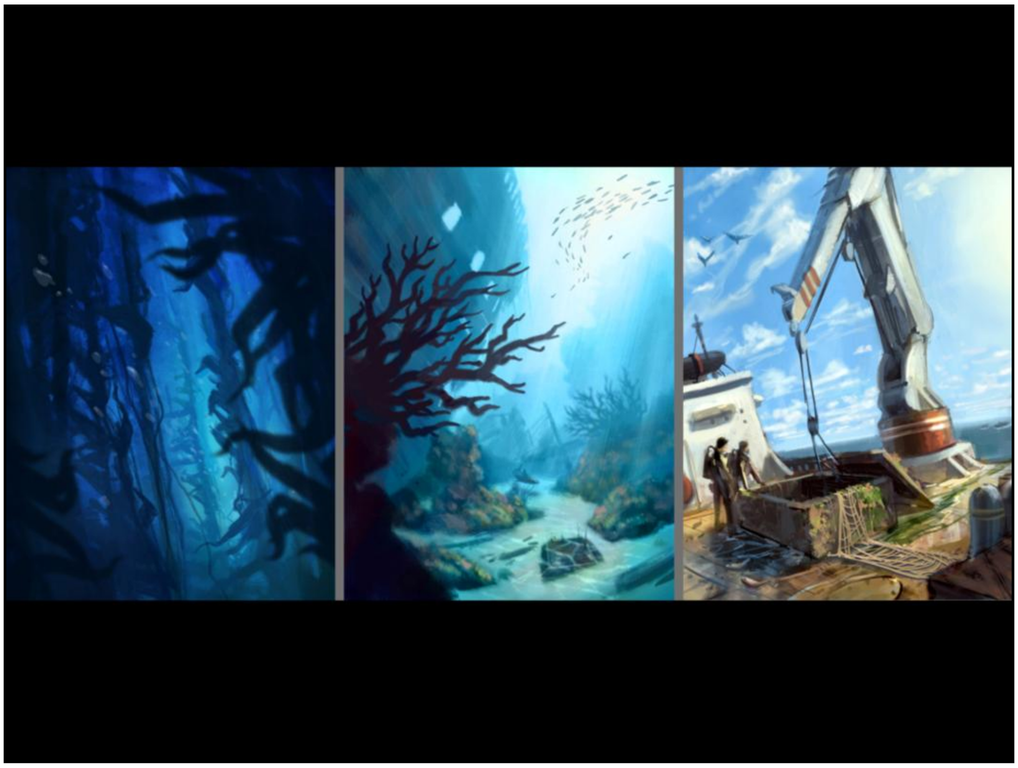
I think we'd all agree that, even if game development is now past its infancy, it's still in its difficult teenage years, and though it's rarely discussed, game production in particular is still struggling to come of age. Why is that?

One highly-placed game consultant friend of mine says that a big part of the trouble is that a lot of game studios still think they’re in the business of software development – when in fact we’re part of the entertainment business.



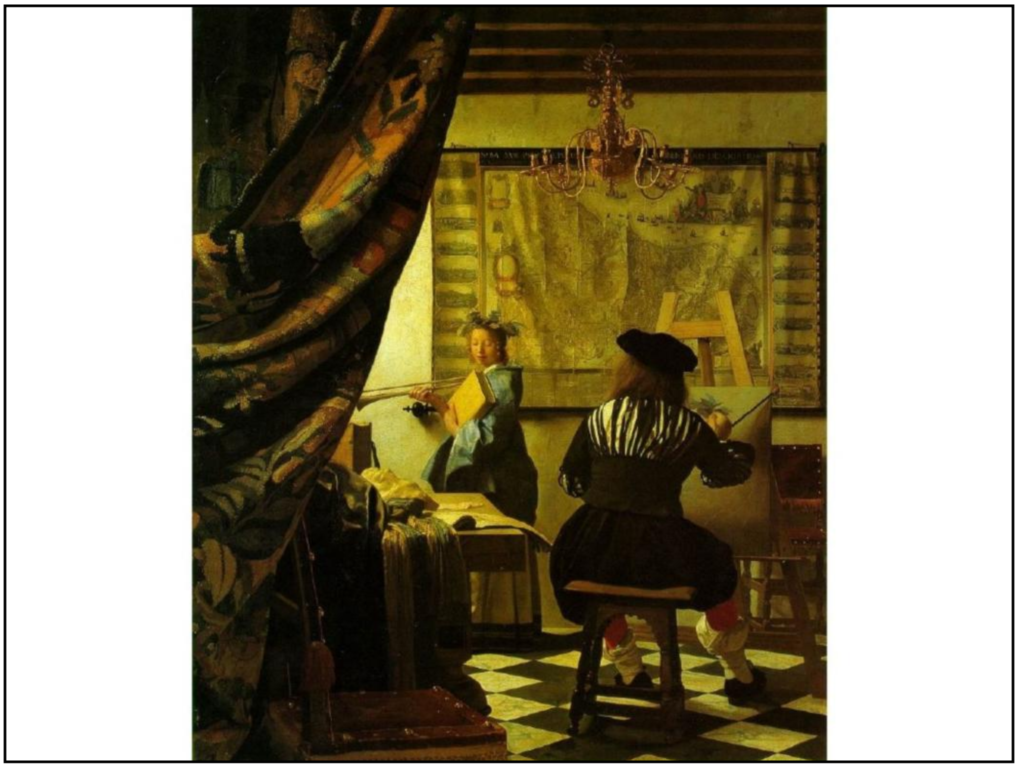
For the past thirty years of game production we've been looking towards the waterfall development methods used by pretty much every major software developer in the world, and we've been dreaming of monolithic specs that define everything up-front, so that we can implement our game on a production line like a car or a washing machine in a way that is predictable in terms of time and the numbers of people needed.

Of course predictable budgets and schedules **are** very important for the full production phase of a project, but the good planning intentions contained in a waterfall approach are often high-jacked by the uncertainties of the process of discovering what's fun about your game.



If you’re making a game in a reliable pattern with well-established game mechanics that you already know work well, then a waterfall approach could work fine.

But what about when you’re struggling to do something new? Something where you’re not yet sure how all the pieces of your game fit together – which parts will prove to be great and need bringing to the fore, which parts don’t work so well, and need back-grounding or removing from the game design altogether?



It’s my belief that the way we work at Naughty Dog has a lot in common with the way a painter makes a painting. We do preliminary sketches, we expand our original ideas, we stick our noses in books and do research, and eventually we’re ready to grab a canvas and draw an outline in charcoal.

We start to fill in the blanks with oil paint, but sometimes when we’re halfway through the painting it takes on a life of its own, and leads us in new directions that we hadn’t anticipated.

UNCHARTED NAUGHTY DOG

O 1

M G 2

W T F 3

S T F U 4

P W N 3 D 5

U R A N 0 0 B 6

L M A O R O T F 7

K T H X B Y E :P 8

IF YOU CAN READ THIS, UR EYEZ R TEH 1337. TTYL.

DEVELOP CONFERENCE 2008

You'd be well within your rights to be thinking right about now that the kind of happy-go-lucky, free-form game production love-in that I've described so far is all very well on paper, but what **about** making a game on budget and on time?

What about game engines that are late to come on line, and running out of time at the end? Well, we don't pretend to have all the answers, but let me tell you how we do it.

Clear Project Goals

On every Naughty Dog project we always like to start out with a very clear, concise summary of the goals of the project and of the game we want to make, so that we have something to refer back to throughout development, to make sure that we haven't veered off-track and lost sight of our original vision.

For *Uncharted: Drake's Fortune*, we wanted to make an all-new character-action game for the PlayStation 3, which exemplified the same kind of leap forward in terms of gameplay and of storytelling that the Jak games had been from Crash Bandicoot.

We wanted it to star a new hero with even wider appeal than our previous heroes had, and interestingly enough, in all the early game ideas we thought up, the hero was always pretty much like Nathan Drake, in terms of his look and his abilities.

We wanted Nate to be a regular guy, rather than an overpowered superhero ninja type, because we figured it would make his interactions with the world more exciting, if he was operating at the very edge of his abilities, and we also thought that it might make him more identifiable and sympathetic.

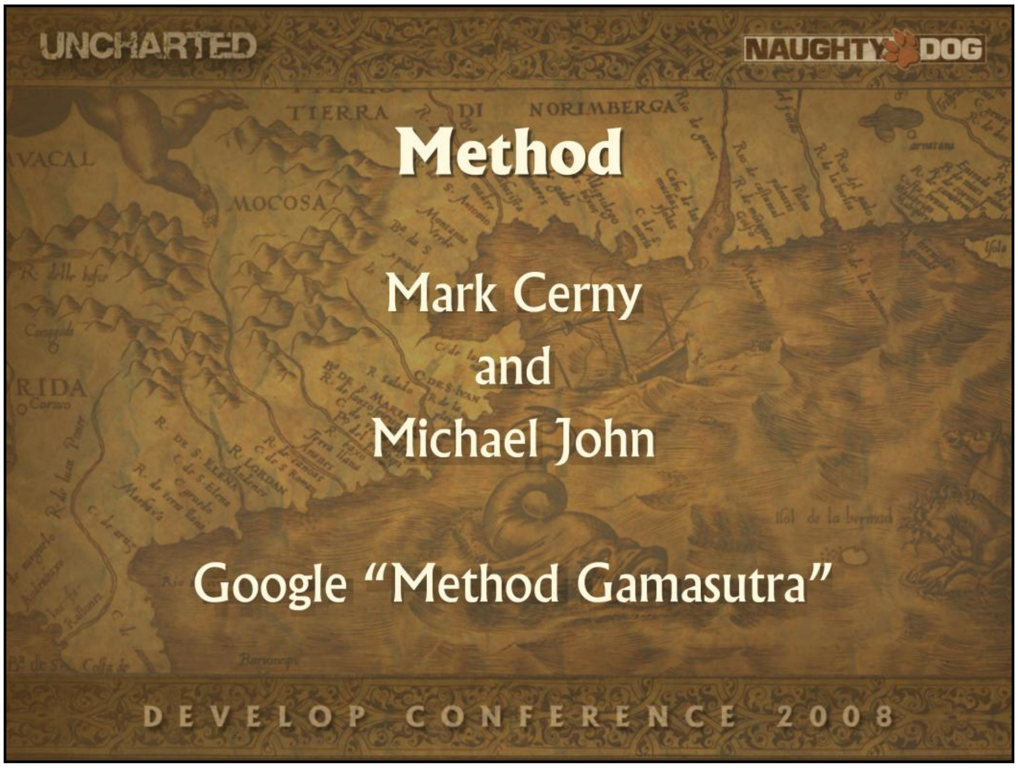
Finally, we just wanted the game to play brilliantly, and have excellent graphics, animation and audio, to showcase the abilities of the PlayStation 3.

So, with these goals in mind, we set out into pre-production.

Pre-production

Even now, the pre-production phase of a game development cycle is rarely defined clearly, and is often misunderstood and undervalued. It’s so undervalued, in fact, that many studios skip it altogether and head straight into production, and we’ve found it often to be the case that when a production is messed up, it’s because pre-production was skipped.

To help us define Pre-production it’s useful to look at something called Method, which is a game development strategy laid out by game consultants Mark Cerny and Michael John in the mid-1990s.



UNCHARTED

NAUGHTY DOG

Method

Mark Cerny
and
Michael John

Google “Method Gamasutra”

DEVELOP CONFERENCE 2008

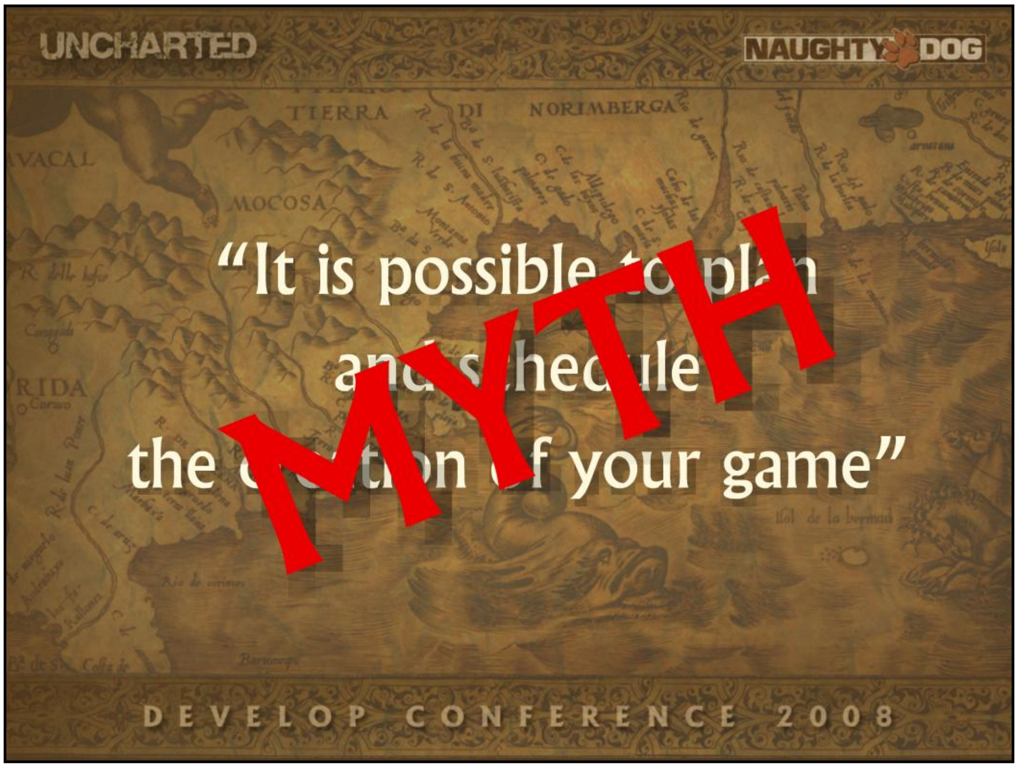
Mark and MJ are old friends and colleagues of both Naughty Dog and Insomniac, and have a string of great development credits to their names. They’re both very hands-on developers, and not just managers.

You can hear Mark Cerny describing Method in detail if you Google “Method Gamasutra”

Method sought to describe a healthy and working structure for developing video games, character action videogames in particular, which was something that didn’t really exist at the time. We don’t follow Method closely at Naughty Dog, and we often stray from it, but it’s a useful set of ideas to help us discuss aspects of good game development.

Pre-production

Method says that pre-production is *very* different from full production. It’s the phase where you’re trying to innovate in your game design and “capture lightning” – a time when you’re searching for that “Eureka” moment.



UNCHARTED

NAUGHTY DOG

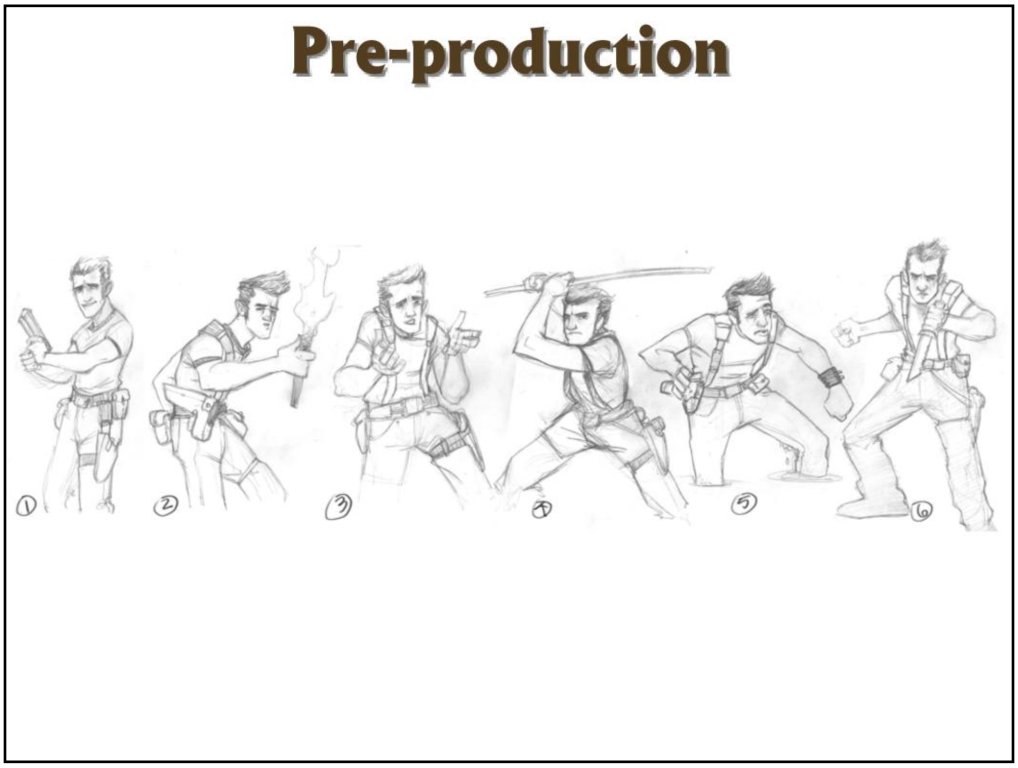
“It is possible to plan and schedule the creation of your game”

MYTH

DEVELOP CONFERENCE 2008

In addition, Method says that the idea that “It is possible to plan and schedule the creation of your game” is a myth.

Pre-production, the time when you create (as opposed to implementing) your game is a process of managed chaos – and it must be allowed to be so, or it won’t really work properly. This means throwing traditional ideas about preordained task lists, set schedules and weekly milestones out of the window.



Pre-production

Now don't panic – those ideas come back during full production.

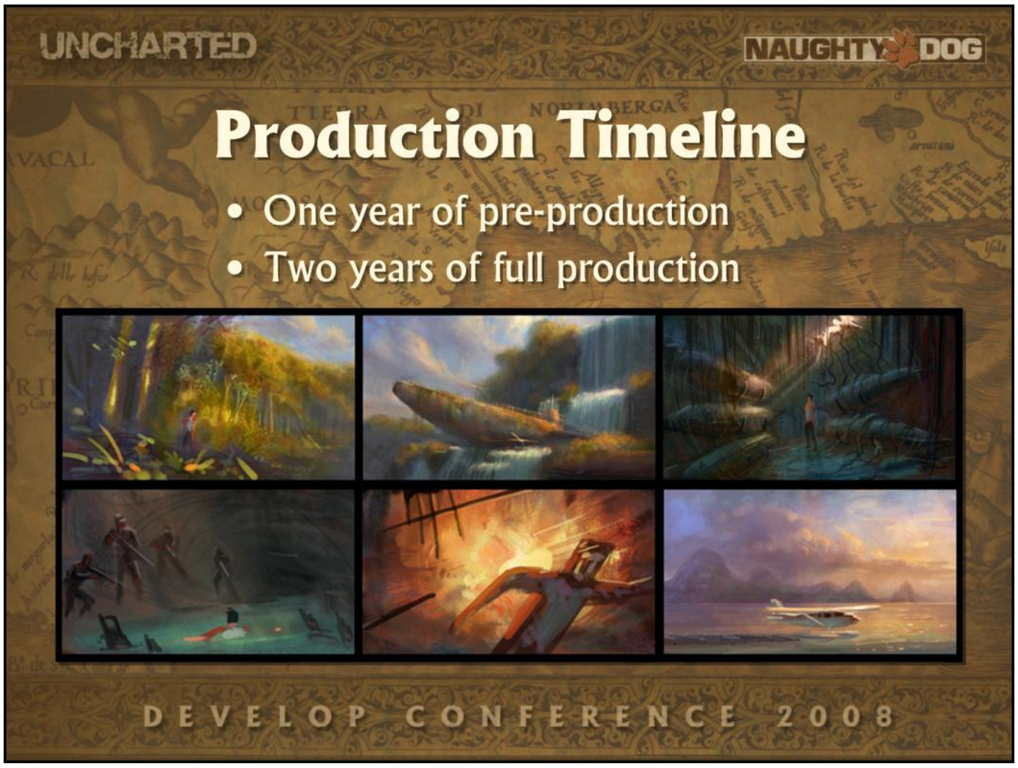
But pre-production is a time of free-form exploration and discovery – like these very early sketches of Nathan Drake show - and should have very little in the way of conventional deadlines. It's even become known within Sony Computer Entertainment of America as "sheltered pre-production," to reflect this blue-sky nature.

Of course, this also implies a high level of trust between developer and publisher - something that I'm happy to say that we definitely have in our relationship with Sony. Trust from publisher to developer - and vice-versa - is something that I think all of us in the industry should be doing our best to foster in our working lives, for all sorts of good reasons that will help our industry grow and be more successful.

Pre-production

Pre-production should be done by a core group of very senior developers – the best people that you have at your company. What frequently happens with pre-production is that it falls to just anyone who's available – often junior people who have just been hired or who can be pulled off another project.

But this core team will, in pre-production, determine everything that's important about your game, and will most likely go on to become your team leaders during Production - so they'd better be among the best people you have at your company.



UNCHARTED NAUGHTY DOG

Production Timeline

- One year of pre-production
- Two years of full production

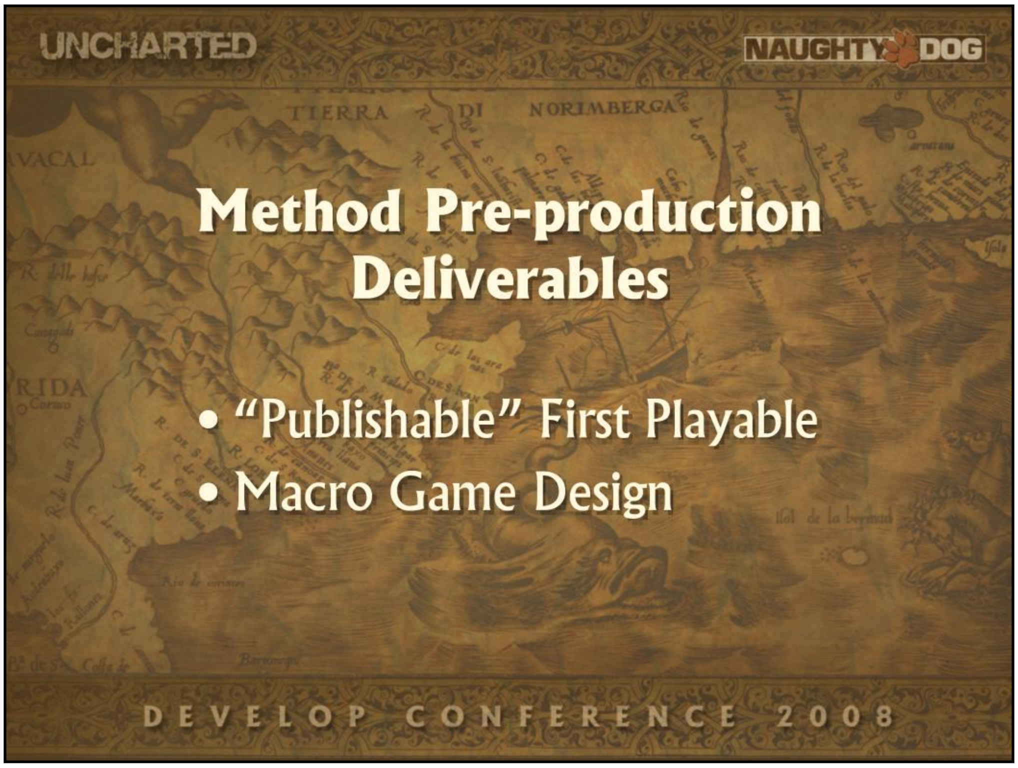
DEVELOP CONFERENCE 2008

For *Uncharted*, we had a bit longer than a full year of pre-production time, representing more than a third of our total three-year development cycle.

Our pre-production team was about fifteen people strong, and composed of a few veteran Naughty Dogs, including Amy Hennig, *Uncharted*'s Game Director and Bruce Straley, one of the game's two Art Directors, along with some new recruits who had been hired for their expertise in various next-gen fields.

This included quite a few programmers and artists from the movie visual effects industry, whose skills in high-fidelity graphics and animation were invaluable in helping us make the most of the powerful capabilities of the new PlayStation 3 hardware.

At GDC this year I discussed our preproduction process in detail in an *Uncharted* post-mortem called “Amazing Feats of Daring”, and if you’d like to hear more about how we developed the idea for the game, you can download the slides and text of my talk from the Naughty Dog web site, along with a bunch of other great talks by some of my fellow ‘Dogs.


UNCHARTED NAUGHTY DOG

Method Pre-production Deliverables

- “Publishable” First Playable
- Macro Game Design

DEVELOP CONFERENCE 2008

So, even though pre-production isn’t scheduled in a conventional way, in Method there **are** two firm deliverables for pre-production. They are:

A “Publishable” First Playable

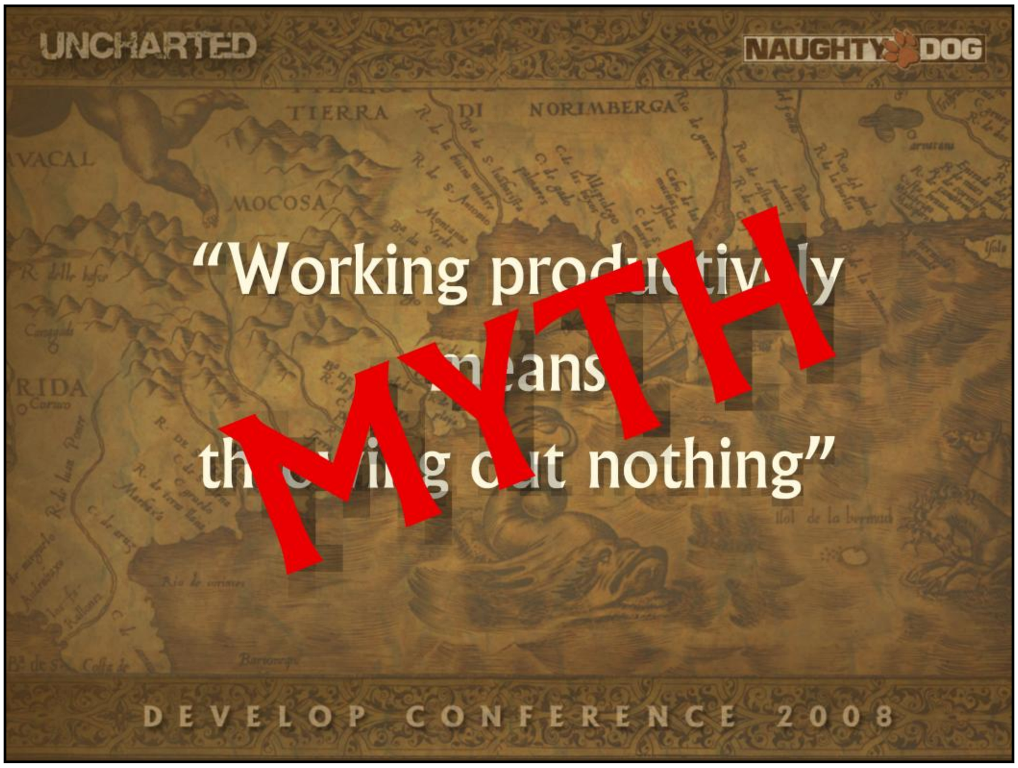
A Macro Game Design

"Publishable" First Playable

The first of these is a playable level that is representative of all the core elements of your game, and which is polished to the point where it's almost publishable.   This is an idea that has taken hold here and there in the industry as part of Pre-production, and often gets talked about as a "vertical slice" or the "X" of a game.

The pre-production team creates this playable level by making about five prototype levels, each of which gets thrown away after it's been built.

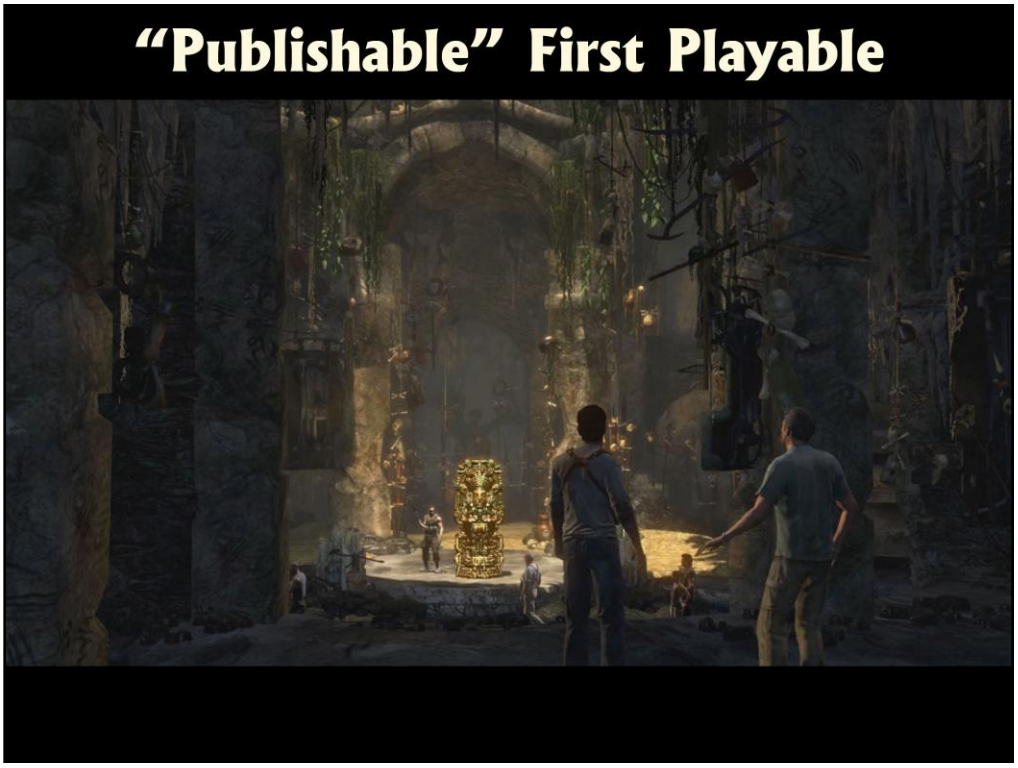
I'm not going to go into detail about the publishable first playable, because it wasn't really how we did things for *Uncharted* – and we try not to ever stick dogmatically to any given approach.


UNCHARTED
NAUGHTY DOG
“Working productively means throwing out nothing”
MYTH
DEVELOP CONFERENCE 2008

But there is something interesting to mention that relates to building levels that you then throw away, and that is that this part of Method seeks to explode another myth: “Working productively means throwing out nothing”.

There’s this idea in game development, often coming from the game team themselves, that effort should only go towards stuff that’s going to end up in the game. I can understand that people who are working very hard don’t want their work to be wasted, but I also think that it’s an attitude that holds us all back.

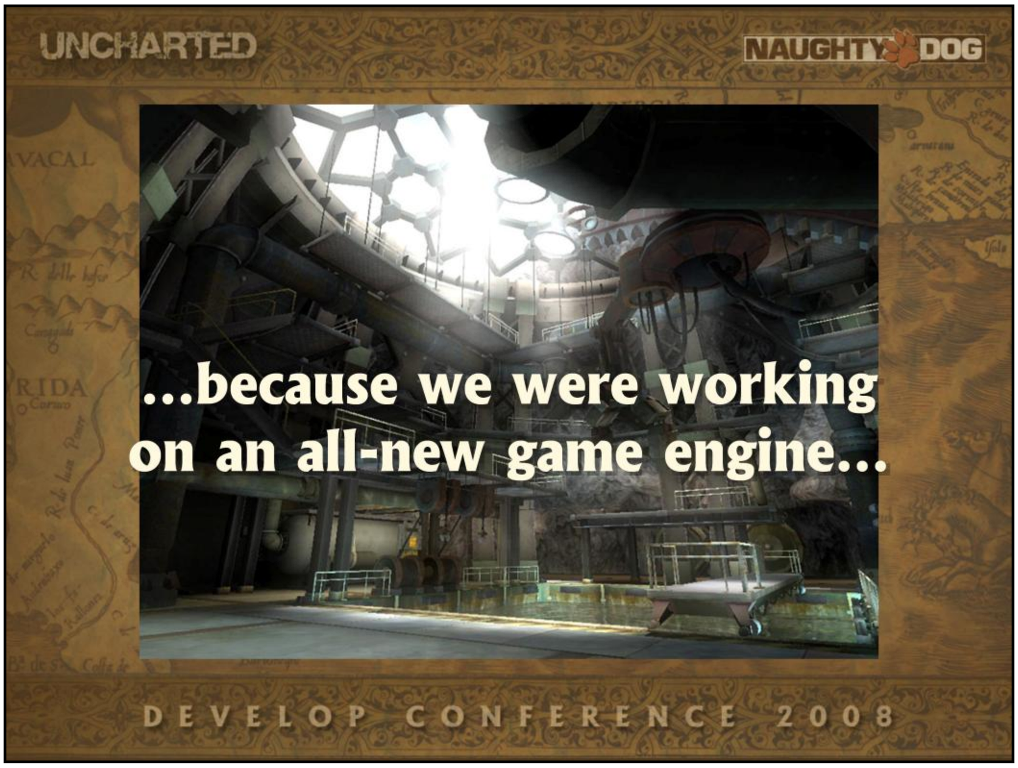
By building playable levels that get junked, you’re actually doing two things.



“Publishable” First Playable

The first is discovering your game design – which of your initial ideas work and which don’t. There’s no other way to make these discoveries than by building something. Even if you can’t make something interactive, pre-visualization movies are tremendously powerful, as you’ll see in a moment.

The second thing you’re doing is developing the concrete realities of your construction methodologies, in particular, your art pipeline, and making artwork components that you can use to make the “real” levels of the game. The art staff is probably the biggest part of the team, and they really need to have their approach down by the time full production begins. For *Uncharted*, that meant getting familiar with our new tools, and the arcane system of instancing which is the basis of our big, beautiful environments.


UNCHARTED

NAUGHTY DOG

...because we were working on an all-new game engine...

DEVELOP CONFERENCE 2008

So on *Uncharted*, we didn’t make a shippable first playable at the end of pre-production. We had set out to create entirely new technology, starting from zero lines of code – we were completely rebuilding our game engine, our tools and our build pipeline from the ground up.

By the time we left pre-production, while we had built some demo environments that helped the artists learn what they needed to springboard them into production, we were still working on the basics of the character animation and control system.

UNCHARTED
NAUGHTY DOG
DEVELO

To help us visualize our game, we got creative. Instead of a playable level, we made a short movie based on the paper game design that we'd been doing.

We put it together from still concept art and some pre-rendered visualization movies of the player-character based on the motion-capture tests we'd been doing, all held together with a lift from the soundtrack to Peter Weir's excellent *Master and Commander* movie.

Green Light Movie

DEVELOP CONFERENCE 2008

I think it's an interesting look into a way of getting at the guts of your game when you can't just go ahead and build a level, so let's take a look at it.

So even though we didn't have a publishable first playable, this movie gave us a very clear direction for the game, and enough visuals to sell ourselves on the idea that what we were aiming for would work.

As we've been working on the PlayStation 3 we've found over and over again how hugely valuable pre-visualization of any kind is, whether it's concept art, animation pre-viz, or character modeling tests. It helps you set targets for the rest of the game, and lets you try out new ideas quite a bit.

As a result of making this movie, we received a Green Light from Sony, and were able to move ahead into full production.

UNCHARTED

NAUGHTY DOG

Macro Game Design

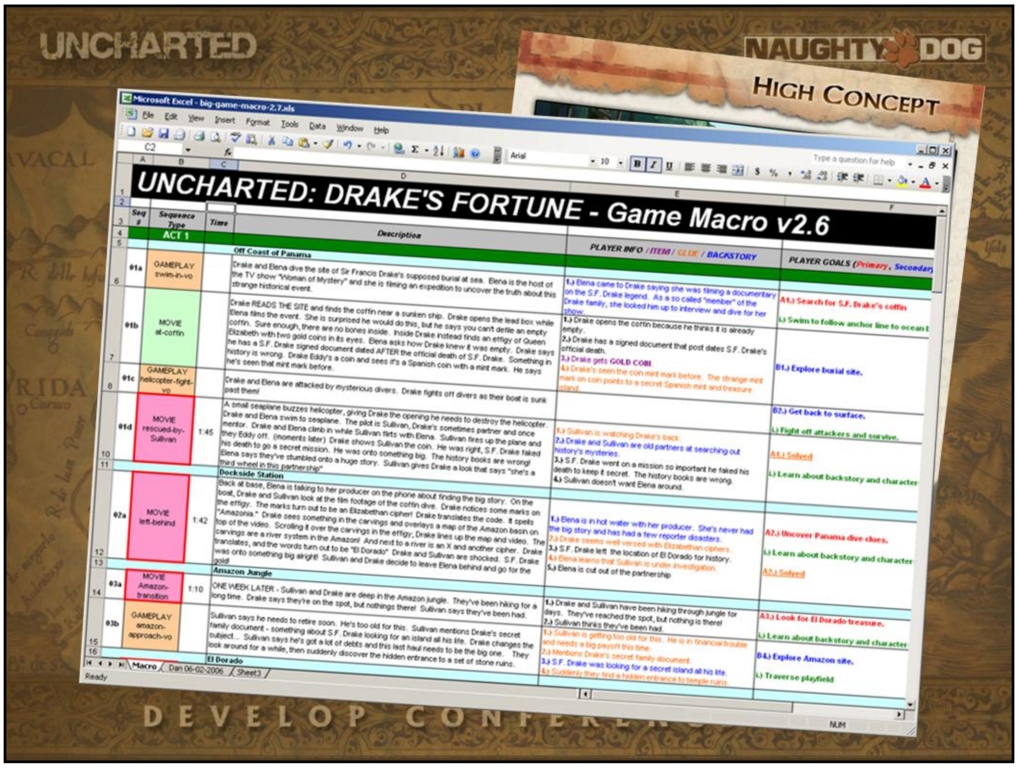
DEVELOP CONFERENCE 2008

We've always used the idea of a Macro Game Design at Naughty Dog when we plan our games, but I don't think it's an idea that's in common use in the industry. In Method, it's the second of the two concrete deliverables for the end of Pre-production.

Think of the Macro like an overview map of the whole British Isles. You can see the big cities, some mountains and the motorways and major roads – but you can't see the London Eye, or Edinburgh Castle, or Newent, the little town in north Gloucestershire where I grew up.

But you can plan a journey across Britain using a map like this, and you can figure out the details of your travels as you go along.

So our Macro design was made up of two main documents…



Microsoft Excel - big-game-macro-2.7.xls

UNCHARTED: DRAKE'S FORTUNE - Game Macro v2.6

Seq #	Sequence Type	Time	Description	PLAYER INFO / ITEM / CLUE / BACKSTORY	PLAYER GOALS (Primary, Secondary
	ACT 1				
			Off Coast of Panama		
01a	GAMEPLAY swim-in-vo		Drake and Elena dive the site of Sir Francis Drake's supposed burial at sea. Elena is the host of the TV show "Woman of Mystery" and she is filming an expedition to uncover the truth about this strange historical event.	1.) Elena came to Drake saying she was filming a documentary on the S.F. Drake legend. As a so called "member" of the Drake family, she looked him up to interview and dive for her show.	A1.) Search for S.F. Drake's coffin i.) Swim to follow anchor line to ocean b
01b	MOVIE at-coffin		Drake READS THE SITE and finds the coffin near a sunken ship. Drake opens the lead box while Elena films the event. She is surprised he would do this, but he says you can't defile an empty coffin. Sure enough, there are no bones inside. Inside Drake instead finds an effigy of Queen Elizabeth with two gold coins in its eyes. Elena asks how Drake knew it was empty. Drake says he has a S.F. Drake signed document dated AFTER the official death of S.F. Drake. Something in history is wrong. Drake Eddy's a coin and sees it's a Spanish coin with a mint mark. He says he's seen that mint mark before.	1.) Drake opens the coffin because he thinks it is already empty. 2.) Drake has a signed document that post dates S.F. Drake's official death. 3.) Drake gets GOLD COIN. 4.) Drake's seen the coin mint mark before. The strange mint mark on coin points to a secret Spanish mint and treasure island.	B1.) Explore burial site.
01c	GAMEPLAY helicopter-fight-vo		Drake and Elena are attacked by mysterious divers. Drake fights off divers as their boat is sunk past them!		B2.) Get back to surface. i.) Fight off attackers and survive.
01d	MOVIE rescued-by-Sullivan	1:45	A small seaplane buzzes helicopter, giving Drake the opening he needs to destroy the helicopter. Drake and Elena swim to seaplane. The pilot is Sullivan, Drake's sometimes partner and once mentor. Drake and Elena climb in while Sullivan flirts with Elena. Sullivan fires up the plane and they Eddy off. (moments later) Drake shows Sullivan the coin. He was right, S.F. Drake faked his death to go a secret mission. He was onto something big. The history books are wrong! Elena says they've stumbled onto a huge story. Sullivan gives Drake a look that says "she's a third wheel in this partnership"	1.) Sullivan is watching Drake's back. 2.) Drake and Sullivan are old partners at searching out history's mysteries. 3.) S.F. Drake went on a mission so important he faked his death to keep it secret. The history books are wrong. 4.) Sullivan doesn't want Elena around.	A1.) Solved i.) Learn about backstory and character
			Dockside Station		
02a	MOVIE left-behind	1:42	Back at base, Elena is talking to her producer on the phone about finding the big story. On the boat, Drake and Sullivan look at the film footage of the coffin dive. Drake notices some marks on the effigy. The marks turn out to be an Elizabethan cipher! Drake translates the code. It spells "Amazonia." Drake sees something in the carvings and overlays a map of the Amazon basin on top of the video. Scrolling it over the carvings in the effigy, Drake lines up the map and video. The carvings are a river system in the Amazon! And next to a river is an X and another cipher. Drake translates, and the words turn out to be "El Dorado" Drake and Sullivan are shocked. S.F. Drake was onto something big alright! Sullivan and Drake decide to leave Elena behind and go for the gold!	1.) Elena is in hot water with her producer. She's never had the big story and has had a few reporter disasters. 2.) Drake seems well versed with Elizabethan ciphers. 3.) S.F. Drake left the location of El Dorado for history. 4.) Elena learns that Sullivan is under investigation. 5.) Elena is cut out of the partnership	A2.) Uncover Panama dive clues. i.) Learn about backstory and character A2.) Solved
			Amazon Jungle		
03a	MOVIE Amazon-transition	1:10	ONE WEEK LATER - Sullivan and Drake are deep in the Amazon jungle. They've been hiking for a long time. Drake says they're on the spot, but nothings there! Sullivan says they've been had.	1.) Drake and Sullivan have been hiking through jungle for days. They've reached the spot, but nothing is there! 2.) Sullivan thinks they've been had.	A3.) Look for El Dorado treasure. i.) Learn about backstory and character
03b	GAMEPLAY amazon-approach-vo		Sullivan says he needs to retire soon. He's too old for this. Sullivan mentions Drake's secret family document - something about S.F. Drake looking for an island all his life. Drake changes the subject... Sullivan says he's got a lot of debts and this last haul needs to be the big one. They look around for a while, then suddenly discover the hidden entrance to a set of stone ruins.	1.) Sullivan is getting too old for this. He is in financial trouble and needs a big payoff this time. 2.) Mentions Drake's secret family document. 3.) S.F. Drake was looking for a secret island all his life. 4.) Suddenly they find a hidden entrance to temple ruins.	B4.) Explore Amazon site. i.) Traverse playfield
			El Dorado		

Macro / Dan 06-02-2006 / Sheet3

Our Concept Bible described *Uncharted* at the top level with a lot of text and images. It briefly described the game mechanics we planned to use, along with information about the cast of characters in the game, and an outline of the story.

Our main Macro document was in a spreadsheet and showed the gameplay and story beats broken down in more detail, organized into a list of levels and sub-levels. It showed which levels were going to use which mechanics, in particular calling out any special mechanics the levels would use, like the Truck Chase and Jet Ski levels.

So the Macro is something like a game design document, but it's a lot more concrete than any game design I've ever written, because it has enough of a level of abstraction that you don't waste time devising details that later get changed. It's also based on all your concrete pre-production work, and so has been pretty well thought-through.

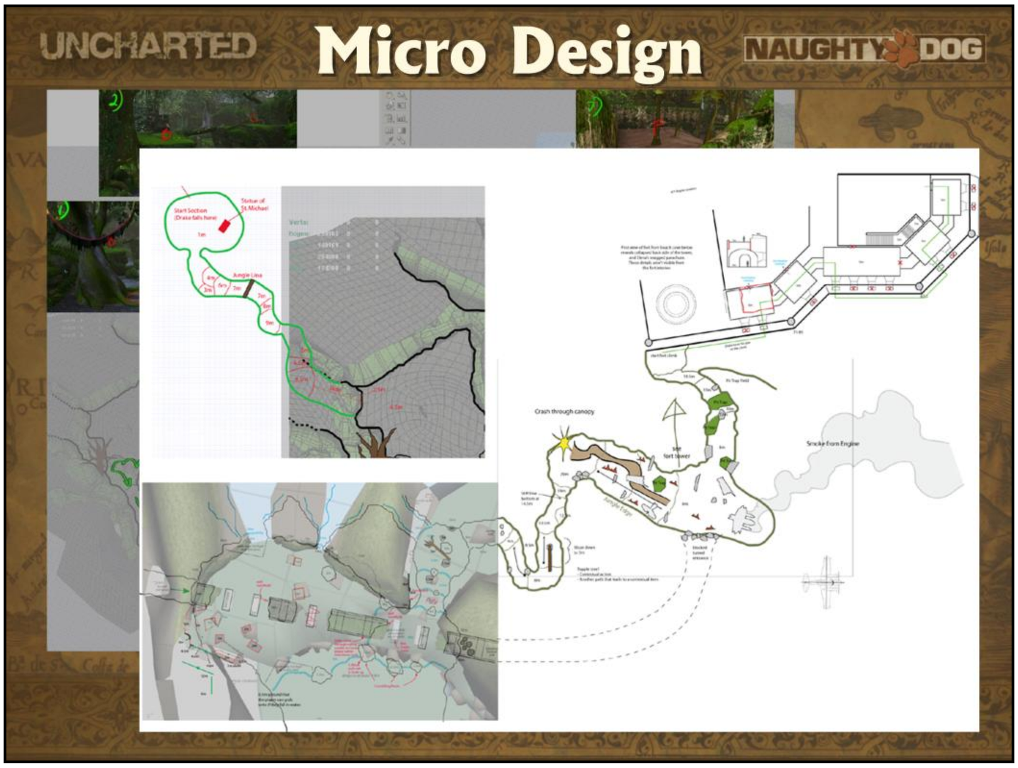
UNCHARTED

Micro Design

NAUGHTY DOG

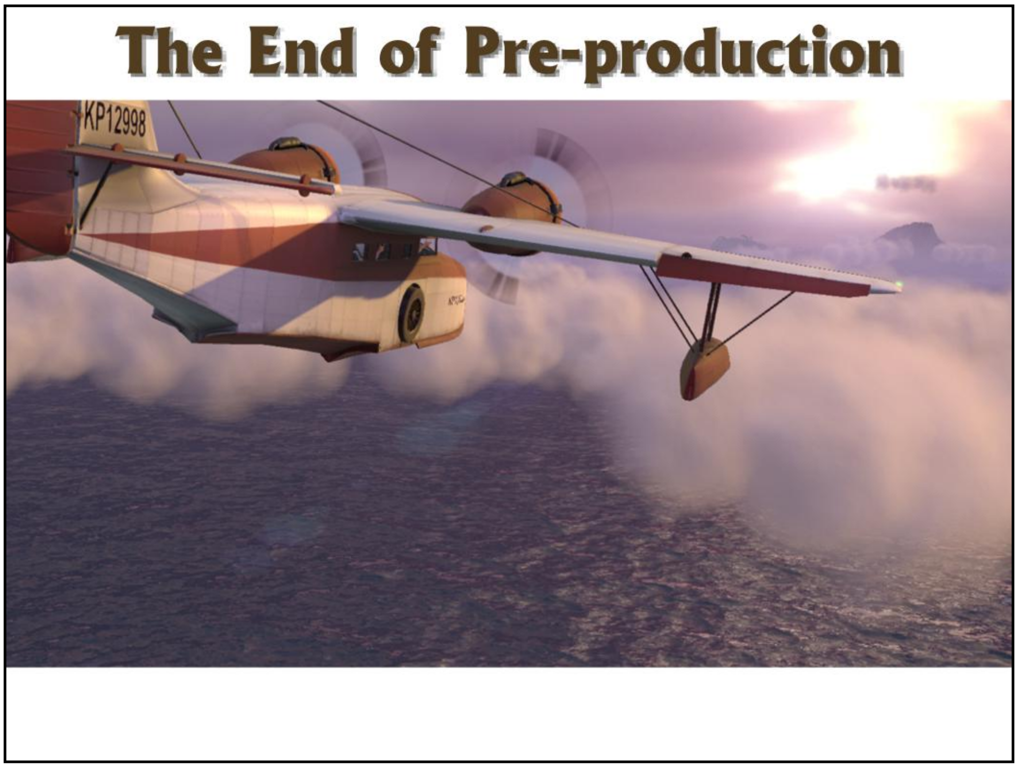
DEVELOP CONFERENCE 2008

As with every Naughty Dog game, the Micro design – that is, the microscopic detail of the game design - is figured out just ahead of time by the game designers and other team members. It’s a bit like Gromit laying out the train tracks in front of himself in *The Wrong Trousers*.



The Micro design includes any and all detailed work we need to do in planning out the gameplay - like enemy behaviors, the layout of the levels, the rules of the special mechanics, the positions of the objects in the levels, and so on.

This has the great benefit of allowing us nearly always to be sure that the detailed, time-consuming design work that we do is not going to get thrown away. Our time is precious, and we want to waste as little of it as possible.



The End of Pre-production

An important part of Pre-production is what happens at the end. When the Macro documents have been written, and the finishing touches have been put to whatever demo level or pre-visualization movies you made, it's time to present everything to the people sponsoring the project. In our case, it was our producers and executives at Sony.

At this time, it's up to the project sponsors to decide whether they want to Green Light the project, or give some steering commentary to nudge the project in a new direction, or cancel it.

An important thing to remember is that canning a project after pre-production does not necessarily mean that the pre-production team has failed. If the results of pre-production have been evaluated correctly, it simply means that they have proven that the project isn't viable.

Pre-production isn’t cheap. It might cost several million pounds, for a big project. So it might sound like money spent on pre-production of a project that then gets canceled has been wasted.

In fact, when you think of the much greater cost of bringing a project to full completion which either isn’t up to the high standards of game design and production quality that the market demands in 2008, or which can’t find a place in the market because it can’t reach enough players who want to play it, you can see that money spent deciding whether the project should be allowed to be brought to completion has been well spent.

In Mark Cerny's words, "…by being cost-*inefficient* during pre-production, you are actually being cost-*efficient* regarding the actual production of your game, since a solid pre-production phase creates a (map for a) production that's much more efficient and predictable in terms of time and money."

So if the project does get cancelled at the end of pre-production, you only blew a few million. You could blow a lot more with a production that's screwed.

Full Production

So, armed with the results of our pre-production work, we went into full production near the start of 2006. Full production lasted nearly two years. This might seem like a lot of time, but we still found ourselves very crunched at the end of the project, because of how long it took us to get everything going.

We went about production in a way that I think many of you would recognize - we drew up schedules based on the art tests that we'd done, matched people with tasks that need doing, and simply started to build.

Concentric Development

We develop concentrically, which is to say that we implement the fundamentals first to a good level of polish. For *Uncharted*, this was Nathan Drake's traversal mechanics, and the basics of how he interacts with the world.

Then we work outward, iterating on secondary and tertiary mechanics, like the gunplay, melee combat, and special mechanics.

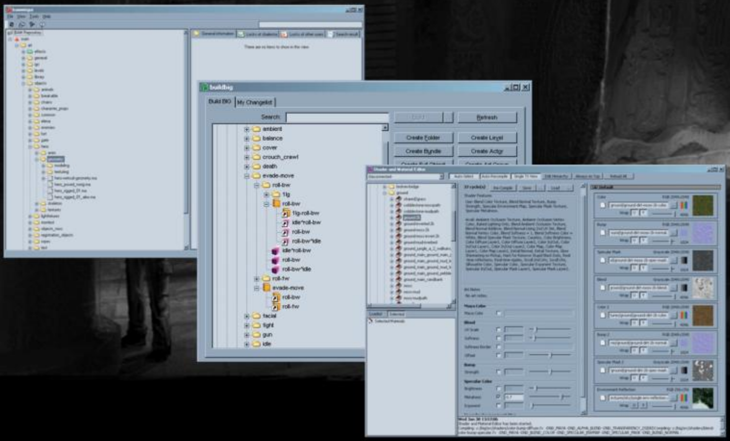
We try to build modularly, plugging in components that can be ditched if they don't work out. An example of this for *Uncharted* was underwater swimming, where we decided that we didn't have enough time to polish the mechanic until it was really fun, and identified that if we pulled it, there wouldn't be much impact on the rest of the game's design. Of course, we try to ditch stuff that fails as early as we can, so that we don't waste too much time on it.

Get It In the Game Quickly

In all of our post-mortem discussions about the game we felt that, when it came to the tools and technology of *Uncharted*, we should have got more basic stuff in and working earlier.

Making Naughty Dog games had always been about quickly getting solid stuff working in the game, but we had somehow lost sight of that amongst all the pre-production planning, and as the team grew. The tipping point for us came a few months into production.

Tools Rethink



Like I said, we’d developed all-new tools for *Uncharted*, including an asset management system, our first GUI build tool, and a shader and material editor.

But there were numerous problems with the tools – they were slow and had other usability issues - and people were very frustrated by them while we were making the trailer that we showed at E3 in 2006.

Basically, we had tried to be too clever. We’d designed very complex tools that took convoluted approaches to solving every last problem that anyone had ever had with each kind of tool. To make things worse, we’d gotten distracted by these lofty aspirations, and had left it very late to implement a tools pipeline that let people actually build and run around in their levels.

We didn’t waste any time crying about all the spilt milk, though, and simply went back to our old familiar, and often Linux command-line-driven *Jak and Daxter* ways of doing things.

Starting once again nearly from scratch, we got a bunch of basic tools up and running very quickly. They weren’t very pretty, or even very fully-featured beyond the basics, but they were fast and, in most cases, reliable.

From then on, things got better and better, and we began to get really good traction on building out the game.

Getting on with making the game

is the best way to make it

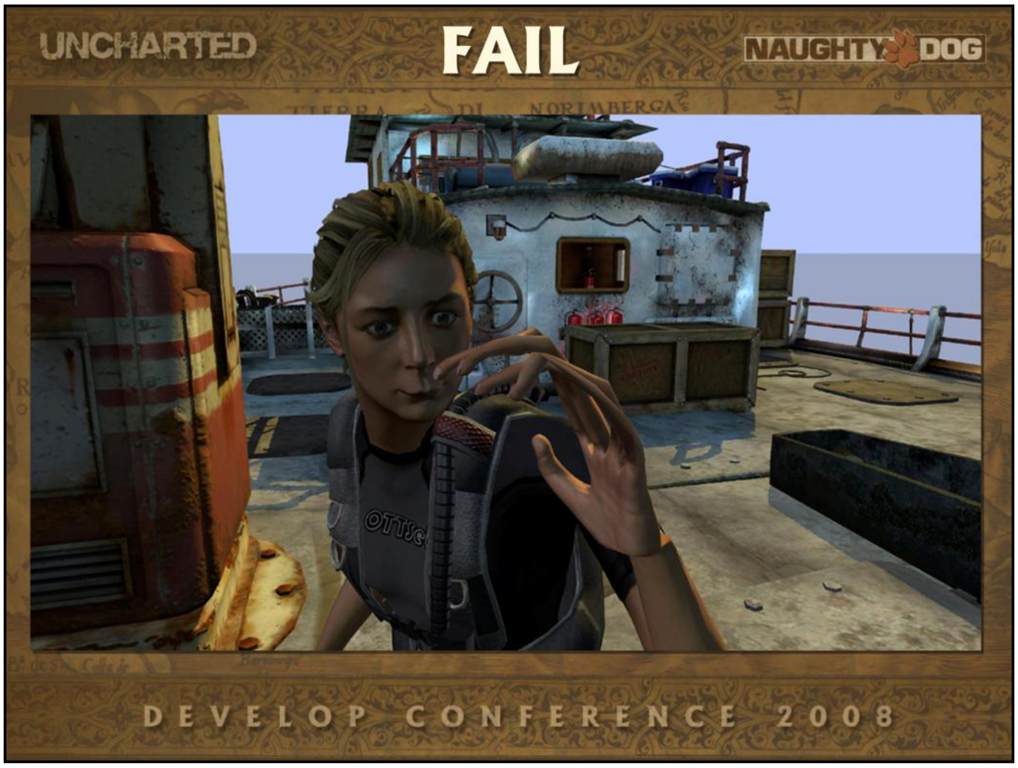
Our biggest takeaway from this experience, and one that we now hold to strongly, was: getting on with making the game is the best way to make it.

Theorizing about process and tools only takes you so far. Getting on with making the game is how we hone our tools and gameplay. If people are always waiting for even simple tools, and no-one can pick up a controller and work out ideas on the screen, the useful time for making the game slips away.

Build something bare-bones that gets the job done, and then expand and refine the things that work well or show potential. People become more creative within the structure of a limited tool or game mechanic that's solid, simply because they can do work and iterate.

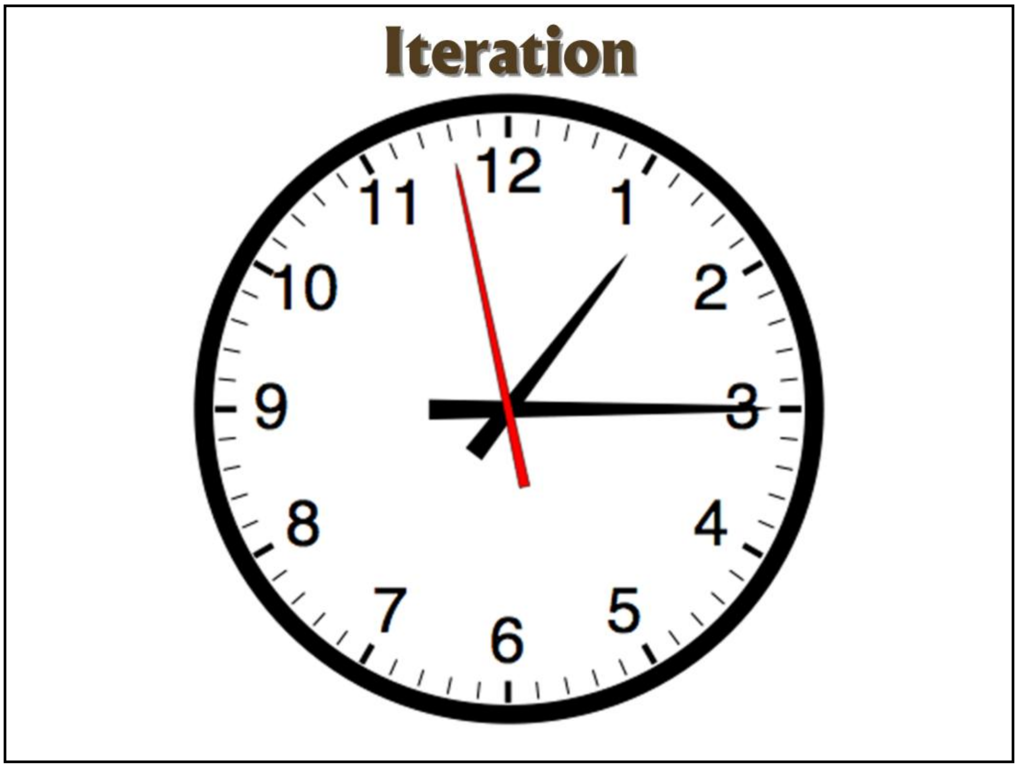
By now, I think everyone understands what an iterative art game development is.

We make progress by implementing our best ideas in the game, play-testing them on our colleagues and on members of the public in formal play-tests, and making changes to the game based on our observations of what worked and what didn't.



Of course, iterating like this implies failing at first. We never criticize someone who’s tried something that’s failed, and we live by the rapid prototyping credo: “Fail early, fail often”.

We want to fail early, while we can still afford to, and fail often so we can learn as much as possible.

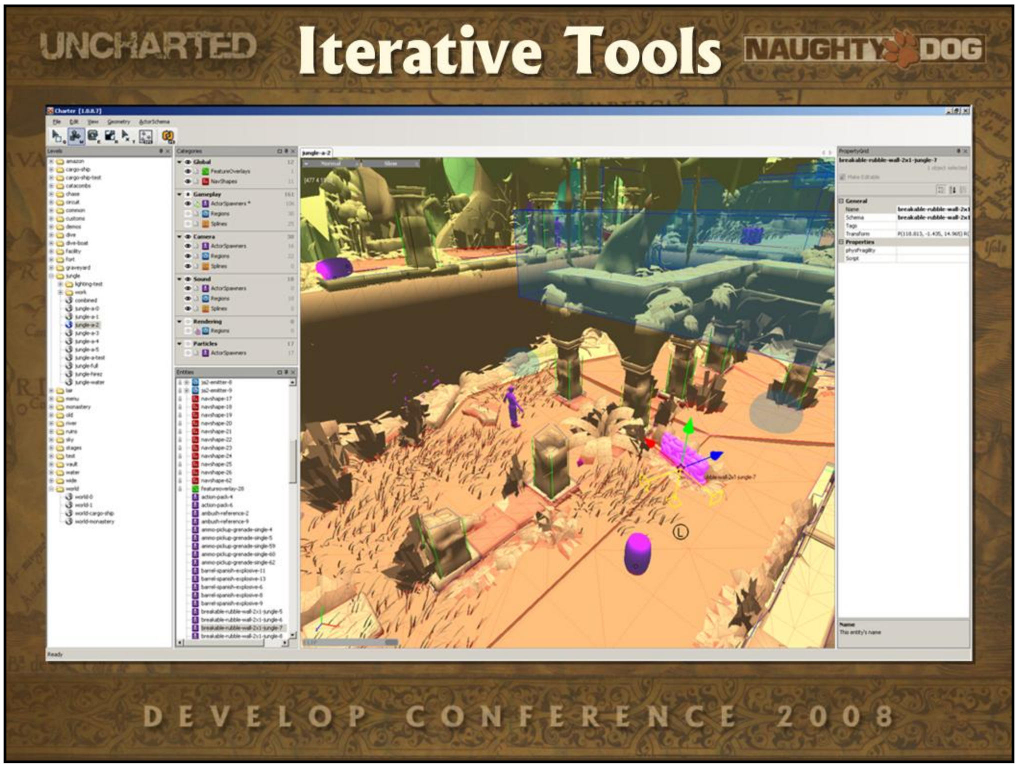


Iteration

You might say to me, “Listen, I just don’t have the time on my project to iterate like that. We need to do it once and get it out the door.”

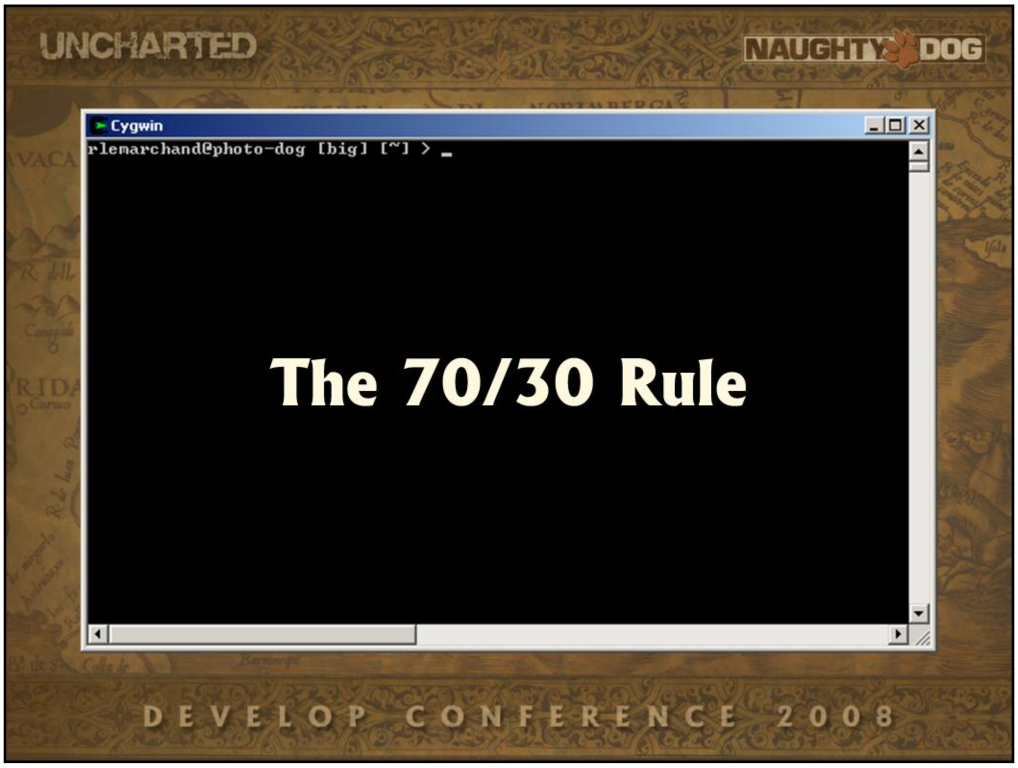
My reply to you is that if you’re trying to make something excellent, you don’t have time ***not*** to iterate. You’re going to fall short of your best if you don’t. No amount of pre-planning can make your game great, so if it’s greatness you want, you’d better scope your project and include some iteration time.

Iterative Tools



As everyone who has ever made a game will realize, the iterative process can be faster or slower depending on the tools that you're using, and so at Naughty Dog we've always had a focus on making tools that allow us to make changes and see them in the game as rapidly as possible.

So we've always gone the extra mile with our tools to make sure that our game designers can iterate quickly with them. For example, Charter, our level editing tool that you can see here, lets us add or move stuff around in a level and reload it dynamically, rather than having to re-run the whole game. Saving even a dozen seconds for each change you make when you're trying to get something just right can add up to weeks and months saved over the course of the project.



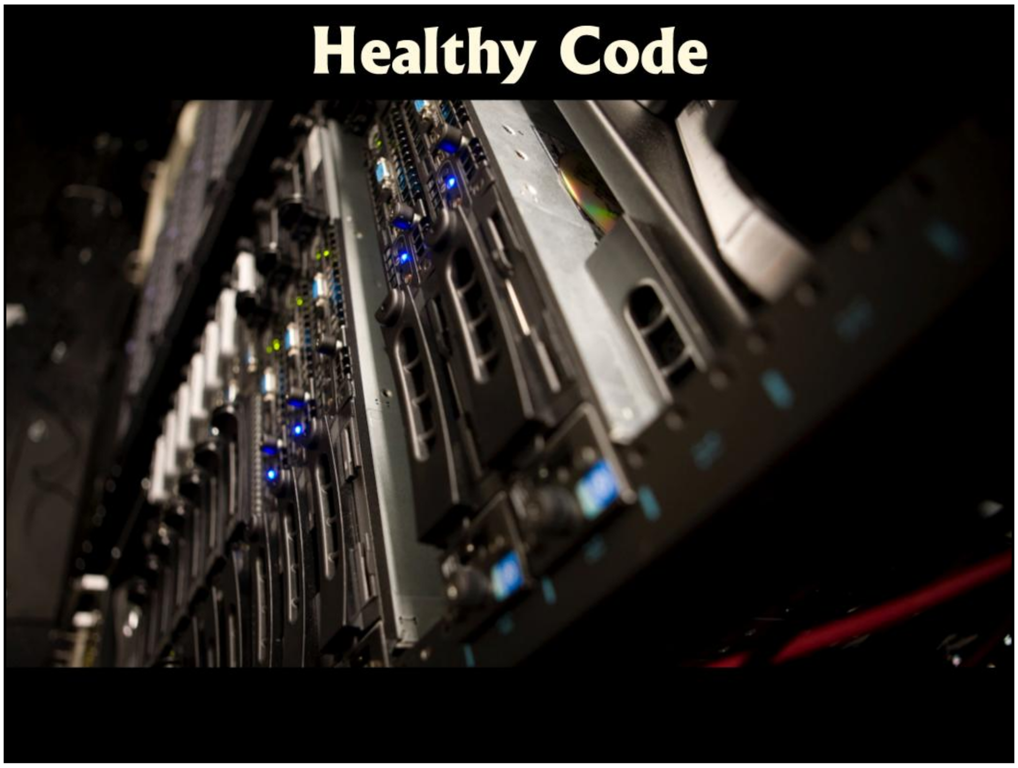
```
rlemarchand@photo-dog [big] [~] > _
```

The 70/30 Rule

UNCHARTED
NAUGHTY DOG
Cygwin
DEVELOP CONFERENCE 2008

I can’t emphasize this enough. Michael John said in a blog post a couple of years ago that:

“My rule of thumb on tools construction is that if the tools team has 100 RPG-style points to allocate to tools development, and the points must be distributed between tools usability and iteration time, they should allocate *30 points to usability, and 70 points to iteration time*. Programmers can create all the slick GUI tools they want, and they are all a good idea... but even the most awkward and opaque tools can eventually be learned; the time wasted in (slow) iteration can never be recaptured.”



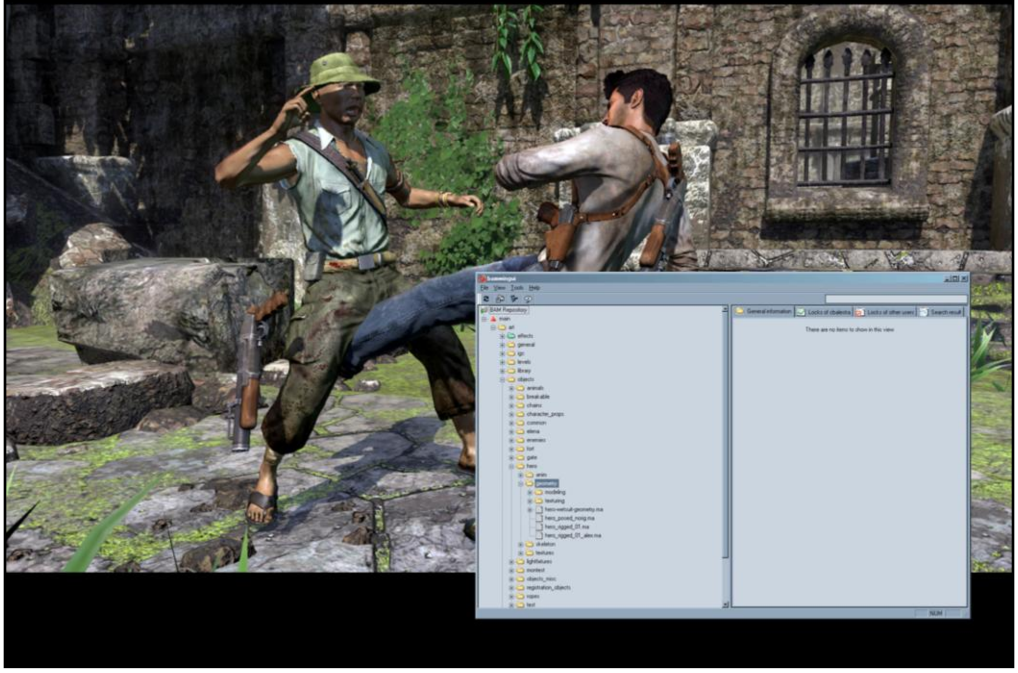
Healthy Code

Another thing we do that can sometimes get overlooked by developers is that we always keep our code running as healthily as possible. As you probably know, it's terribly frustrating to work on a game that's often broken, and unstable code makes a production move very slowly.

Of course, this can be a struggle, particularly at the beginning of the project when the codebase and feature list are growing very rapidly.

But it's a philosophical mindset of our programming team that when faced with a choice between quickly fixing a bug and implementing a new feature, they fix the bug first. This is particularly important when it comes to making demos and play-test discs with a minimum of farting around.

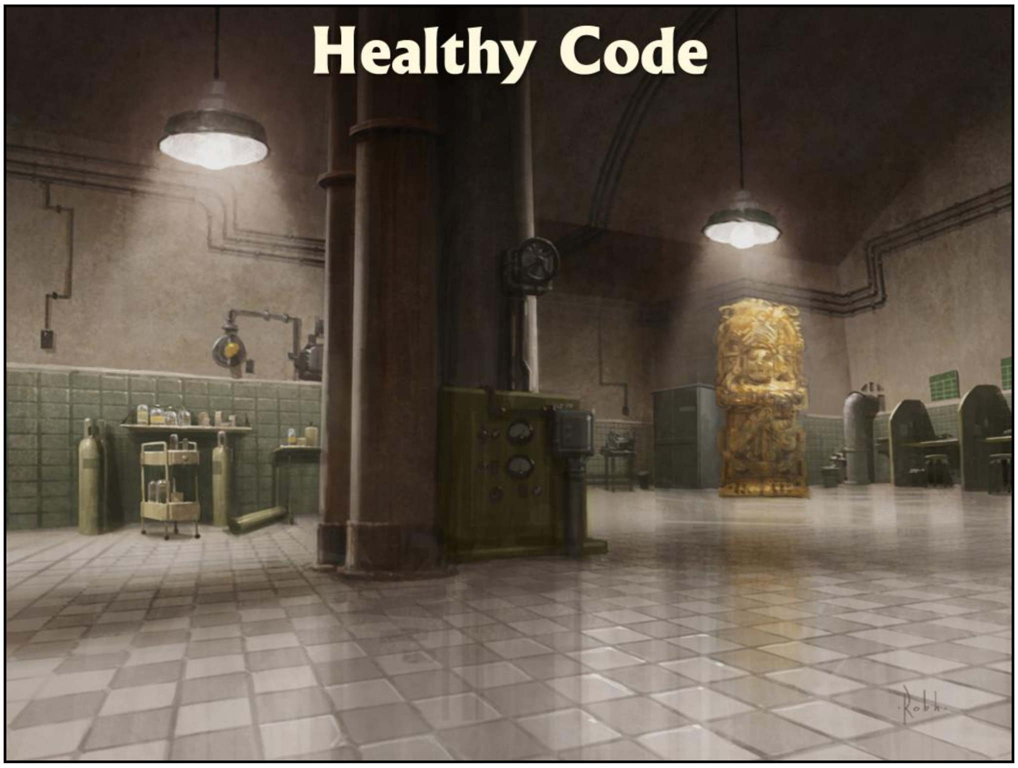
Healthy Code



Demos are always a pain in the arse to make, of course, and they can even become destructively time-consuming if you make too many, but if the code is always kept running healthily it becomes much easier to excerpt a section of the game and turn it into a demo for whoever needs to look at it, whether it’s the press or our producers at Sony.

Something that we do that might seem counterintuitive is that our asset management tool automatically syncs everyone immediately to the latest version of the game code and content. This means that if somebody checks something in that breaks the game, everyone knows about it immediately and mobilizes to fix it.

Healthy Code



Now I know that especially on large inexperienced teams this can be impractical – you don’t want your game to break all day every day - and we’re admittedly lucky in having a team experienced enough that *Uncharted* didn’t usually break more than a few times a day, and was usually only broken for as long as it took to publish a fix and rebuild the code.

The upside is that we’ve found that it creates an atmosphere where people are very careful about what they do check-in, and have more of a tendency to double-check their work for quality.

Frequent Deadlines

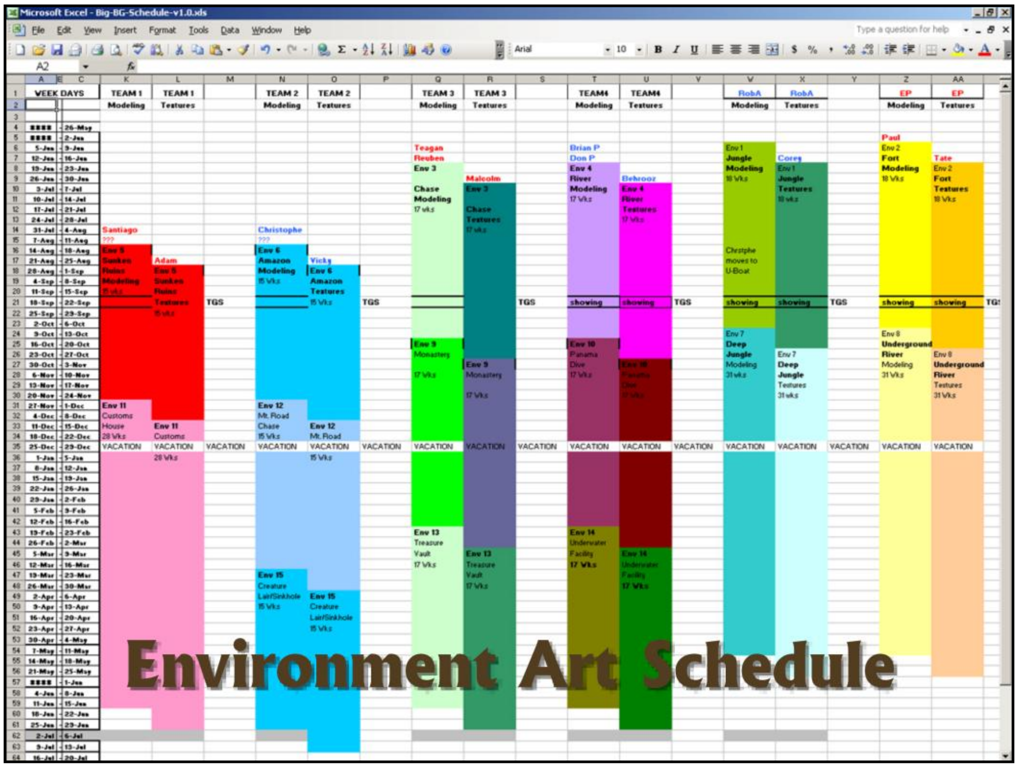
We’re driven by frequent deadlines at Naughty Dog, which we find to be the best way to keep our project on track. Our company presidents and game directors are always keeping an eye on the “big picture” of the project, of how much we have left to achieve and in what order, and they set a series of deadlines to help us get there bit by bit, usually on a four to six-week basis.

We also try and give each other very frequent reviews of the work that we’re doing. We’re constantly looking at each other’s work so that we can give feedback immediately, since it’s much easier to incorporate notes that you’re given while you’re still in the flow of working on something.

Frequent Deadlines

There’s a strong sense in our team culture that we have to meet our deadlines, and we always try to remind people of how important they are, even if they’re only internal milestones. I think that once you get into a mindset that slippage is OK, it’s very hard to get back out of it, and anyone who’s ever been on a death march will know just how hard it is when the goal posts move.

So as deadlines appear, everyone pulls together, and we do a lot of running around to make sure that everyone’s in the loop and understands what they’re expected to achieve.



Microsoft Excel - Big-BG-Schedule-v1.0.xls

	A		C	K	L	M	N	O	P	Q	R	S	T	U	V	V	X	Y	Z	AA
1	WEEK DAYS			TEAM 1	TEAM 1		TEAM 2	TEAM 2		TEAM 3	TEAM 3		TEAM4	TEAM4		RobA	RobA		EP	EP
2				Modeling	Textures		Modeling	Textures		Modeling	Textures		Modeling	Textures		Modeling	Textures		Modeling	Textures
3																				
4	####	-	26-May																	
5	####	-	2-Jun																Paul	
6	5-Jun	-	9-Jun							Teagan			Brian P			Env 1			Env 2	
7	12-Jun	-	16-Jun							Reuben			Don P			Jungle	Corey		Fort	Tate
8	19-Jun	-	23-Jun							Env 3			Env 4			Modeling	Env 1		Modeling	Env 2
9	26-Jun	-	30-Jun								Malcolm		River	Behrooz		18 Vks	Jungle		18 Vks	Fort
10	3-Jul	-	7-Jul							Chase	Env 3		Modeling	Env 4			Textures			Textures
11	10-Jul	-	14-Jul							Modeling			17 Vks	River			18 wks			18 Vks
12	17-Jul	-	21-Jul							17 wks	Chase			Textures						
13	24-Jul	-	28-Jul								Textures			17 Vks						
14	31-Jul	-	4-Aug	Santiago			Christophe				17 wks									
15	7-Aug	-	11-Aug	???			???													
16	14-Aug	-	18-Aug	Env 5			Env 6									Chrstphe				
17	21-Aug	-	25-Aug	Sunken	Adam		Amazon	Vicky								moves to				
18	28-Aug	-	1-Sep	Ruins	Env 5		Modeling	Env 6								U-Boat				
19	4-Sep	-	8-Sep	Modeling	Sunken		15 Vks	Amazon												
20	11-Sep	-	15-Sep	15 vks	Ruins			Textures												
21	18-Sep	-	22-Sep		Textures	TGS		15 Vks	TGS			TGS	showing	showing	TGS	showing	showing	TGS	showing	showing
22	25-Sep	-	29-Sep		15 vks															
23	2-Oct	-	6-Oct																	
24	9-Oct	-	13-Oct													Env 7			Env 8	
25	16-Oct	-	20-Oct							Env 9			Env 10			Deep			Underground	
26	23-Oct	-	27-Oct							Monastery			Panama			Jungle	Env 7		River	Env 8
27	30-Oct	-	3-Nov								Env 9		Dive	Env 10		Modeling	Deep		Modeling	Underground
28	6-Nov	-	10-Nov							17 Vks	Monastery		17 Vks	Panama		31 wks	Jungle		31 Vks	River
29	13-Nov	-	17-Nov											Dive			Textures			Textures
30	20-Nov	-	24-Nov								17 Vks			17 Vks			31 wks			31 Vks
31	27-Nov	-	1-Dec	Env 11			Env 12													
32	4-Dec	-	8-Dec	Customs			Mt. Road													
33	11-Dec	-	15-Dec	House	Env 11		Chase	Env 12												
34	18-Dec	-	22-Dec	28 Vks	Customs		15 Vks	Mt. Road												
35	25-Dec	-	29-Dec	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION	VACATION
36	1-Jan	-	5-Jan		28 Vks			15 Vks												
37	8-Jan	-	12-Jan																	
38	15-Jan	-	19-Jan																	
39	22-Jan	-	26-Jan																	
40	29-Jan	-	2-Feb																	
41	5-Feb	-	9-Feb																	
42	12-Feb	-	16-Feb																	
43	19-Feb	-	23-Feb							Env 13			Env 14							
44	26-Feb	-	2-Mar							Treasure			Underwater							
45	5-Mar	-	9-Mar							Vault	Env 13		Facility	Env 14						
46	12-Mar	-	16-Mar							17 Vks	Treasure		17 Vks	Underwater						
47	19-Mar	-	23-Mar				Env 15				Vault			Facility						
48	26-Mar	-	30-Mar				Creature				17 Vks			17 Vks						
49	2-Apr	-	6-Apr				Lair/Sinkhole	Env 15												
50	9-Apr	-	13-Apr				15 Vks	Creature												
51	16-Apr	-	20-Apr					Lair/Sinkhole												
52	23-Apr	-	27-Apr					15 Vks												
53	30-Apr	-	4-May																	
54	7-May	-	11-May																	
55	14-May	-	18-May																	
56	21-May	-	25-May																	
57	####	-	1-Jun																	
58	4-Jun	-	8-Jun																	
59	11-Jun	-	15-Jun																	
60	18-Jun	-	22-Jun																	
61	25-Jun	-	29-Jun																	
62	2-Jul	-	6-Jul																	
63	9-Jul	-	13-Jul																	
64	16-Jul	-	20-Jul																	

Environment Art Schedule

This is the way that we schedule environment art creation at Naughty Dog. We don't really believe in Gantt Charts – Microsoft Project and all that. With a production approach like ours that involves so much change, they really just make for busy-work, and for tasks that are measured in weeks, this kind of schedule in a spreadsheet is perfect.

In fact, we use spreadsheets for a lot of things; for example, the Environment and Animation Leads also use spreadsheets to track work and coordinate different team members, and probably more than half of our design documents are spreadsheets.

Make Up Your Own Rules

I wanted to briefly mention that we always try not to follow any preconceived ideas about the right and wrong ways to do things, and we always try to bring “fresh eyes” to our processes to make sure that they’re geared towards making the game as great as it can possibly be.

An example of this came as we worked on our motion-capture sessions, both for the gameplay and the in-game cut scenes that we made for *Uncharted*. Because this was our first experience using motion capture, we had an open mind about how to do things, and our approach ended up owing more to the way a movie or theater production is staged than the way most people in the games industry do mo-cap.

We used the same actors for both mo-cap and dialogue, which allowed us to workshop the scenes on the motion-capture stage, and in fact many of our favorite moments in the game were suggested by our fantastic actors and our great motion capture director, Gordon Hunt.

Make Up Your Own Rules

Another unusual aspect of the way we do things is that we have an in-house editor – someone with movie-editing experience, who works with our animators on the cut scenes, made our Green Light movie, and creates our trailers and other promotional movies.

When they were working together on the cut scenes, our animators and editor would pass animatics back and forth, constantly refining the cuts and timing of the scene until it was just right. Just one more example of keeping an open mind to new processes that help make the finished game really rock.

Making Tough Calls

So what happens when your iteration has taken longer than you expected, some of your game mechanics needed much more development that you'd anticipated, and the levels need an extra few weeks of polish to make them awesome?

This is when it's important that your project's leaders can make good decisions on the fly, and make tough calls when they're needed.

Making Tough Calls

For example, we really started to hit a crunch in the last few months of *Uncharted*, and realized that we didn't have time to make all the levels as big as we'd planned them to be. We acted quickly, and asked the level's designers to roughly halve the size of the Customs House and Facility levels, since those were the ones that had had the least environment art built for them, and where we thought the pace of the game would suffer least.

We also reallocated some of our art resources, moving object artists that had some available cycles onto these environments, and that helped us get everything done on-time and to quality.

Happily, it turned out we were right about the pacing, and in fact, people say that they love the pace of the game in those sections, so even though they were a bit painful at the end, we ended up glad that we made the cuts.

Making Tough Calls

So when it's all coming down, you have to be able to make the right trade-offs between the rigid constraints of resources and time, and the malleable constraint of scope – in other words, which things from your macro design you choose to implement, and what you have to reduce in size or ditch altogether.

That usually involves some very difficult decisions, but for your team to succeed, you need someone who can make those tough calls. If you haven't got too far ahead with your micro-design, it becomes easier to make these kinds of choices, but it's always tough, and it's definitely a skill that all the great game designers share.

One nice rider to this is that Evan told me that even when we're losing something that we really liked, and that feels like it was core to the game design, he always has faith that the team will still find a way to make the game amazing.

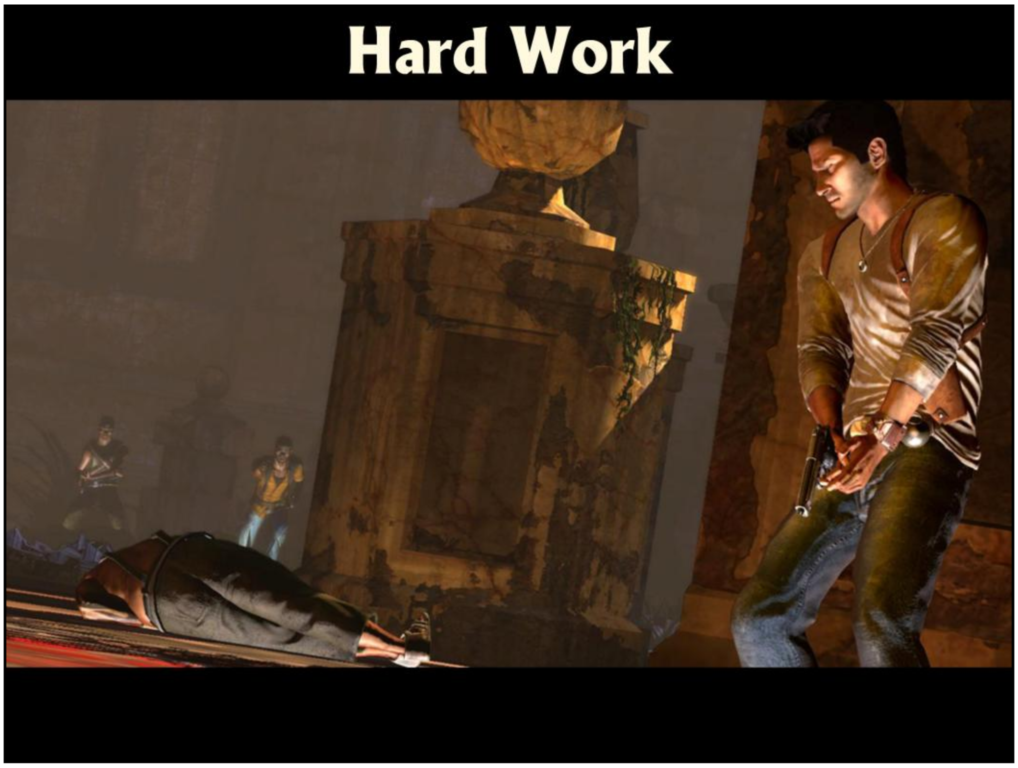
Massive Attack Trailer

MASSIVE ATTACK
MEZZANINE

DEVELOP CONFERENCE 2008

Just before Gold Master, we made a trailer using the track “Dissolved Girl” by Massive Attack, of whom we’re huge fans.

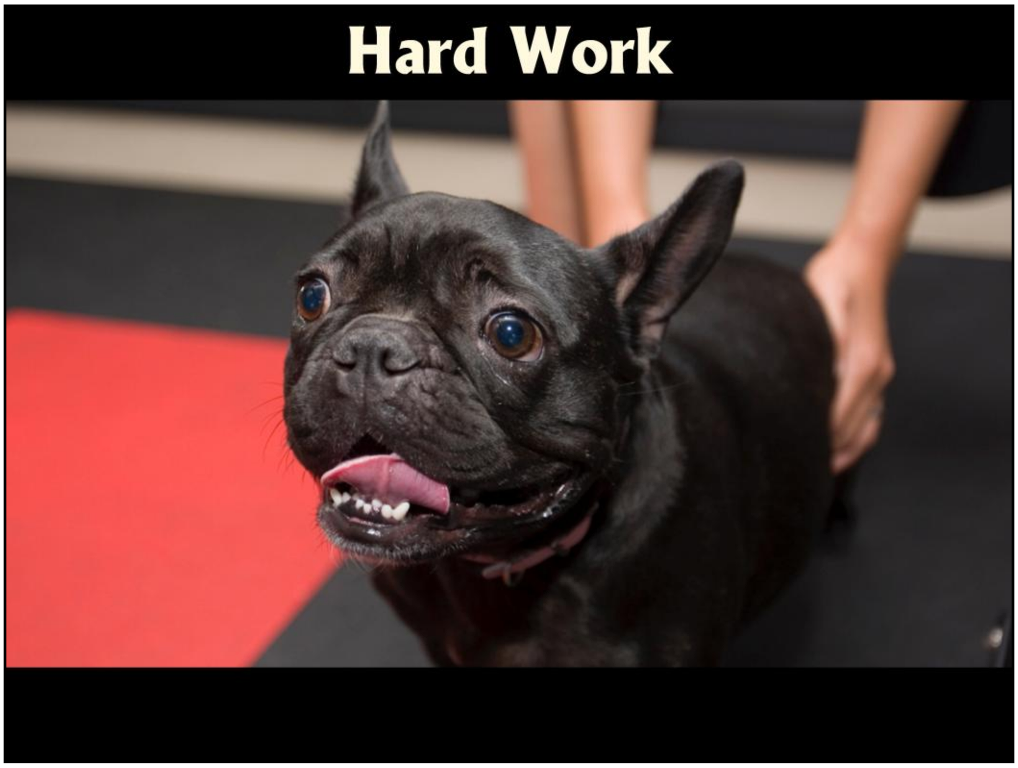
We’d liked all of our trailers a lot, but we thought this one was something really special, and I’d like to show it to you now.



Hard Work

Hopefully everything that I've talked about so far has already given you the impression that we put a lot of energy into our work at Naughty Dog, because we think that, more than anything else, hard work is the thing that makes our games great.

Walt Disney coined the term "plussing" as a way of making a great idea even better. By telling his artists and animators to "plus" it, even when they thought they'd nailed a piece of work, Disney got that extra edge when it came to quality. We like to think that we uphold that tradition, and though it takes extra energy and work, it's worth it.



Hard Work

I must admit that this means some crunch near deadlines and at the end of the project, and plenty of the Dogs – including Pogo, our studio mascot – are so into our work that you'll often find us at our desks late at night or at the weekend.

However, we've got a handle on crunching over the years, partly because we realize that after a few weeks of crunch, your productivity drops back down to that of a normal work week – or even less. We're also all getting a little older, and many of the Dogs in our kennel now have puppies to hang out with in the evenings.

Hard Work

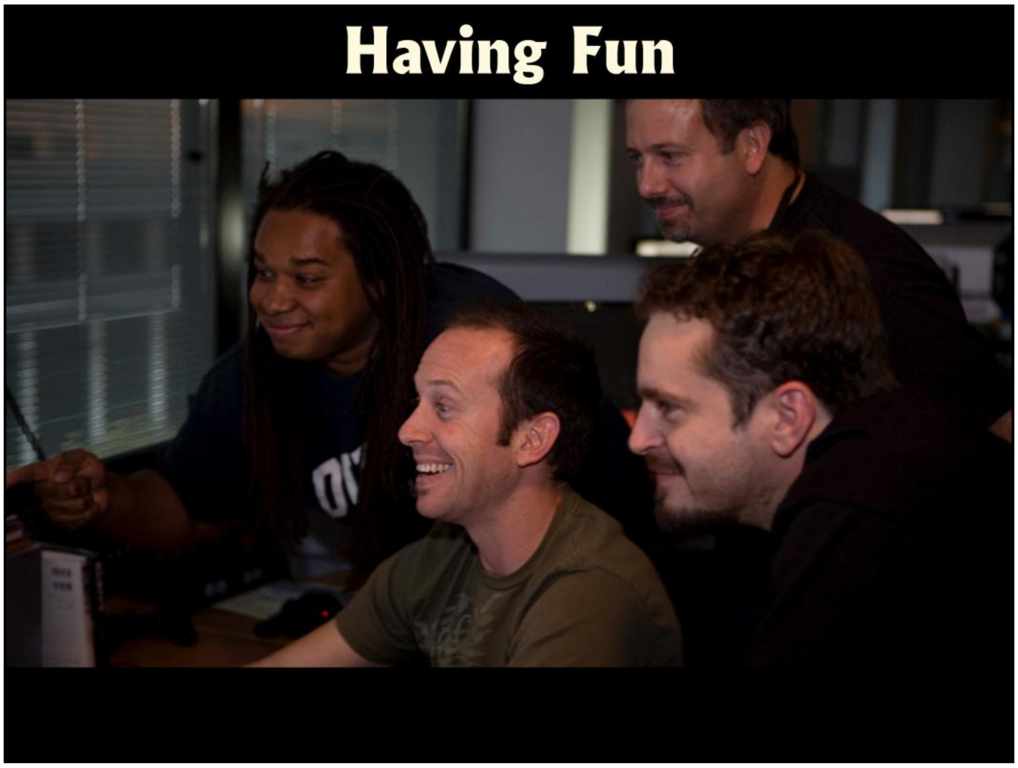
What we much prefer is to work in a very hard and focused way for a full forty hours a week. When you walk around Naughty Dog during core hours it's often very quiet – and it's the sound of people intently focused on what they're doing. We all try to have healthy working habits, focusing only on work things during work hours.

Forty hours a week is a lot of time, and it's amazing what you can achieve when you stay very focused for those forty hours.

Having Fun

So I feel like I've said quite a lot that relates to the "working hard" theme of my talk, but what about "having fun"? There's a saying in game development circles that "you have to have fun in order to make fun".

This seems right to me: if the development team are enjoying their work, their happy, engaged mindset will trickle down into the content of what they're building, and the game that results will be that much better.



Having Fun

People joining our studio usually find our approach to game development very refreshing, and everyone at Naughty Dog agrees – when you work in the fast, loose, passionate, professional way that we work, it’s just naturally more fun.

It’s about going home after eight hours of focused work with a feeling of accomplishment, because you can see the concrete progress that you’re making towards the shared goals of the team.

It’s the difference between slogging away to someone else’s spec on something that you don’t really care about, and working on something that you’re passionate about and in charge of.

It’s a feeling of being valued and listened to by your team-mates and the people in charge of the company, and of being trusted to do a great job.

We have a lot of fun at Naughty Dog, and hopefully it shows in our games.

Now we don't pretend that we have the only answers, or the final answer, to the question of how to make great games. All I can hope for is that some of the things I've mentioned today will sound useful to you in your work, and you can pick and mix to find the approaches that work best for you.

But I just wanted to come here today and tell you all that *Uncharted: Drake's Fortune*, the great big expensive flashy game that we've received so much praise for, and which we consider to be the best game that we've ever made, was not created on a big slick Californian game development factory-line - but that it was jammed together by a bunch of passionate, smart, hard-working people from all around the world, working together day by day in a fast, loose way to make something that they considered brilliant.

That it's still possible to make big, awesome games with a garage-developer mentality in 2008. That there's a lot to look forward to in the future, for all of us, in terms of making great games by working hard while having fun.

“Working Hard and Having Fun”
How Naughty Dog Made
UNCHARTED
DRAKE'S FORTUNE

Thanks for listening!

Richard Lemarchand
Lead Game Designer
Uncharted: Drake’s Fortune
www.naughtydog.com

DEVELOP CONFERENCE 2008

Thanks very much for your time.