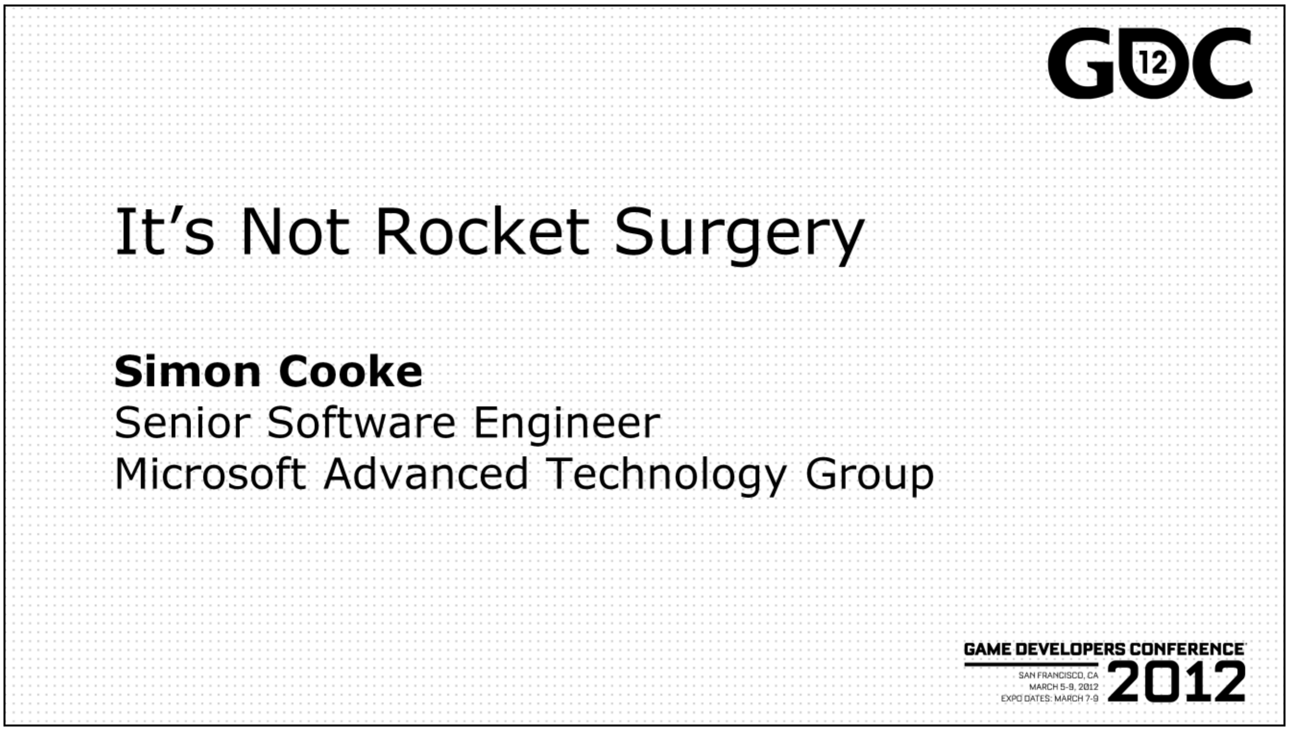


GDC 12

It’s Not Rocket Surgery

Simon Cooke
Senior Software Engineer
Microsoft Advanced Technology Group

GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 5-9, 2012
EXPO DATES: MARCH 7-9
2012

So this talk is the distillation of 10 years of managing varying size teams in a whole variety of different situations, including working with teams across and on other continents. Hopefully I’m not going to ramble too much, but it’s packed with information. Ideally, at the end of it, I’ll have given you a bunch of tools for your toolkit, and some concrete scenarios you will most likely come across while shipping a game, with solid ideas on how to conquer them.

You may, however, be wondering why you should listen to an Engineer about Production Methodology. *(Hint: I’m weird, and don’t pigeonhole well).*

So why would you want to listen to me about Producing?

Well, I've shipped a lot of both consumer and games software over the years.

If you've heard of me, it's because you were a reader of Your Sinclair magazine at one point in time (cue embarrassing caricature), where I had a monthly column for about a year when I was 17. Or, you've heard of the SAM Coupe – a home computer that nearly no-one had, where I was pretty well known and worked on ports of Prince of Persia and Lemmings.

Spin forward a few years omitting a lot of stuff, and I was at Sierra as a Lead working on Generations Family Tree – yes, we had about 50 CDs in each box! – and shipped a LOT of SKUs – 120 or so to be precise over 3 years. I also ran the Photo Lab project which shipped in Print Artist and other Sierra apps.

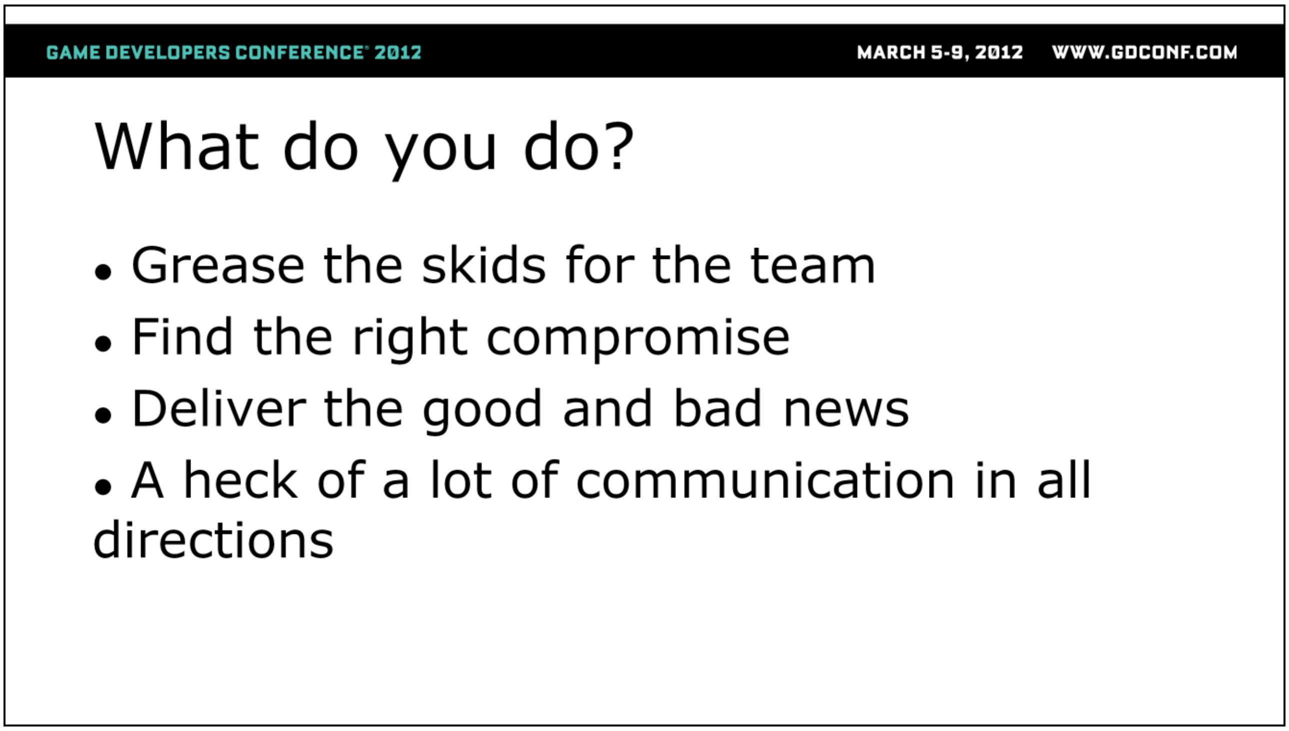
After Sierra collapsed, I worked on scientific instruments for a while, and then dived back into the games industry in 2005 to join Surreal Software, where I was Lead Tools/Engine Programmer for Suffering: Ties That Bind, helped out with Localization for the Russian version, and then moved onto This Is Vegas, where I was Lead Tools, then Lead Gameplay Programmer, then one of the Principle Technology Engineers on the engine team.

I ended up leaving because … well… I was engaged to the programming producer and it's weird to be in that kind of work relationship when you're going home together at night, so she gave me an ultimatum ☺

Cue X-Ray Kid Studios, where I was Director of Engineering, and also Business Development Manager. After working on game designs for about 18 games, and pitching to several large companies, I ended up leaving (we ran out of money). X-Ray Kid are still going, and do some awesome work – check them out, they're good folks (http://www.x-raykid.com).

So now I'm in Xbox – well, ATG, which covers all of Microsoft's Game Technology initiatives, where I make game developers' lives better.

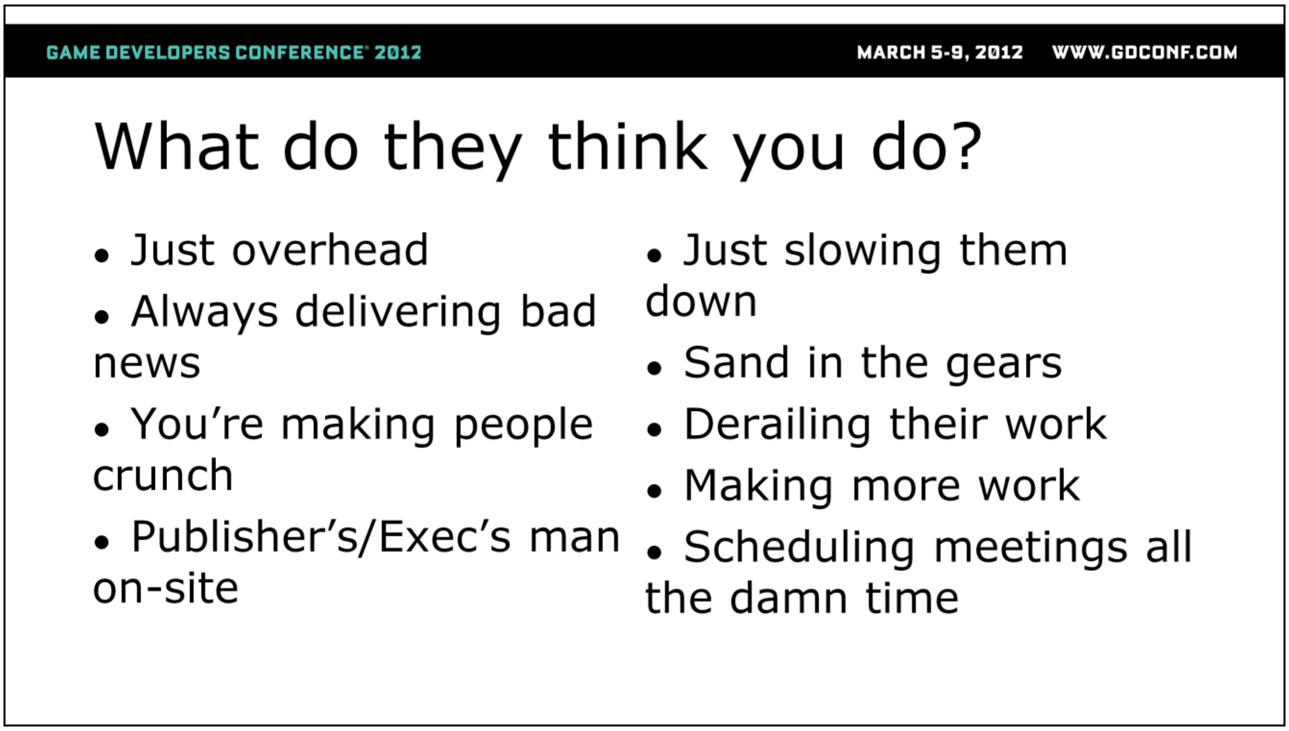
(see also: http://www.linkedin.com/in/fleetingshadow , http://accidentalscientist.com if you want to know more… older stuff - still somewhat relevant - is on http://simoncooke.com)



What do you do?

- Grease the skids for the team
- Find the right compromise
- Deliver the good and bad news
- A heck of a lot of communication in all directions

Here's what you think you do… I think you'll agree it's pretty accurate in a vague kind of way. It feels about right. (Even though it's a bit handwavy)

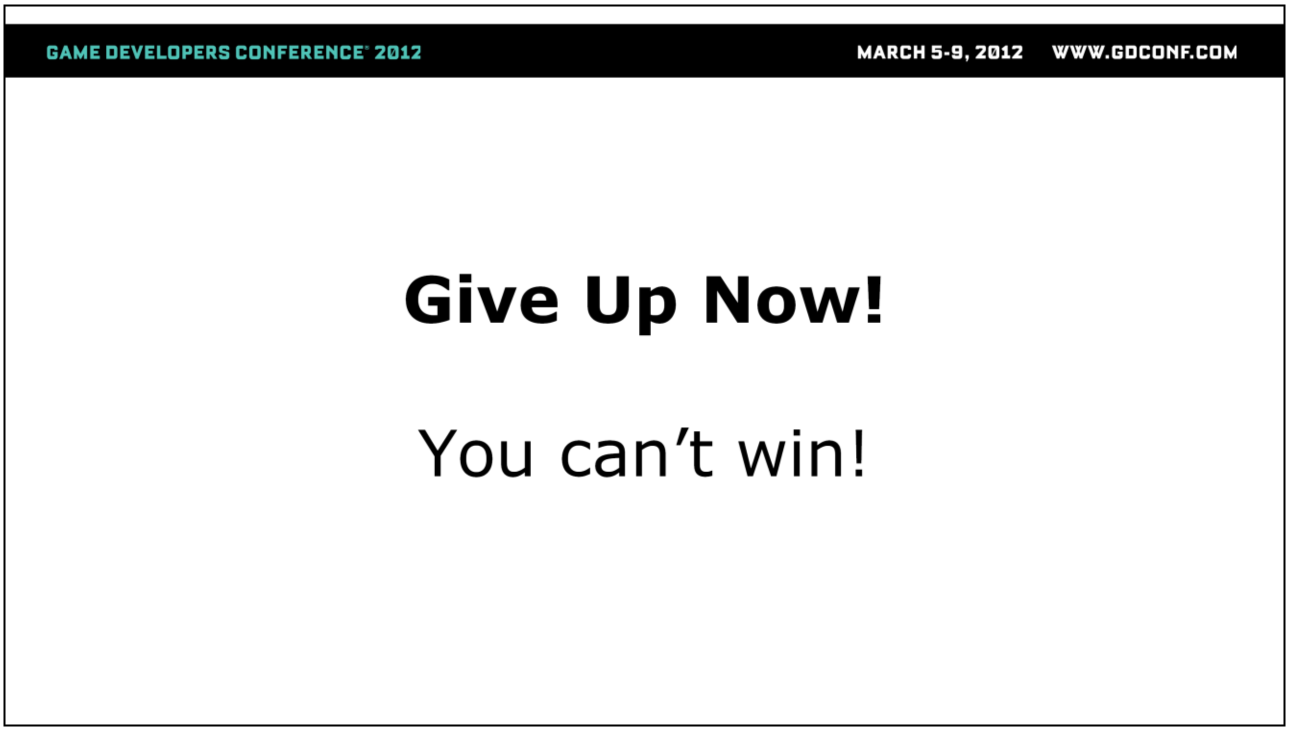


What do they think you do?

- Just overhead
- Always delivering bad news
- You're making people crunch
- Publisher's/Exec's man on-site
- Just slowing them down
- Sand in the gears
- Derailing their work
- Making more work
- Scheduling meetings all the damn time

Here's what everyone else on the team might think you do, especially if they've had a bad producer in the past (they do exist! And people have been burned by them!), or if you're dealing with a smaller company where the founders had to do everything at one point or another, and can't figure out why you're there.

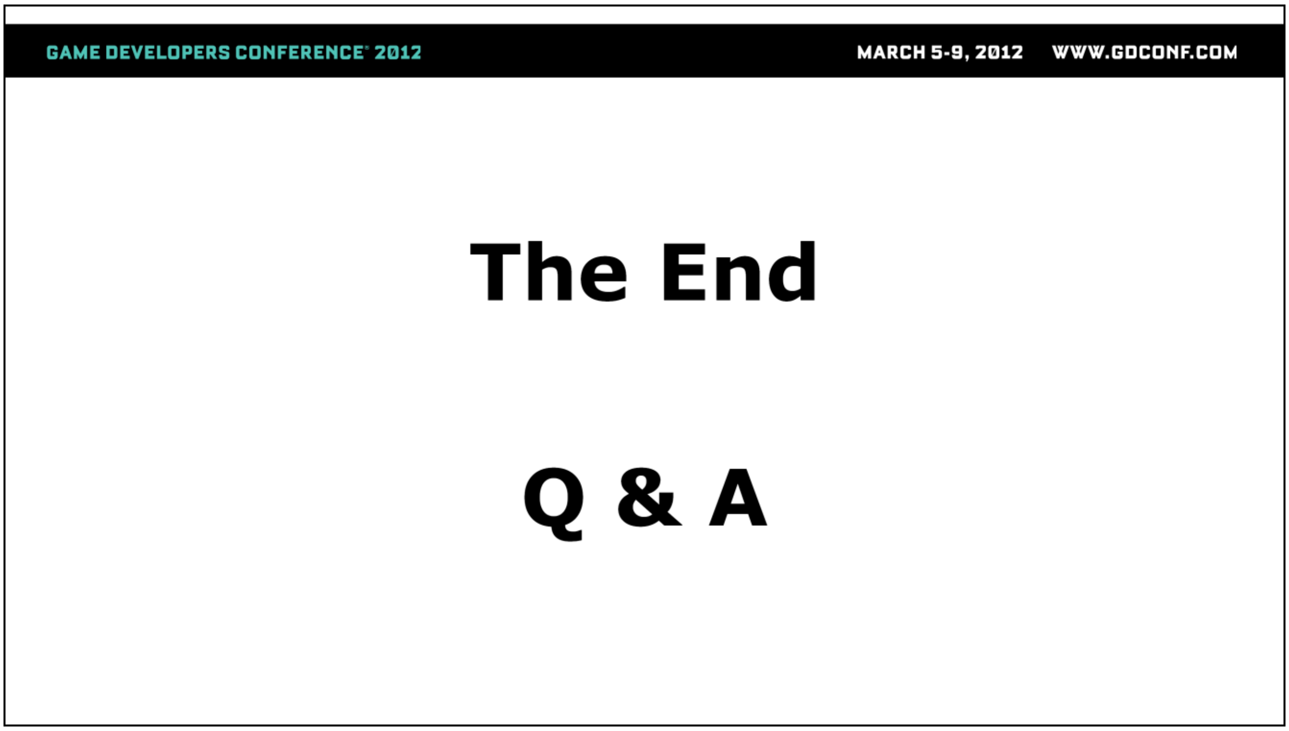
(The answer of course is that it's not a small company any more, and you're picking up a LOT of work they used to do, so that they can focus on keeping the company afloat).



Give Up Now!

You can't win!

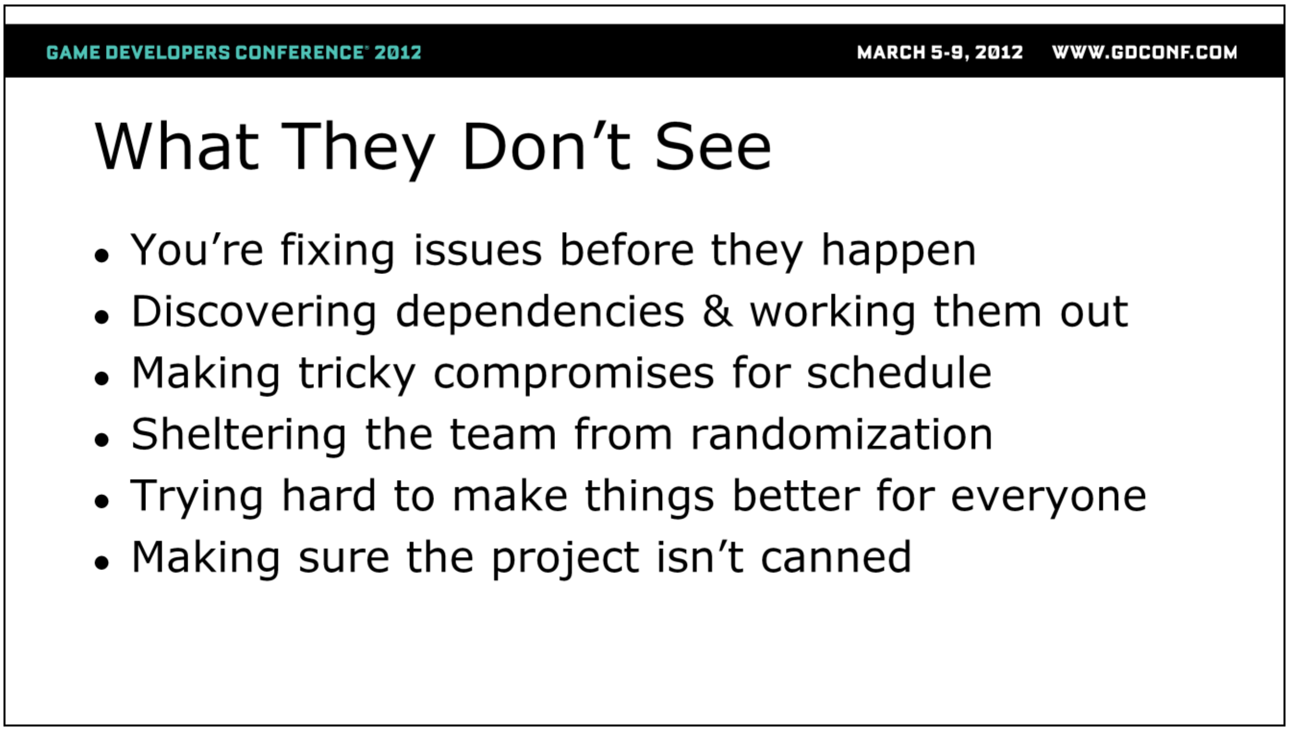
You can't fix this. It's impossible. I've seen it tried over and over again.



The End

Q & A

Only kidding!



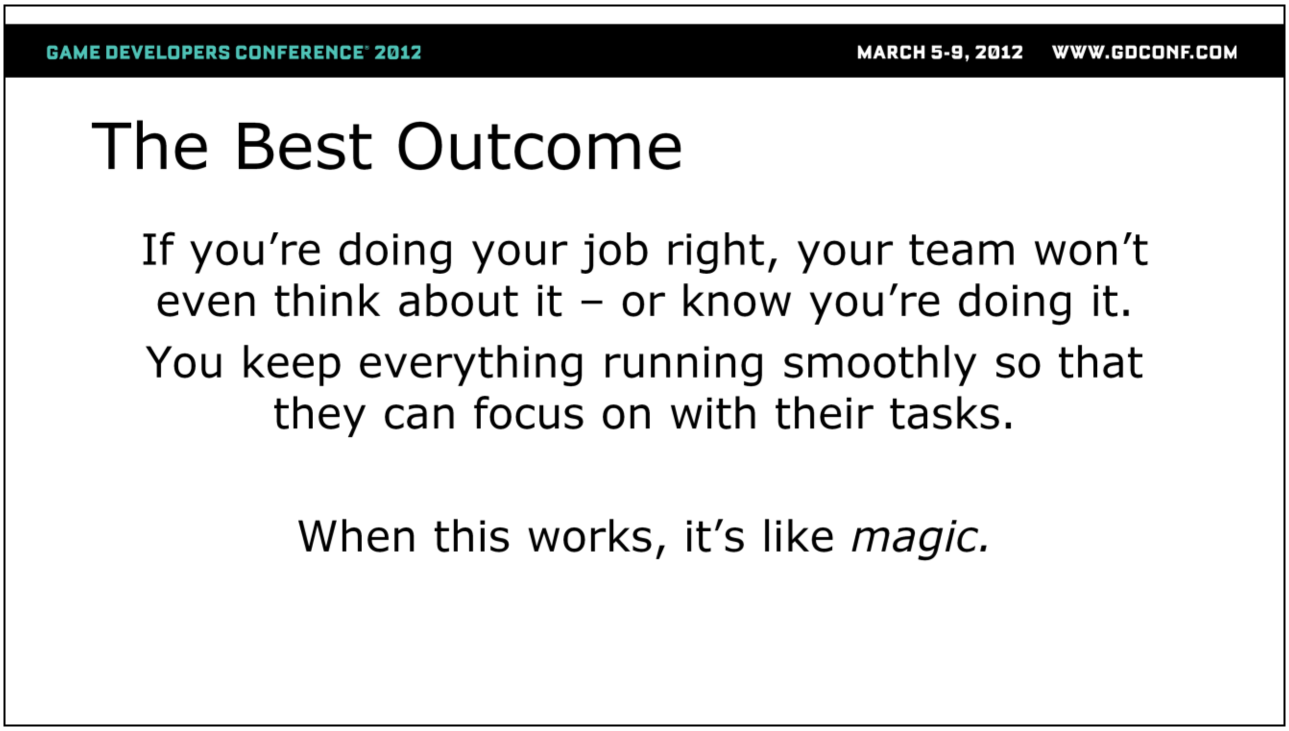
What They Don't See

- You're fixing issues before they happen
- Discovering dependencies & working them out
- Making tricky compromises for schedule
- Sheltering the team from randomization
- Trying hard to make things better for everyone
- Making sure the project isn't canned

All of these are things you do every day that are hard to stick your finger on and say "here! Look! I did THIS!":

Moving roadblocks, making sure the path ahead is clear, compromising on everything to make sure you hit the usually immovable schedule, fielding weird requests from execs and publishers alike, and generally, trying to get everyone through it all unscathed. It's a lot of stuff.

The annoying thing is… say you're sheltering the team from randomization – you'll bat away maybe 1000 randomizations and weird requests…. but you let through 3 of them. That's a pretty great score, all told, but the team never sees the 1000 things you moved out of the way for them – they only see the three you couldn't. And they think "Oh my god, you let three through!". This is horribly unfair, and a hard perception to change. And you *have* to let some through, or else the people pushing on you will start to see you as an obstacle rather than a team player…


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

The Best Outcome

If you're doing your job right, your team won't even think about it – or know you're doing it.

You keep everything running smoothly so that they can focus on with their tasks.

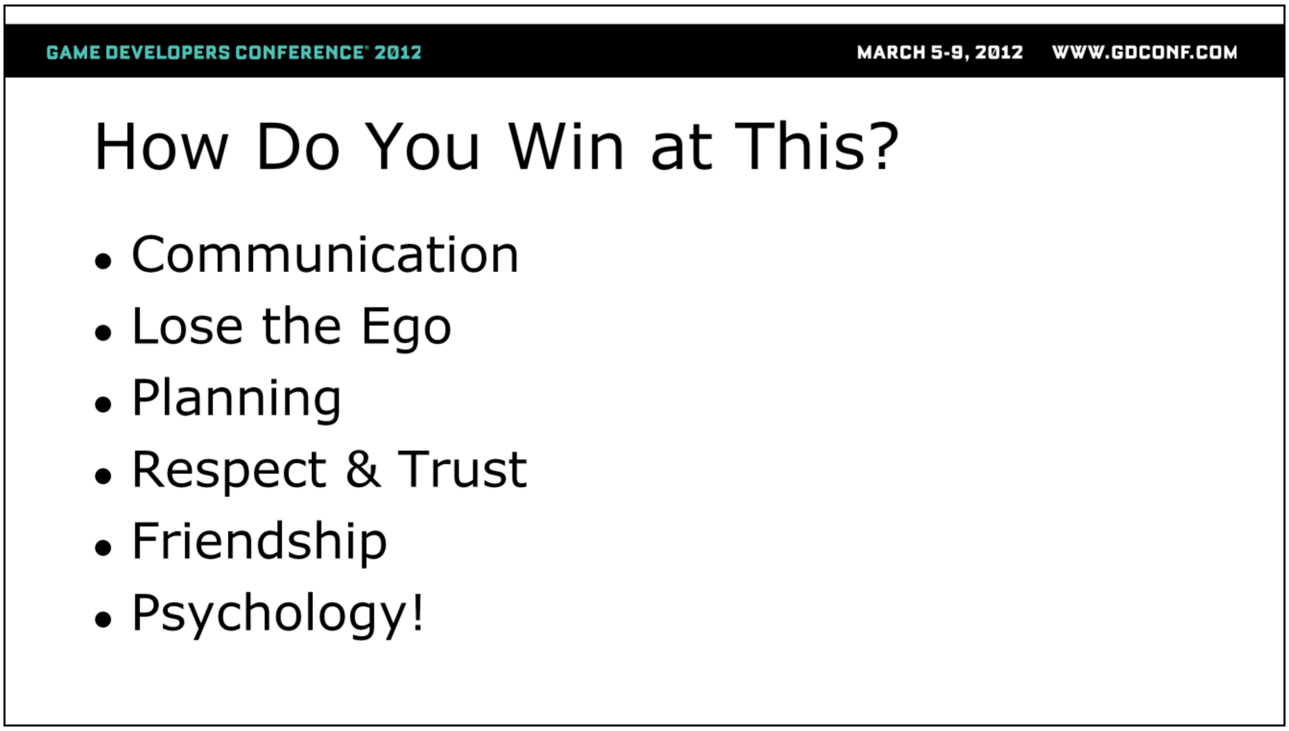
When this works, it's like *magic.*

This, by the way, is why I always – always – seek out the producers on a project when I join a new company. I know that they will make my life easier, my teeth whiter, and my job a lot happier to show up for every morning. And how do I know this? Because I *pay attention*.

(My wife Darci Morales – an awesome producer http://www.linkedin.com/in/darcimorales – wrote this slide. It's true).

One of the best producers I've ever known – Steve Freeman (ex-Sierra - http://www.linkedin.com/in/stevenjfreeman) – used to describe his job as one that if everything was 100% on track and going smoothly, you should be able to walk into his office and find his feet up on his desk, a cup of coffee in his hand, and a video game running on his computer. Because it's only when things *aren't* going smoothly, and things that are on fire, that you would ever see anything else.

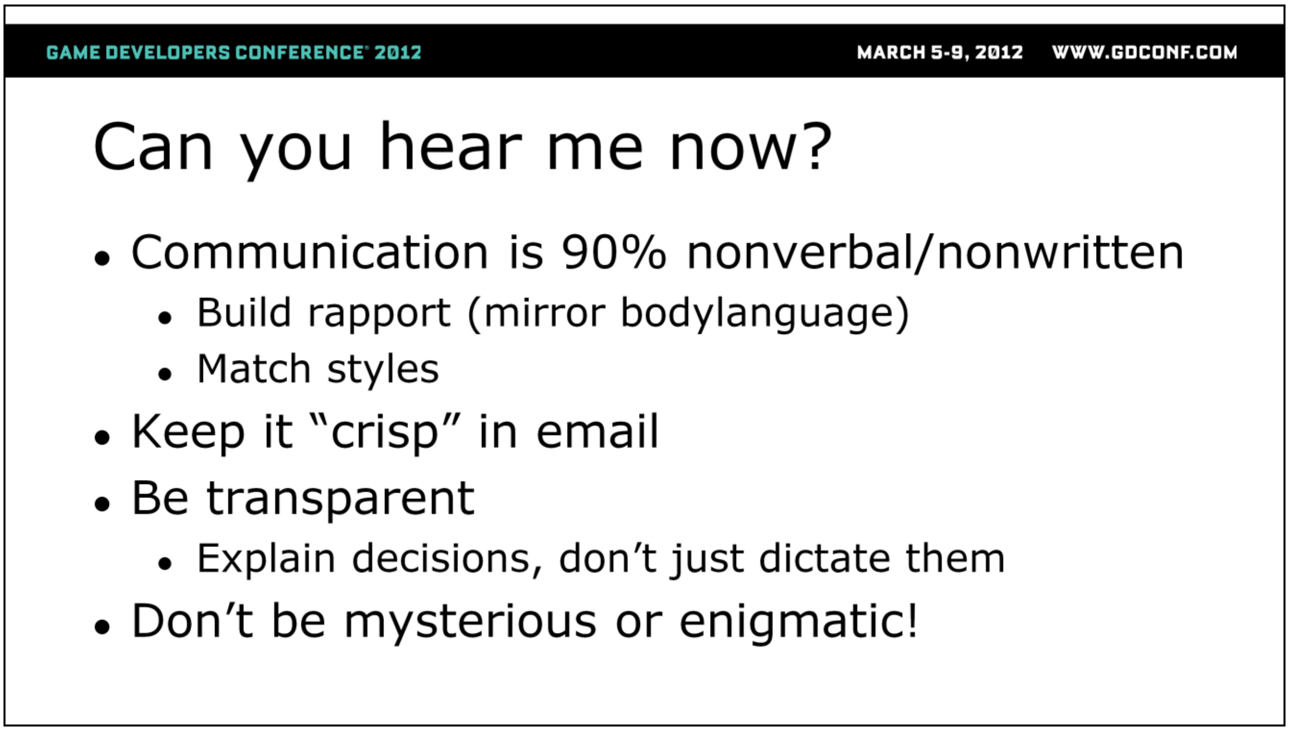
(With this explanation, he dismisses and handwaves away the fact that it’s a LOT of work and upkeep to keep things in this state… but to other people, that’s what it looks like).



How Do You Win at This?

- Communication
- Lose the Ego
- Planning
- Respect & Trust
- Friendship
- Psychology!

You can change people's perceptions, and pretty much every problem you come across is going to be one of Communication, in some form or another. So here's some things to throw into your Producer toolbag to help you along the way…


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Can you hear me now?

- Communication is 90% nonverbal/nonwritten
  - Build rapport (mirror bodylanguage)
  - Match styles
- Keep it “crisp” in email
- Be transparent
  - Explain decisions, don’t just dictate them
- Don’t be mysterious or enigmatic!

Communication is pretty much everything.

Rapport:

You can build rapport by mirroring body language - http://en.wikipedia.org/wiki/Rapport . Most people do this automatically, but if you’re feeling put-upon or threatened, you might find yourself not doing it. The trick is, force yourself. If they shift position, shift to match a few seconds later. But be subtle about it. Very quickly, you’ll be disarming them, and the conversation will flow much better.

Match Styles:

People have a variety of different communications styles – some are direct (“Be brief, be bright, be gone”), some value the building of a relationship via conversation as important as the conversation itself. Some people are linear thinkers, others bounce around from A to Q to D to Z. Figure out the communication style of the person you’re talking to - try to match your style to theirs. It’ll make things smoother.

Keep it Crisp:

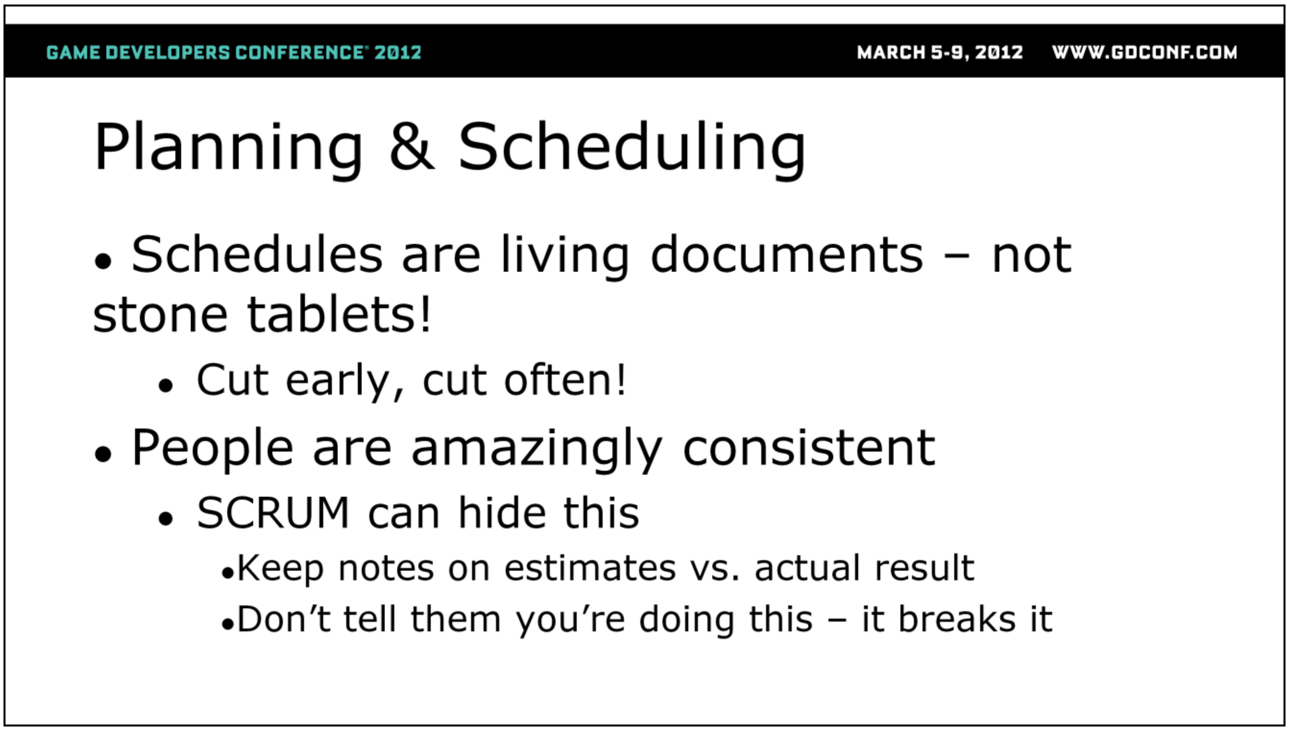
Assume that no-one ever reads more than the first paragraph of your emails. Front-load any important information there, or people won't read it.

Be Transparent:

It's hard to argue with a decision when people can see the conflicting factors you had to balance. They might disagree with the decision, but they'll be less likely to mess with it when they see why that particular compromise was made.

Don't be Mysterious:

The worst thing you can do is send off a meeting request without explaining why – it just makes people fear you. Worst example of this I've experienced was a Studio Head sending me an email saying "Meet me in my office at 11am tomorrow". He just wanted a chat. *I* thought I was going to be fired…


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Planning & Scheduling

- Schedules are living documents – not stone tablets!
  - Cut early, cut often!
- People are amazingly consistent
  - SCRUM can hide this
    - Keep notes on estimates vs. actual result
    - Don't tell them you're doing this – it breaks it

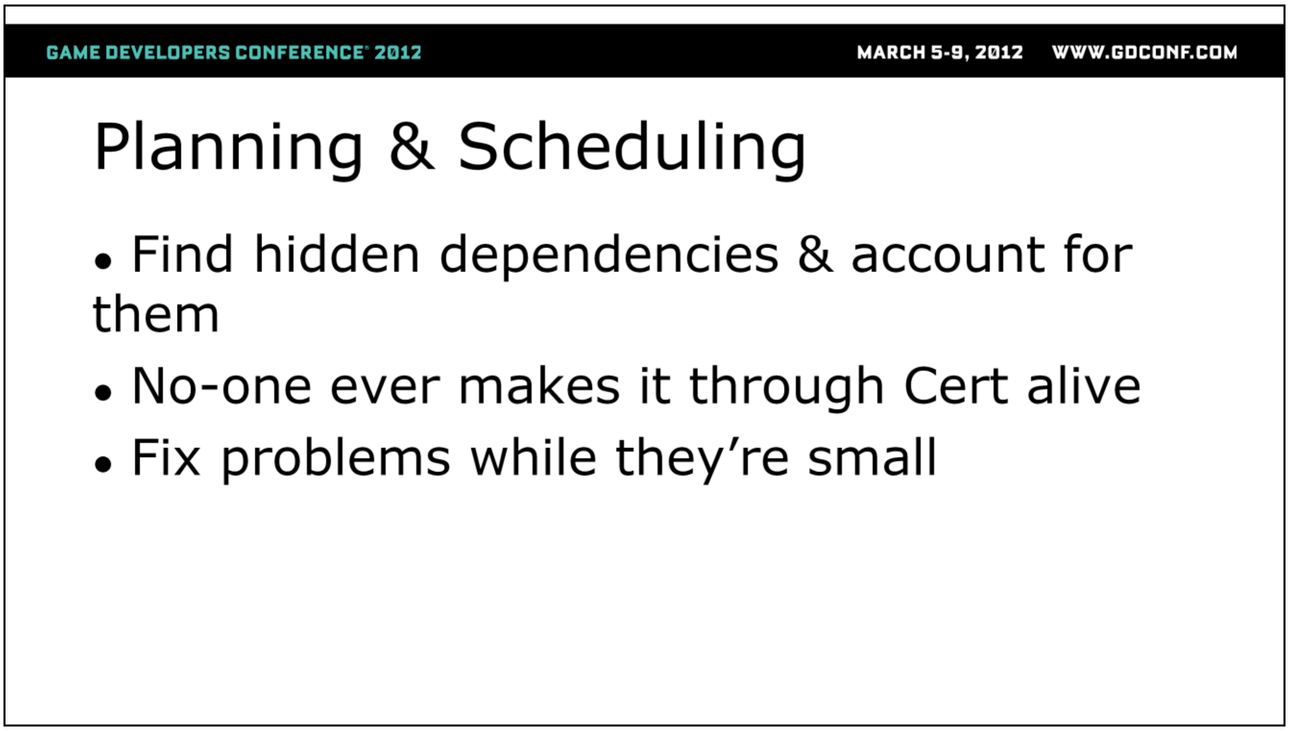
If there's one thing I can encourage you to know and take to heart, it's that schedules are living documents. They're not made to be written, and then adhered to unthinkingly.

No, seriously, you CAN change the schedule, and you should, regularly – or it'll be totally wrong and out of date. No-one has a crystal ball; at the beginning of a project, you'll know nothing about how it's going to turn out. Near the end? You'll know a lot more. Planning should match this - sticking to a bad schedule will cause huge issues. And if you have to make changes, you can even go to publishing/execs and offer them options, if you know what to do to switch it up.

People are very consistently wrong in their estimates, to the point where you can look at how long someone says it'll take to do something vs. how long they actually take, and calculate a fudge factor for the task.

But don't tell them what that fudge factor is… all this will do is futz with their estimates, and make them wrong *again*. Just collect the data, and work it into a schedule you keep in private. You'll find that schedule gets more and more accurate as time goes on, and you have more data to work with.

(SCRUM is horrible for this; depending on how you implement it, you'll lose that data… it also seems to encourage poor estimation up-front, by requiring split-second decisions on how long a task might take).



GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

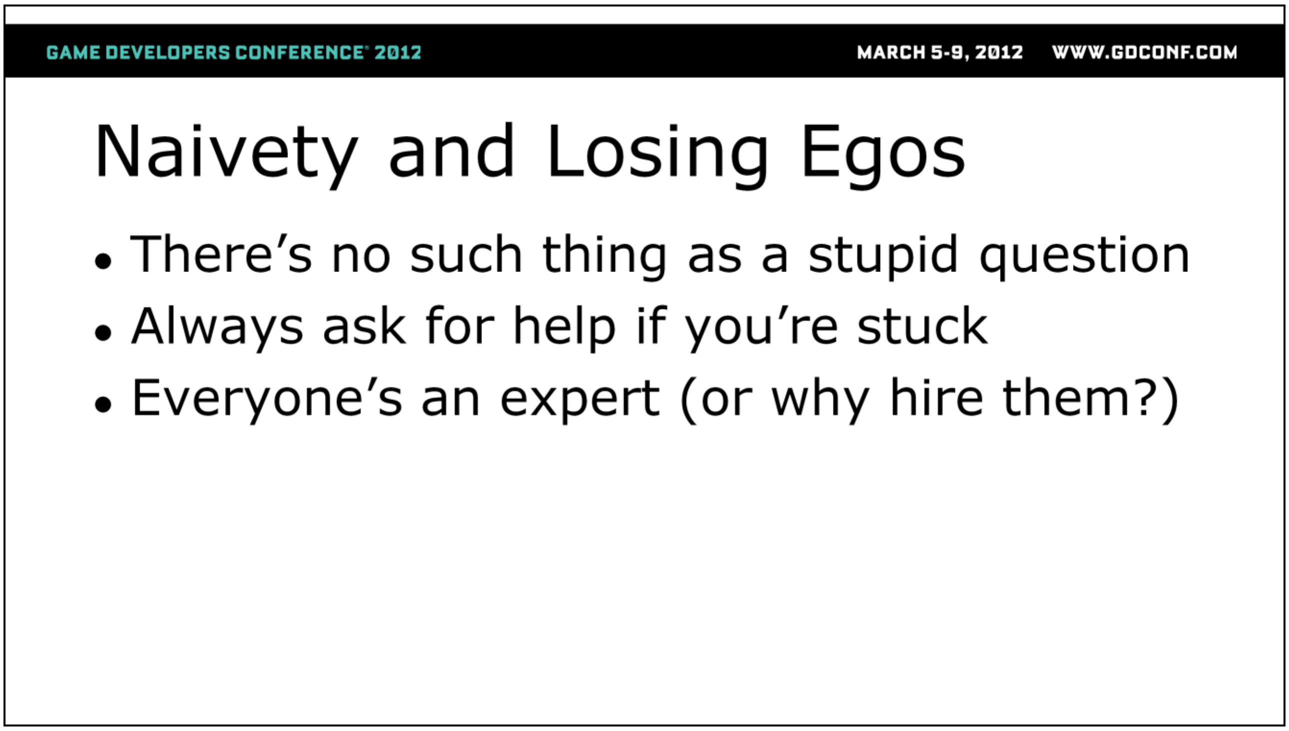
Planning & Scheduling

- Find hidden dependencies & account for them
- No-one ever makes it through Cert alive
- Fix problems while they're small

Everyone's in their own little hole, working on their own thing. You own the big picture, so you get to break it out and figure out the knock on effects. A programmer checking in a new feature "just because" at the end of a project has a knock on effect on all other departments – whether they think it will or not. You need to be figuring out the impacts.

Cert always takes 2 weeks. Most people fail at least once, every time. This means 4 weeks minimum. Just in case you fail twice, schedule 6 weeks for Cert.

Fix problems while they're small. ALWAYS. Don't put it off – they just snowball and get bigger.



Naivety and Losing Egos

- There's no such thing as a stupid question
- Always ask for help if you're stuck
- Everyone's an expert (or why hire them?)

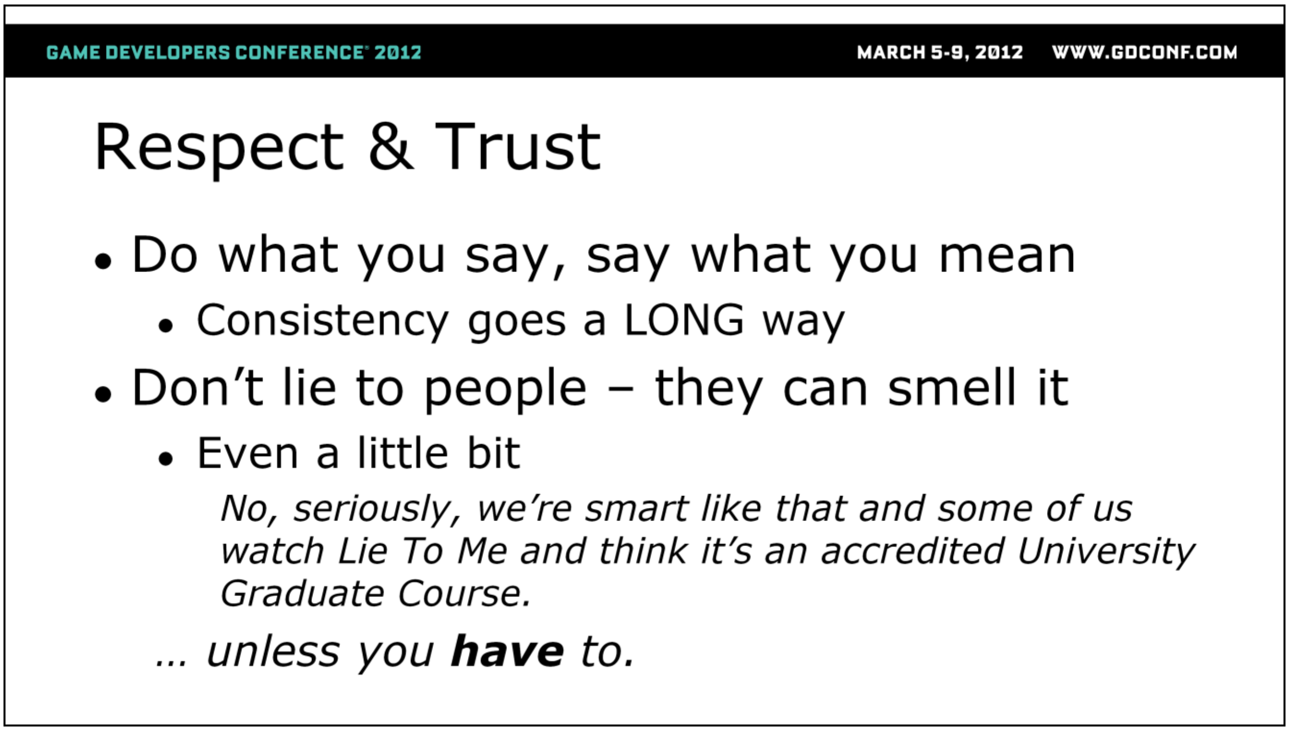
No such thing as stupid question – just bruised egos and feeling stupid.

Get over that it's pointless.

The best teams do this a lot in EVERY discipline. Collaboration is better than someone grinding their gears.

You hire specialized people for a reason – so that they know their stuff. If you ask questions, you'll learn more, and be able to ask better more intelligent questions. And experts love to talk about what they know.

If you don't, you'll just hit more and more problems.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Respect & Trust

- Do what you say, say what you mean
  - Consistency goes a LONG way
- Don't lie to people – they can smell it
  - Even a little bit

    No, seriously, we're smart like that and some of us watch Lie To Me and think it's an accredited University Graduate Course.

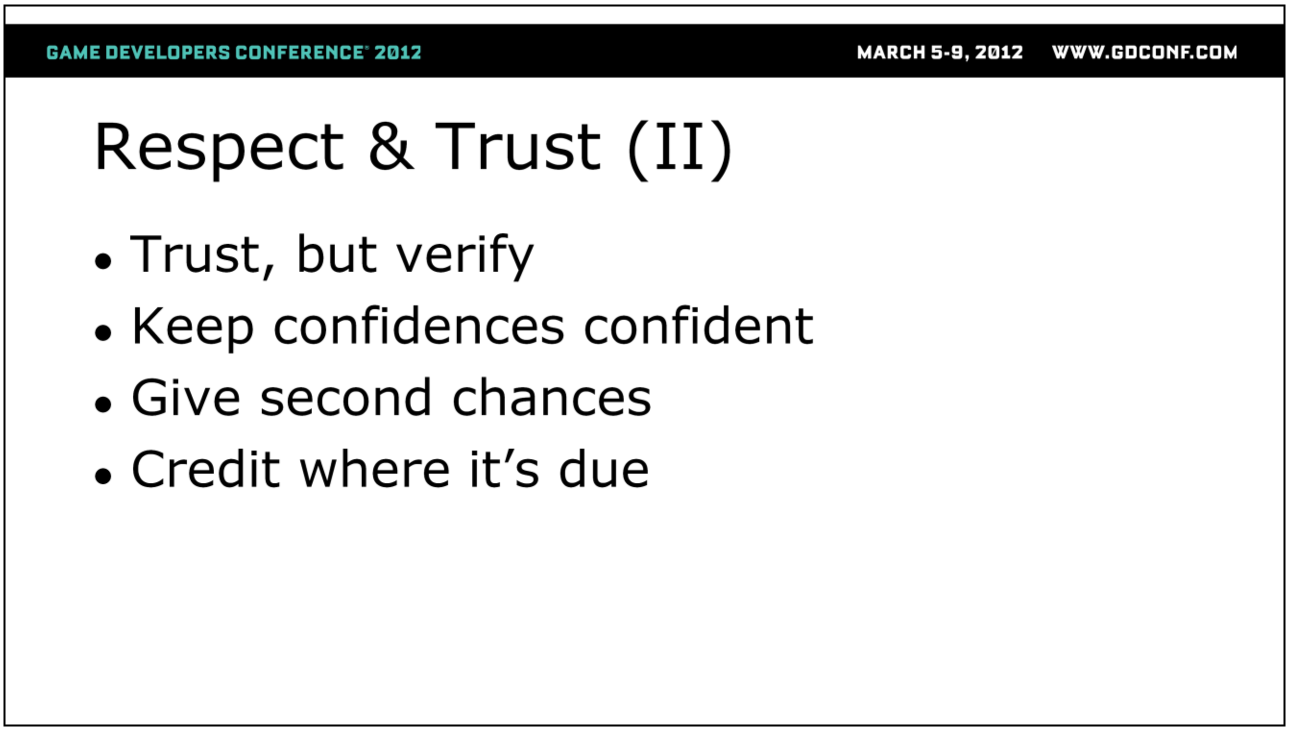
  *… unless you **have** to.*

Trust is another word for credibility.

Can't always be honest with the team about everything – information overload.

Scenario: Someone from team comes up and asks you a question about layoffs.

You know who is, and what's coming down the line. You can't be completely honest about it – because you need to protect the project. If in doubt, say you don't know or can't talk about it – delicately. Discretion!


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Respect & Trust (II)

- Trust, but verify
- Keep confidences confident
- Give second chances
- Credit where it’s due

Near the end of projects in particular, you’ll have people coming to you a lot, giving you information in private.

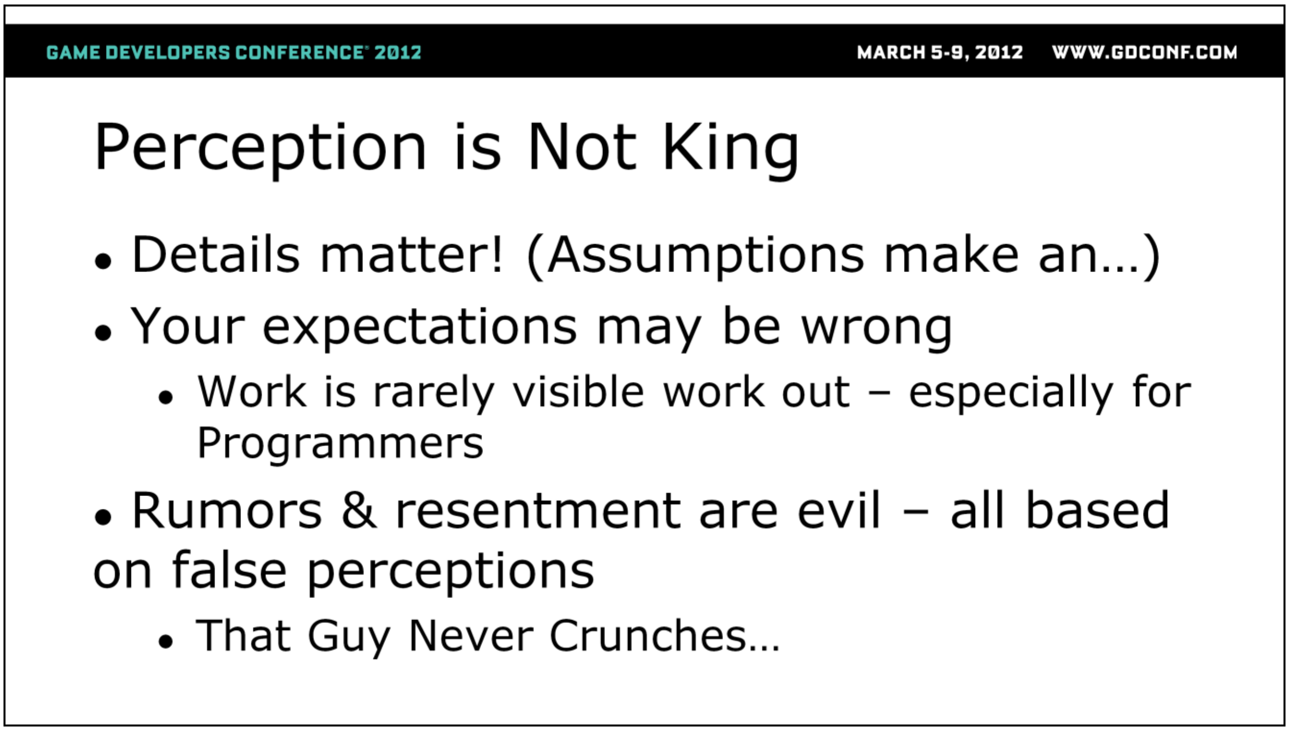
If this is people bitching about others on the team, trust that they’re being honest, but they may be over-inflating the problem (or misidentifying it). VERIFY first, and try to find a win-win solution for everyone – never go off half-cocked and shoot the person who is being blamed without looking into it.

Similarly, you’ll probably be privy to a lot of other kinds of information (people wanting to leave, interpersonal issues etc). Never break that confidence – you can provide information regarding it to others to affect change, but do it in a way that doesn’t identify the confidant – or people will very quickly not talk to you about important things ever again.

If you don’t give second chances, we have a name for what you are – a psychopath. Everyone makes mistakes, and EVERYONE in the moment tries to do the best they possibly can in the situation they’re in. Occasionally they get it wrong.

Everybody does it. So give second chances – people learn from mistakes, and improve. Consider it an investment in the future.

Credit where it's due: Always give credit for things, and always credit it to the people who did the work. This reflects well on them, and ***on you***. If you steal credit – or people in the trenches never get it for their work, it'll go sour very quickly, and people will hate you.



Perception is Not King

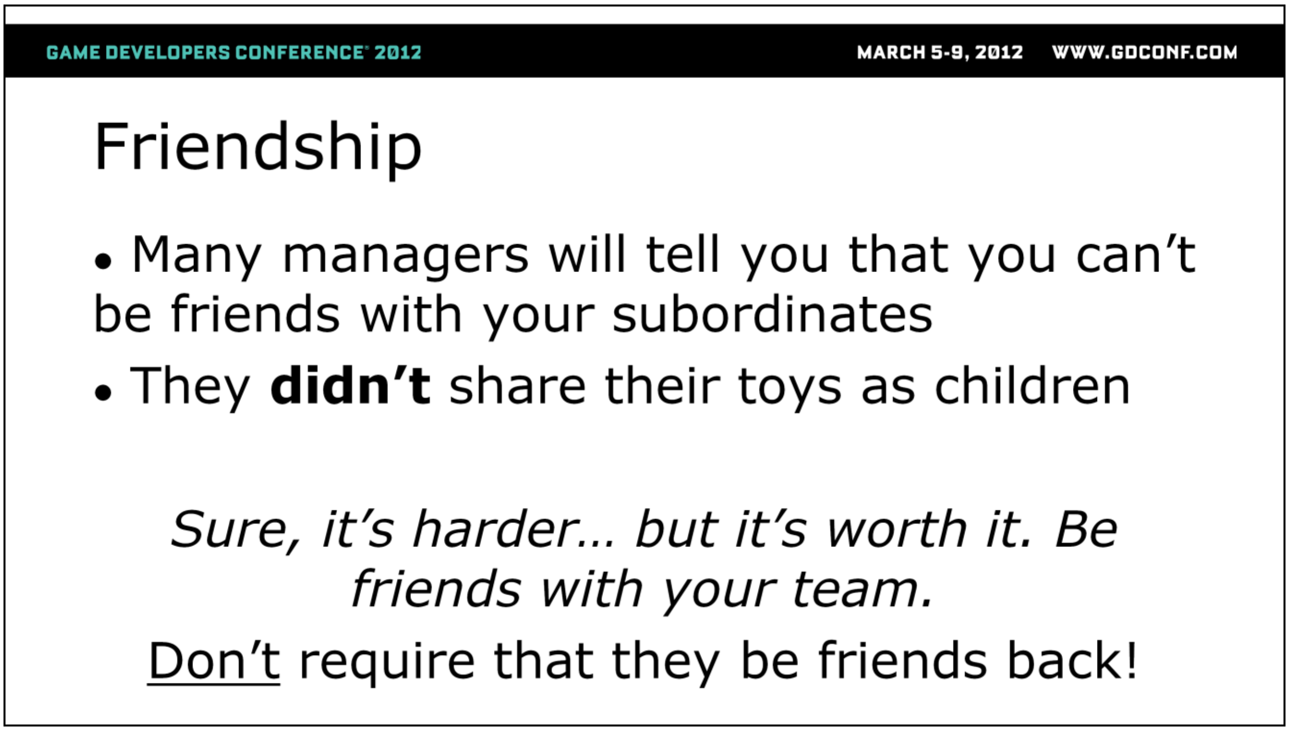
- Details matter! (Assumptions make an…)
- Your expectations may be wrong
  - Work is rarely visible work out – especially for Programmers
- Rumors & resentment are evil – all based on false perceptions
  - That Guy Never Crunches…

One guy on a team I ran was awesome at writing clean code. I put him in for a very high score on that in his review, and it was knocked back to me, with a claim that he wrote the most ***buggy*** code on the team. I went through every bug in the database with my manager, and showed exactly what each one was, and what happened. None of them were caused by **his** code. The only way I could do this was because I always paid very close care and attention to what was going on with my team.

Also, most people expect work to progress steadily on a project, from beginning to end, in a straight line. This is false; effort expended is sinusoidal, with people crunching, working hard for bursts, and then relaxing. And especially in programming, tasks rarely close out until they're completely finished. So you see very little happening at first, and then a huge rush at the end as things come online. This is normal – but it scares execs.

One of the ways Unreal has done really well in the market is by allowing the art team to rush ahead of programming and show continual progress on the project. This is great – until you're ¾ of the way through the project, and all of a sudden, you start slowing down as the art changes become smaller, and programming has to catch up. Still, it's useful – because otherwise you have to explain the way projects work to business folks who might not agree…

The guy who shows up at 6am and leaves at 6pm might be resented by the other guys who show up at 10am and leave at 9pm. But that 6am guy is doing more work, and crunching harder – they're just on a different schedule and never see it. Watch for this kind of thing and counter it.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Friendship

- Many managers will tell you that you can't be friends with your subordinates
- They **didn't** share their toys as children

Sure, it's harder… but it's worth it. Be friends with your team.

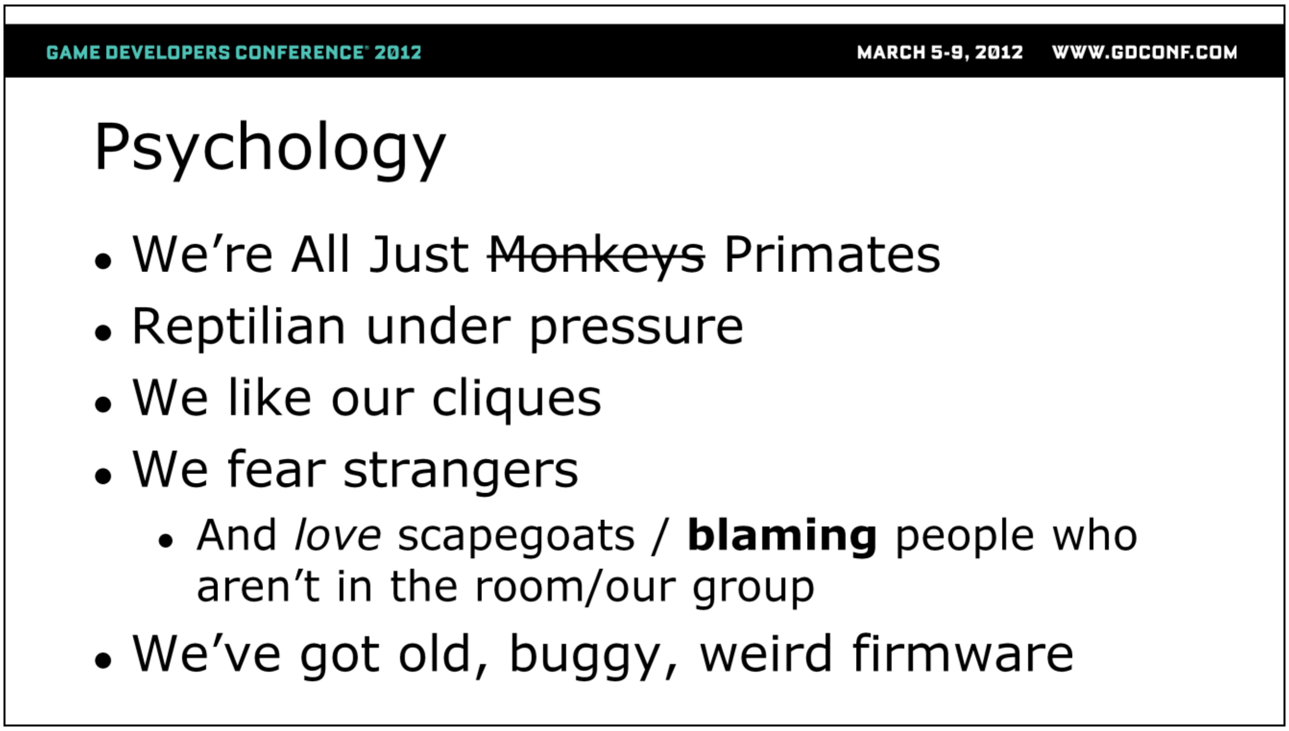
<u>Don't</u> require that they be friends back!

This is the biggest pile of bullshit I've ever heard. And okay, so maybe that "not sharing toys as children" comment goes a bit far… but still

Friends pull together for one another. Why wouldn't you be friends with the people you work with? The only reason I've ever been able to come up with is to protect yourself from having to make hard decisions which could hurt your relationship with your friend.

The solution to this is simple. Set the tone of your conversation at the start.

And just because you want to be friends with everyone on the team, doesn't mean that everyone will be your friend. Your relationships with people might suffer because of your responsibilities to your position.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Psychology

- We're All Just ~~Monkeys~~ Primates
- Reptilian under pressure
- We like our cliques
- We fear strangers
  - And *love* scapegoats / **blaming** people who aren't in the room/our group
- We've got old, buggy, weird firmware

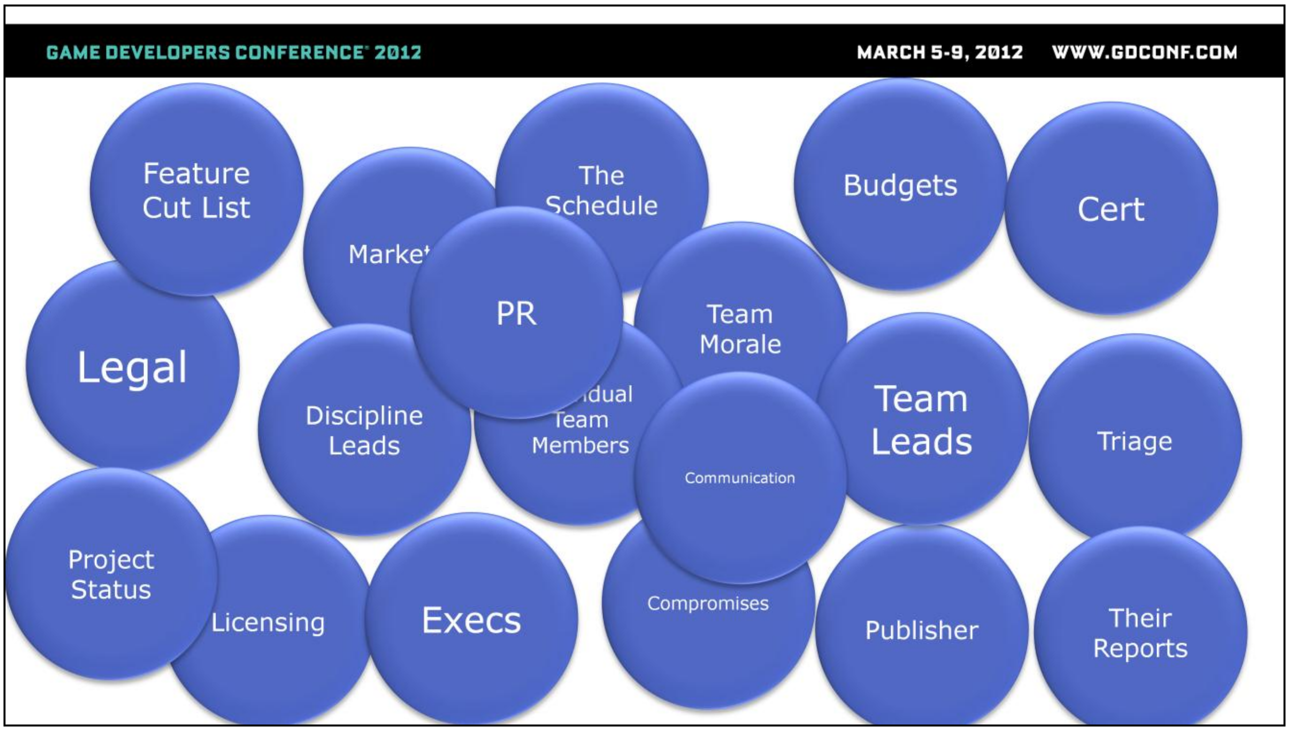
I'll come back to this one in a bit. But an ounce of psychology will get you VERY VERY FAR in this industry. Especially if you're willing to play staff shrink.

Oh, and strangers in this context = anyone who isn't a **close** friend, especially during crunch. When people are under pressure, they retreat to their safe group of compadres. Everyone else can and may get thrown under the bus as a result. Watch for it.

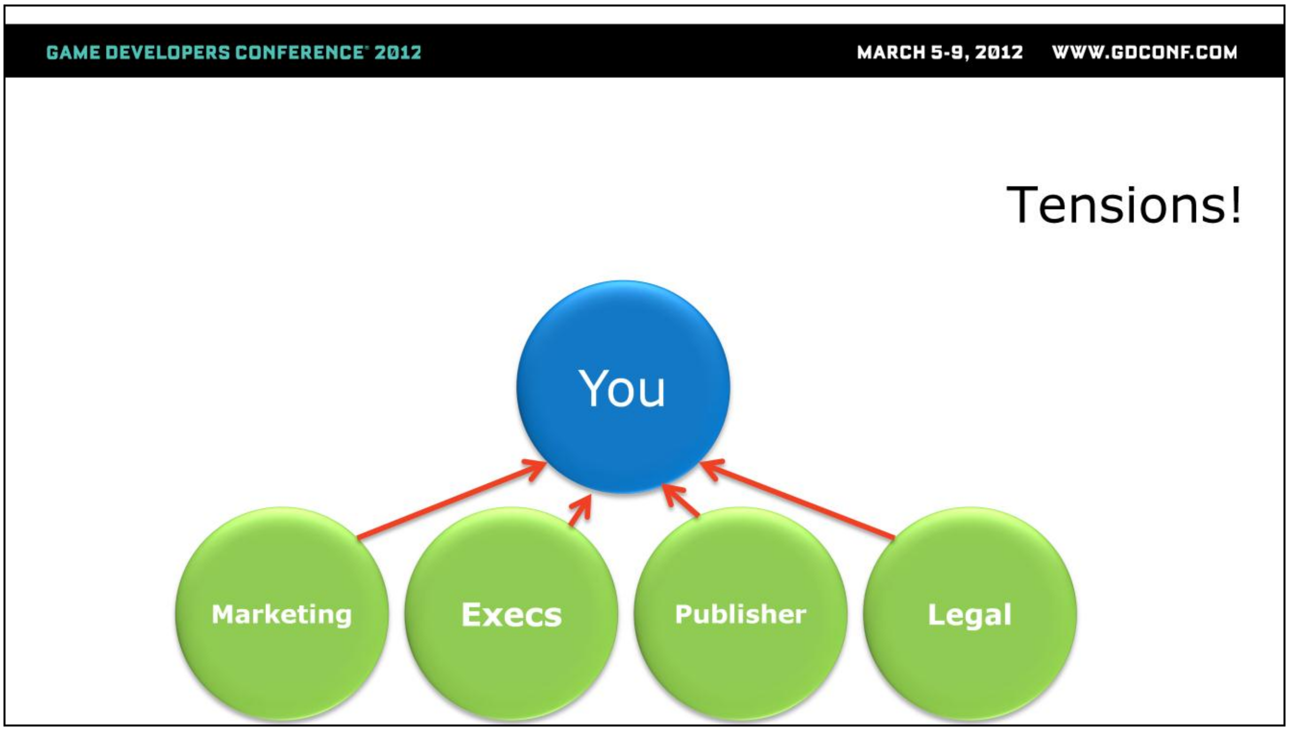
And our firmware doesn't make sense. It was designed for tribal villages and groups of apes running around in forests millenia ago. But if you know where the holes are, you can plan for it.

small teams

Here's a few things you'll see working on small teams…



This is all the things that, at varying points during a project, you as a producer may need to juggle. Scary, huh?



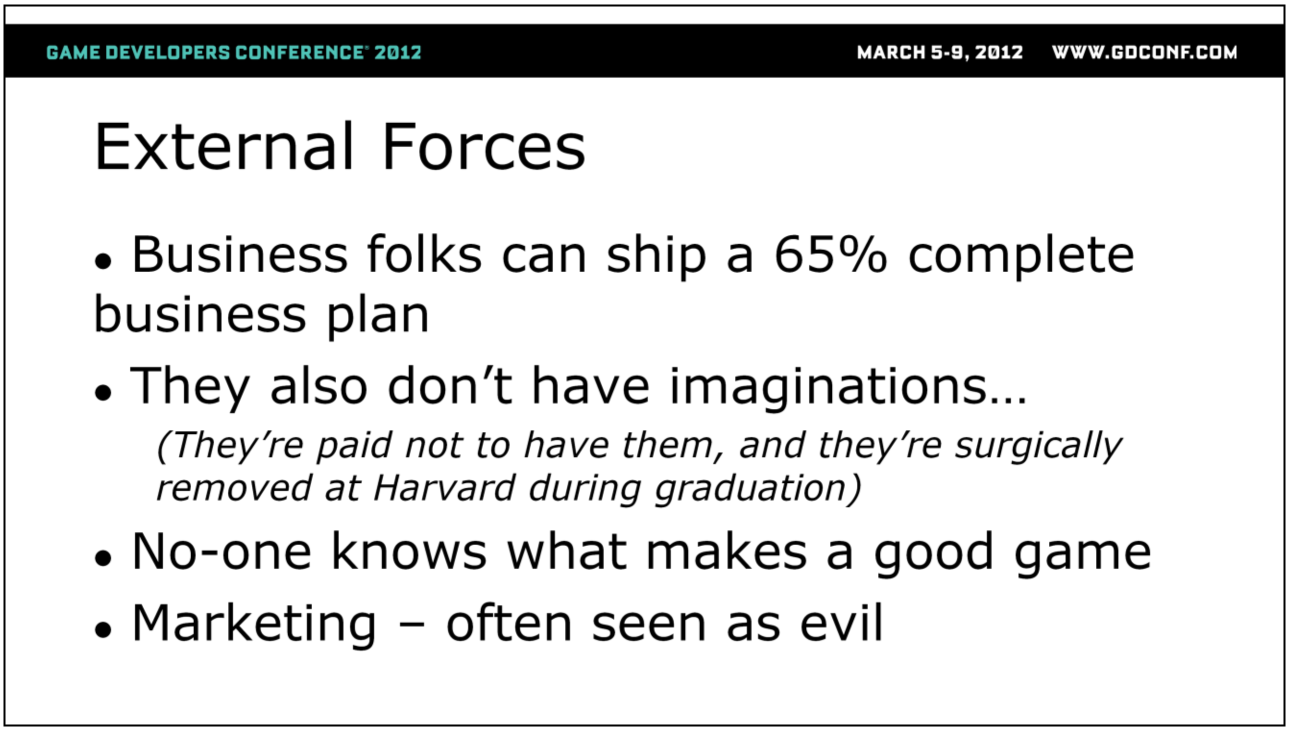
GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Tensions!

Let's break down the tensions you'll see on any given team…

Here's the external ones – the ones you generally don't have much sway over.

Except you do! These are ***business*** relationships, which means you *can* and *should* <u>***negotiate***</u> with them! It's just a different relationship than the one you'll have with your team. Treat them as customers/investors, and you'll have a much better time of things. And work collaboratively - they want to succeed as much as you do.



GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

External Forces

- Business folks can ship a 65% complete business plan
- They also don't have imaginations…
  (They're paid not to have them, and they're surgically removed at Harvard during graduation)
- No-one knows what makes a good game
- Marketing – often seen as evil

You can ship a 65% complete business or marketing plan and still succeed. You can't ship a 90% complete game – it doesn't run! So you may need to educate business folks on this – if you don't, you'll just be left scratching your head at some of the weird requests you get and start flipping the bozo bit on them. They're not – they just do radically different things to you.

No imaginations – hard data rules. Business runs on hard data. So be ready to paint pictures, show demo's, storytell, and more to get your point across and build consensus. Expecting them to fill in the blanks won't work.

No-one knows = minimize risk, and stick to what works…

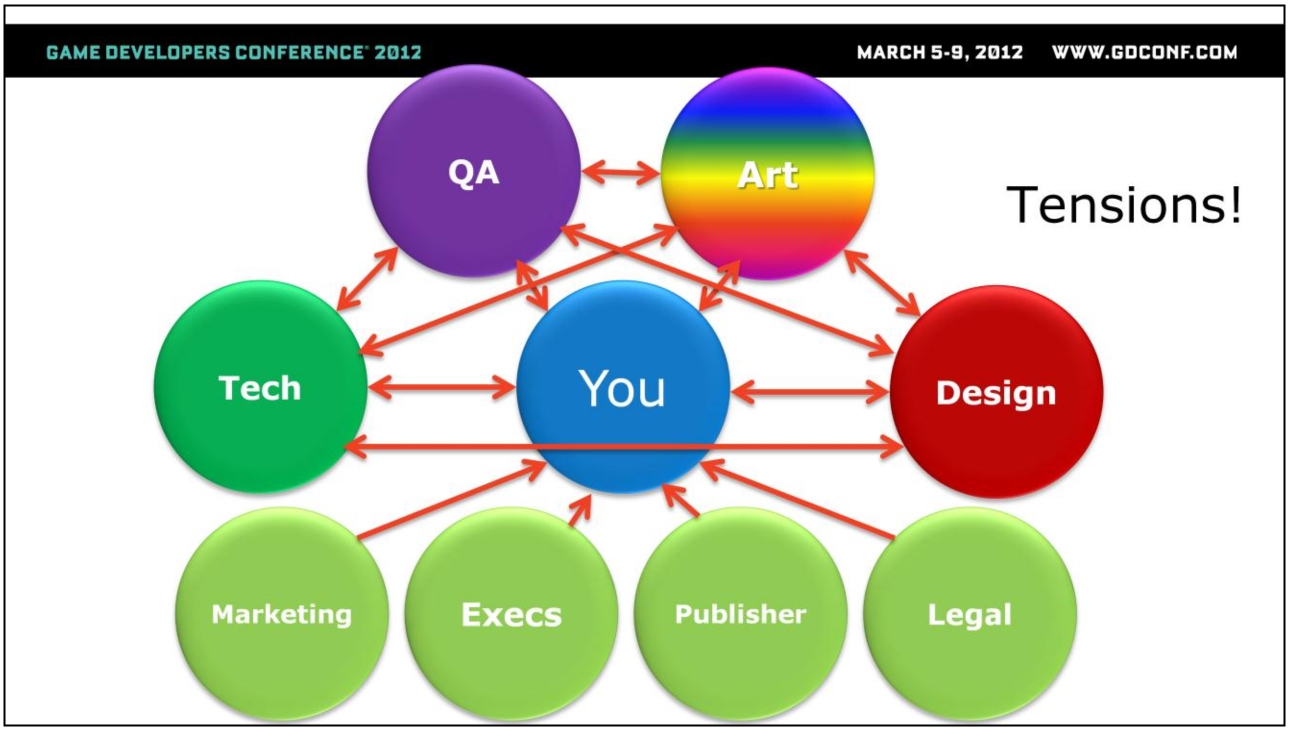
What we do know:

- Franchises sell – by about the 3rd one in the cycle, user awareness is there to build a snowballing following.
- Multiplayer appears to increase games sales – possibly by cutting off the used game market at its knees if your friend

wants to play with you

- There’s more… talk to your business folks, listen closely, and see what you can do if it makes sense for your game. But ask them for the data so you can verify it yourself.

Marketing – they’re your friends, seriously. But you can only do it if you collaborate closely with them, bring them in early, and recognize that they have to sell the game to various people, including retailers, distributors, and so on – not just end users (who are ***your*** focus). Build a good relationship, collaborate, challenge them where necessary, and make them a part of your team – remember, their budget is probably at least as large as yours for building the entire game. It doesn’t have to go poorly.



You're in the middle, and you act as the hub for all the teams.

Of course, they're all talking to each other anyway.

Design wants an environment you can enter a building in, with interactive objects inside.

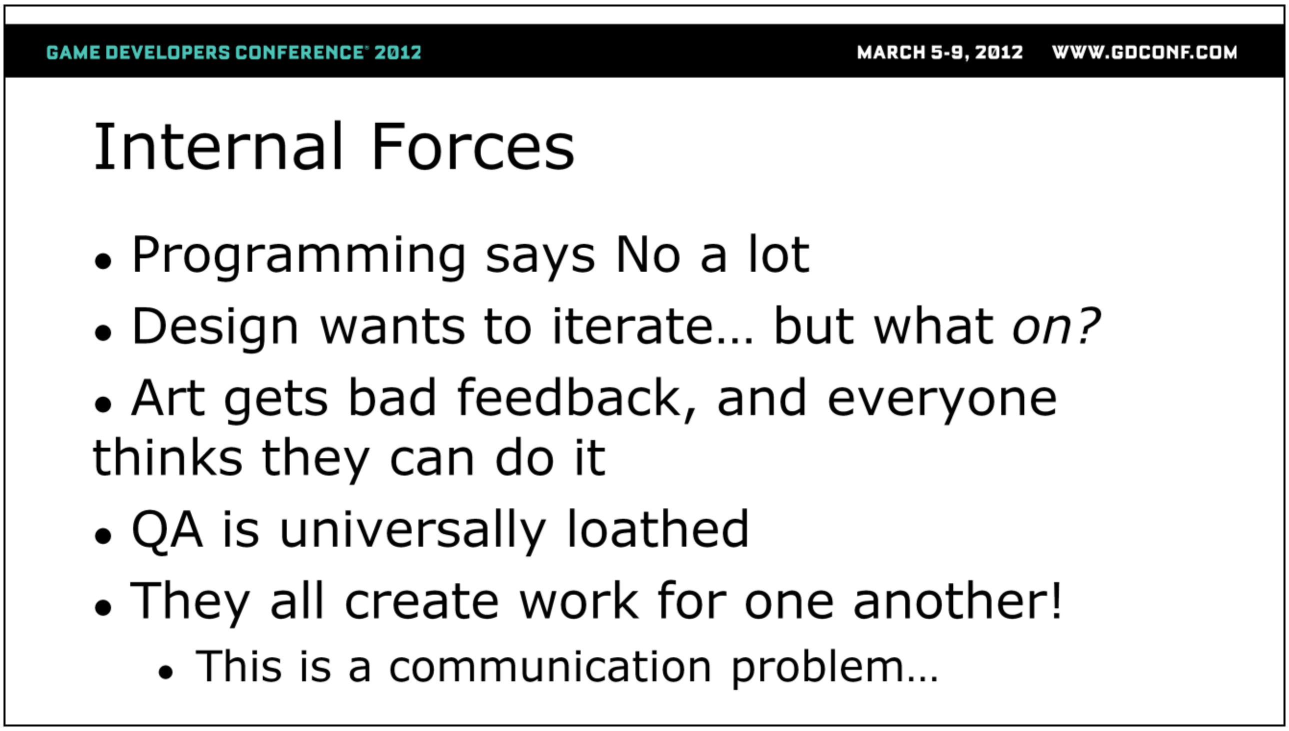
Art says we don't have the tech to do that for this game. Talk to programming.

Programming says oh, but we do!

Art says Argh! No, we don't!

… all of this is just a communications issue.

The problem is that any demand from any other team is seen as adding more work. Helping other teams is seen as adding more work. ALL TENSIONS are caused – in one way or another - by some other team not thinking carefully, and accidentally (or inconsiderately) adding more work for other teams. If you can insert yourself into this mix, and balance requests and regulate how this happens without stemming impromptu collaborations between teams, you’re ahead of the game.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Internal Forces

- Programming says No a lot
- Design wants to iterate… but what *on?*
- Art gets bad feedback, and everyone thinks they can do it
- QA is universally loathed
- They all create work for one another!
  - This is a communication problem…

Some other things you'll see:

Programmers have to say NO a lot. They're the guardians of consistency and coherency, and have to make lots of things work *together*, which is harder than making them work at all. So we get used to saying No early – and forcefully – to keep things on track. Except eventually, we start doing this automatically and that's bad.

Design often provide a very fuzzy up front design, and want to sharpen it by iterating. But if the initial design is too fuzzy, there's nothing concrete there to iterate on. This can be very frustrating for programmers and others. The best designers I've seen know how to concretely and completely specify at least v1 of a game design/feature before throwing it over the wall. This gets harder if you're running fast trying to keep up with pressure from the schedule.

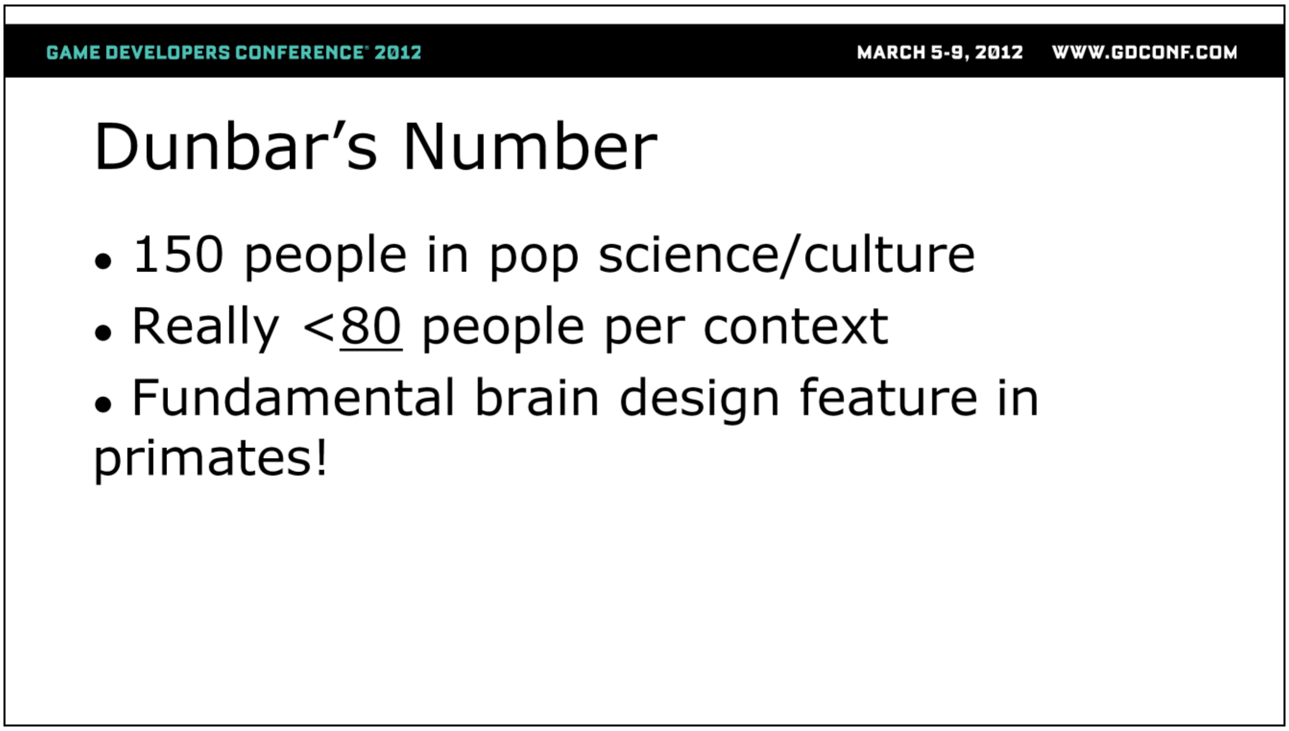
Art – everyone can see it, so everyone has an opinion. Even if that opinion is "Those pants look *assy"*. Make sure any art feedback is appropriate, well thought out, and has more depth to it than just personal taste.

QA is always hated – which is sad, because they should be your ***best friends***. They’re there to make YOU look good.

GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

BIG TEAMS

What happens if you grow? From maybe 40 people to 120?



GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Dunbar’s Number

- 150 people in pop science/culture
- Really <80 people per context
- Fundamental brain design feature in primates!

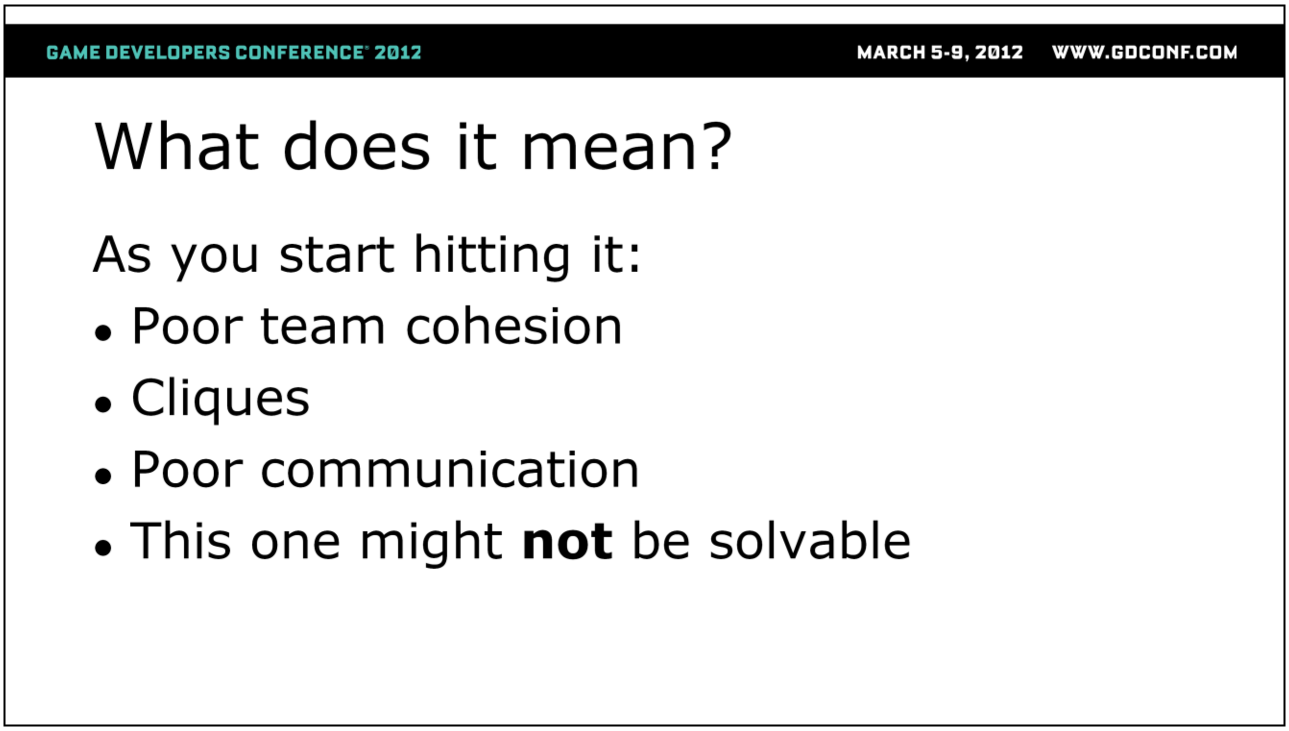
Okay, now you’re screwed, because you’re going to begin to run up again Dunbar’s number.

http://en.wikipedia.org/wiki/Dunbar's_number

It’s the number of people who you can know in a given context and consider to be “friendly” rather than strangers or an enemy. And it’s built into our brains – it’s a function of cortex size in primates.

The popular science version of this is 150 people, but that’s wrong – because that’s for situations where your survival depends on other people. The actual figure you should be using for companies and teams is about 80.

Different people have different capacities; from experience I can tell you from watching who I knew while our team grew on This Is Vegas that mine tops out at about 98 people (I made it my mission to know everyone). My wife’s? I’ve not seen it top out yet – she goes to at least 128.

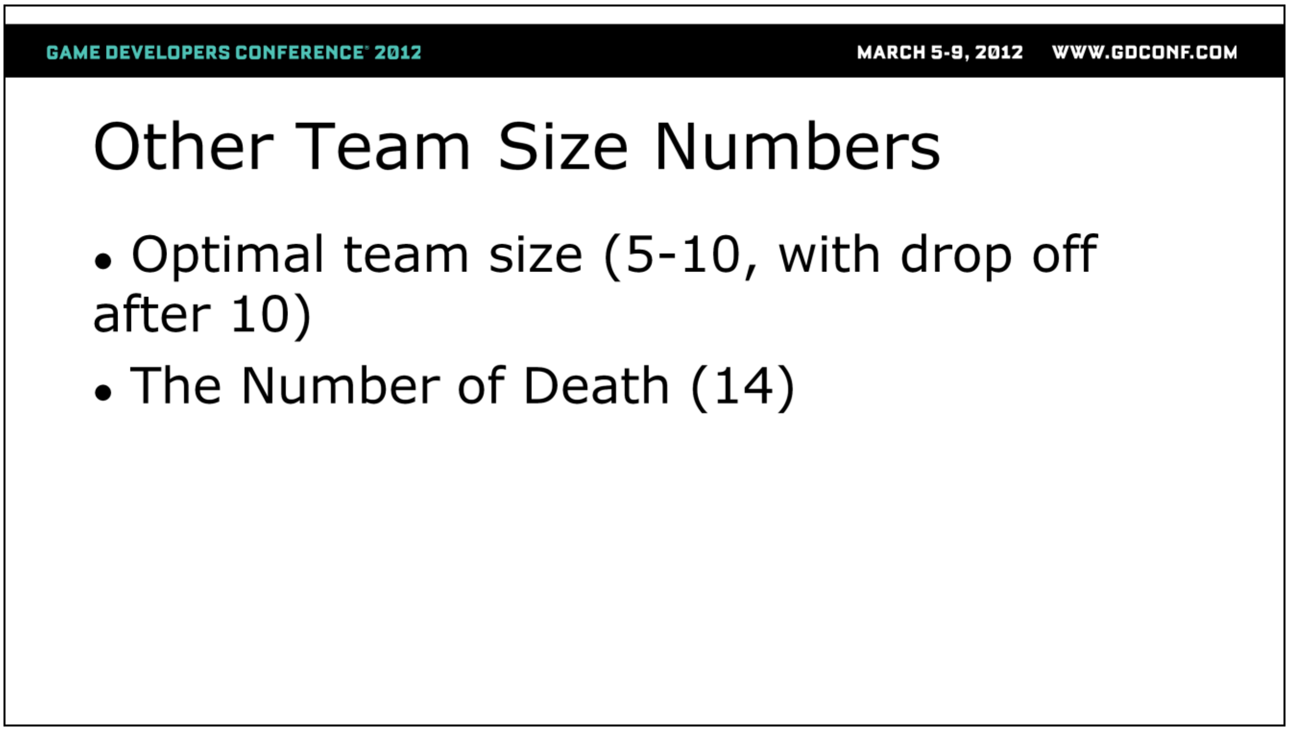

GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

What does it mean?

As you start hitting it:

- Poor team cohesion
- Cliques
- Poor communication
- This one might **not** be solvable

Unfortunately, this can cause lots of problems. The only way around this is to break teams into smaller groups, with structured communications between them, and processes in place to lessen its effect.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Other Team Size Numbers

- Optimal team size (5-10, with drop off after 10)
- The Number of Death (14)

Teams are weird… here's how they work.

1 person – not a team. Just a solo rogue agent.

2 people – they will never agree (usually), or they'll work well together, but it'll require a huge emotional/energy commitment which limits productivity.

3 people – that's one solo rogue agent, and two people who override them and agree totally. (This is why the military sends people out in groups of 3 on missions – two snipers to hit the target, and one whose job is solely to watch their backs and make sure they're not discovered).

4 people – that's two opposing groups of two.

Once you hit 5 people, things start motoring – now you actually have a team dynamic. This works great until about 10 people, and then things rapidly tail off until you hit 14.

No-one know why 14 is a bad number, but you end up with people who just plain don't agree on anything, and don't work together well at all. It's actively detrimental. And then it crawls back up. It's the uncanny valley for team sizes. Once you get past this number, it's probably because you've split the group into multiple teams.

(Brook's Law may possibly account for this - # of communciation pathways = $n(n - 1) / 2$, where n is the number of people).

They figured this one out by watching groups of people playing MMOs – specifically, Ultima Online.

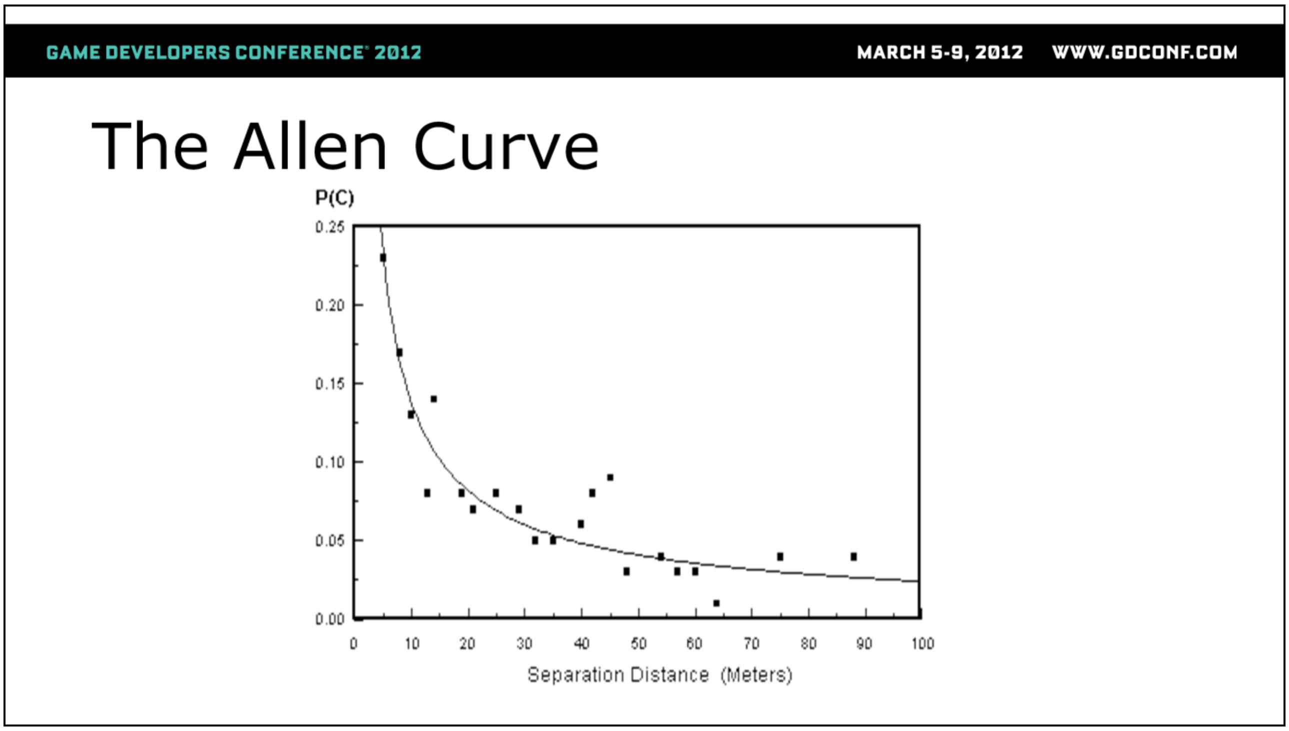
Note: these figures aren't hard and fast rules – they're kind of fuzzy, and your mileage may vary.

Note: before putting presenting this talk, I knew that 8 was also a bad number in some circumstances, but couldn't find data to back that up. Now I have:
http://www.santafe.edu/media/workingpapers/08-12-055.pdf . King Charles I's committee of state had 8 members. They didn't get anything done. 8 is a bad number for a mix of decision makers who aren't necessarily collaborating, and have equal stake. That's one to **avoid** for meetings.

http://en.wikipedia.org/wiki/The_Mythical_Man-Month

http://www.lifewithalacrity.com/2004/03/the_dunbar_numb.html

http://www.santafe.edu/media/workingpapers/08-12-055.pdf - Parkinson's Law Quantified: Three Investigations on Bureaucratic Inefficiency; Peter Klimek, Rudolf Hanel, Stefan Thurner

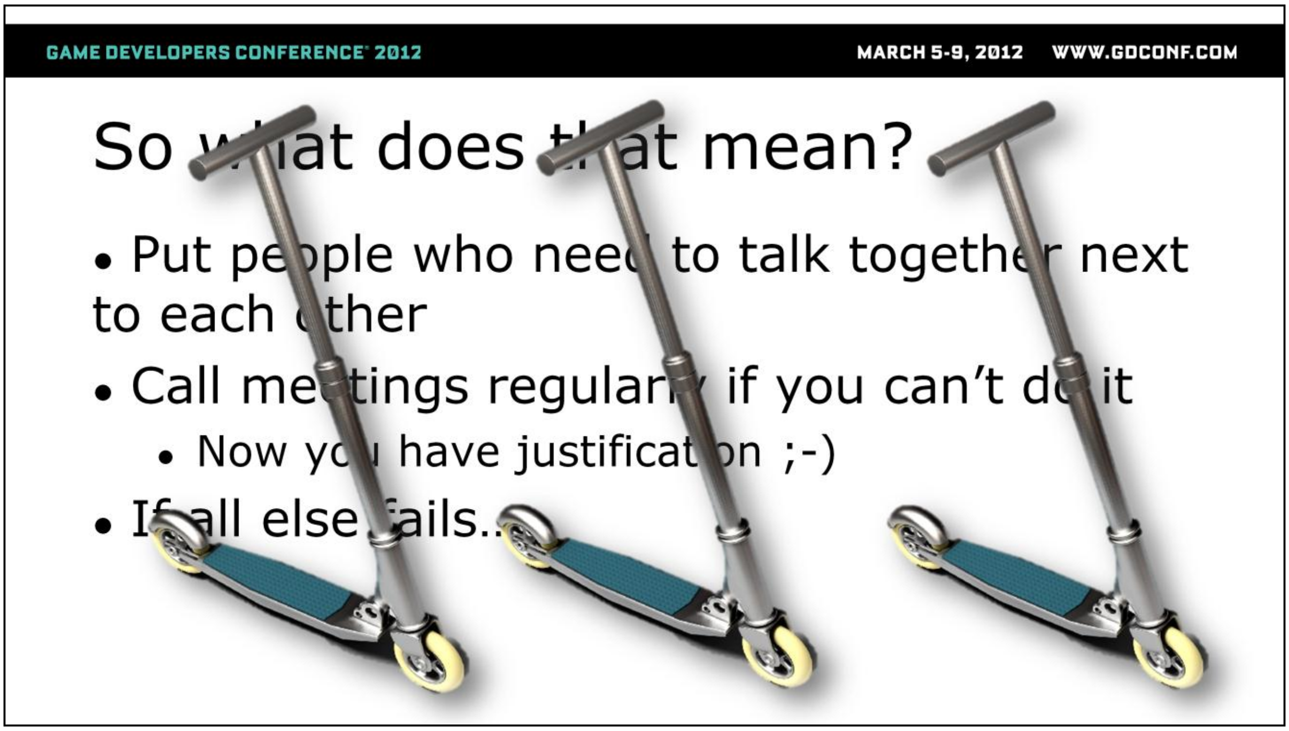


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

The Allen Curve

While we're on the topic of random numbers affecting communication… So this guy Professor Thomas Allen did a study at MIT on engineers shows a reduction in communication with distance. And it's not helped by email, text messages, Facebook or anything else – with a few exceptions, you want people who need to talk to each other to sit less than 50m away from one another (164 feet).

Unless you don't want them to talk to each other… in which case, go ahead! Put them further than that, and you'll nuke communication.



GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

So what does that mean?

- Put people who need to talk together next to each other
- Call meetings regularly if you can't do it
  - Now you have justification ;-)
- If all else fails...

So if you want people to communicate, they have to be within 164ft/50m of each other.

If you can't manage this, you're going to need to structure things around it, and call lots of meetings – NOW YOU HAVE JUSTIFICATION FOR THEM! ☺ (this also applies as team sizes grow, of course).

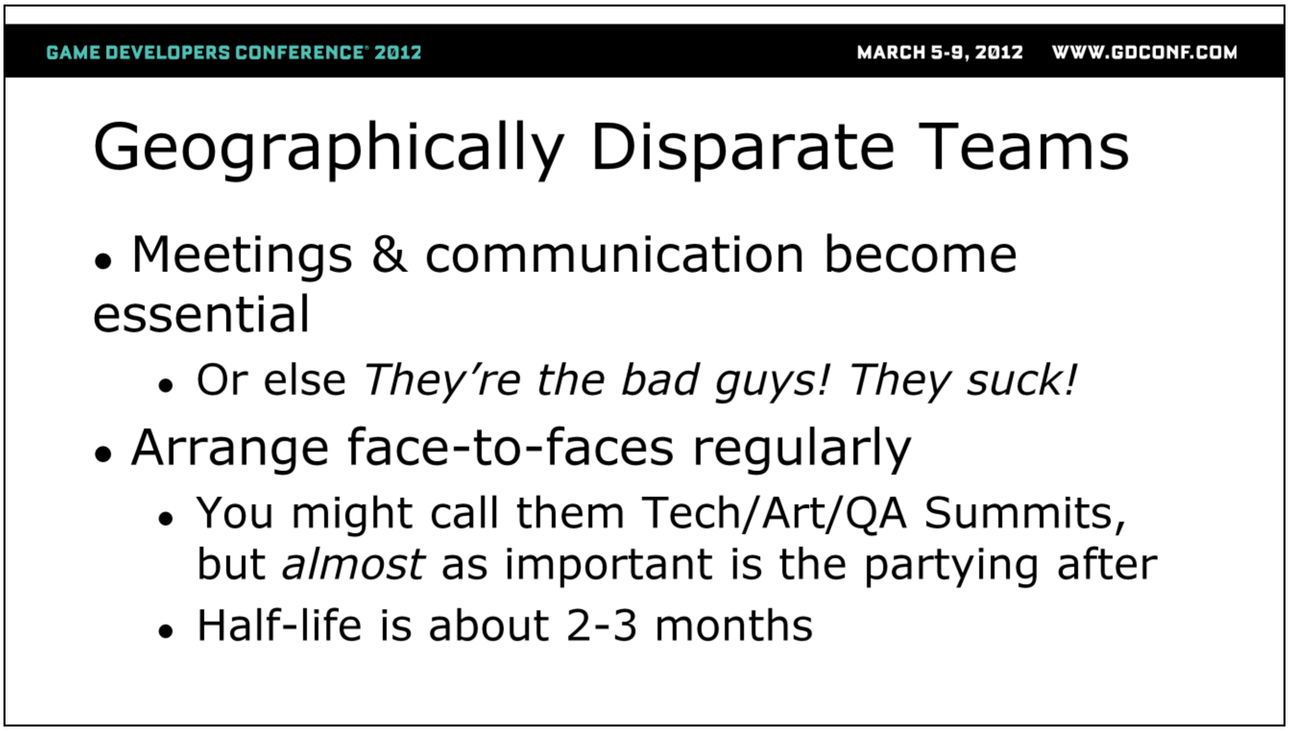
What worked fantastically at Surreal was… Scooters. We had two bays of scooters, one at each end of the office. Need to talk to someone? Pick one up, scoot around, put it back when you're done. People were whizzing around the office at high speed in no time – and it short circuited the problem (to a degree). Way better than email/facebook. If you have a large team, invest in scooters. They're fun too ☺

GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

REALLY BIG TEAMS

This is where your team is split across multiple divisions, companies, or a continent – or several continents. Either way, you're way out of Dunbar territory here, and are on your own…

The two best pieces of advice I can give you here are…


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

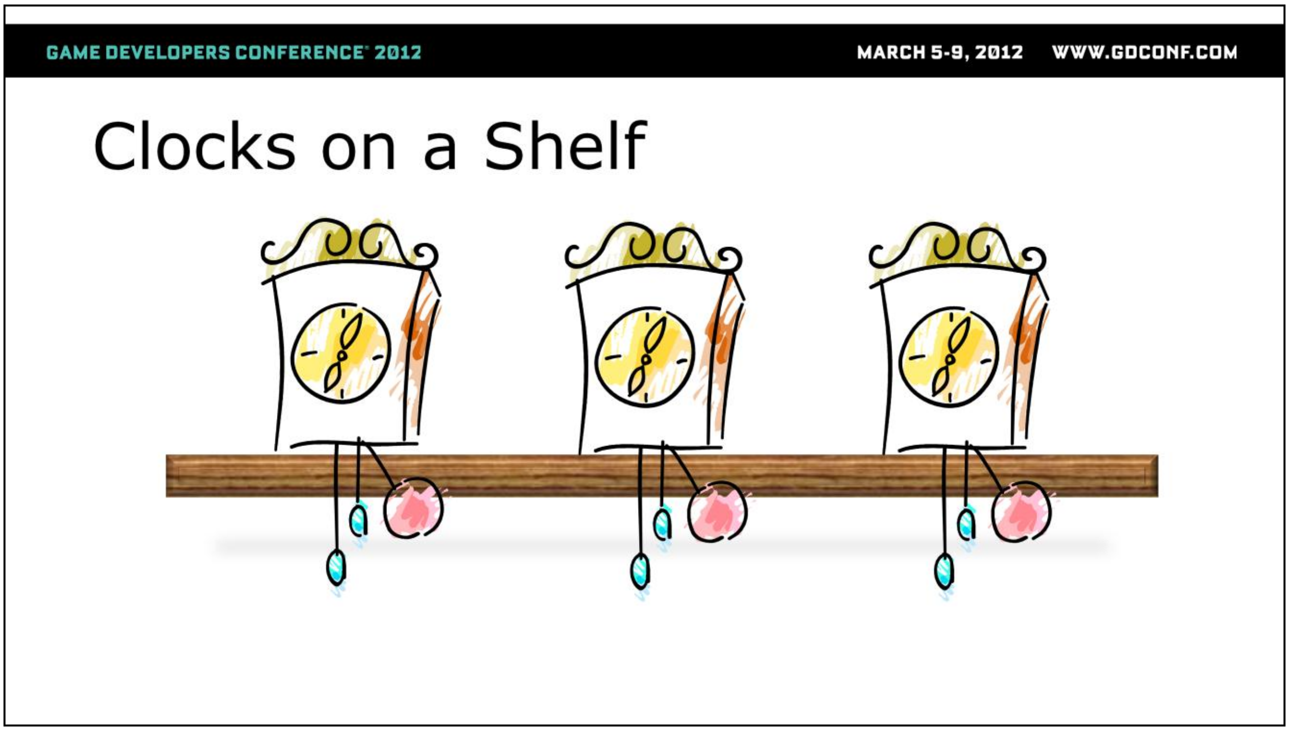
Geographically Disparate Teams

- Meetings & communication become essential
  - Or else *They're the bad guys! They suck!*
- Arrange face-to-faces regularly
  - You might call them Tech/Art/QA Summits, but *almost* as important is the partying after
  - Half-life is about 2-3 months

You can't have these people in your tribe if you never see them. They'll become the enemy quickly – because they're not "real". (If you can't smell them, they're not real). Phone calls and emails don't fix this. Video conferences don't fix it well – they help a little, but not much.

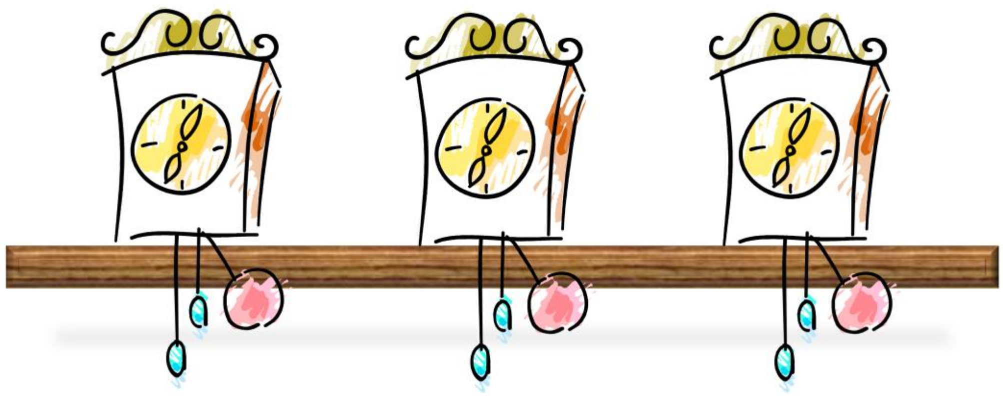
You need to have face-to-face meetings to fix this. Midway did this once or twice a year – and quickly, teams we were relying on who before the meeting were ***holding us up and causing problems for us, those idiots!*** … became "***oh my god, you're having that problem too? This sucks for everyone, friend, we're all in this together!" teams.***

Problem: It only works for about 2-3 months (gleaned from just watching teams start bitching again), and then you have to rinse and repeat – it wears off. So do it often, as regularly as you can afford.



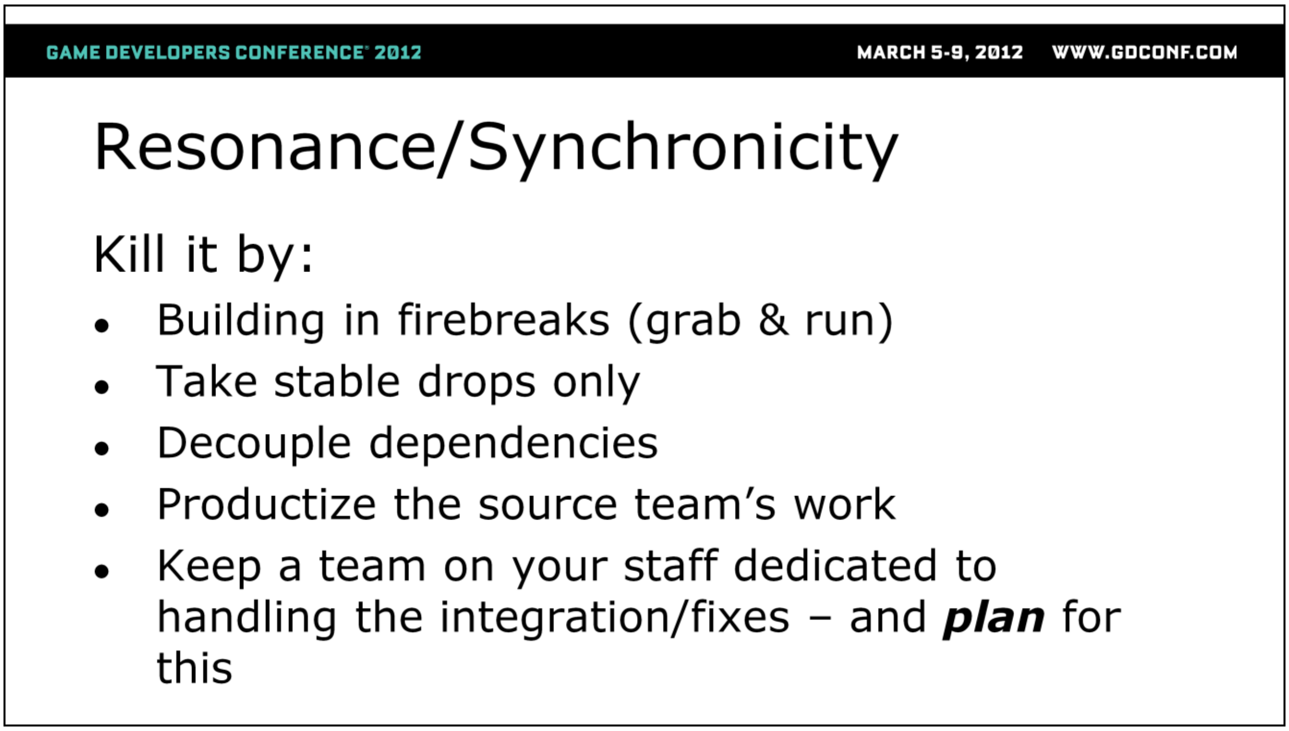
GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Clocks on a Shelf



Pendulum clocks on a shelf synchronize with one another over time, no matter how thin the shelf is. It's because they start to resonate, and the microscopic vibrations (and thin coupling) between them cause them lose/gain energy until they're all going to the same beat. (This is an actual physical property – you can do the experiment yourself).

The same thing will happen on projects where teams are dependent on other teams, and then they'll ***all ship at the same time!!!*** unless you actively guard against it.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Resonance/Synchronicity

Kill it by:

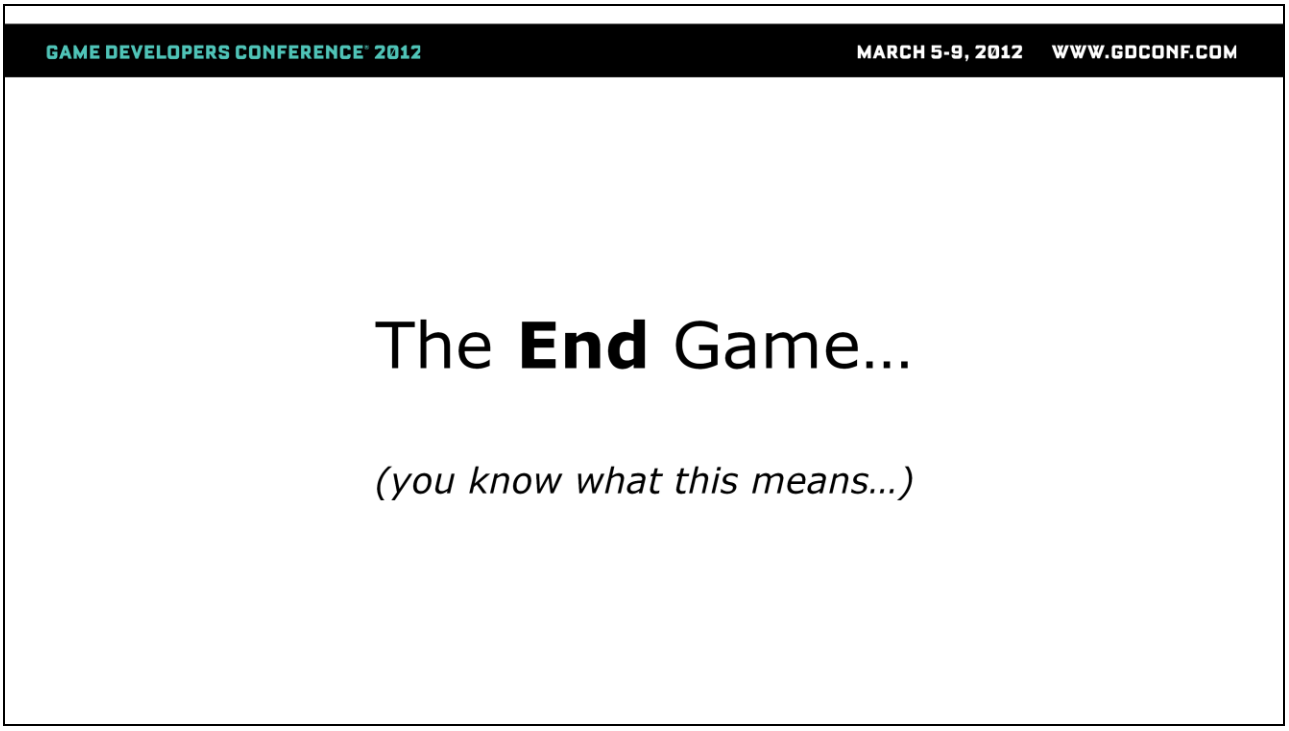
- Building in firebreaks (grab & run)
- Take stable drops only
- Decouple dependencies
- Productize the source team's work
- Keep a team on your staff dedicated to handling the integration/fixes – and ***plan*** for this

Firebreaks: Grab a stable drop. That's the last drop you ever take. After that, you're on your own.

Decouple dependencies – find ways to break the dependencies between the teams, and actively do so.

Productize – respect that you've now got a tech team creating a standalone product for consumption by others – and decouple *them* from the game team they *were* on. This allows all dependent game teams to ship separately.

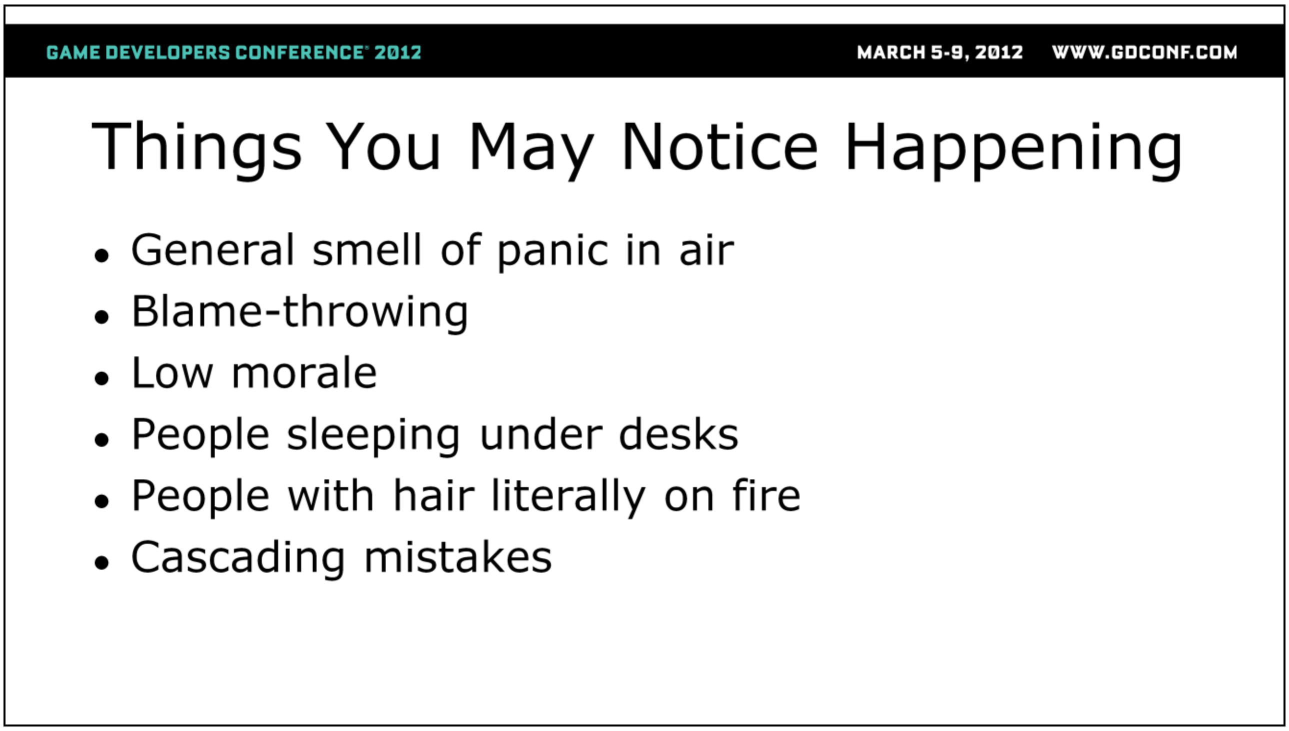
Integration ALWAYS takes longer than you hope or expect, and is painful. And you'll have that work to do whenever you have anything in your project that looks or smells like, or actually IS middleware. So plan for it, and dedicate resources on the team to integrate the tech, ***and to know it inside and out***.



The **End** Game…

(you know what this means…)

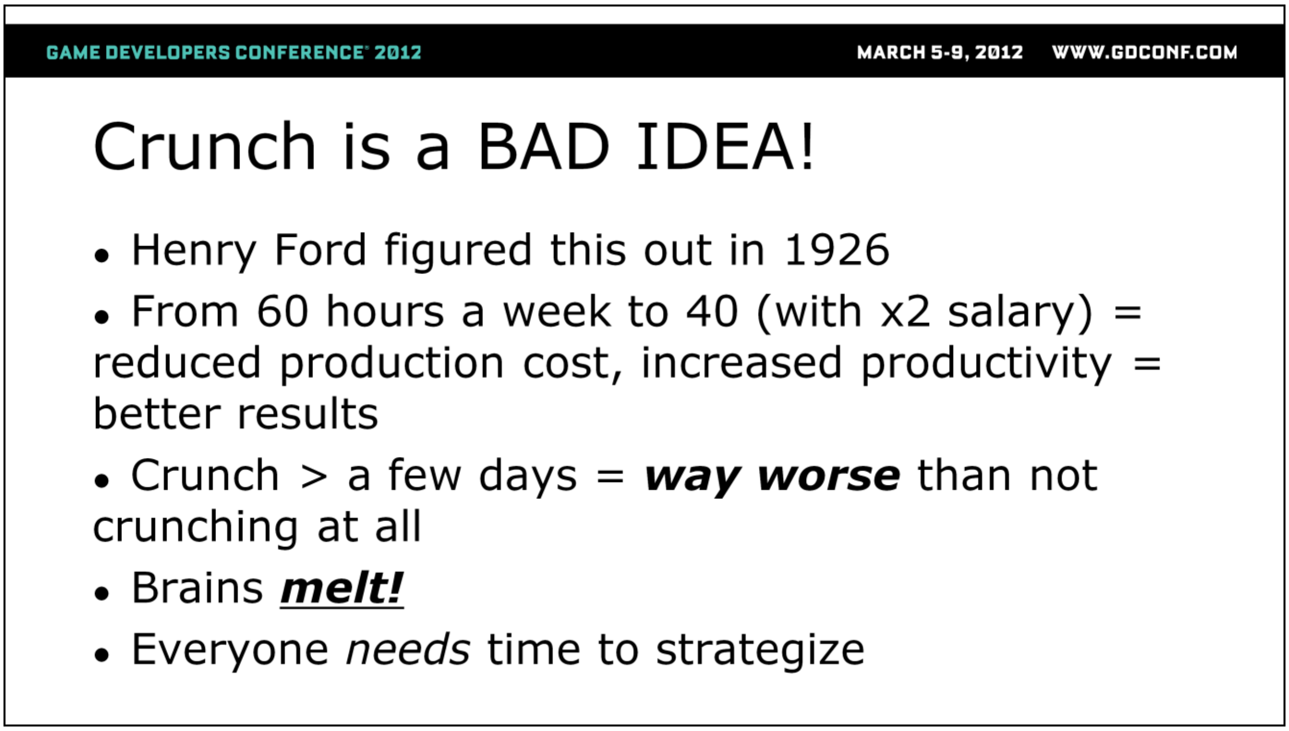
Oh yes. Crunch.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Things You May Notice Happening

- General smell of panic in air
- Blame-throwing
- Low morale
- People sleeping under desks
- People with hair literally on fire
- Cascading mistakes

- Suffering Ties That Bind – team crunch 3-9 months
- At this point, panic set in. Design blaming programming. Programming blaming QA. QA blaming art. Lots of blame going around.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Crunch is a BAD IDEA!

- Henry Ford figured this out in 1926
- From 60 hours a week to 40 (with x2 salary) = reduced production cost, increased productivity = better results
- Crunch > a few days = ***way worse*** than not crunching at all
- Brains ***<u>melt!</u>***
- Everyone *needs* time to strategize

http://www.igda.org/why-crunch-modes-doesnt-work-six-lessons – Evan Robinson

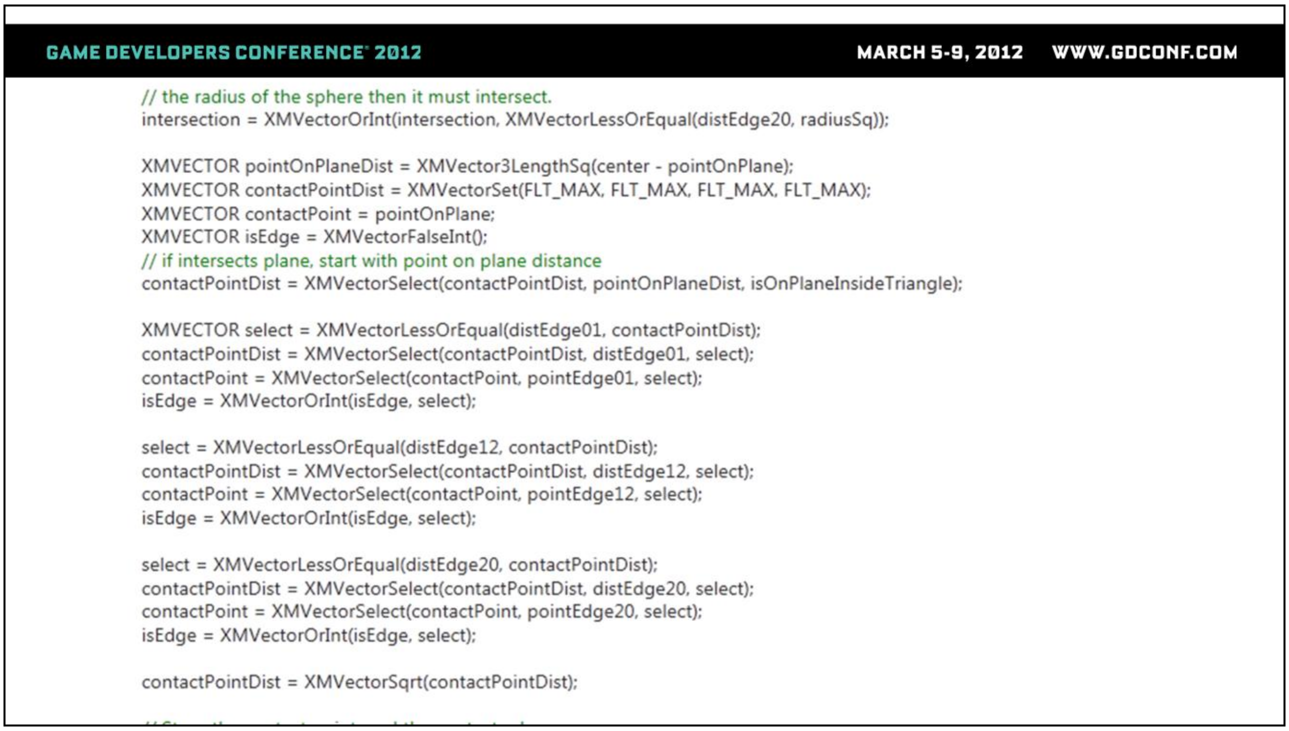
Ford made more money by REDUCING the work on his people – because they were working past efficiency. They were also working on an ***assembly line*** which requires little actual hard thinking – just repetition.

This is an easier environment than a game team, where everyone has to be sharp and on their game. (Jury is still out on if 6 hour days is better than 8 for "knowledge work").

If you crunch for more than a couple of days, you'll hit diminishing returns really quickly, and if you measure it, you'll probably find that it's causing more problems than progress. It actively hurts your project – even if it feels like the right thing to do.

Everyone's brains malfunction when they're tired. Mistakes are made. Humans have a pretty lousy grip on accurate reasoning at the best of time.

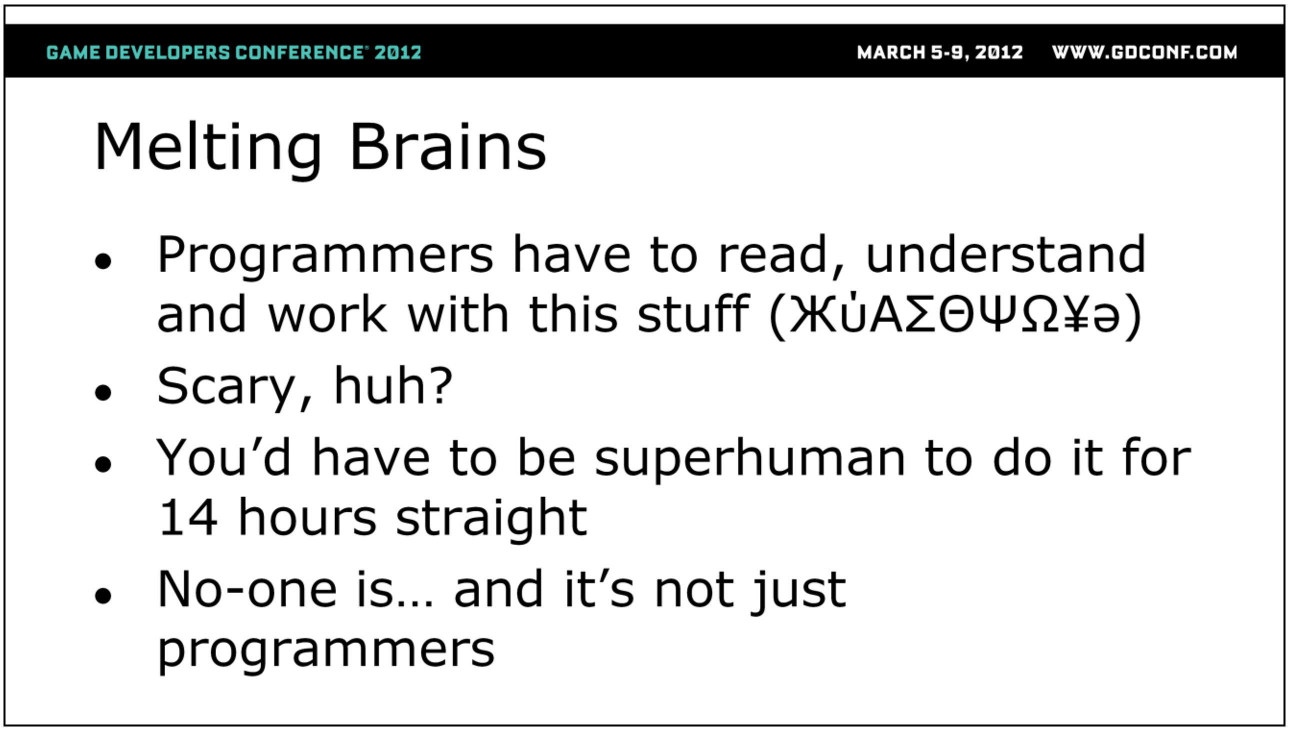
And everyone needs down-time to strategize – that is, to think about what you’re doing, and look at it from a broader view to see how you can do it better/what the side effects of it are. Running at full tilt the whole time actively disables your ability to do this. If you’re drained to the point where you’re only able to react – you can’t strategize… heck, you can’t even think *tactically*, then you’re going to hit problems.



GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

```
// the radius of the sphere then it must intersect.
intersection = XMVectorOrInt(intersection, XMVectorLessOrEqual(distEdge20, radiusSq));

XMVECTOR pointOnPlaneDist = XMVector3LengthSq(center - pointOnPlane);
XMVECTOR contactPointDist = XMVectorSet(FLT_MAX, FLT_MAX, FLT_MAX, FLT_MAX);
XMVECTOR contactPoint = pointOnPlane;
XMVECTOR isEdge = XMVectorFalseInt();
// if intersects plane, start with point on plane distance
contactPointDist = XMVectorSelect(contactPointDist, pointOnPlaneDist, isOnPlaneInsideTriangle);

XMVECTOR select = XMVectorLessOrEqual(distEdge01, contactPointDist);
contactPointDist = XMVectorSelect(contactPointDist, distEdge01, select);
contactPoint = XMVectorSelect(contactPoint, pointEdge01, select);
isEdge = XMVectorOrInt(isEdge, select);

select = XMVectorLessOrEqual(distEdge12, contactPointDist);
contactPointDist = XMVectorSelect(contactPointDist, distEdge12, select);
contactPoint = XMVectorSelect(contactPoint, pointEdge12, select);
isEdge = XMVectorOrInt(isEdge, select);

select = XMVectorLessOrEqual(distEdge20, contactPointDist);
contactPointDist = XMVectorSelect(contactPointDist, distEdge20, select);
contactPoint = XMVectorSelect(contactPoint, pointEdge20, select);
isEdge = XMVectorOrInt(isEdge, select);

contactPointDist = XMVectorSqrt(contactPointDist);
```

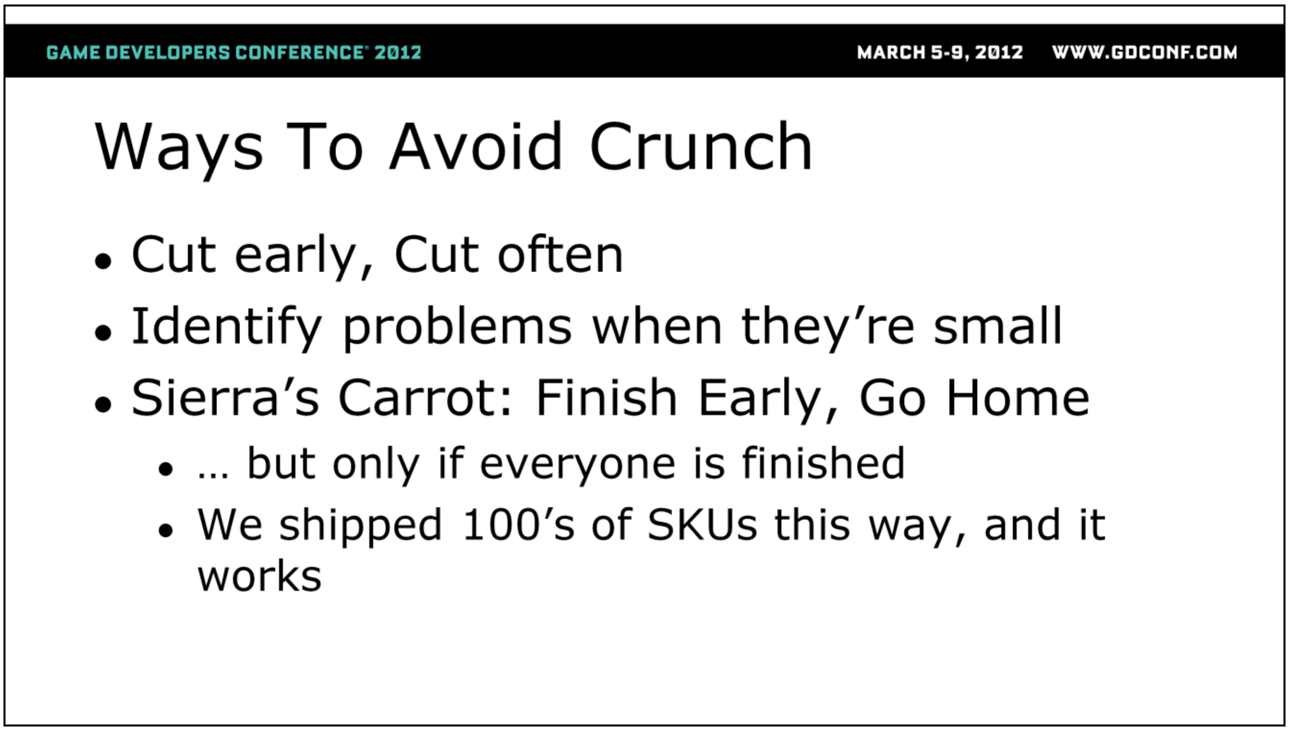
Code part of the Windows 8 Marble Maze Sample, Copyright © 2011 Microsoft Corp. (I wrote this code about 6 months ago, and I couldn't tell you what it means… can you imagine reading that when you're tired?)


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Melting Brains

- Programmers have to read, understand and work with this stuff (ЖὐΑΣΘΨΩ¥ə)
- Scary, huh?
- You'd have to be superhuman to do it for 14 hours straight
- No-one is… and it's not just programmers

Applies to artists, QA, everyone. We all need downtime, and we all need sleep.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Ways To Avoid Crunch

- Cut early, Cut often
- Identify problems when they're small
- Sierra's Carrot: Finish Early, Go Home
  - … but only if everyone is finished
  - We shipped 100's of SKUs this way, and it works

Your biggest advantage here is cutting features early and often from your schedule. So front-load the important stuff, and let less-important stuff slip off the end. Sounds like the SCRUM backlog? Yep, but you don't need to buy into the whole SCRUM methodology here – just reschedule at least once a week, and trim from the end.

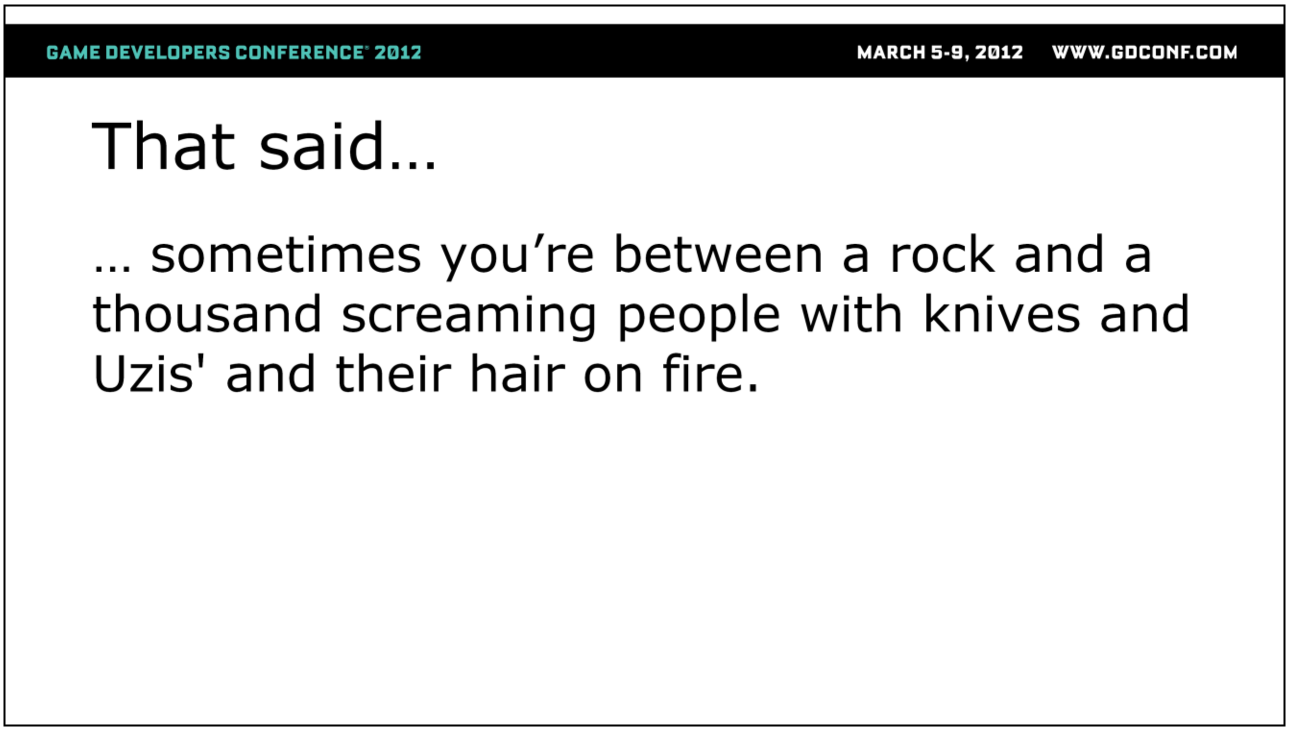
Related to this: identify – AND FIX! – problems while they're still small. Otherwise they grow, and can require herculean efforts to fix later.

Sierra's Carrot – more accurately, Fred Shean's carrot (http://www.linkedin.com/pub/fred-shean/0/598/713) was invented by Fred Shean at Sierra. The idea is this:

- The project is budgeted, planned out, and has a fixed ship date. And you're reasonably certain you should be able to make that date (if not, revise schedule and stop kidding yourselfe).

- If EVERYONE is finished before that ship date, all that extra time is treated as paid vacation.
    - It has to be taken right then (it's not added to your vacation time to be used randomly)
    - Everyone has to be finished – if you're finished and someone else isn't? Help them across the finish line!
- The business guys balk a little, but if you explain to them that they're losing no money (it's already budgeted for and spent!), and in return they get better accuracy on shipping on that date, it's a ***no brainer***.
    - Especially when you point out that people will need a few weeks to recover from shipping anyway, so all you're eating into is this less productive period of time.
    - ***Some people actually <u>decline</u> this time off, and show up at work anyway…*** (so you lose even less) (yes, it's very surprising, but it really happens)

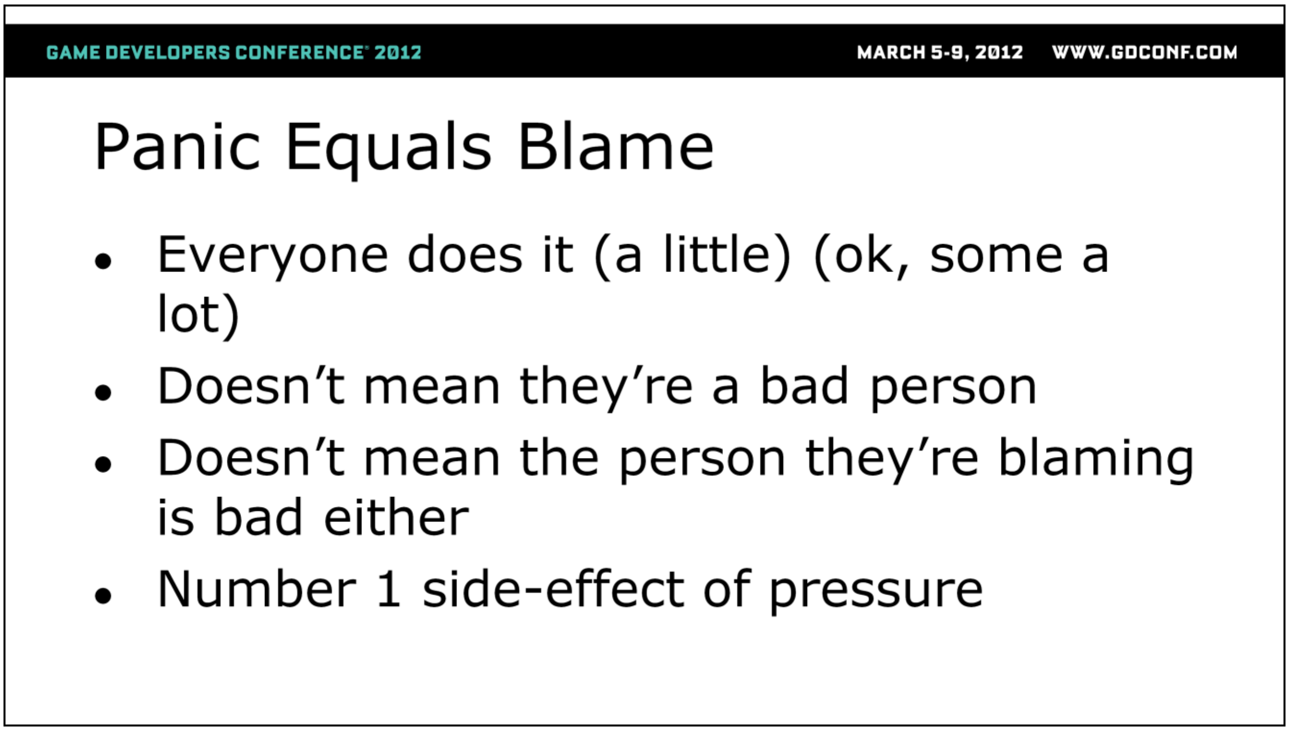
It works. We did ship 100s of SKUs on time this way. And it IMPROVES collaboration, unlike a number of other mechanisms you can use, which actually only improve *competition.*


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

That said…

… sometimes you're between a rock and a thousand screaming people with knives and Uzis' and their hair on fire.

If it's your first time at Crunch Club, you *have* to Crunch.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

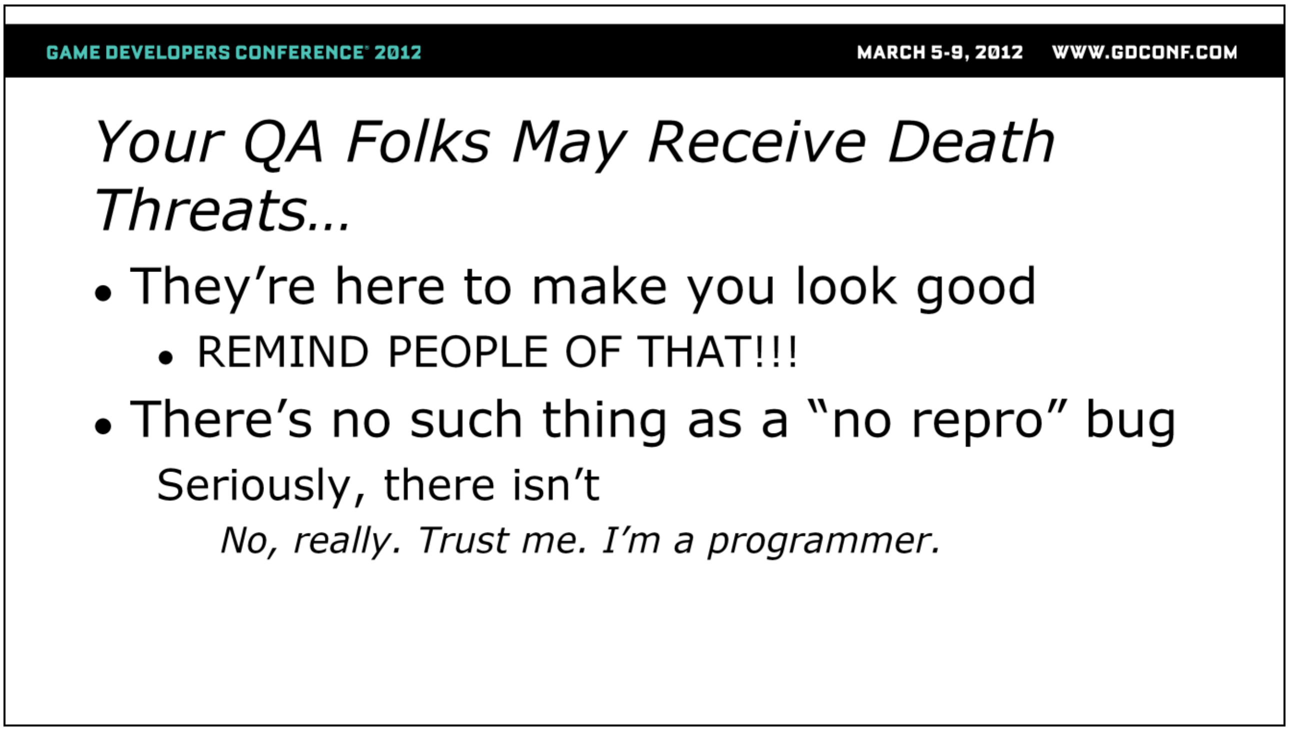
Panic Equals Blame

- Everyone does it (a little) (ok, some a lot)
- Doesn't mean they're a bad person
- Doesn't mean the person they're blaming is bad either
- Number 1 side-effect of pressure

This goes back to this whole reptilian psychology thing. When we are under pressure, we tend to panic – and blame anyone who isn't part of our inner circle. (Or sometimes, the weakest person in our inner circle).

Watch for this. Look for it. And *defuse* it. Sometimes it's a simple as negotiating a treaty, compromise or collaboration between the people who are at war. Or just explaining why things are going screwy, and how it's really not the other person's fault.

We all turn into toddlers under pressure. We whine. We act out. We throw tantrums. A lot of the time, we just want someone to hug us and make us feel better. You can do that by being an expert listener for your team during crunch, and providing the right brightly printed cartoon band-aids.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Your QA Folks May Receive Death Threats…

- They’re here to make you look good
  - REMIND PEOPLE OF THAT!!!
- There’s no such thing as a “no repro” bug

  Seriously, there isn’t

  No, really. Trust me. I’m a programmer.

It’s easy to see why people on game teams can fall into the trap of hating QA. After all, if they hadn’t found 100 new bugs in it, you’d have shipped by now… They’re just holding you up!

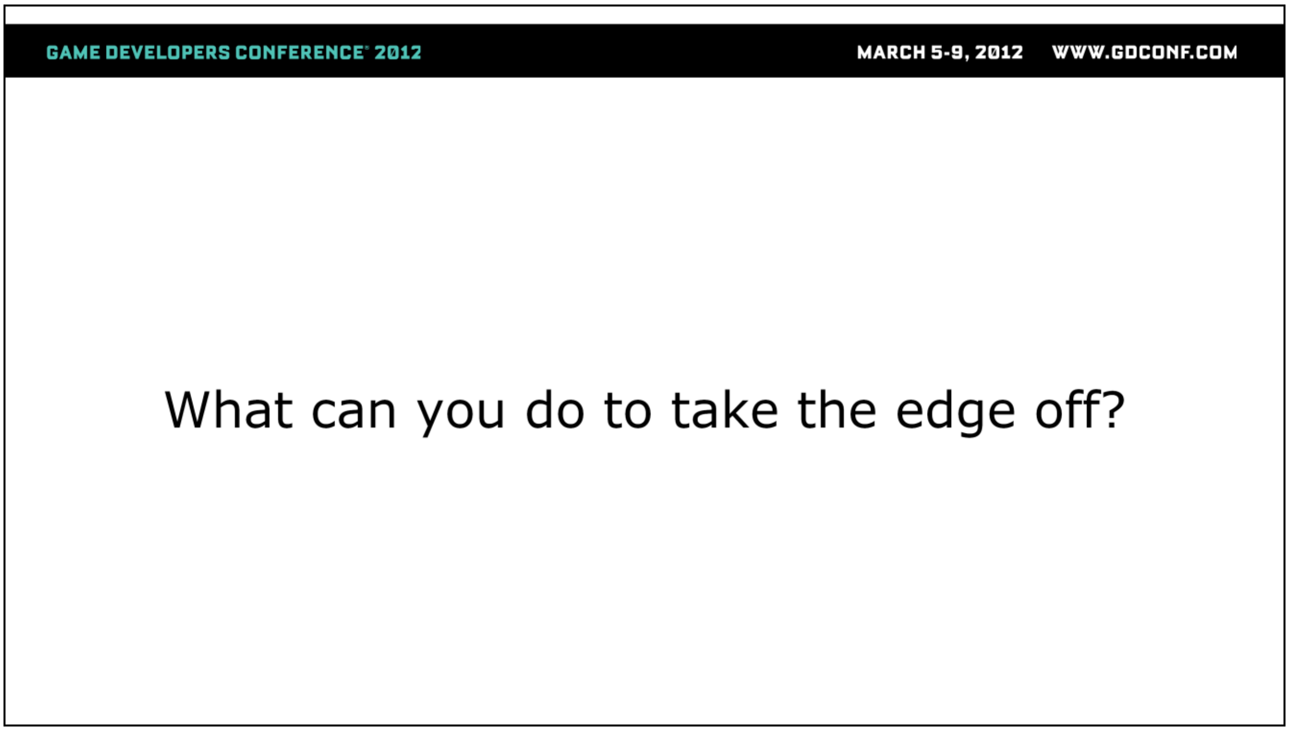
QA is your last best defense against looking like an idiot in the eyes of your customer. They’re there to make you look good – not to slow you down, not to make you look bad. They would LOVE to have a bug-free, complete game to work on, because it would mean they can stop playing it and play something new that they’ve not seen over and over and over and over and over…. (repeat until you’re sick).

You should always encourage your team to look at QA as your backstop. They’re not there to catch crappy work on your side – so do your best work, your own testing, and NEVER just “throw it over the wall”. They’re there to pick up the things you missed. They’re partners – not in conflict with you.

No repro = not trying hard enough. Seriously. Most of the time it’s something *hard* to repro, but it *really*is a horrible life destroying bug. The end of Suffering: Ties That Bind’s ship cycle was full of this (surprisingly, mostly due to corrupt content builds on our servers… that took a while to figure out).

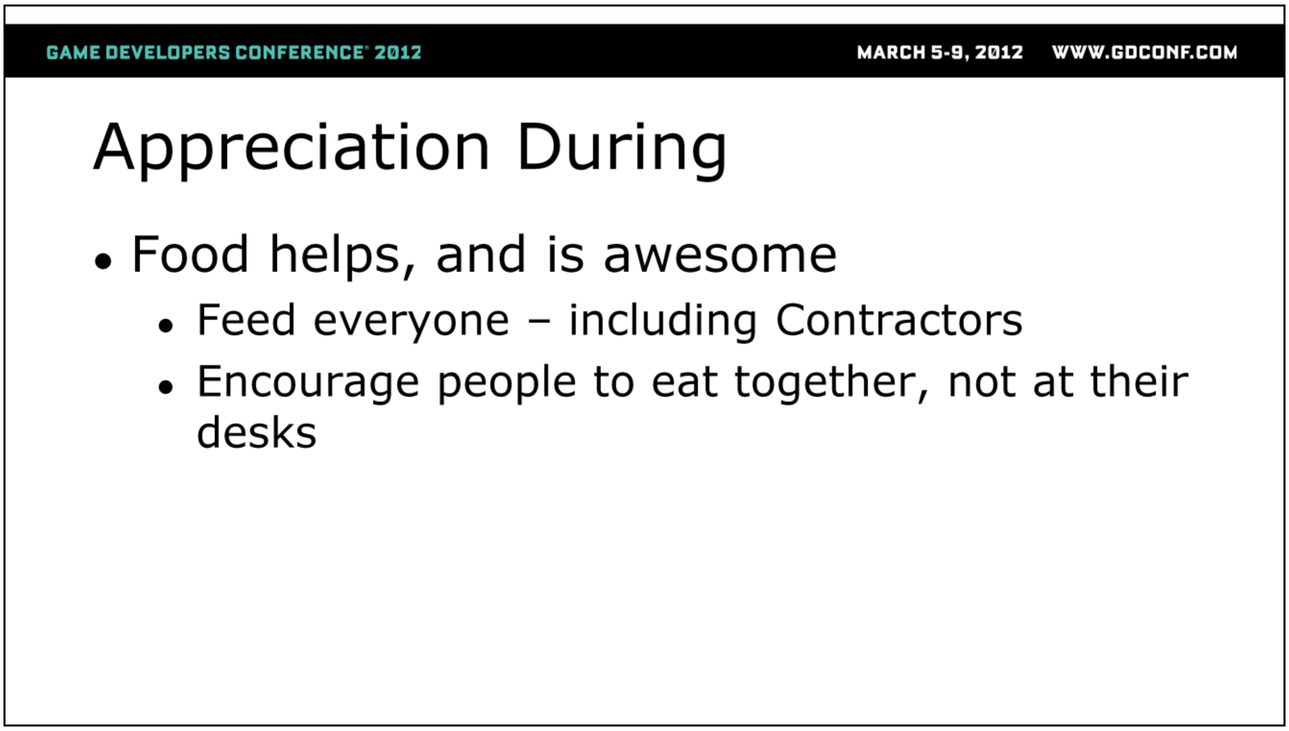
Get the programmer out of their chair and over to the QA person’s desk. Get the QA person to repro it for the programmer. If they can’t, then the QA person closes it out as No Repro. Too often, under pressure, No Repro means “I couldn’t be bothered trying it for more than 30 seconds”.

(Mind you, it also may be symptomatic of poor QA bug reporting skills – either way, getting people out of their chairs and over to each others’ desks will fix this problem).



What can you do to take the edge off?

Here’s some tricks for helping the team through crunch…

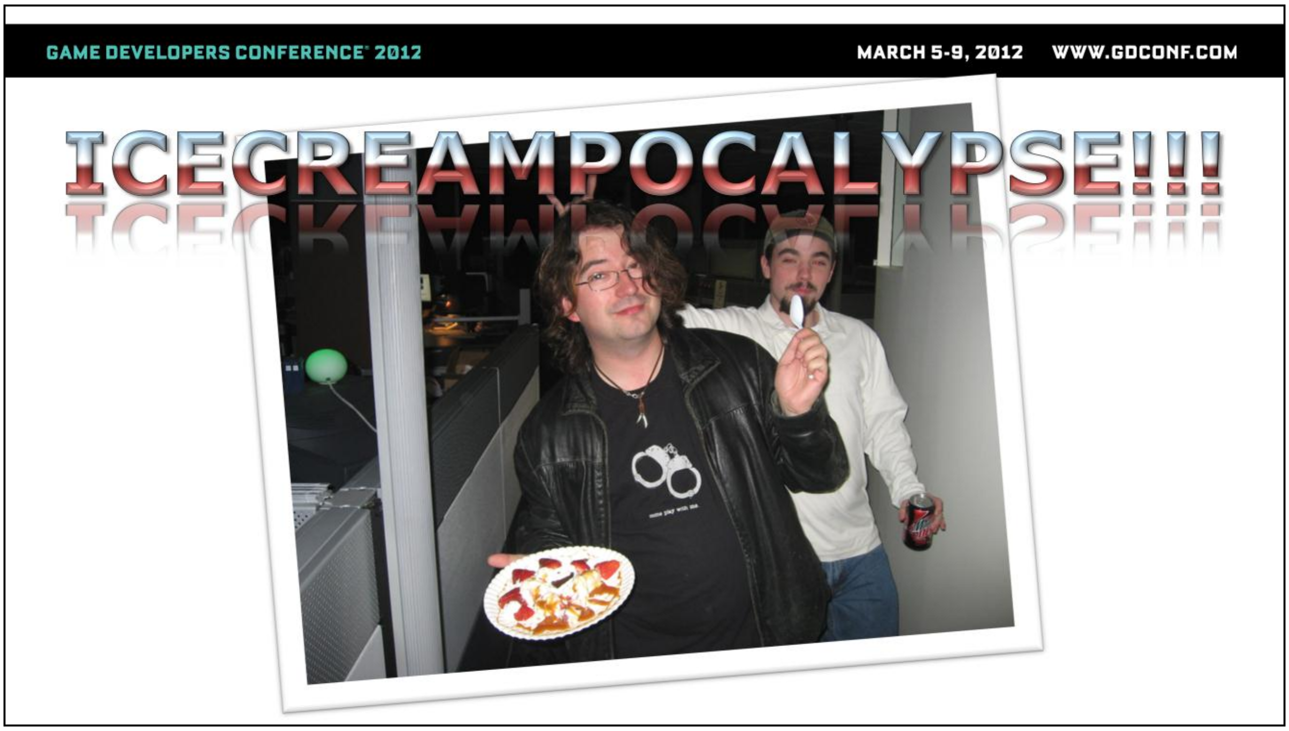

GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Appreciation During

- Food helps, and is awesome
  - Feed everyone – including Contractors
  - Encourage people to eat together, not at their desks

If you don’t feed contractors because “they’re not really part of the team”, you’re just plain rude and deserve to be ostracized. Food is social, not just fuel. Unless you *want* to kill your team’s relationship with people you’re apparently relying on to ship a game… in which case, go ahead. Don’t feed them.

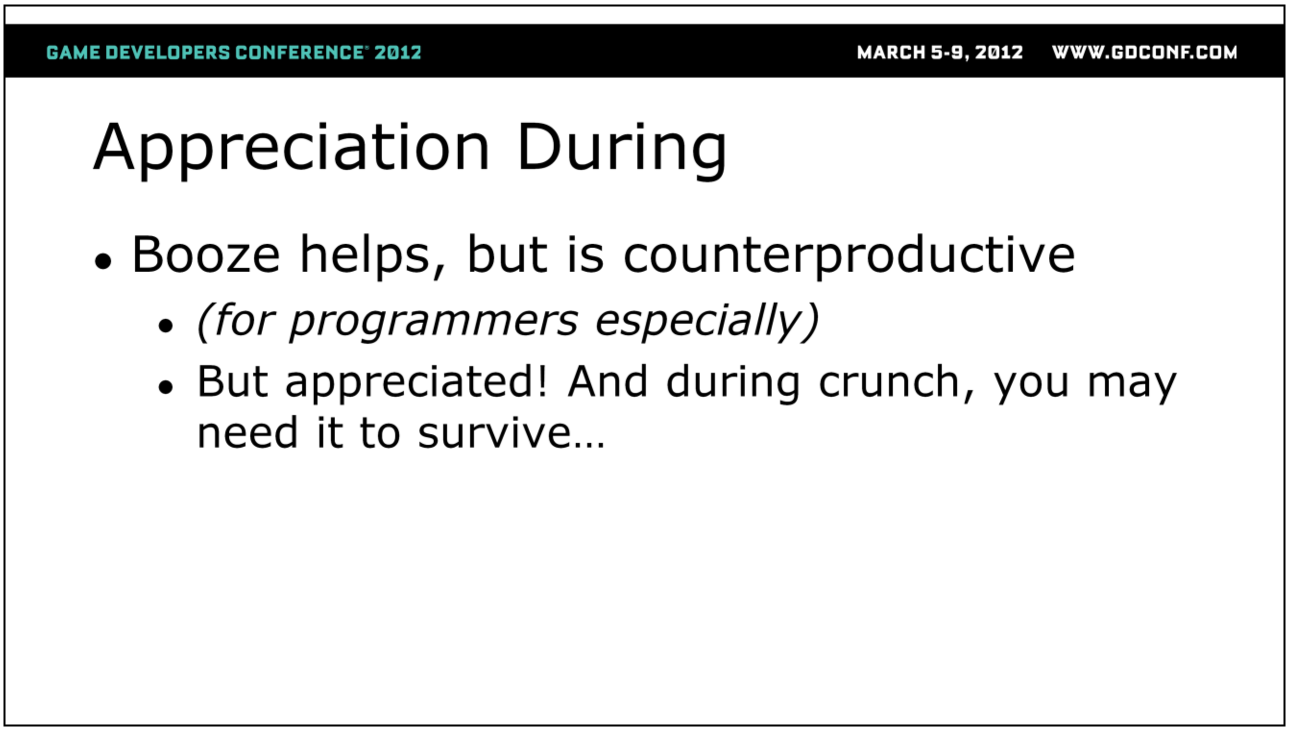
And it’s also a great way to give people downtime, cement team dynamics, and otherwise improve morale. So try to encourage people to take 30 minutes and eat together. Don’t force them, but at least nudge them in that direction. It’s worth it.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

ICECREAMPOCALYPSE!!!

Seriously, it's the least you can do. And it's tax deductible. Make sure you *switch it up* though, and don't order pizza every night. Give people choices, and let them pick what they want to eat.

One night I brought in several tubs of flashy ice-cream, whipped cream, caramel and chocolate sauce, and a bunch of fresh fruit. It went down well.

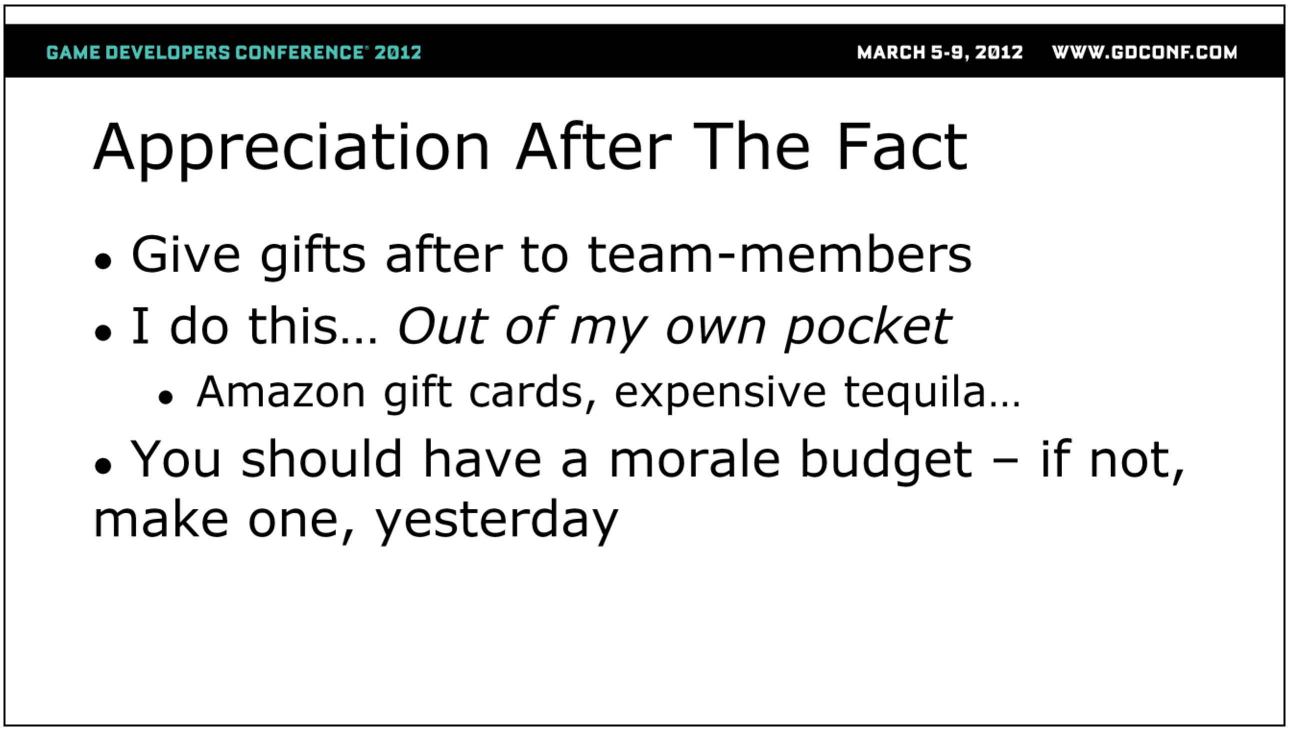


Appreciation During

- Booze helps, but is counterproductive
    - *(for programmers especially)*
    - But appreciated! And during crunch, you may need it to survive…

Booze is a double-edged sword. Programmers are especially debilitated by it, so use sparingly and carefully. But if you're stuck in the office during crunch, shots at 11pm are probably not a bad idea, and may help your team survive. It certainly seemed to help at Sierra and Surreal.

At Surreal, there was a bar downstairs. Most people would pop down for a couple of drinks and then head back up to work towards the end of a long night.

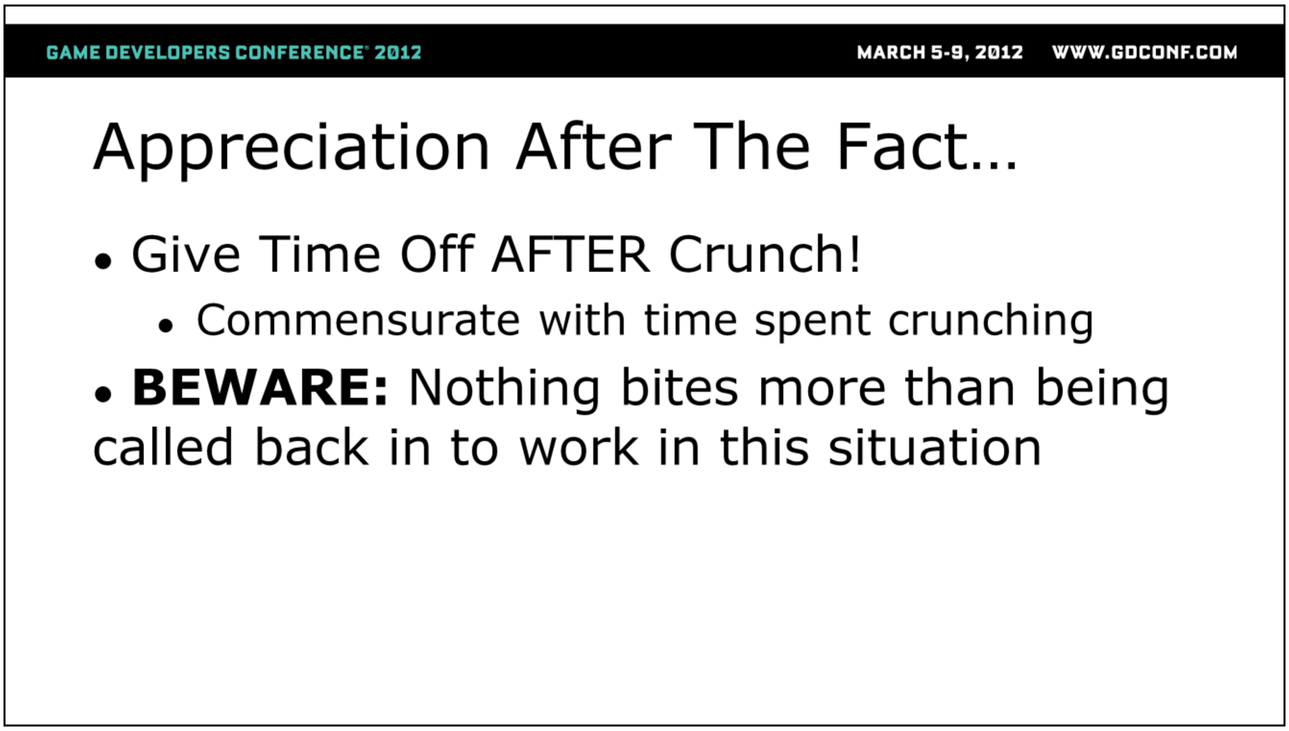

GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Appreciation After The Fact

- Give gifts after to team-members
- I do this… *Out of my own pocket*
  - Amazon gift cards, expensive tequila…
- You should have a morale budget – if not, make one, yesterday

Publishers pay for morale budgets as line-items in budgets. They're expected, so create one and use it liberally. Saving the money doesn't buy you anything.

I've always done this out of my own pocket as a mark of respect for people on my team. People on other teams have complained, wondering where their gifts are. I point out that it was coming out of my salary, and the complaints quickly went away…


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Appreciation After The Fact…

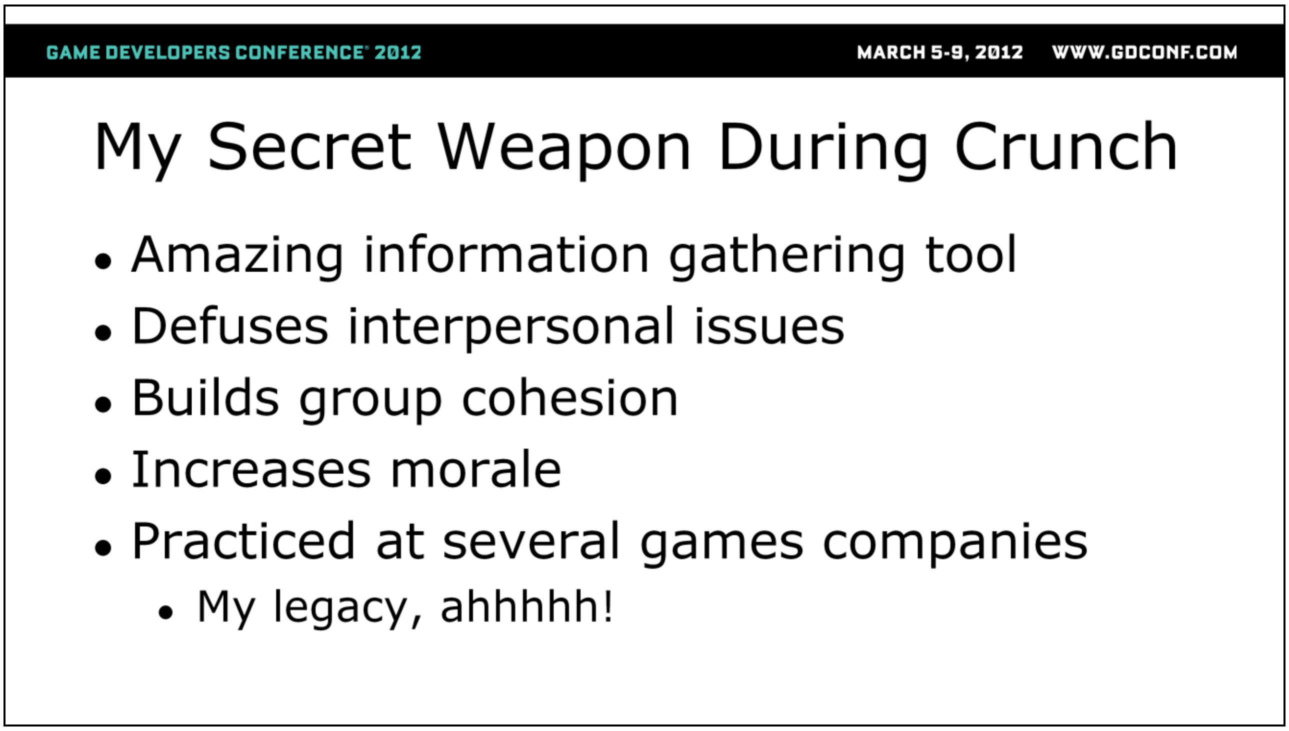
- Give Time Off AFTER Crunch!
  - Commensurate with time spent crunching
- **BEWARE:** Nothing bites more than being called back in to work in this situation

If you give people time off, make sure that it's:

- At least two days AFTER any big dog & pony shows you crunched for
- At least two WEEKS after you've officially gotten through Cert.

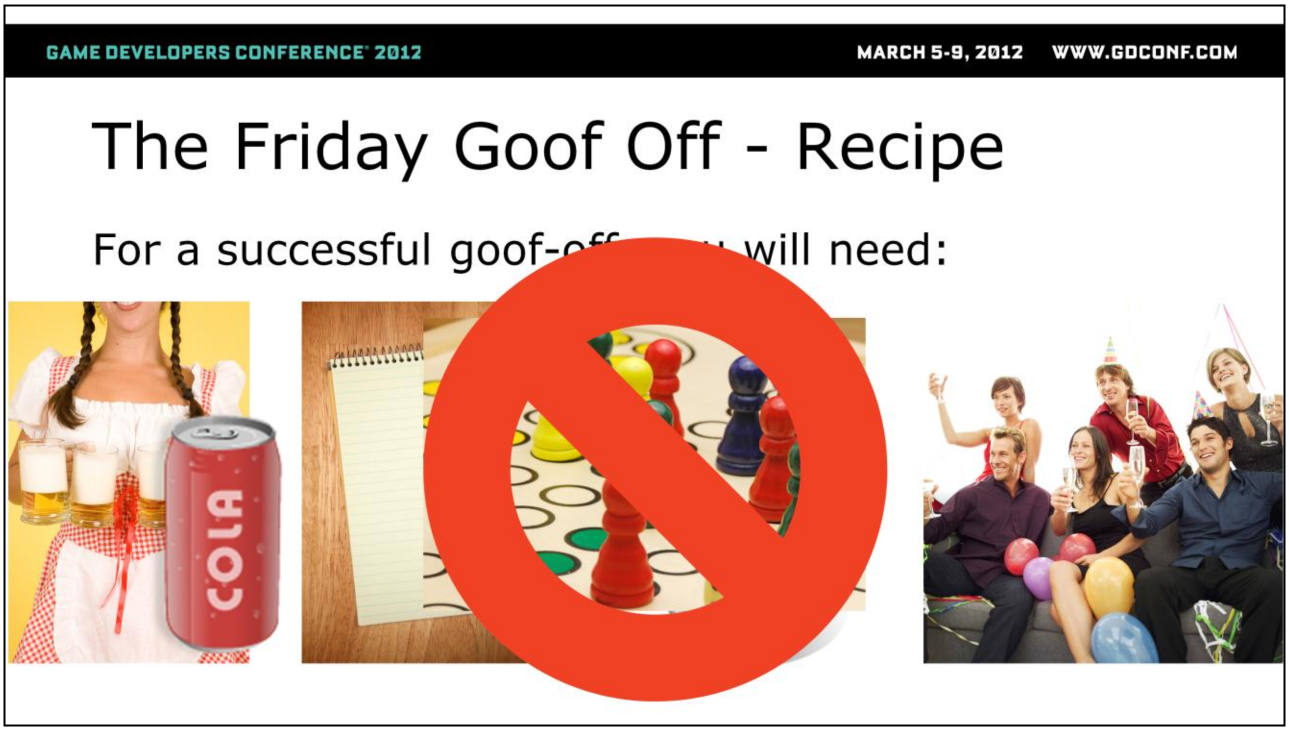
Otherwise, you're risking having to ask people to come back during that time off. (Worse is if you do this, then ***don't give the people you called back in extra time off to make up for it).***

So either stagger it (let senior/junior folks go immediately, but keep leads around), or hold off until you ***know*** you're in the clear.



My Secret Weapon During Crunch

- Amazing information gathering tool
- Defuses interpersonal issues
- Builds group cohesion
- Increases morale
- Practiced at several games companies
    - My legacy, ahhhhh!



GAME DEVELOPERS CONFERENCE 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

The Friday Goof Off - Recipe

For a successful goof-of[illegible] will need:

The recipe:

Beer (or soda)

Notepad + pen

Soccer ball (optional)

Team members

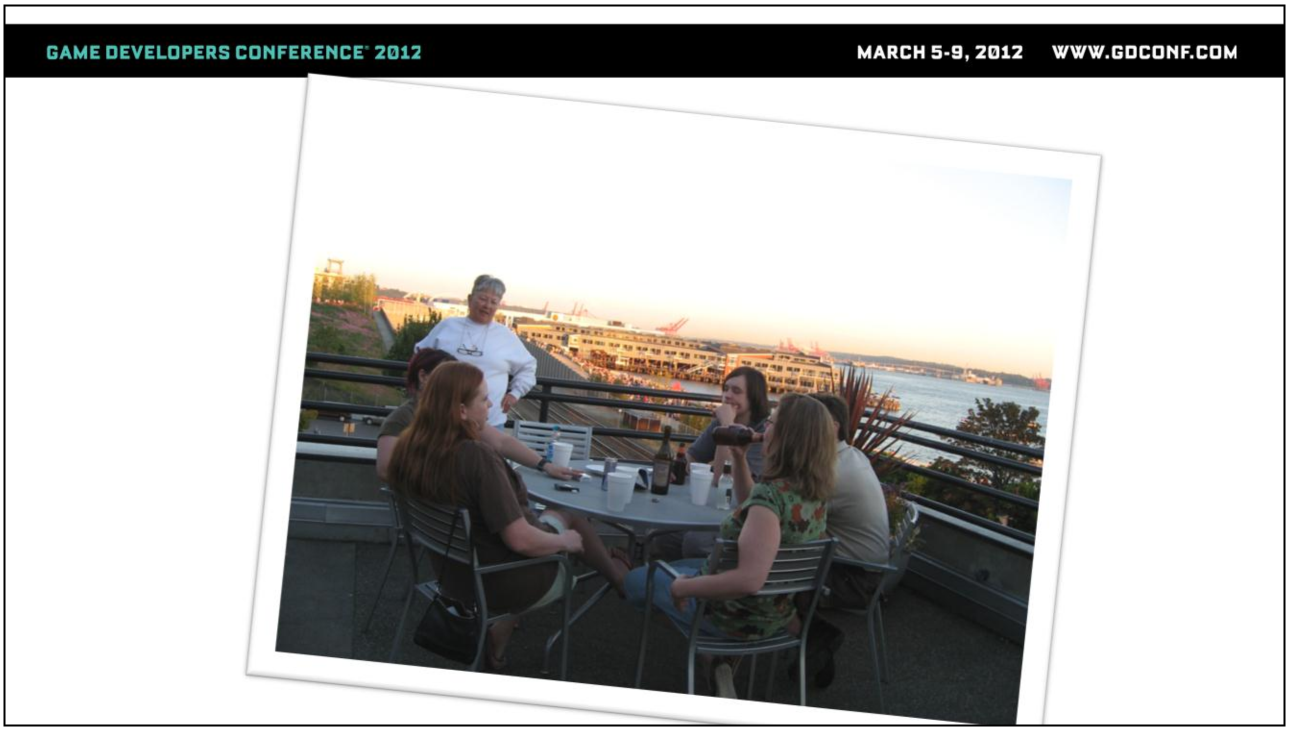
NO GAMES! The focus has to be on collaborative communication – board games break this down.

GAME DEVELOPERS CONFERENCE® 2012

WWW.GDCONF.COM

Originally, it started as a way to get out in the sun – I'd decided that I wasn't going to crunch through the summer again, and as you can imagine, sun is really important if you live in Seattle.

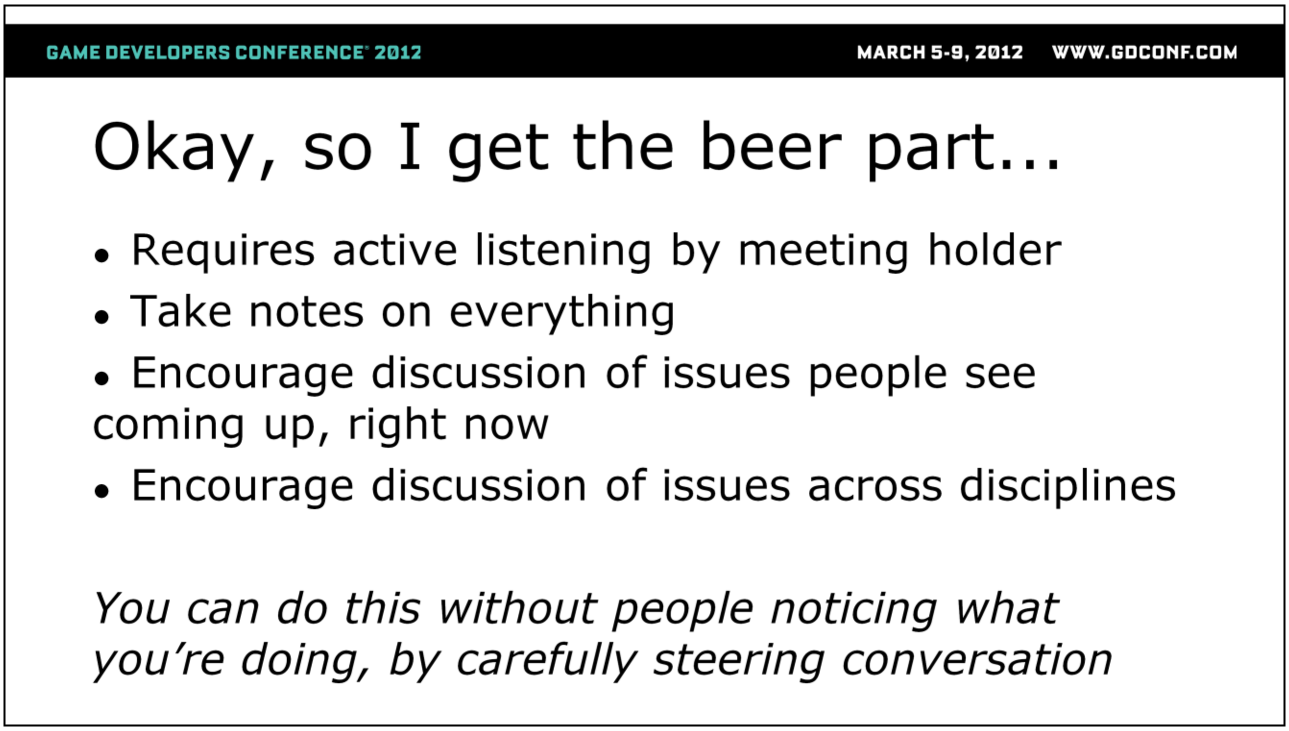
So we grabbed a soccer ball and I grabbed a notepad, and we just started kicking it around. And I noticed that I was getting way better and more information from my team than before – and not just the usual stuff you get in status meetings. So I carried on doing it….


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Version 2.0 in the new Surreal building – no Soccer Ball because we'd risk people going over the edge and dropping three stories to the train tracks below.

But ultimately, still the same thing:

- Away from earshot of other people.
- Focus on talking and chatting.
- A deliberate break from the normal environment.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Okay, so I get the beer part...

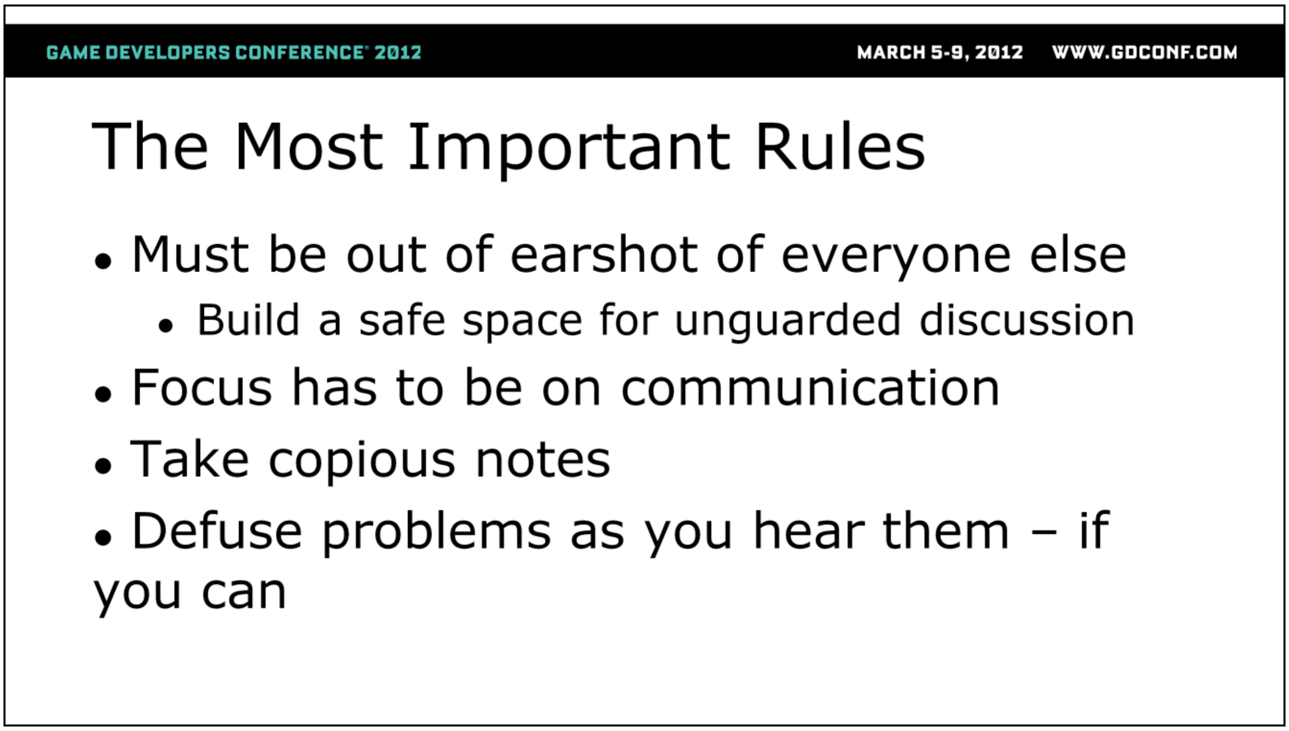
- Requires active listening by meeting holder
- Take notes on everything
- Encourage discussion of issues people see coming up, right now
- Encourage discussion of issues across disciplines

You can do this without people noticing what you're doing, by carefully steering conversation

If you're good, you can get people to talk about all of the issues they see – and problems with interpersonal dynamics within and outside of the team – and note them down.

You'll also get them to share ideas, share solutions to problems (hey, I've worked on that… when we get back in, let me help you), and build morale and team happiness.

It's *not* break time. It's not a break. It's a team meeting. But it *looks* like a break. It's sneaky.


GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

The Most Important Rules

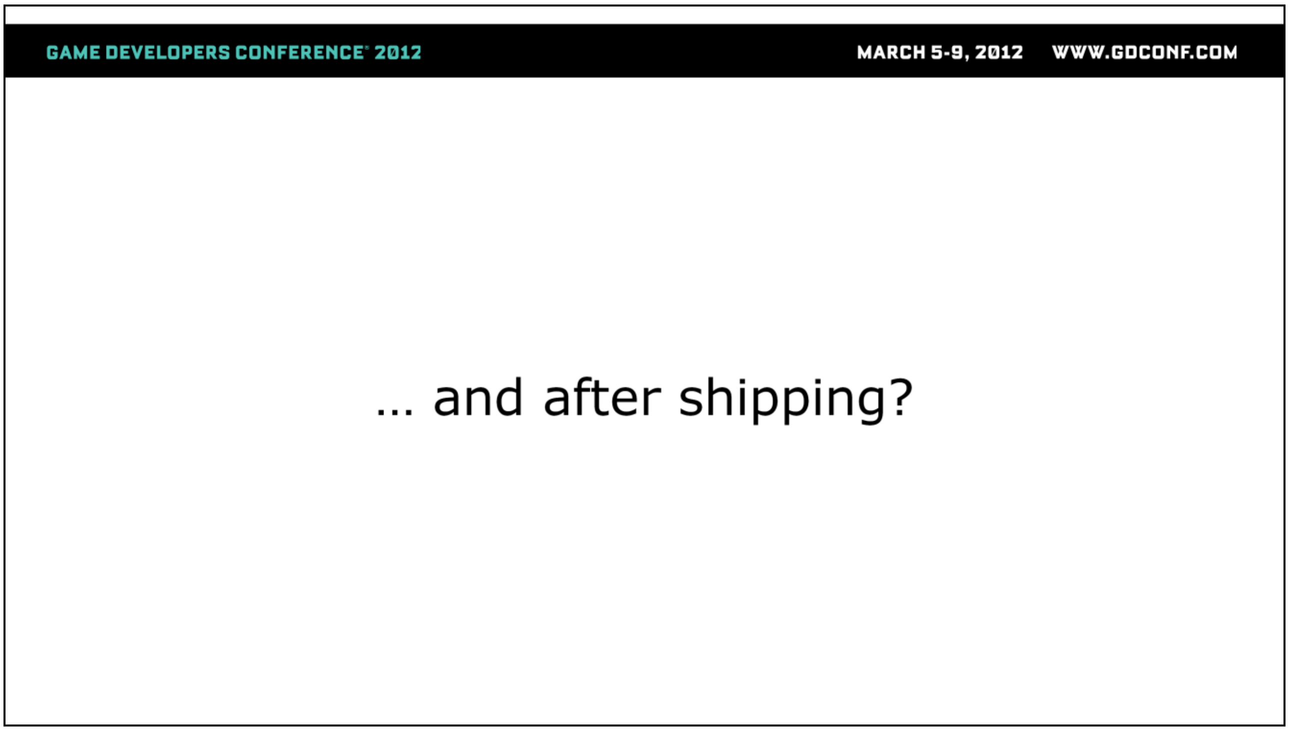
- Must be out of earshot of everyone else
  - Build a safe space for unguarded discussion
- Focus has to be on communication
- Take copious notes
- Defuse problems as you hear them – if you can

You need that safe space for communication if you're going to hear the bitching. You need to hear the bitching so that you can either take notes, and go off and fix the problem… or you can take notes, and then defuse the bitching by explaining the rationale, and providing suggestions for how *they* can fix the problem and make things smoother on their side.

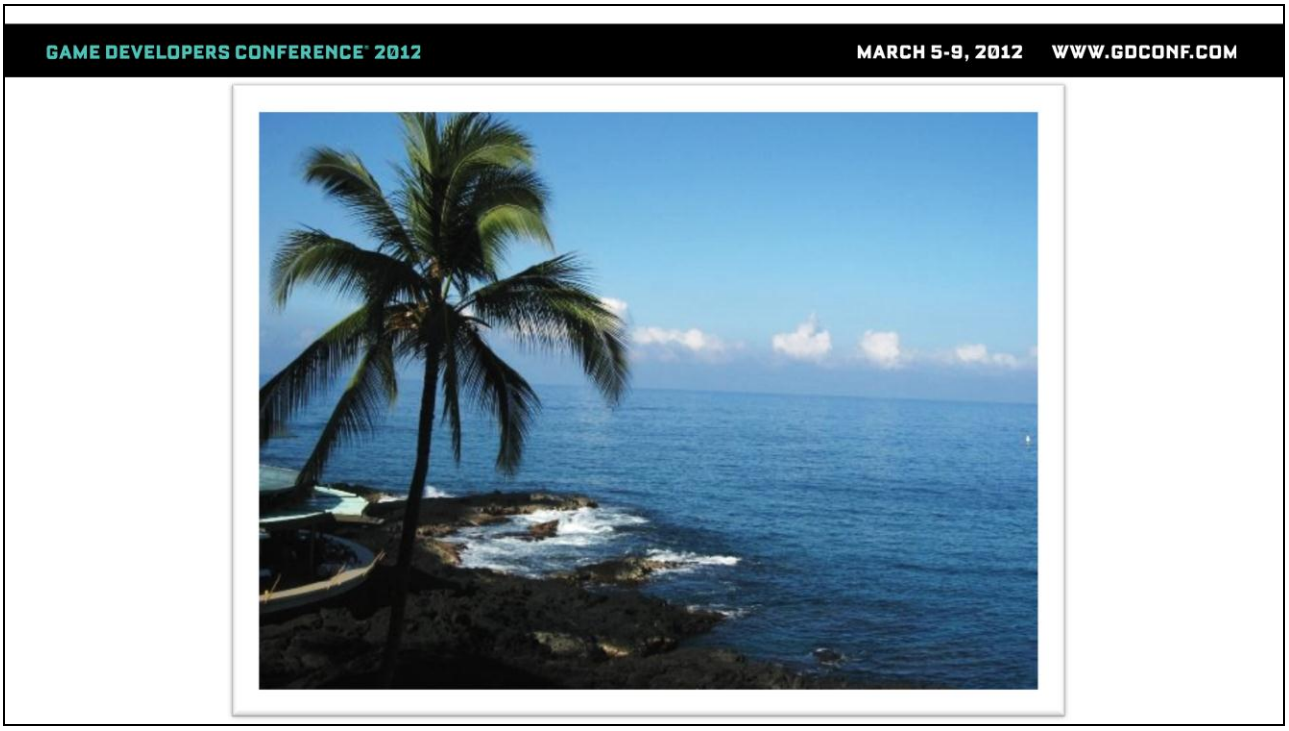
Games get in the way of this, because if someone brings a board game, (1) it's immediately competitive, and (2) the focus shifts towards the game. You want people *unengaged in tactical thinking*, or what's bugging them won't come out. It's very much like a group psychology session.

And yes, you as meeting holder should take notes. Lots of them. Every little thing. Try to be nonchalant and hide it if you can.

Either way, this really helps especially in high pressure situations. You get a lot of bang for your buck with this one, and it's practiced at several companies, especially by people I used to manage.

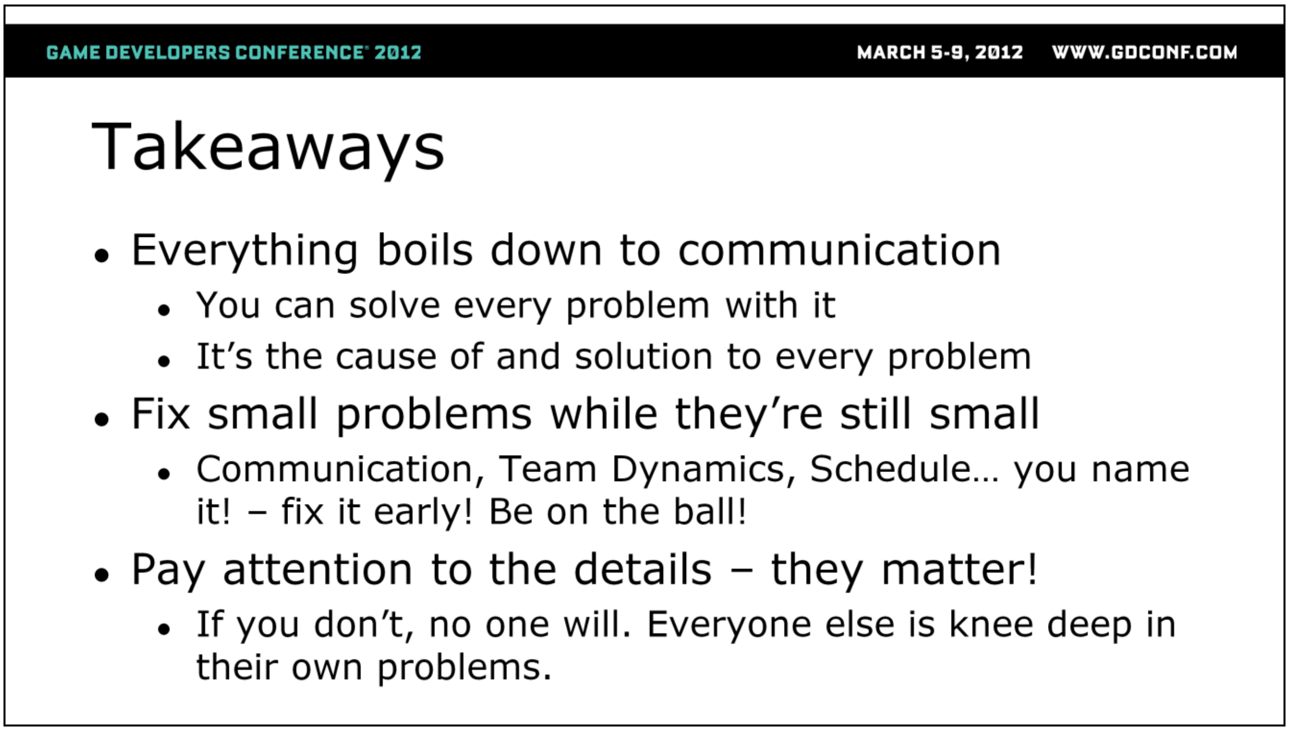


… and after shipping?



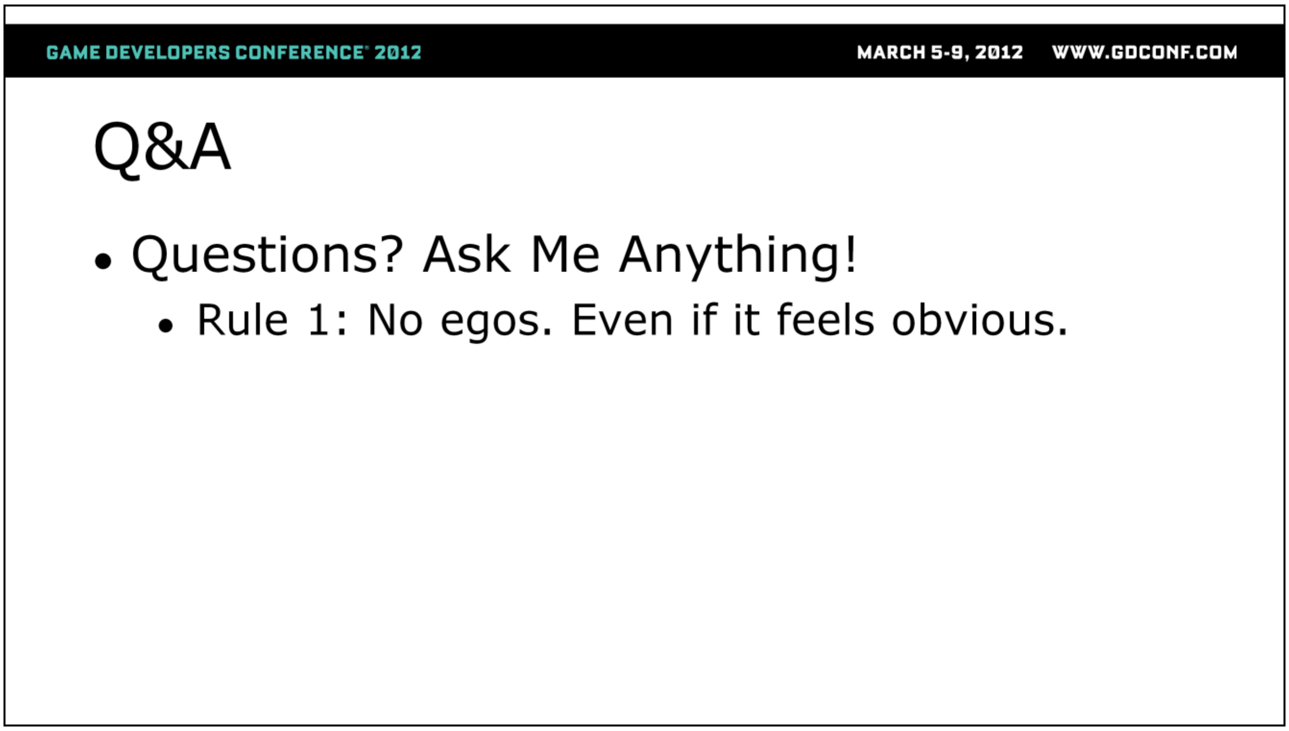
GAME DEVELOPERS CONFERENCE® 2012 MARCH 5-9, 2012 WWW.GDCONF.COM

Everyone should relax… nothing stressful or schedule driven for a month or two. You need to wind down. Take a vacation. You've earned it!



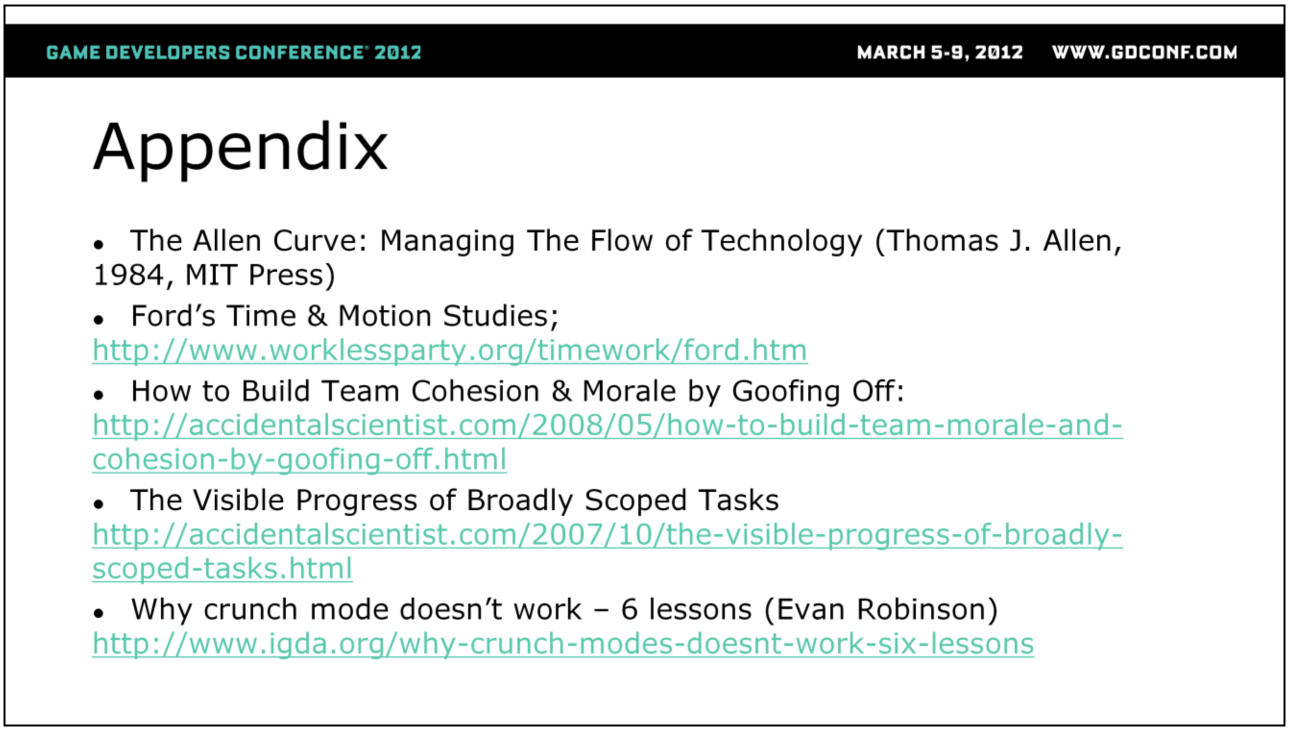
Takeaways

- Everything boils down to communication
  - You can solve every problem with it
  - It's the cause of and solution to every problem
- Fix small problems while they're still small
  - Communication, Team Dynamics, Schedule… you name it! – fix it early! Be on the ball!
- Pay attention to the details – they matter!
  - If you don't, no one will. Everyone else is knee deep in their own problems.



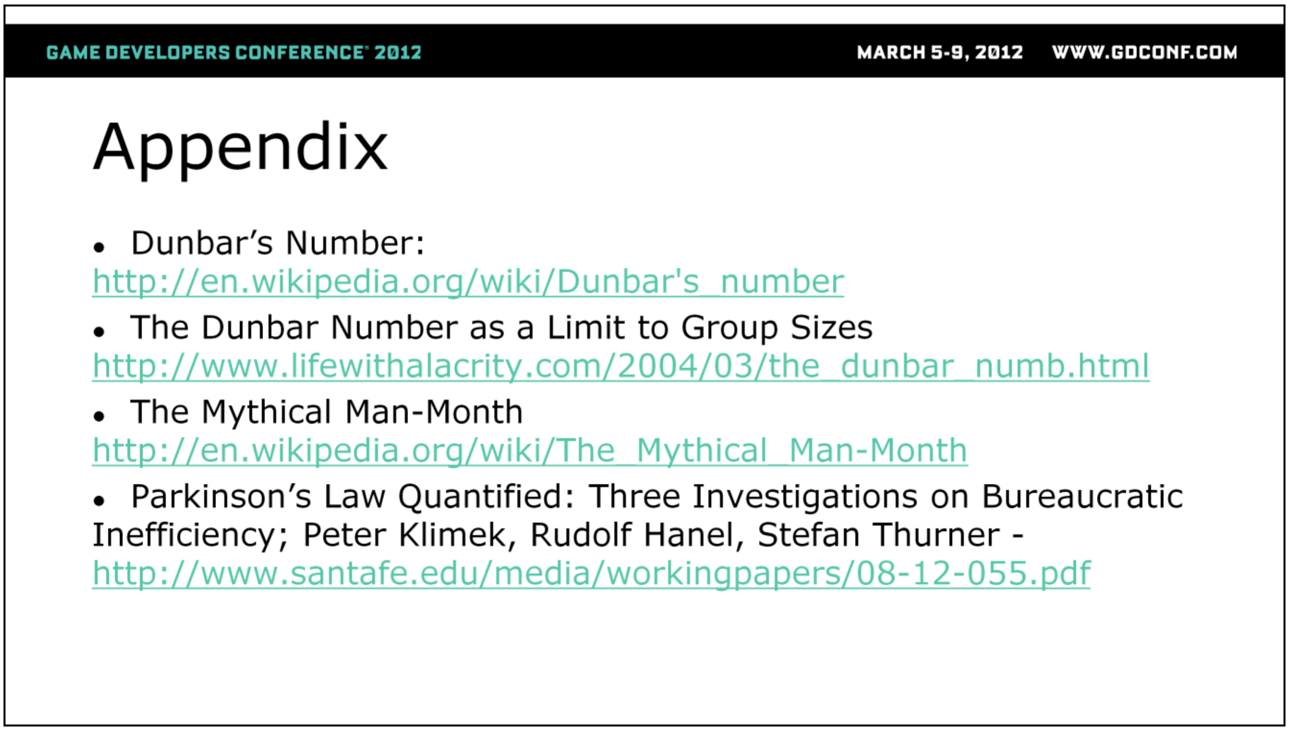
Q&A

- Questions? Ask Me Anything!
  - Rule 1: No egos. Even if it feels obvious.



Appendix

- The Allen Curve: Managing The Flow of Technology (Thomas J. Allen, 1984, MIT Press)
- Ford's Time & Motion Studies; http://www.worklessparty.org/timework/ford.htm
- How to Build Team Cohesion & Morale by Goofing Off: http://accidentalscientist.com/2008/05/how-to-build-team-morale-and-cohesion-by-goofing-off.html
- The Visible Progress of Broadly Scoped Tasks http://accidentalscientist.com/2007/10/the-visible-progress-of-broadly-scoped-tasks.html
- Why crunch mode doesn't work – 6 lessons (Evan Robinson) http://www.igda.org/why-crunch-modes-doesnt-work-six-lessons



Appendix

- Dunbar’s Number: http://en.wikipedia.org/wiki/Dunbar's_number
- The Dunbar Number as a Limit to Group Sizes http://www.lifewithalacrity.com/2004/03/the_dunbar_numb.html
- The Mythical Man-Month http://en.wikipedia.org/wiki/The_Mythical_Man-Month
- Parkinson’s Law Quantified: Three Investigations on Bureaucratic Inefficiency; Peter Klimek, Rudolf Hanel, Stefan Thurner - http://www.santafe.edu/media/workingpapers/08-12-055.pdf