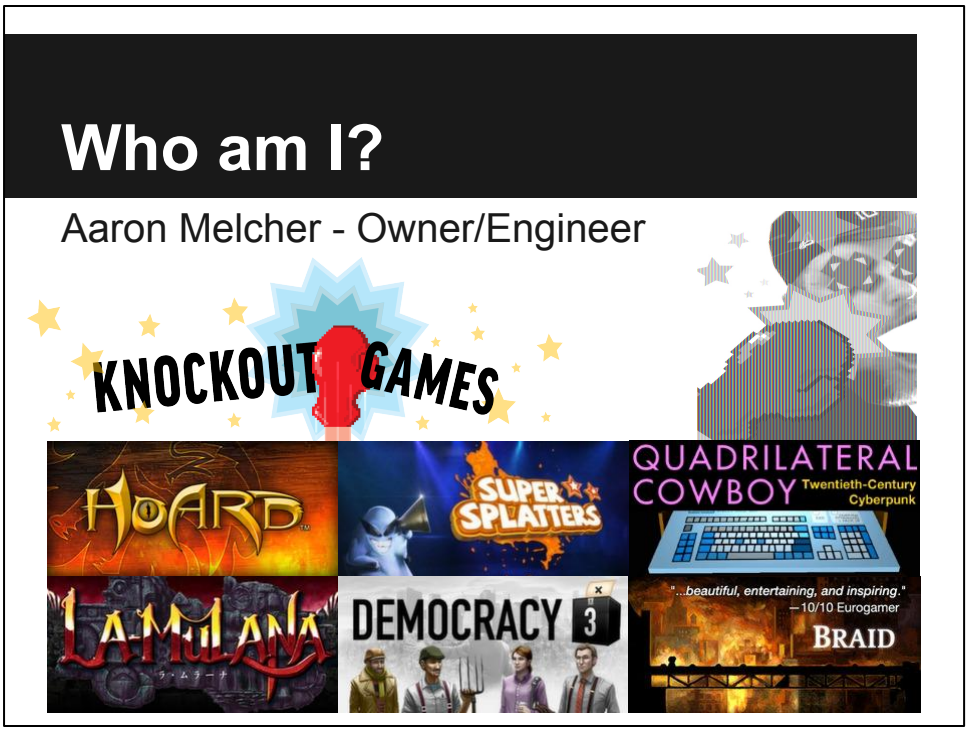
Your code sucks, deal with it.



Who am I?
I'm Aaron Melcher. Owner and Engineer at Knockout games. My days are spent porting video games from around the globe to platforms they don't exists on. It's awesome and I love it.

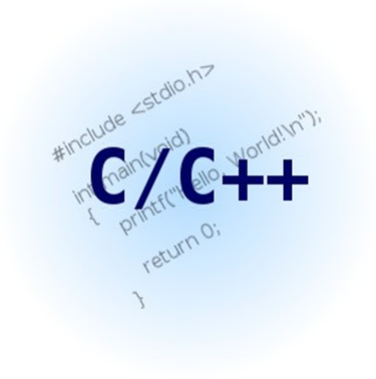
What else?

What else have I done?

I got my start developing Bar Games. I would later go on to work on bargain bin hits, AAA PC games and I even helped ship a game called Snapshot.

Bias info

FULL SAIL
UNIVERSITY



Some disclaimers. I have a gamedev degree from FullSail, which isn't a real degree. I've only worked on games. I've worked with many different languages, but have mainly coded with C/C++.

So...

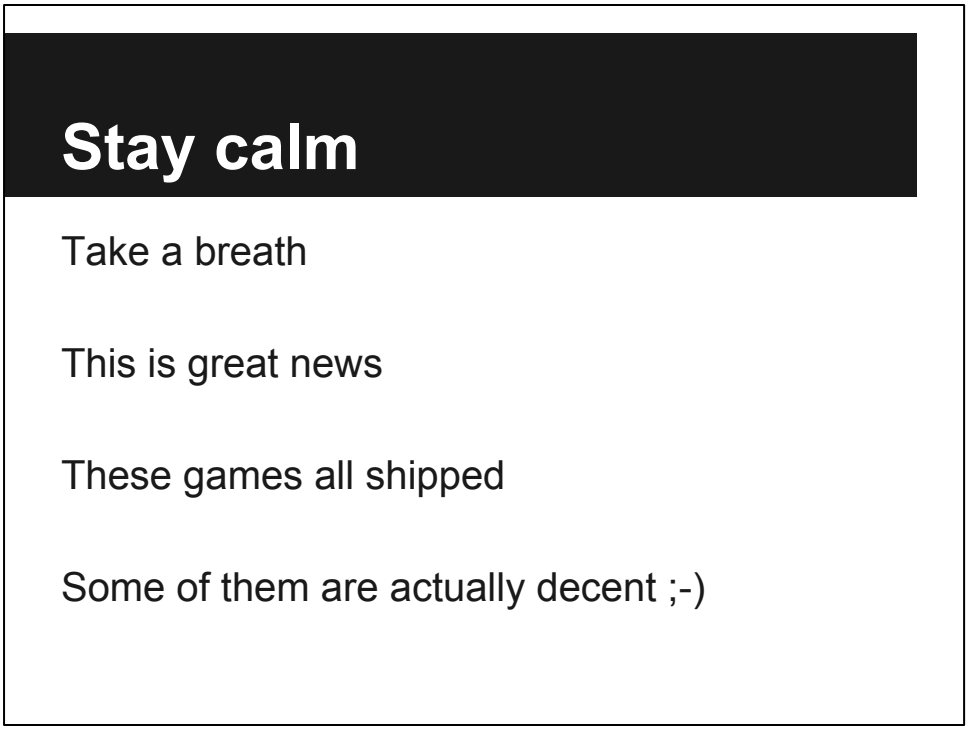
I’ve seen a bit of code

and

Wait for it

90% of that code SUCKS

So
I’ve worked on a bit of code
and
wait for it
90% of that code SUCKS



Stay calm guys and gals
Take a breath
This is great news
These games all shipped
Some of them are actually decent ;-)

The Point

Don’t Focus on Writing the Best Code

Let’s get to the point.

Don’t focus on writing the best code.

The Point

You Will Fail

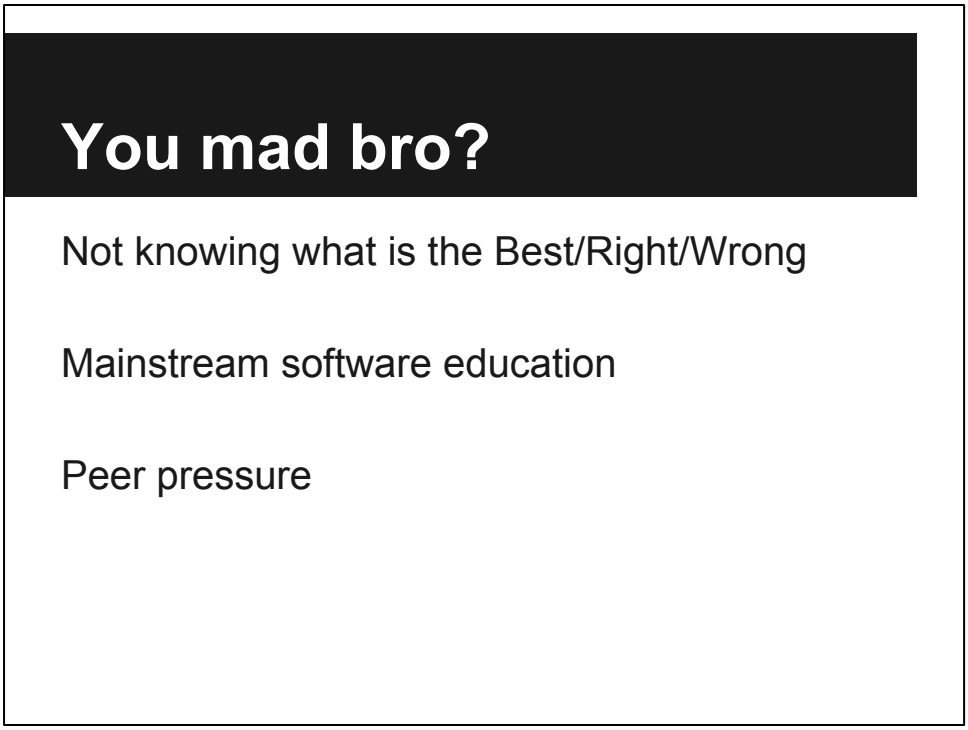
To will fail

The Point

Instead Focus on Writing Simple Code

Instead focus on writing simple code.

Now before I move on, I'd like ask everyone to recite the next words you see. Ready, Set, Go!



audience: You mad bro?

Yeah I'm mad. As if programming isn't hard enough you have to worry about doing things the "Right" way. When you're first starting out you don't know what is right or wrong. So what do most people do? Seek mainstream education. Problem with that is it's focused towards large production software engineering. Most video games shouldn't match this description.

Eventually you might work with another experienced engineer. He will peer pressure you to adopt his "best" practices. Problem is he is pulling his information from the same sources you are trying to understand. There is too much interpretation going on to make heads or tales of everything.

IT SUCKS

Clarification

Best Code

- Elegant OOP
- New lang features
- Platform fanciness
- Security
- Coding for future
- Boilerplate code

Simple Code

<= None of that

Lets quickly clarify.

When I say “Best” code I mean things like this. I know, I could add a few more acronyms. All of these things increase the complexity of code, but there are benefits right? RIGHT?

Now let’s clarify what I mean by “Simple” Code. Yeah… basically none of that.

How to deal

1. Keep things simple
   a. Prefer coarse granularity
   b. How much work to change/replace a system?
2. Minimize keystrokes
   a. Avoid too much boilerplate code
   b. Don’t write noisy comments
3. Don’t believe everything you hear or read*
   a. Try things out and review
   b. Keep techniques that improve productivity

*Including this talk

Here is how I deal.

Keep things simple. This really boils down to how spread out your code is. How much code do you have to change to tweak or add to a system?

Minimize keystrokes. My hands hurt, please don’t make me write a ton of code to extend or change a system. Also save yourself some keystrokes by avoiding gabby commenting. Summarizing blocks of code is great, summarizing a single line of code is a waste of time.

Don’t believe everything you hear or read, including this talk. Try out things and check in on yourself to ensure development is actually improved. Are you just typing more code for the same result? Track yourself.

Use case: the Newb

- Has little or no programming knowledge
- Is able to understand the gist of what needs to get done
- Implements shoddy but simple code
- Can get stressed out by the details

Lets quickly examine two stereotypical programmer archetypes.

First there is the newb. We've all been there and we all continue to be newbs at many things in our field. They're dumb, they make mistakes, but they are passionate and the job gets done, some how.

Use case: the Smarty

- Has read about design patterns
- Written a few factories and message systems
- Knows how to craft a family tree of classes with the best of them
- Writes their code to withstand the apocalypse and make the world a better place

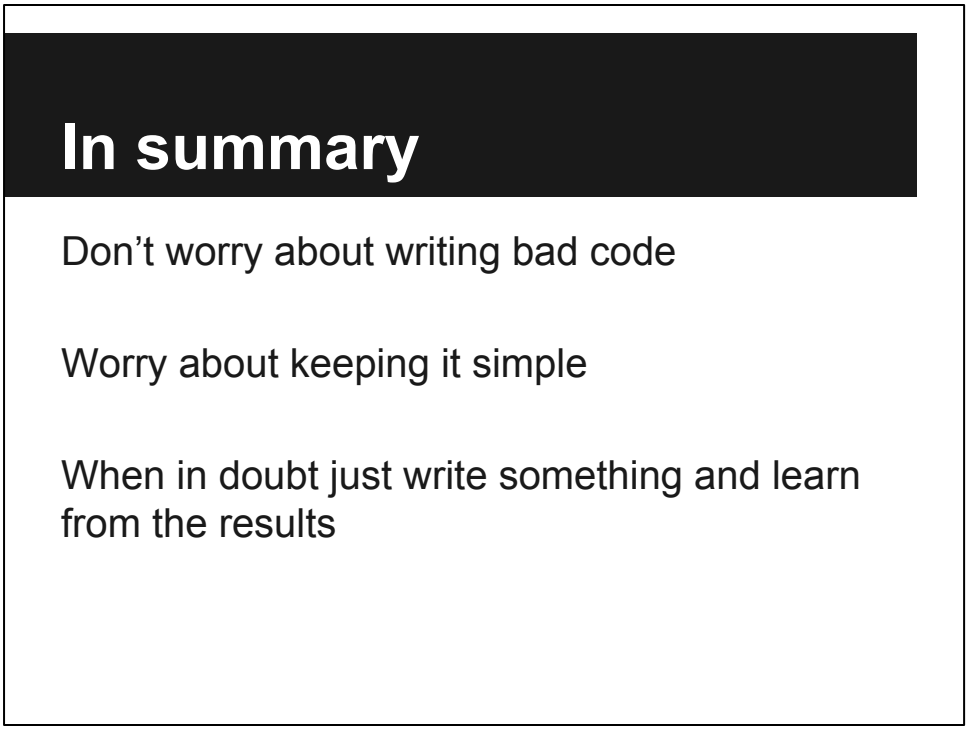
Second there is the Smarty.

These folks, bless their little hearts. They study up on the latest design patterns, write fancy systems and their code has so much error checking the cockroaches will be able to maintain it after the nukes drop.

Lets compare

Newb	Smarty
● Takes longer, but gets the job done	● Designs code to account for everything
● Fixes issues that were not planned for	● Fixes odd issues with their complex system
● Seeks mentorship for harder problems	● Rewrites their system when things change
● Smarties grumble at their code	● When I see their code I grumble
● **Changing their code is quick**	● **Changing their code takes extra time**

-Lets compare
-Its a tough road for the newb. They have some challenges, but they overcome. Their code seems kind of janky, but they don't know enough to really make anything all that complicated. Smarties hate their code, it's just icky.
-Smarties have an even tougher road. If they're at the top of their game they will take just as much time to complete their work as the Newb. The bonus of course is that the code is pretty, has lots of files and will support everything + the kitchen sink in terms of functionality. I hate this code, for the most part.
-WHY Aaron? Why do you hate functionality and well designed code?
-Well since the Newb's code is simple, making tweaks and iterating is easy.
-While Smarty's code is another story. Not only do I have to wrap my head around their complicated systems. I have to deal with the ripple effect of any changes I make across all the systems that are connected.
-All the extra energy the Smarties put towards elaborate designs is much better focused on maintaining the Newb workflow paired up with their increased coding skill.



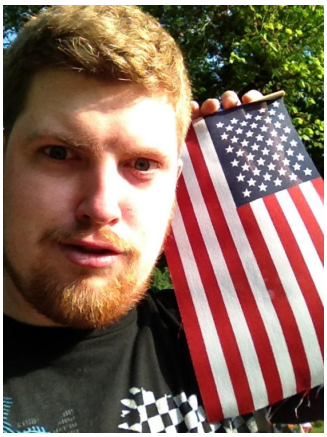
So in summary

Don’t worry about writing bad code, it’s going to happen regardless

Instead worry about keeping things simple so you can easily tweak functionality. Games are so mushy you can’t predict or design for everything. Simplicity is the only tool we can utilize to fight against the unknowns in game development.

Above all else, if you’re unsure, just write something and see how it goes. All of what I’ve presented is my opinion, in the end you have to do what feels right to you.

Thanks



Aaron Melcher
@knockoutgames
aaron@knockoutgames.co
http://knockoutgames.co

Lets continue the conversation. This my online presence. Thanks for your time, feel free to connect.