Writing robust code

November 2010

What is ”robust” to me?

This.

It works.

It is not very complicated.

You know that it is not going to let you down at a critical moment.

Has this ever happened to you?

"I thought it would take 3 days to do this, but ... something came up."

"We have a milestone on Friday ... and everything stopped working this morning."

What's that? Surprises during development?

Why robust code is important

Surprises mess with the planning.
Surprises make you work weekends, or miss milestones.
Surprises are bad.

Robust code => fewer surprises.

Strategies for writing robust code

"I have been asked to implement feature X"

- Do we even need the feature?
- Who will use the feature?
- Is the overall code-design good enough?
- How do I test the code?

Do we even need the feature?

If you don't know why the feature is needed, then you are programming blindly.

That is bad. Real bad.

You can't do a good job if you don't know why you are doing it.

Who will use the feature?

The person that will use the code is your customer.

Customers can be in other programmers in the team, or artists, or actual players.

If you implement the feature, but your customers cannot use it - because in reality they needed something else - then the feature is useless.

Good code-design

Rule of thumb:
If it is simple to explain to someone else, then it is a good design.

Other people should be able to guess how your code works … and get it right at the first guess!

Good code design - robustness

Rule of thumb:

If you can convince someone else that your code can never ever

- leak memory
- get stuck in an infinite loop
- return garbled results

… then it is good code design

Good code design - self-review

Play devil's advocate:
”What if this pointer is 0?”
”What if this loop will run just 0 or 1 times?”
”What if that memory allocation fails?”
”What if there is no player object yet?”

Try to figure out ways in which your code can fail, and then **re-design** it so that it is foolproof

Relax.

We'll continue in 5 minutes.

Let's look at some code

<check out texture streamer examples>

Relax.

We'll continue in 5 minutes

Let your compiler help you!

Build with warnings-as-errors: all code MUST compile with 0 warnings

- if there is a warning.. then you just introduced it!

Sometimes benign, but it pays off in the long run

Let your computer help you!

Create unit tests: describe more complex relationships in the code, which the compiler doesn't know how to check for

Making good unit tests is difficult; learn how to do it on a small scale first

Let your build system help you!

It's OK to sometimes break the build by accident, because the build system compiles your code all the time. Just make sure – if you break the build, fix it quickly.

This becomes more important on multi-platform projects with a lot of SKUs.

Catching errors during runtime

Asserts are your friend.
Like compiler errors, but during runtime.

They test code+data in conjunction.

Runtime errors

Errors come in two forms:

Errors you should find & fix before you ship the game – bugs, that is
Example: a datafile is missing on the DVD

Errors that occur during normal execution
Example: the user removed the DVD during loading

If you're going to crash, crash early.

Let's say that some code is opening a file, but the file open fails. How do you know if the game should crash, or continue?

Crash if: you must fix this before the game ships. Print debug info, and stop!

Continue if: this is expected to sometimes happen after the game has shipped.

Hmm... what about data?

A lot of the stuff said previously also applies to data.

How about: warnings-as-errors when building data?
Unit tests when building data?

Would it be good? Would it be bad?
Which is to prefer?

An ideal to strive for...

… your code has no known bugs in it

… your colleagues can use your code, without needing to ask you for help

… after 6 months, you can change your code without introducing new bugs

… once you have quit, your colleagues can change your code without introducing new bugs

Questions?

Thank you.