
1978
19:19
C++ Pitfalls
ACTIVISION

C++ Pitfalls

- Modern C++'s biggest problem is Old C++
  - There are a lot of sharp edges from C and C++98 and earlier that are still present in the language
- If you use the right subset, C++ is a pretty safe / easy to use language
  - Unfortunately, no way to force use of that subset (if people could agree on it, which they don't).
- Maybe you've only dealt with modern C++
  - If so good for you, you'll already be doing (many of) these things
  - May have problems at boundaries with libraries with a less enlightened view or older sample code
- Here are some simple rules for sticking to a clean modern subset of C++ and avoiding some big sharp edges

No Pointers

- Prefer value types and pass by value if possible, pass by reference if you absolutely need to
  - “SomeClass foo;” vs. “SomeClass* foo = new SomeClass”
  - “void func(SomeClass& thing)” vs. “void func(SomeClass* thing)
- Never use raw pointers (SomeClass*)
  - Always use unique_ptr or shared_ptr to wrap things that must be heap allocated
- Also implies no C-style arrays or strings
  - Use std::vector and std::string
    - Lots of game studios eschew STL due to some issue with allocation but those issues don’t affect you

Use RAII

- Resource Acquisition Is Initialization
  - All resources should be acquired by constructing an object on the stack, and the destructors should release the resource
- Makes code both cleaner and exception safe

Use (Some) C++ Casts

- static_cast<SomeClass*>
  - Not “(SomeClass*)notSomeClass”
- dynamic_cast is cool too
- Mildly avoid const_cast, and really avoid reinterpret_cast unless you are 100% sure you need them

Avoid Multiple Inheritance

- Complicated and poor-performing relative to how much value it gives
  - Lots of weird gotcha's you need to understand (i.e. virtual base classes)
- There are times it is the only reasonable solution to a problem
  - You probably aren't going to hit any of those times

Avoid Exceptions

- Exceptions are a good thing in general
  - Used to have some problems, particularly in games
  - Still do have a few problems
- A large, well designed C++ app probably SHOULD use exception
- However getting correct error handling and recovery behaviour is hard enough that it’s not worth bothering with for your projects
- Just assert() and fail immediately on errors.

Some Caveats

- Not all of this applies in general
  - No Pointers, Use RAII, Use C++ Casts probably do
  - No MI and No Exceptions are more “in my opinion” or “for this project”
- Other organizations you might work for in the future may have a different set
  - “The good thing about standards is that there are so many to choose from.” — Andrew S. Tanenbaum
- Games tend to be pretty mired in Old C++
  - You might need to learn how to deal with some of the old crud eventually
  - No reason it has to be today though