Game Programming Patterns_Revisited_Singleton

* singleton usually does more harm than good & is overused

"Ensure a class has one instance, and provide a global point of access to it."

* sometimes a class can't perform if >1 instance, ex if it ints w/ an ext. system that maintains global state

① private ctor + instance_ member variable
② public static instance() method to lazy instantiate & return instance

Pros: never initialized if not needed & intialized @ runtime & can subclass it.
Cons: it's global (makes it difficult to reason abt code)
(encourages coupling)
(aren't concurrency friendly

"... Singleton IS global state — it's just encapsulated in a class"

pure functions: functions that don't access or modify global state

* ex. one logger, but what if we want to write to different logs? w/ singleton you'll now need to modify the class & every line of code
* ex. for games, lazy loading takes control away from dev, it helps to control when objects are initialized.
  ↳ can choose to have a static class instead w/ simpler syntax

→ do you need the class?

"Many of the singleton classes I see in games are "managers" — those nebulous classes that exist just to babysit other objects."
↳ poorly designed singletons are often "helpers" to add functionality onto another class, but you can move that code to the obj. itself.

→ single instance w/o global access
↳ can check if the class has been created & fail if it already has
↳ an assert() is a contract in your code that if broken should be fixed asap, before it results in a bug.
→ convenient access

Convenient access cont'd

→ pass it in
→ get from base class w/ wide but shallow inheritance
→ " " something already global
→ " " a service locator

Dependency Injection: dependencies are pushed in to the code that needs it through params

Cross-cutting Concern: things like logging that appear scattered in codebase

Law of Demeter: aka principle of least knowledge

① each unit should only have limited knowledge about other units: only units "closely" related to the current unit
② Each unit should only talk to its friends; don't talk to strangers
③ Only talk to your immediate friends

*look into Subclass Sandbox + Service Locator patterns