C++ coding conventions

- prefer reference as an argument of functions
- prefer reference instead of pointer whenever possible
- use shared_ptr or unique_ptr.
  - avoid naked new / delete
- make ownership explicit
- use expect for pre-condition and ensure for post-condition
- use ensure for class invariant
- use exception for exceptional error
- use result template for return value if it returns error
  - make result boolean to check failure
- use config structs to pass options
- use structs to hold related data
- use types to wrap simple types and force semantics on it
- use constness whenever possible
  - use const parameters
  - use const functions if it is const
  - remember lock is mutable in const functions
- use constexpr for const values
- use the same namespace as class for free helper functions
- use fluent style when proper
- use declarative style when possible
- use operators to provide convenience
- use std::* if it exists
  - use stl for containers
  - use algorithms, use std::to_string, and etc.
- use lambda instead of bind / functor
- use using instead of typedef
- use auto when possible
- use decltype when possible
- use rvalue and move carefully
  - don't use it when performance is not critical
- do document parameters, return value and exceptions
- do unit test
  - do functional test as unit test if possible

- do review code
- do review design

practice conventions

- write unit test for all classes, especially for library classes.
    - aim for 100 % code coverage, especially for library classes.
- write function test for game
    - aim 100% code coverage for important classes and for all classes as much as possible
- make design simple
- make code look simple and easy to read