

Engine System Update #1

In general, In order to make a networking connection system. I have studied windows socket api, and their examples. I have read some articles about network system design about and C++ concurrency.

I have studied the basic usage of windows socket following Microsoft tutorial[1]. They have an simple example of communication under tcp protocol. I will rap those functionalities with a class, and provides platform independent interface for later usage. I have read a section talking about networking multiplayer game loop[2], which mentioned a *client-on-top-of-server* pattern. In general, *client-on-top-of-server* pattern will have the client send user inputs to the server, and the server will do all the logic and send information about the game. For my system the server will return inform about states all the objects that can be updated by the server to the clients.

I have studied the first two chapter of a book[3] talking about the threads and concurrency of C++ standard library. There are lots of implementation details discussed in this book, which could be very helpful to know considering I am going to implement a Multi-Threaded system.

Screenshot of notes I have taken

avoid local pointer used in thread. when you detach it.

Solution: do hard copy on variables threads use.

join and detach might be skipped if exception occurs.

local variable got destructed in the inverse order of their declaration.

detach and join can be called once when thread is joinable.

Pass argument to thread

using a callable object in thread will get the object copied.

Pointer got Pointer copied (address of the variable)

reference does not work.

In general, by default the arguments are copied (deep copy) into internal storage, where they can be accessed by newly created thread of execution, and passed to the callable object or function as rvalues as if they were temporaries. (which would not work with non-const reference)

will lambda functions get copied to the new thread (Yes?)

Reference

- [1] <https://docs.microsoft.com/en-us/windows/win32/winsock/getting-started-with-winsock>
- [2] Section 7.7, Gregory, J. (2017). Game engine architecture. AK Peters/CRC Press.
- [3] Williams, A. (2012). C++ concurrency in action: practical multithreading. Manning Publ.