**Bonsik, Kong**

[Kongbong23@gmail.com](mailto:Kongbong23@gmail.com) / skype : [kongbong23@gmail.com](mailto:kongbong23@gmail.com) / <http://kongbong.github.io>

**Experience**

**Neowiz games / Bless project**

Seoul, Korea / 2010 – current

* Server side lead programmer. Team manager since 2013.
* Improved performance and memory usage.
* Made whole server code from low-level core systems to top-level features.
* Made a DB caching server (C#, ORM DB).
* Made DB procedures and DB schedulers.
* Made a login server that integrated with neowiz authority server (TCP and JSON web query).
* Made a processes monitoring tool (C#, WPF).
* Made a chatting program for in-house chat.

**Nexon / Mabinogi project**

Seoul, Korea / 2008 – 2010

* Content programmer.
* Programming front and back-end both.
* Made many game features (skills, pets, mini-games, and main story contents)
* Achieved the most earning in Mabinogi history with a pet that I made.

**NDoors / Kungpa project**

Seoul, Korea / 2007 – 2008

* Back-end programmer.
* Made a match making server.
* Made a dummy client that is for increasing performance.
* Made a processes monitoring tool.
* Made a UDP hole-punching logic.

**Education**

**Sangmyung University, Game design, Master’s degree**

**Seoul, Korea. 2004 – 2006**

**Graduating paper: A Study on the Requirements of Long-term Playable Strategy Game**

**Konkuk University, Electronic engineering, Bachelor’s degree**

**Graduation portfolio: Camera device using serial port**

**Using techniques: C, C++, x8086, MFC, CMOS, OrCad**

**Project Management Professional** 2009 expired

**Skills**

C++, C#, MSSQL, Node.js, Python, Ruby, Java

**More details what I did in Bless project**

**I improved max current-user count from 2500 to 5000 on same machine.**

* RSS (Receive side scaling) enable (solved server lag)
* Divided threads into worker threads and network threads.
* Improved async tasks serializer performance (Improved 9 times copy operation to 1 time)
* Solved lock-free queue’s corruption problem.
* Removed unnecessary interlocked operations. (It caused a cache lock)
* Improved performance and memory usage using profilers
  + Vtune, in-house profiler (<https://github.com/kongbong/Simpleprofiler>), Windows performance analyzer, Unreal’s Memory profiler

**I improved a thread local memory allocator.**

* We had an in-house memory allocator.
* It use thread local memory block queue like glibc.
* It had an inequality problem that some thread’s queue were always grown up.
* I solved it by having a global queue.
* And I improved its performance using lock-free queue.

**I made NPC AI decision maker using BT (behavior tree)**

* We had an NPC AI decision maker using lua script.
* But, It caused many problem (Unskilled designer made a performance consuming scripts)
* I made a behavior tree builder using C#. Designer can write AI using that tool.
* BT has an advantage making acting scene easily.

**I made a DB caching server (using C#)**

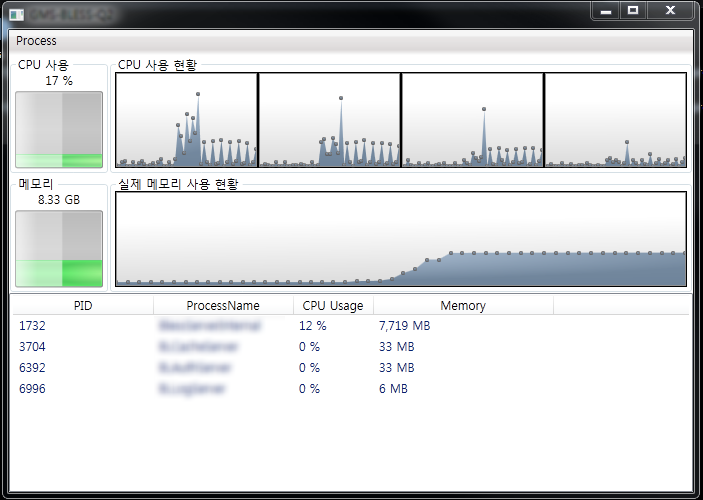
* It is a kind of OODB. It analyzed its scheme objects using C# reflection, and it made its tables and stored procedures.
* Many tasks can do on single transaction.
* Independency tasks do in parallel.

**I made main game systems.**

* Quest system.
* Inventory system
* Skill system.
* Event driven programming structure (<https://github.com/kongbong/EventListener-for-c>)

**I made many server operating Tools**

* Protocol generator
  + It parse protocol defined file (xml format) and make C++ headers, cpp files and C# files for client and servers (login server, world server and DB caching server)
  + It can be extended by modifying config file.
  + It made by ruby script.
* Content Builder
  + Transform Excel data file to xml data file.
  + It parse xsd scheme file and make C++ headers, cpp files and C# files for client and server that read xml files and make data structure instances.
  + It read Excel data file, transform xml file and check validation using XSLT check rule file.
* Monitoring Tool (C# WPF)
  + Server processes monitoring tool
  + It consist agent, server, client – 3 tier architecture.
  + Agent collect target processes CPU and memory usage. And send it to server.
  + It support console In/Out to target server. Monitoring client can view target process’s console out and can input console Input command.
  + It support collect current stack trace. So, we can analyze dead lock and infinity loop problems.
  + It support remote kill and start process.



* Bless Chat (C# WinForm)
  + In-house chatting program.
  + Group chat, 1 : 1 chat, send image, send clipboard, make a group, emphasis group message, view last messages (it save messages for 6 months), dice and vote macro.
  + It use SQLite File DB.

