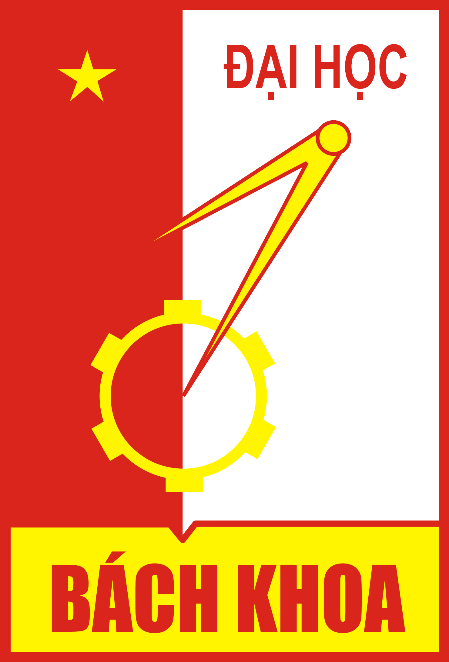
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND TECHNOLOGY



NETWORK PROGRAMMING PROJECT

SWORD, SHIELD, SHOTGUN

A 3D online multiplayer game

Pham Le Duc – 20161128

Nguyen Tuan Hai – 20161311

Hanoi, 2020

# Overview:

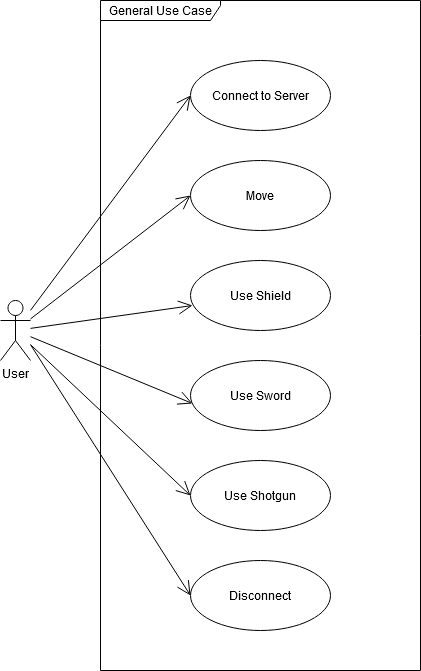
Sword, Shield, Shotgun is a player versus player game where players are thrown into an arena and have to battle each other, the last one to survive wins the game.

Player can use three equipment:

* + Sword: close range, one hit kill
  + Shield: block 1 attack, long cooldown
  + Shotgun: high range, low damage

# Functionalities:

Player has can do the following actions:

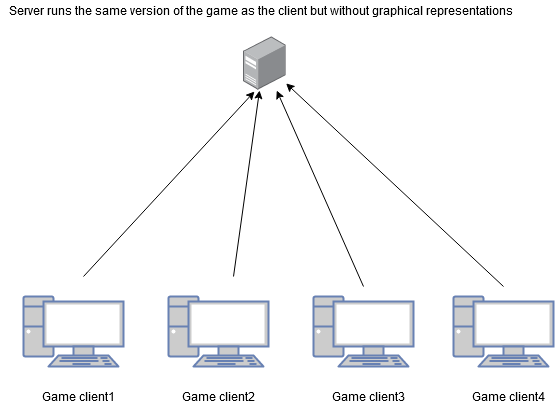


# Network Architecture:

The game will use server – client model, utilizing both TCP and UDP protocol. The server will use asynchronous socket (non-blocking).

The server will handle all the game logic using an identical version of the game as on client side.

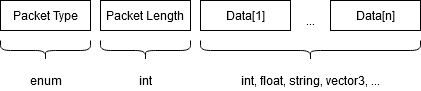
The server will receive/send data in the same interval as the game’s frame per second.



# Message Design:

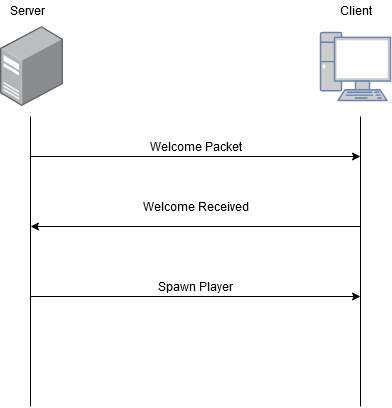
Convert data into byte array and send. Received data will be converted back into original data type to process.

General Message Format:

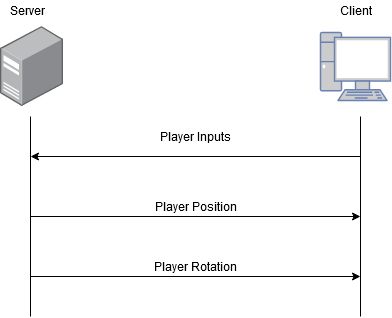


* + - Packet Type: enum constants include: spawnPlayer, playerPosition, playerRotation, playerHealth, playerDisconnected. With each type of enum, the packet format will be different.
    - Packet Length: include the length of the packet at the beginning for easier data handling. Only String type needs the packet length, and each data index will have its own length.
    - Data[1] … Data[n]: the main content of the packet, can take any type and will be converted back to the respective type when the packet gets handled
    - Eg: spawnPlayer5user118username, where spawnPlayer is a package type, for this type, it is required to send data in string with each index having a specific length
    - Eg: playerPosition578, where 5 is the position in the x-axis and 7 is the position in the y-axis, 8 for the z-axis, both have 4 bytes (because they are integers). Here we do not need to append the string length.

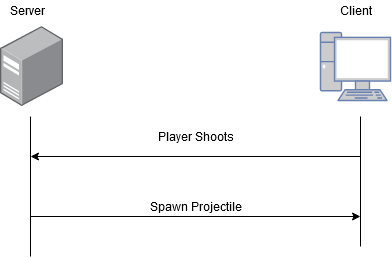
Packet request – response relationship:



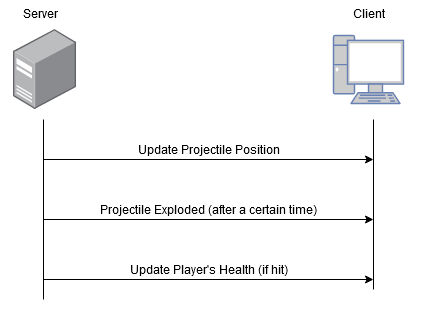
*When client connected successfully*

**

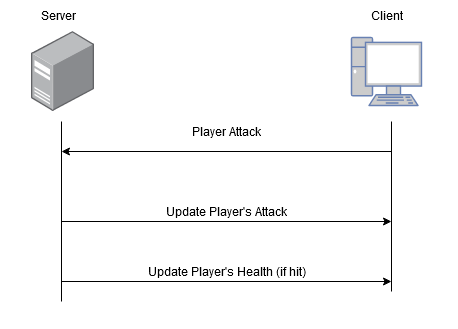
*Update player’s movement and position*

**

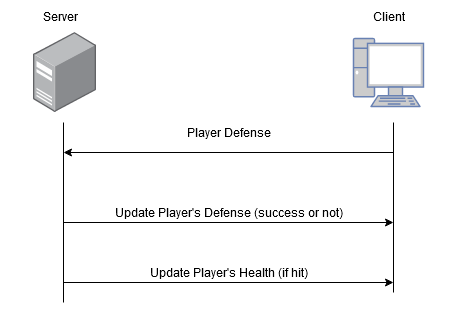
*When player shoots with shotgun*

**

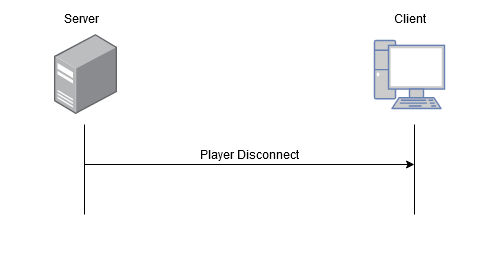
*Bullet hit check after player shot*

**

*When player attacks using sword*

**

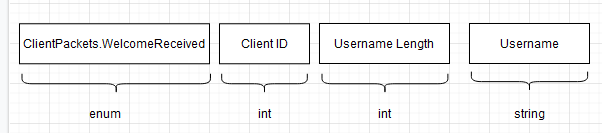
*When player defense using shield*

**

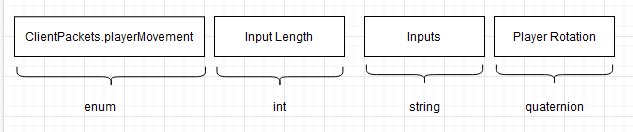
*When a player disconnects from the game*

Client Messages:

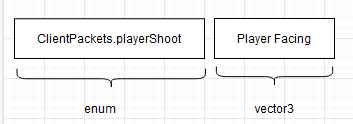
* + - Connected Successfully (let the server know that the client received the welcome package):
      * Format:



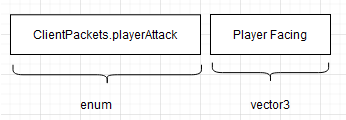
* + - * Example: ClientPackets.welcomeReceived 1 5 user1
    - Send Player Input (for server to calculate actions and position/rotation/state of the player):
      * Format:



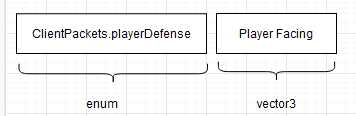
* + - * Example: ClientPackets.playerMovement 20 awasdwadsawd<mouserotate>awdda<mouserotate> 3 4 5 2
    - Player Shoot (send the facing vector so that the server knows where to spawn the bullet):
      * Format:



* + - * Example: ClientPackets.playerShoot 1 2 3
    - Player Attack (send the facing vector so that the server knows where to do the hit checking):
      * Format:



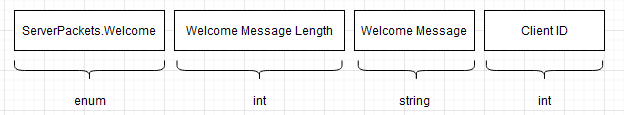
* + - * Example: ClientPackets.playerAttack 1 2 3
    - Player Defense (send the facing vector so that the server knows where to check if the defense successful):
      * Format:



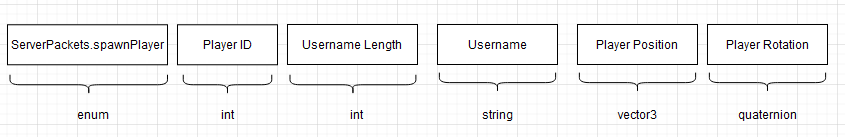
* + - * Example: ClientPackets.playerDefense 1 2 3

Server Messages:

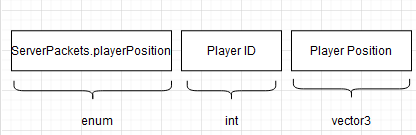
* + - Connect Response (welcome message):
      * Format:



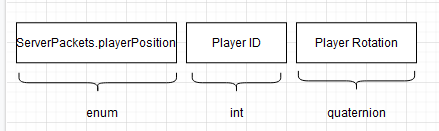
* + - * Example: ServerPackets.Welcome 21 Welcome to the server 1
    - Spawn Player:
      * Format:



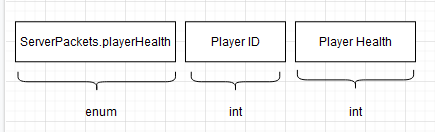
* + - * Example: ServerPackets.spawnPlayer 1 5 test1 0 0 0 1 1 1 4
    - Player Position Update (send to all clients on the server):
      * Format:



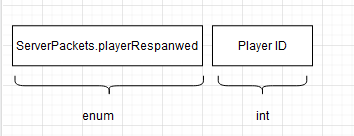
* + - * Example: SeverPackets.playerPosition 1 3.4 1.45 3.56
    - Player Rotation Update (send to all clients on the server):
      * Format:



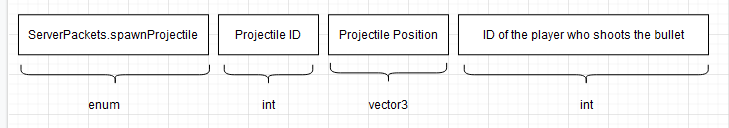
* + - * Example: ServerPackets.playerRotation 1 5.6 32.4 51.5 65.3
    - Player Health Update (send to all clients on the server):
      * Format:



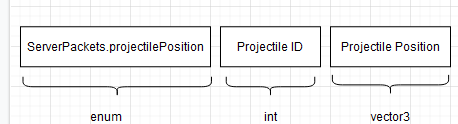
* + - * Example: ServerPackets.playerHealth 1 2
    - Player Respawned (send to all clients on the server):
      * Format:



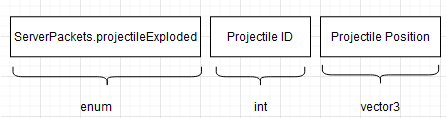
* + - * Example: ServerPackets.playerRespawned 1
    - Spawn Projectile (spawn a bullet when player shoots shotgun):
      * Format:



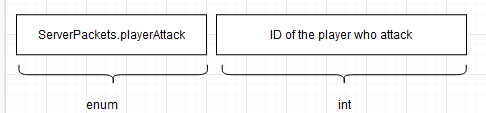
* + - * Example: ServerPackets.spawnProjectile 1 3 4 7 3
    - Projectile Position Update:
      * Format:



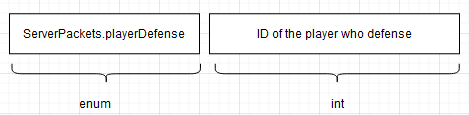
* + - * Example: ServerPackets.projectilePosition 1 3 2 5
    - Projectile Exploded:
      * Format:



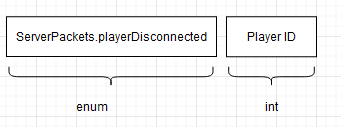
* + - * Example: ServerPackets.projectilePosition 1 3 2 5
    - Player Attack (when player use sword to attack):
      * Format:



* + - * Example: ServerPackets.playerAttack 1
    - Player Defense (when player use shield to defense)
      * Format:



* + - * Example: ServerPackets.playerDefense 1
    - Player Disconnect:
      * Format:



* + - * Example: ServerPackets.playerDisconnected 1