Course:	**INFO-6016 - Network Programming**
Professor:	**Lukas Gustafson**
Final Project:	**4 Player Networked Game**
Due Date:	**Sunday December 17 2023, at 11:59 PM**

Description and Purpose

Create a simple multiplayer networked arena shooting game in C++ using OpenGL. The game should have networking as its core component, and be server authoritative.

The game should allow a player to control a unique character in the game. The player is able to move the character along the xz axis, and shoot a bullet in the direction the player is facing.

Player Controls Example:

- **W** - Forward
- **S** - Backward
- **A** - RotateLeft
- **D** - RotateRight
- **Space** - Fire

When the player presses the fire key, a bullet is fired from the player's position, and moves in the direction the player is facing. This is a single shot game, where each player only has 1 bullet. The bullet is returned to the player if it has hit an opponent or if after 2 seconds has passed. The player should only be able to fire their weapon if they have their bullet.

The camera can be 3rd person or static, but must include the ability to change to a birds eye view camera to be able to see all players in the arena.

This assignment can be done alone or in a group of up to 3 members.

I suggest using https://trello.com to manage the work that gets done on this project.

You **must** include a ReadMe that includes instructions for building and running your project. You **must** include a video of your project working, and showing each feature you have completed.

Details

You are tasked with creating a 4 player multiplayer game. Your game must feature a player controllable object that is able to move on the xz plane. Your character must be able to fire a weapon in the direction they are facing.

Protocol (7 marks)

1. Create a protocol that uses either your own buffer or google protocol buffers as its serialization and deserialization method **(2 marks)**
2. Must use the same source files on the client and server **(2 marks)**
3. Must implement a protocol similar to the one below **(3 marks)**

Here is an example of a protocol you may use as a reference (in pseudo code)

```
message UserInput {
        int32 id = 1,
        int32 input = 2; // [FORWARD, BACKWARD, TURN_LEFT, TURN_RIGHT, FIRE]
}

message GameScene {
        int32 id = 1,
        repeated Players players = 2;
        repeated Bullets bullets = 3;
}

message Player {
        int32 state = 1 // [IS_ACTIVE, IS_CONNECTED, HAS_AMMO]
        vector3 position = 2;
        vector3 velocity = 3;
        quaternion orientation = 4;
}

message Bullet {
        int32 state = 1; // [ACTIVE]
        vector3 position = 2;
        vector3 velocity = 3;
}
```

Game Server (25 marks)
Must run an update loop, to handle player input, collision detection, bullet state, etc, at a reasonable rate (preferably 60 HZ)
Must only send new game state to clients five times per second (5 HZ)

1. Must use **UDP (5 marks)**
2. Must be able to handle up to 4 simultaneously connected players**(10 marks)**
    a. Each player must be moved from client input
    b. Each bullet must be updated on the server.
3. Implement **Server Reconciliation** for all packets **(10 marks)**

Game Client (35 marks)
Must render the game at a reasonable rate (preferably 60 FPS)
Must only send input data to the server five times per second (5 HZ)

1. Must use **UDP (5 marks)**
2. Implement **Client-Side Prediction** for the player controlled character **(10 marks)**
3. Implement **Reconciliation** for all packets received **(10 marks)**
4. Implement **Dead Reckoning** for non-player controlled characters **(10 marks)**

Gameplay (30 marks)
1. Player must be able to move around the game via updates from the server **(10 marks)**
2. Player must be able to shoot **(5 marks)**
3. Player must be able to die if shot **(5 marks)**
4. Player must be able to respawn by pressing a button **(10 marks)**

Assignment must be done in Git. Commit messages should be small and sweet. **(3 marks)**

Bonuses (40 marks)
Bonuses are only available if all other project requirements are completed properly.

1. Optimize your packets using techniques discussed in class. **(10 marks)**
2. Interpolate players positions. **(10 marks)**
3. Server renders the scene. **(10 marks)**
4. Allow the user to toggle different coloured copy of each object to show: **(10 marks)**
    a. Last Server Update position.
    b. Client Predicted Position of the player controlled character.
    c. Dead Reckoning Position of the server controlled objects.

Plagiarism:

- While you may freely “borrow” mine (or anyone other) code but you code should be “sufficiently” different from mine.
- In other words, you cannot simply use an existing game engine (or part of a game engine) to complete this assignment; it should be either completely new of **significantly modified**.

Grading Scheme

1. Late assignments will not be accepted.
2. You **MUST** include a ReadMe.txt to explain how to build and run your project. I will **NOT** accept a project *without* one.
3. If your code does not even compile, I will not mark it. Period. This will get you a mark of zero (0)
4. If your code does not build (i.e. linker error) and run (i.e. no crazy run-time crash that is unexpected), I may investigate this further, but only if there is some simple problem and/or very slight and/or very obvious (and easy to fix) configuration error.

Item	**Mark**
Protocol	7
GameServer	25
GameClient	35
Gameplay	30
Git	3
Bonuses:	
Optimization *(Position or Velocity to be awarded marks)*	10
Toggle Debug Objects View	10
Server Renders the scene	10
Interpolate Players	10
Total possible	100
Total possible (with bonuses)	140

Project Corrections

If any corrections or changes are necessary they will be posted to the course web site and you will be notified of any changes in class. It is your responsibility to check the site periodically for changes to the project. Additional resources relating to the project may also be posted.