ICNFinal Report

Group 3: b07901032, b07901037, b07901055

I. Code Survey

- Client side: under folder Assets
    - Game Content
        - Player_Interaction, Player_Prefab, Progress_Bar, Weapon

    These folders are mostly related to Unity and game content.
    - Connection_Script:
        - Client, ClientHandle, ClientSend, Packet, PlayerManager, GameManager, ThreadManager

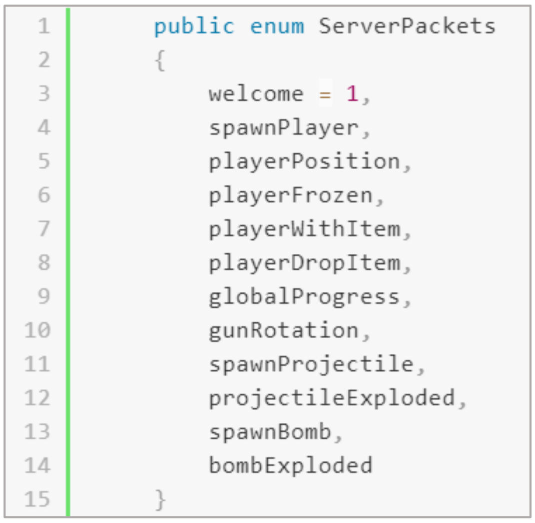
    These files are found in Library/Collab/Download/Assets, which should be a clone of tutorial code. These files are related to client handling and sending packets.
- Server side: under folder GameServer
    - Server/Handle/Send/Packet, Player/Projectile/Bomb, ThreadManager, Program(main)

    These files was cloned from tutorial code. However, since our game content was distinct from the game in tutorial, we have to do minor modification to make it work.

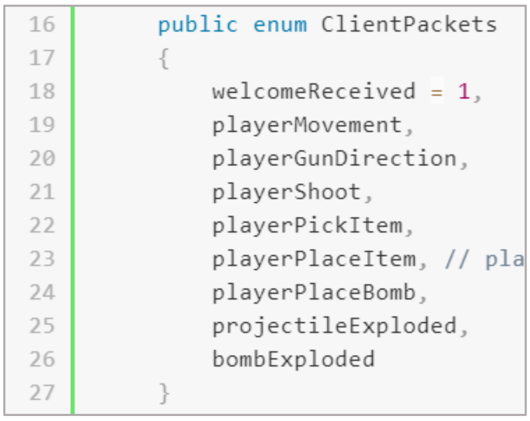
II. Tasks and Approaches: Server Side

- Coordinate what should be included packet based on events that will happened in the game.

Packet type:



```
        public enum ServerPackets
        {
            welcome = 1,
            spawnPlayer,
            playerPosition,
            playerFrozen,
            playerWithItem,
            playerDropItem,
            globalProgress,
            gunRotation,
            spawnProjectile,
            projectileExploded,
            spawnBomb,
            bombExploded
        }
```



```
        public enum ClientPackets
        {
            welcomeReceived = 1,
            playerMovement,
            playerGunDirection,
            playerShoot,
            playerPickItem,
            playerPlaceItem, // pla
            playerPlaceBomb,
            projectileExploded,
            bombExploded
        }
```

Events:

Packets to Send

Events	First Step ClientSend/ ServerHandler	Second Step ServerSend/ ClientHandler	Connection method
Game start	tcp connection request	welcome	tcp
Player spawned	welcomeReceived	spawnPlayer remember to sync	tcpToAll

Player move	playerMovement
	player.position(Vector3)
		playerPosition
		player.id(int)
		player.position(Vector3)
			udpToAll

Player gun rotation	playerGunDirection
	player.gunRotation(Quaternion)
		gunRotation
		player.id(int)
		player.gunRotation(Quaternion)
			udpToAll

Player shoot	playerShoot
	projectile.position(Vector3)
	projectile.rotation(Quaternion)
		spawnProjectile
		projectile.id(int)
		projectileOrigin(Vector3)
		projectileDirection(Quaternion)
			tcpToAll

Player place bomb	playerPlaceBomb
	bombOrigin(Vector3)
		spawnBomb
		bomb.id(int)
		bomb.position(Vector3)
			tcpToAll

Player with item	playerPickItem
	item((int)PROGRESS)
		playerWithItem
		player.id(int)
		item((int)PROGRESS)
			tcpToAll

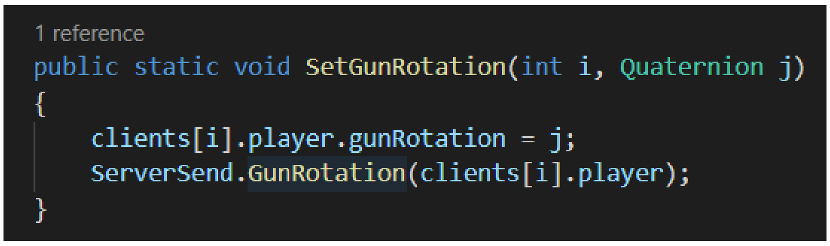
Player place down item at factory	playerPlaceItem	playerDropItem
		player.id(int)
		globalProgress
		PROGRESS(int*4)
			tcpToAll

Player got hit	projectileExploded
	projectile.id(int)
		projectileExploded
		projectile.id(int)
		playerFrozen
		player.id(int)
		playerDropItem
		player.id(int)
			tcpToAll

Player bot bombed	bombExploded
	bomb.id(int)
		projectileExploded
		bomb.id(int)
		playerFrozen
		player.id(int)
		playerDropItem
		player.id(int)
			tcpToAll

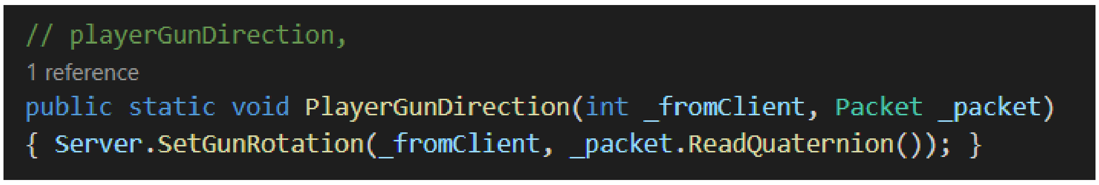
- Based on above information to create interfaces in Server.cs/ ServerHandle.cs/ServerSend.cs

For example, if we want to handle a packet about a client updating player's gun direction, we write a set function in Server.cs that will modify player's properties under client and broacast to other clients using interface in ServerSend.cs. After that, we called it from ServerHandle.cs to handle when there is relating packet incoming.



```
1 reference
public static void SetGunRotation(int i, Quaternion j)
{
    clients[i].player.gunRotation = j;
    ServerSend.GunRotation(clients[i].player);
}
```

^ In Server.cs



```
// playerGunDirection,
1 reference
public static void PlayerGunDirection(int _fromClient, Packet _packet)
{ Server.SetGunRotation(_fromClient, _packet.ReadQuaternion()); }
```

^ In ServerHandle.cs

- Modify tutorial code such as adding Projectile and Bomb objects to suit our game content.

III. Tasks and Approaches: Client Side

- Mostly dealing with Unity: Scenes, Prefabs, missing script, Transform, GameObject, Collider, etc.
- Based on Packet, create interfaces in Client.cs/ ClientHandle.cs/ClientSend.cs that connect packet with our game content. It has major difference from server side because updating infomation in Unity is quit frustrating.
- Divide players on client to local and non-local
- To synchronize timer in all client, we reset timer when new player spawned.

IV. Work Distribution

- 1032: server side code and others
  - Team management
  - Git/hackmd maintenance
  - Server/client packet design
  - Report
- 1037, 1055: client side code
  - Unity scene and prefab understanding and manipulation
  - Single-to-multiplayer game conversion
  - Packet-to-game-content conversion
  - Timer synchronization

V. Reference and Sites

- [Unity Networking Tutorial] (https://youtube.com/playlist?list=PLXkn83W0QkfnqsK8I0RAz5AbUxfg3bOQ5)
- [Github of Tutorial] (https://github.com/tom-weiland/tcp-udp-networking)
- [ICNFinal Github] (https://github.com/howardchou0302/ICNFinal.git)
- [ICNFinal Markdown] (https://hackmd.io/jBB5SW9aSLCRbKnop1MU4A)