네트워크 게임 프로그래밍 프로젝트 추진계획서

2025-10-26

2023184017 백지훈 / 2023184028 임성민 / 2023180007 김도윤

1. 애플리케이션 기획

기존 애플리케이션 기획

교과목: 컴퓨터 그래픽스

작업자: 임성민

프로젝트: OpenGL 을 이용한 3D 1 인칭 FPS 게임(발로란트) 모작

기존 상태: 현재 코드는 모든 로직이 100% 클라이언트에서 실행되는 싱글 플레이어 환경 / 플레이어의 입력, 맵과의 충돌, 적의 피격 및 사망 로직 등 게임 로직과 데이터가 클라이언트에 종속

네트워크 기능 기획

핵심 목표: 기존의 싱글 플레이어 FPS 환경을 TCP, UDP 기반의 실시간 멀티플레이어 데스매치 게임

네트워크 모델: 로그인 TCP 를 사용

실시간 게임 데이터(이동, 사격)는 UDP 를 사용.

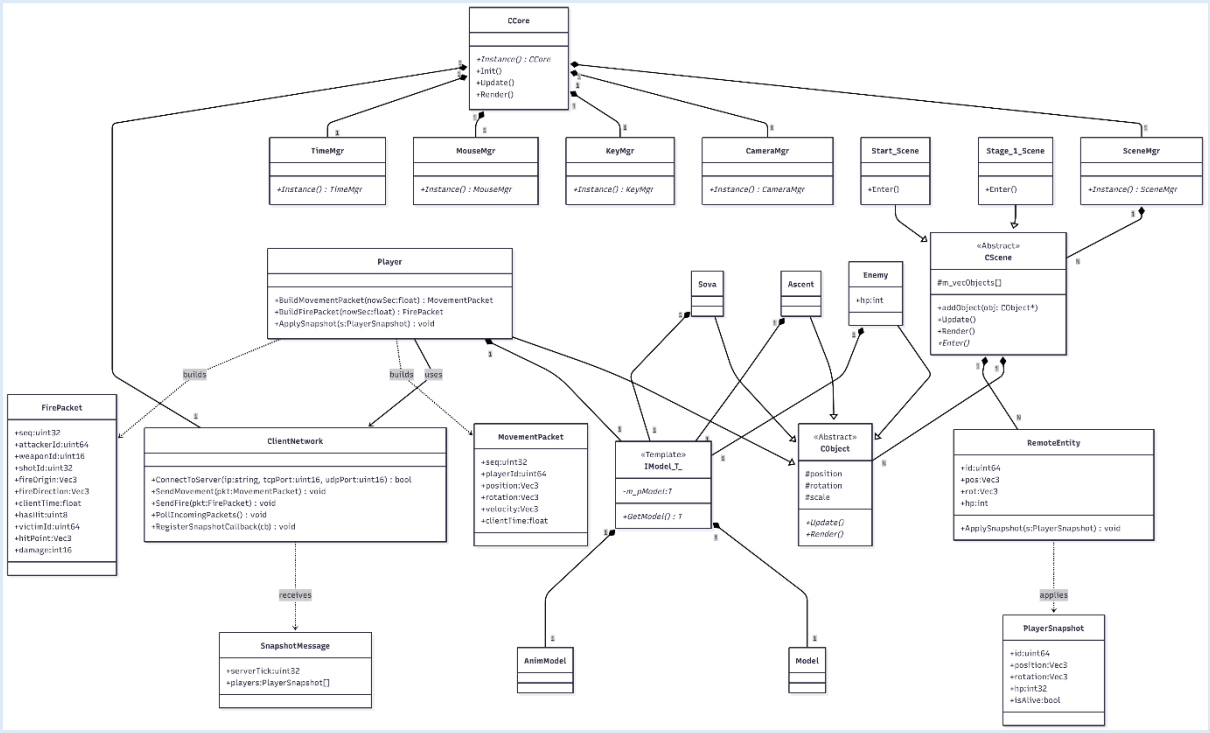
주요 기능

- 다중 클라이언트 접속
- 플레이어 정보 동기화
- 총기 클라이언트 판정 및 동기화

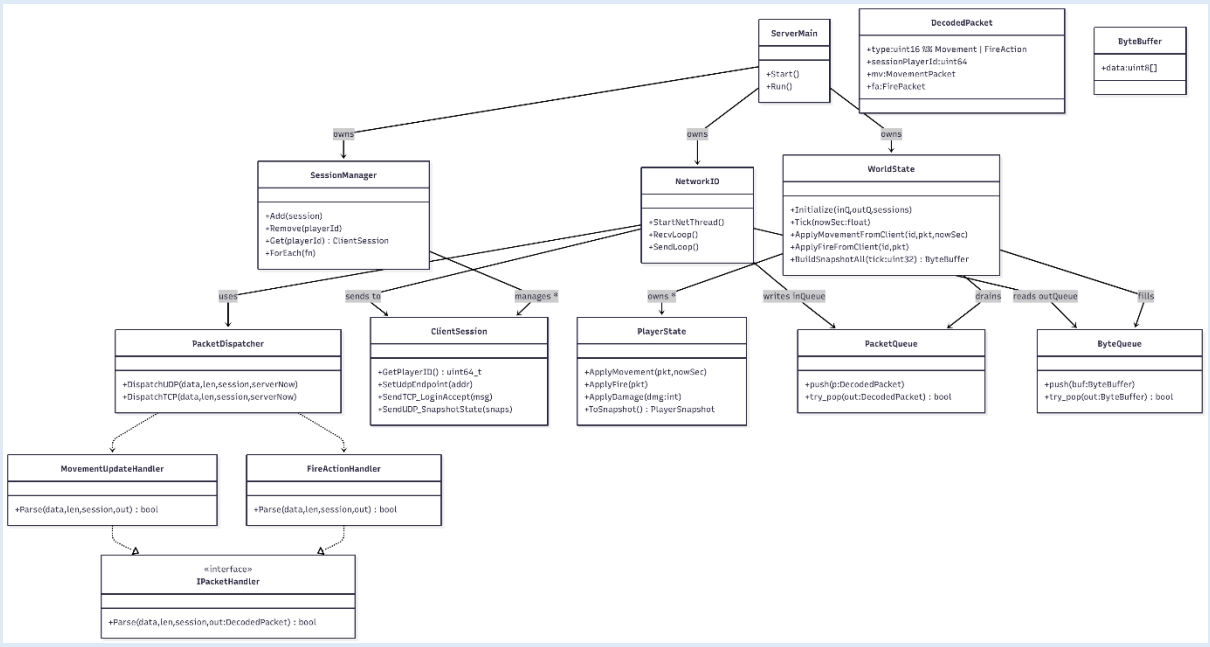
2. Hight-Level Design

플로우 차트

클라이언트



서버



클래스 개요

클라이언트

Player (기존 클래스 / 역할 변경)

- **역할 :** 로컬 이동/충돌/히트스캔을 **계산**해, 그 결과를 서버에 보고, 서버가 브로드캐스트한 상대 플레이어 상태를 반영
- **설명:** 이동 결과를 BuildMovementPacket()을 호출해 입력 패킷 생성
- 발사/명중 결과를 BuildFirePacket()을 호출해 패킷 생성
- 서버에서 S2C_SNAPSHOT_STATE 를 받으면 ApplySnapshot()을 호출 서버가 확정한 위치로 이동
- **주요 처리 패킷:** C2S_MOVEMENT_UPDATE, C2S_FIRE_ACTION, S2C_SNAPSHOT_STATE

ClientNetwork (신규)

- **역할:** 서버와의 **모든 통신(TCP/UDP)을 관리**
- **설명:** ConnectToServer()로 서버에 접속
- Player 가 생성한 패킷을 SendMovement(), SendFire(), SendHeartbeat(), PollIncomingPackets() 서버에 전송
- **주요 처리 패킷:** 모든 C2S, 모든 S2C

서버

NetworkIO (NetThread) (신규)

- **역할:** TCP/UDP 수신/송신 전담 스레드
- **설명:** 수신 바이트를 PacketDispatcher 로 파싱해 큐에 저장
- WorldState 가 만든 플레이어들의 상태 값를 큐에서 꺼내 클라이언트로 UDP 전송
- **주요 처리 패킷:** C2S_LOGIN_REQUEST, S2C_LOGIN_ACCEPT, C2S_MOVEMENT_UPDATE, C2S_FIRE_ACTION, S2C_SNAPSHOT_STATE

WorldState (GameThread) (신규)

- **역할:** 게임 상태 저장/갱신

- **설명:** 틱마다 큐의 데이터를 전부 꺼내 Movement, FireAction 을 PlayerState 에 반영
- 모든 플레이어 상태 값을 큐에 넣음
- **주요 처리 패킷:** C2S_MOVEMENT_UPDATE, C2S_FIRE_ACTION, S2C_SNAPSHOT_STATE

ClientSession (신규)

- **역할:** 클라이언트 1 명과의 **1:1 통신(TCP/~~UDP~~)을 담당**.
- **설명:** 클라이언트의 접속(accept) 시 생성되며, 해당 클라이언트의 소켓 정보와 고유 ID 를 관리합니다.
- ~~수신된 패킷을 RoomManager 로 전달하고, RoomManager 가 보낸 패킷을 해당 클라이언트에게 전송(Send)하는 파이프 역할을 합니다.~~
- **주요 처리 패킷:** C2S_LOGIN_REQUEST, C2S_JOIN_ROOM_REQUEST (처리) / S2C_LOGIN_ACCEPT (전송).

3. low-level design

Client

```
class Player : public CObject
MovementPacket BuildMovementPacket(float nowTimeSec) const;
FirePacket     BuildFirePacket(float nowTimeSec) const;
void ApplySnapshot(const PlayerSnapshot& snap);


class ClientNetwork
bool ConnectToServer(const std::string& ip, uint16_t tcpPort, uint16_t udpPort);
void SendMovement(const MovementPacket& pkt);
void SendFire(const FirePacket& pkt);
void PollIncomingPackets();
const std::vector<PlayerSnapshot>& GetLastSnapshots() const;
```

Server

```
class ClientSession
PlayerID GetPlayerID() const;
void SendUDP_SnapshotState(const std::vector<PlayerSnapshot>& snaps);

class PlayerState
void ApplyMovementFromClient(const MovementPacket& pkt, float serverNow);
void ApplyFireFromClient(const FirePacket& pkt);
void ApplyDamage(int dmg);
PlayerSnapshot ToSnapshot() const;



class IPacketHandler
virtual ~IPacketHandler() = default;
virtual bool Parse(const uint8_t* data, size_t len,
                   ClientSession* session,
                   DecodedPacket& out) = 0;

class MovementUpdateHandler : public IPacketHandler
bool Parse(const uint8_t* data, size_t len,
           ClientSession* session,
           DecodedPacket& out) override;

class FireActionHandler : public IPacketHandler
bool Parse(const uint8_t* data, size_t len,
           ClientSession* session,
           DecodedPacket& out) override;

class PacketDispatcher
void DispatchUDP(const uint8_t* data, size_t len,
                 ClientSession* session, float serverNow);
```

```
class WorldState
void Initialize(PacketQueue* inQ, ByteQueue* outQ, SessionManager* sessions);
    void Tick(float nowSec);


    void ApplyMovementFromClient(PlayerID pid, const MovementPacket& pkt,
float nowSec);
    void ApplyFireFromClient    (PlayerID pid, const FirePacket& pkt);



class ClientSession
{
public:
    ClientSession(SOCKET tcpSocket, SessionManager* manager, PacketQueue*
worldInQueue);
    ~ClientSession();
    void StartThread();
    void SetLoginInfo(PlayerID id, const sockaddr_in& udpAddr);


   PlayerID GetPlayerID() const { return m_PlayerID; }
    sockaddr_in GetUdpAddress() const { return m_UDPEndPoint; }


private:
    static DWORD WINAPI ThreadWrapper(LPVOID lpParam);
   void ReceiveLoop();
   void HandleLogin(const C2S_LoginRequest& pkt);
    void HandleDisconnect();


private:
    SOCKET m_TCPSocket;
    HANDLE m_hThread;
    PlayerID m_PlayerID;
    sockaddr_in m_UDPEndPoint;
```

```
    SessionManager* m_pSessionManager;
    PacketQueue* m_pWorldInputQueue;
};

class SessionManager
{
public:
    SessionManager();
    ~SessionManager();

    void Initialize(PacketQueue* worldInQueue);

    void OnSessionLoggedIn(ClientSession* session, const sockaddr_in& tcpAddr,
USHORT clientUdpPort);
    void OnSessionDisconnected(PlayerID pid);
    std::vector<ClientSession*> GetAllSessions();
    PlayerID FindPlayerIDByUdpAddress(const sockaddr_in& addr);

private:
    CRITICAL_SECTION m_Lock;
    std::map<PlayerID, ClientSession*> m_SessionsByPID;
    PacketQueue* m_pWorldInputQueue;
    PlayerID m_NextPlayerID;
};

class NetworkIO
```

```
{
public:
    NetworkIO();
    ~NetworkIO();

    bool Initialize(USHORT port, PacketQueue* worldIn, ByteQueue* netOut,
SessionManager* manager);
    void Run();

private:
    void HandleTCPAccept();
    void HandleUDPReceive();
    void HandleBroadcast();

private:
    SOCKET m_TCPListenSocket;
    SOCKET m_UDPGameSocket;
    bool m_IsRunning;

    SessionManager* m_pSessionManager;
    PacketQueue* m_pWorldInputQueue;
    ByteQueue* m_pNetworkOutputQueue;
};
```

4. 네트워크 패킷 목록

구분	패킷 이름	설명	주요 필드	프로토콜
로그인	C2S_LOGIN_REQUEST	클라이언트가 서버 접속 요청 및 UDP 포트 정보 전송	playerName : string clientUdpPort : uint16	TCP
	S2C_LOGIN_ACCEPT	서버가 로그인 성공 응답 및 PlayerID 부여	playerId : uint64	TCP
실시간 전투	C2S_MOVEMENT_UPDATE	현재 위치·회전·속도 보고 (클라 계산 결과)	msgSeq : uint32 playerId : uint64 position : Vec3 rotation : Vec3 velocity : Vec3 clientTime : float	UDP
	C2S_FIRE_ACTION	사격 발생 보고	msgSeq : uint32 playerId : uint64 fireOrigin : Vec3 fireDirection : Vec3 clientTime : float	UDP
	S2C_SNAPSHOT_STATE	서버가 현재 룸 내 모든 플레이어 상태 브로드캐스트	serverTick : uint32 players[] : PlayerSnapshot	UDP

5. 애플리케이션 기획

통합 개발 환경 (IDE):

Visual Studio 2022′

통합 개발 환경 (IDE):

- Visual Studio 2022

사용 언어:

- **C++ (C++17 표준 사용)**

핵심 라이브러리:

- 네트워크: WinSock2 (Windows Sockets API)
- 그래픽스: OpenGL, GLEW, FreeGLUT
- 수학: GLM
- 모델 로딩: Assimp

버전 관리 시스템 (VCS):

- Git
- GitHub 리포지토리 주소: https://github.com/limdev59/VALORANT-COPY.git

6. 개발 일정

백지훈	일	월	화	수	목	금
1 주차	11/2 ~~GetPlayerID()~~	11/3	11/4 SendMovement() GetPlayerID()	11/5	11/6	11/7
2 주차	11/9 ~~SendFire()~~	11/10	11/11 ~~MovementUpdateHandler::Parse()~~	11/12 SendFire() MovementUpdateHandler::Parse()	11/13	11/14 NetworkI
3 주차	11/16	11/17	11/18 ~~SendUDP_SnapshotState()~~ ClientSession	11/19	11/20	11/21
4 주차	11/23 ~~BuildSnapshotAll()~~ SessionManager	11/24	11/25 ~~WorldState::Tick()~~ SendUDP_SnapshotState()	11/26	11/27	11/28
5 주차	11/30 ~~SendTCP_LoginAccept()~~	11/31	12/1	12/2	12/3	12/4

김도윤	일	월	화	수	목	금	토
1 주차	11/2 ~~WorldState~~	11/3	11/4	11/5 ToSnapshot() WorldState	11/6	11/7	11/8
2 주차	11/9 DispatchUDP()	11/10	11/11	11/12 ApplyMovementFromClient()	11/13	11/14	11/15
3 주차	11/16 ApplyFireFromClient()	11/17	11/18	11/19 BuildSnapshotAll()	11/20	11/21	11/22
4 주차	11/23 SendToBoth() BuildSnapshotAll()	11/24	11/25	11/26 ApplyFireFromClient()	11/27	11/28	11/29
5 주차	11/30	11/31	12/1	12/2	12/3	12/4	12/5

	~~SendTCP_JoinRoomAck()~~			WorldState::Tick()			

임성민	일	월	화	수	목	금	토
1 주차	11/2	11/3 ~~ConnectToServer()~~	11/4 ConnectToServer()	11/5 ~~BuildMovementPacket()~~	11/6	11/7	11/8
2 주차	11/9	11/10 BuildFirePacket()	11/11	11/12 FireActionHandler::Parse()	11/13	11/14	11/15
3 주차	11/16	11/17 PollIncomingPackets()	11/18	11/19 ApplySnapshot()	11/20	11/21	11/22
4 주차	11/23	11/24 GetLastSnapshots()	11/25	11/26 SendHeartbeat()	11/27	11/28	11/29
5 주차	11/30	11/31 Player UI	12/1	12/2	12/3	12/4	12/5