# 네트워크 게임 프로그래밍

# 추진계획서 V3

2020180030 이현수

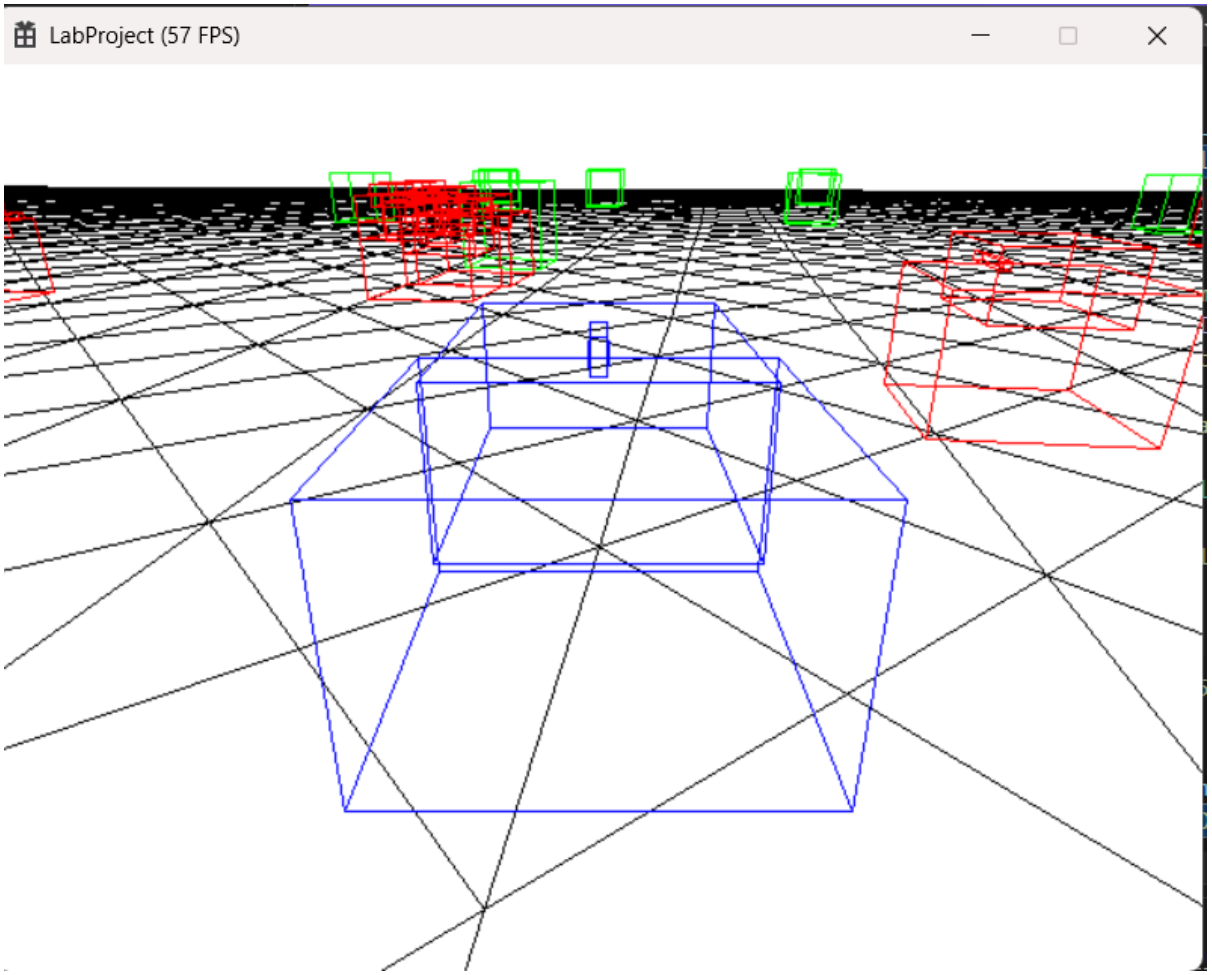
2022182047 신동호

# 목차

**애플리케이션 기획**

3D 게임 프로그래밍 수업 때 이현수가 만든 과제 1 게임 사용

플레이어는 장애물들을 부수고 적 탱크를 타격하여 죽인다.




조작키 : wasd

발사 : spacebar

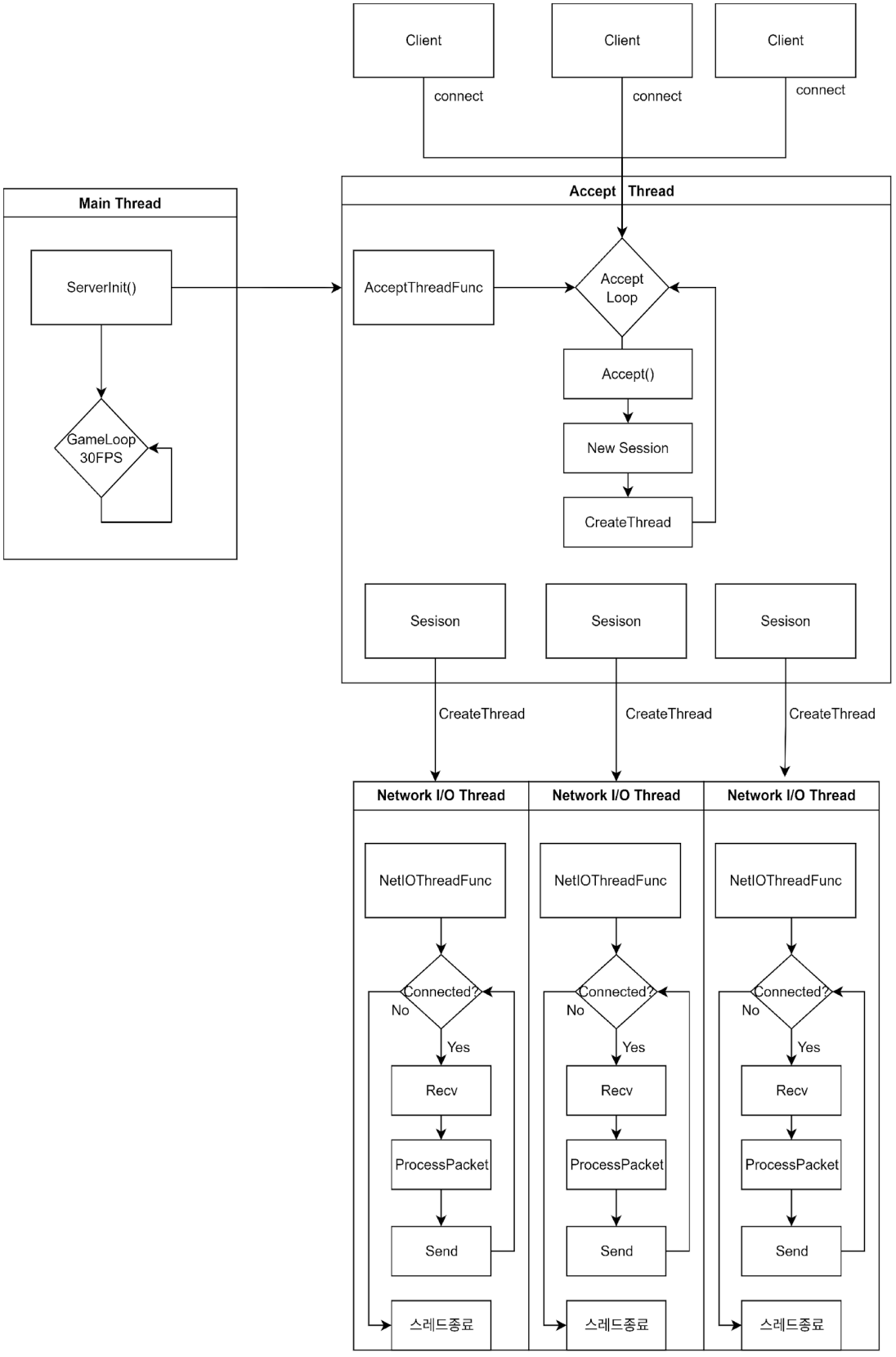
회전 : 좌클릭

핵심 구현

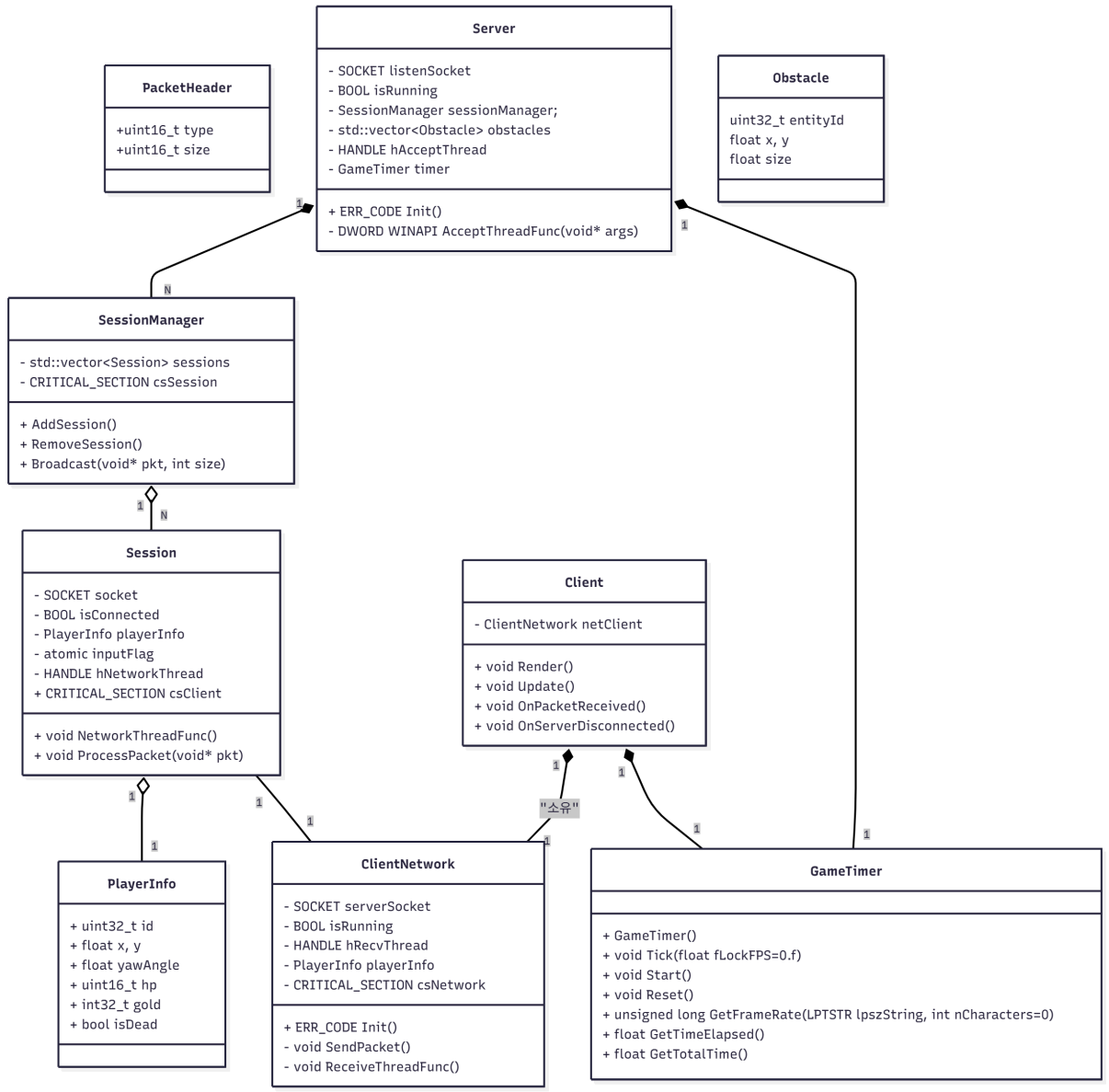
- 클라이언트간 이동 동기화
- 클라이언트간 타격/충돌 처리
- 죽으면 1초 후 부활
- 상단 좌측에 HP, KILL / death 텍스트로 표시
- 죽인 id 표시
- 키를 입력한 간단한 상점 구현
- 오른쪽 상단 킬많이한 순으로 LeaderBoard 표시

## High-level 디자인

플로우차트



클래스다이어그램


Server
- SOCKET listenSocket
- BOOL isRunning
- SessionManager sessionManager;
- std::vector<Obstacle> obstacles
- HANDLE hAcceptThread
- GameTimer timer
+ ERR_CODE Init()
- DWORD WINAPI AcceptThreadFunc(void* args)
PacketHeader
+uint16_t type
+uint16_t size
Obstacle
uint32_t entityId
float x, y
float size
SessionManager
- std::vector<Session> sessions
- CRITICAL_SECTION csSession
+ AddSession()
+ RemoveSession()
+ Broadcast(void* pkt, int size)
Session
- SOCKET socket
- BOOL isConnected
- PlayerInfo playerInfo
- atomic inputFlag
- HANDLE hNetworkThread
+ CRITICAL_SECTION csClient
+ void NetworkThreadFunc()
+ void ProcessPacket(void* pkt)
Client
- ClientNetwork netClient
+ void Render()
+ void Update()
+ void OnPacketReceived()
+ void OnServerDisconnected()
"소유"
PlayerInfo
+ uint32_t id
+ float x, y
+ float yawAngle
+ uint16_t hp
+ int32_t gold
+ bool isDead
ClientNetwork
- SOCKET serverSocket
- BOOL isRunning
- HANDLE hRecvThread
- PlayerInfo playerInfo
- CRITICAL_SECTION csNetwork
+ ERR_CODE Init()
- void SendPacket()
- void ReceiveThreadFunc()
GameTimer
+ GameTimer()
+ void Tick(float fLockFPS=0.f)
+ void Start()
+ void Reset()
+ unsigned long GetFrameRate(LPTSTR lpszString, int nCharacters=0)
+ float GetTimeElapsed()
+ float GetTotalTime()

Low-level 디자인

```
// Error.h
enum class ERR_CODE : short
{
	NONE = 0,

	// 서버 오류
	WSASTARTUP_FAIL = 11,
	SOCK_CREATE_FAIL = 12,
	SOCK_BIND_FAIL = 13,
	SOCK_LISTEN_FAIL = 14,

	ACCEPT_FAIL = 21,

	SEND_FAIL = 26,

	RECV_FAIL = 31,

};

// Obstacle.h
struct Obstacle {
	uint32_t entityId;
	float x, y;
	float size;
};

// PlayerInfo.h
struct PlayerInfo {
	uint32_t id;
	float x, y;
	float yawAngle;
	uint16_t hp;
	int32_t gold;
	bool isDead;
};

// Constant.h
constexpr int SERVER_PORT = 9000;
```

```cpp
// Session.h
class Session {
public:
	DWORD WINAPI NetworkThreadFunc(void* args);

private:
	SOCKET m_socket;
	std::atomic<bool> m_isConnected;
	PlayerInfo m_playerInfo;
	std::atomic<uint8_t> m_inputFlag;

	HANDLE m_NetworkThread;
	CRITICAL_SECTION m_csSession;
};

// SessionManager.h
#include "Session.h"

class SessionManager {
public:
	AddSession();
	RemoveSession();
	Broadcast(void* pkt, int size);

private:
	std::vector<Session> csSession;
	CRITICAL_SECTION csSession;

};
```

```cpp
//Server.h
#include "SessionManager.h"
#include "Timer.h"

class Server {
public:

	Server() : m_isRunning(true)
	{}

	ERR_CODE Init();
	DWORD WINAPI AcceptThreadFunc(void* args);
	// TODO : 지형 정보(장애물, 지형크기) 초기화 함수
	// TODO : ERR_CODE 오류 처리 메서드
private:
	SOCKET m_listenSock;
	std::atomic<bool> m_isRunning;
	SessionManager m_sessionManager;
	std::vector<Obstacle> m_obstacles;
	HANDLE m_acceptThread;
	GameTimer timer;
};
```

```
// Timer.h
#pragma once
//-----------------------------------------------------------------------
-------
// File: CGameTimer.h
//-----------------------------------------------------------------------
-------

const ULONG MAX_SAMPLE_COUNT = 50; // Maximum frame time sample count

class GameTimer
{
public:
      GameTimer();
      virtual ~GameTimer();

      void Tick(float fLockFPS = 0.0f);
      void Start();
      void Stop();
      void Reset();

    unsigned long GetFrameRate(LPTSTR lpszString = NULL, int
nCharacters=0);
    float GetTimeElapsed();
      float GetTotalTime();

private:
      double                                          m_fTimeScale;

      float                                       m_fTimeElapsed;

      __int64
m_nBasePerformanceCounter;
      __int64
m_nPausedPerformanceCounter;
      __int64
m_nStopPerformanceCounter;
      __int64
m_nCurrentPerformanceCounter;
    __int64
m_nLastPerformanceCounter;

      __int64
m_PerformanceFrequencyPerSec;
```

```
    float
m_fFrameTime[MAX_SAMPLE_COUNT];
    ULONG                                   m_nSampleCount;

    unsigned long                       m_nCurrentFrameRate;

      unsigned long                         m_FramePerSecond;

      float                                 m_fFPSTimeElapsed;


      bool                                  m_bStopped;
};
```

패킷정의

```
enum PacketType : uint16_t {
    CS_INPUT            = 101,
    CS_BUY_HP           = 102,
    CS_BUY_ATK          = 103,
    CS_BUY_HEAL         = 104,

    SC_SNAPSHOT         = 201,
    SC_ENTITY_SPAWN     = 202,
    SC_KILL_EVENT       = 203,
    SC_DEAD_EVENT       = 204,
    SC_RESPONSE_EVENT   = 205,
}

struct PacketHeader {
    uint16_t type;
    uint16_t size;
};

struct InputPacket {
            uint32_t id;
            uint32_t tick;
    uint8_t inputFlag;
    int16_t yawAngle;
};

struct SnapshotPacket {
            uint32_t tick;
            uint16_t count;
            PlayerInfo players[];
}
struct EntitySpawnPacket {
            uint16_t count;
            Obstacle obstacles[];
}

struct KillEventPacket {
            int32_t killer_id;
            int32_t killed_id;
}

struct DeadEventPacket {
            int32_t tick;
}
```

```
struct ResponeEventPacket {
            float x, y;
}
```

**팀원 별 역할분담**

이현수 – Server Send/Recv, 이동 동기화, 사망, 부활 처리, Session Pool 구조 변경

신동호 – 장애물 위치, 타격, 입력 동기화, 게임 콘텐츠

**개발환경**

Visual Studio 2022 , C++, Winsock2 API, GitHub, Notion

**개발일정**

**이현수**

| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| | | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5<br>Server::Init() | 6 | 7<br>Recv, ProcessPacket, Send | 8<br>Server Game Loop |
| 9 | 10 | 11<br>클라이언트 지형 동기화 | 12 | 13 | 14<br>이동 동기화 (Tick 보정) | 15 |
| 16 | 17 | 18<br>여러 플레이어 동기화 SnapshotPacket | 19 | 20 | 21<br>사망/부활 Dead/Respone Packet | 22 |
| 23 | 24 | 25<br>ServerConfig.ini 추가 | 26 | 27 | 28<br>SessionPool 로 변경 | 29 |
| 30 | 1 | 2 | 3 | 4 | 5<br>DeadLine | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |

**개발일정**

신동호

| 일 | 월 | 화 | 수 | 목 | 금 | 토 |
|---|---|---|---|---|---|---|
| | | 28 | 29 | 30 | 31 | 1 |
| 2 | 3 | 4 | 5 | 6<br>AcceptThread (New Session) | 7<br>Client Network( Connect, Send, Recv) | 8 |
| 9 | 10 | 11<br>Client SendEntitySpawnPacket / 장애물 위치 동기화 | 12 | 13<br>입력 패킷 InputPacket | 14 | 15 |
| 16 | 17 | 18<br>타격 동기화 | 19 | 20 | 21<br>BUY_HP, BUY_ATK, BUY_HEAL | 22 |
| 23 | 24 | 25<br>KillEvent (UDP 사용) | 26 | 27 | 28<br>LeaderBoard | 29 |
| 30 | 1 | 2 | 3 | 4 | 5<br>Dead Line | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |