네트워크 게임 프로그래밍 추진계획서

네트워크게임프로그래밍

2018180009 김시인

2018182034 전수민

2019182046 김미령

목차

1. 게임 기획

* 게임 개요(게임 이름, 장르, 소개 등)
* 게임 플레이
* 오브젝트나 UI소개, 조작법

1. 개발 환경
2. High Level Design

* 전체적인 게임 플로우 다이어그램
* 서버 다이어그램
* 클라 다이어그램
* 서버 클라 동작 다이어그램

1. Low Level Design

* 객체 설명(멤버변수, 멤버변수), 서버 함수 설명, 클라이언트 함수 설명

1. PM

* 팀원 역할분담
* 전체 개발일정

#1. 게임 기획



(게임 실행 화면)

게임 이름: 3D 헬기 대전게임 (3D게임 프로그래밍2, 전수민)

게임 소개: 다른 플레이어와 미사일을 쏴서 hp를 깎아 경쟁하는 게임.

최대 3명까지 동시에 접속 가능하고, 플레이어가 미사일에 맞아 HP가 모두 닳았을 경우 플레이어 게임 종료

조작 키: WASD(전진, 후진, 좌우 이동), P(미사일 발사), QE(좌우 회전), 34(상하 회전), 12(상승, 하강)

오브젝트: 지형, 전투기(플레이어), 미사일

UI: HP바

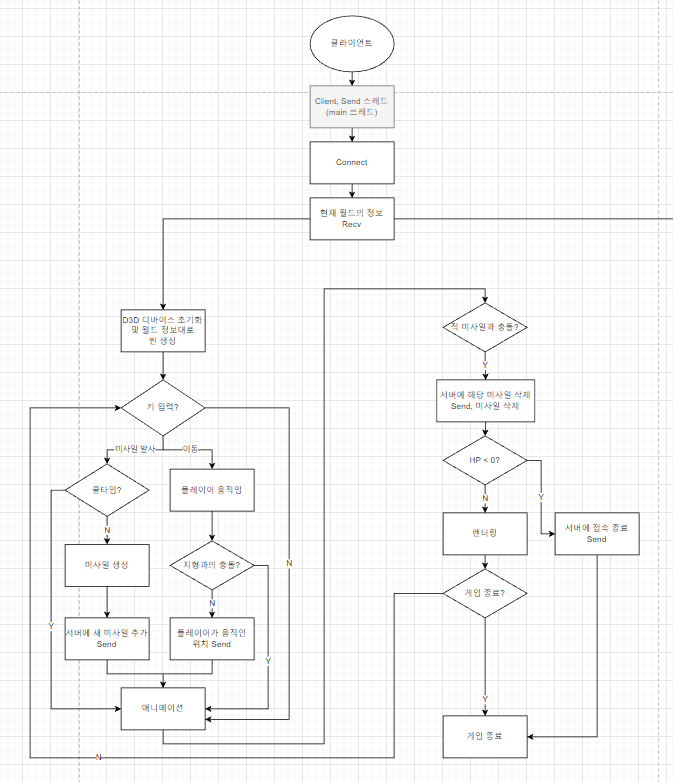
#2. 개발 환경

*   화살이(가) 표시된 사진

  자동 생성된 설명
* Visual studio C++ - TCP / IP - DirectX 12

#3. High Level Design

[클라이언트의 Flow chart]



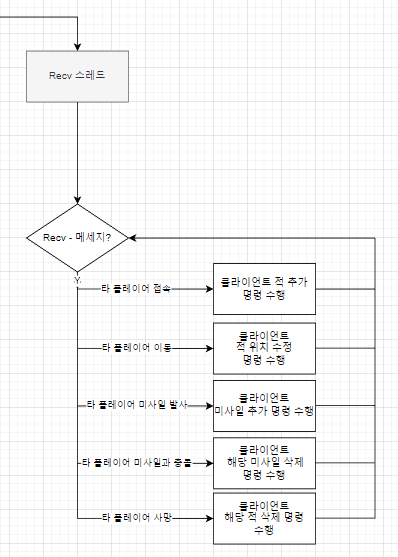
클라이언트는 크게 게임진행, Send를 동시에 하는 쓰레드와 Recv하는 쓰레드 두개로 구성.

Send의 경우 non-blocking 이기 때문에 클라이언트가 진행되는 메인 쓰레드와 같이 두고, Recv와 Send의 순서가 정해져 있지 않기 때문에 실시간 입출력을 구현하기 위해서 Recv 쓰레드를 따로 분리했다. 클라이언트가 처음 게임에 접속 시 2초의 무적 시간을 갖게 된다.

\* Client, Send 쓰레드 ( main 쓰레드 ) 의 기능

1. 서버에 접속한다.
2. 현재 월드 정보를 Recv해서 반영해준다.
3. 새로 접속한 플레이어는 2초간 무적으로 보인다.
4. Recv 전용 쓰레드를 하나 생성한다.
5. 메시지가 왔는지 지속적으로 확인하고 메시지가 있을 경우 해당 패킷을 읽어 임계영역을 설정 후 클라이언트에 플레이어 혹은 미사일 정보를 수정하고 임계영역을 해제한다.
6. 서버에 새 미사일을 추가했다고 Send를 한다.
7. 서버에 플레이어가 움직였다고 Send를 한다.
8. 임계영역 설정 후 플레이어와 미사일 간의 충돌체크를 프레임마다 하고, 충돌 시 HP를 깎고 다시 푼다.
9. 애니메이션과 렌더링을 각각 임계영역 설정 후 진행한 다음 푼다.
10. 서버에 충돌한 미사일을 삭제하라고 Send를 한다.
11. 체력이 없을 경우 서버에 플레이어가 죽었다고 Send를 하고 클라이언트를 종료한다.

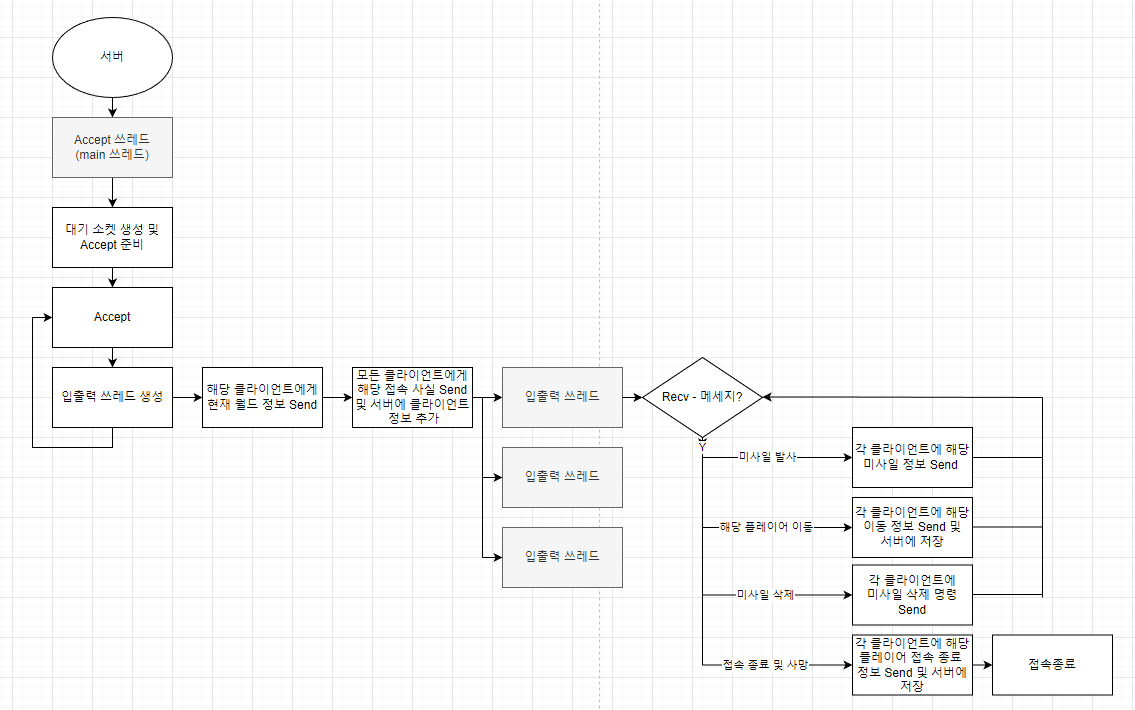
플레이어 정보들과 미사일 정보는 독립적으로 수행되어도 상관없기 때문에 각각 따로 임계영역을 설정해준다.



\*Recv 쓰레드

1. 메시지가 왔는지 지속적으로 확인하고 메시지가 있을 경우 해당 패킷을 읽어 임계영역을 설정 후 클라이언트에 플레이어 혹은 미사일 정보를 수정하고 임계영역을 해제한다.

[서버의 Flow chart]



서버는 새 클라이언트의 접속을 받는 Accept(메인 쓰레드), 접속을 받을 때 마다 생성되는 입출력 쓰레드로 나누어진다. 서버 내에는 미사일 정보는 따로 없고, 각 클라이언트의 정보를 갖고 있다.

서버의 동기화

새 플레이어가 접속해 새 입출력 스레드가 생성되고나서 월드정보를 보내줄 때, 해당 플레이어의 이동이나 접속 종료 및 사망시에 임계영역을 걸고 clients내 정보를 읽거나 쓴 후 바로 해제해 준다. 미사일의 경우 따로 저장해두는 것이 아닌, 패킷을 받자마자 바로 모든 다른 플레이어에게 전달해주지만, clients 내 행렬정보를 읽기 때문에 마찬가지로 임계영역을 설정 해준다.

\* Accept 쓰레드

1. 접속을 받을 준비를 한 후 계속해서 accept를 받고 새 입출력 스레드를 생성한다.

\* 입출력 쓰레드

1. 이미 서버에 접속해 있는 모든 클라이언트에게 해당 접속 정보를 Send한다.
2. 해당 클라이언트에게 현재 월드 정보를 Send하고 서버에 그 정보를 추가한다.
3. 메시지가 왔는지 계속해서 확인하고, 받은 메시지에 따라 모든 클라이언트에게 해당 정보를 Send하고, 임계영역을 설정하고 서버에 미사일이나 플레이어의 정보를 수정 후 임계영역을 해제한다.

#4. Low Level Design

[패킷 구성]

텍스트이(가) 표시된 사진

자동 생성된 설명

\* 클라이언트->서버

서버에서 클라이언트와 connect시 클라이언트 식별id를 key값, 소켓이 포함된 SESSION클래스를 value값으로 하는 unordered\_map<USHORT, SESSION> 에 추가하고, 새로 만들어지는 입출력 쓰레드에게 고유한 클라이언트 id를 인자 값으로 준다. 후에 패킷이 도착했을 때 그 쓰레드 내에 클라이언트 id값이 있으므로 그 값으로 보낸 클라이언트의 세션 객체를 찾는다.

- CS\_MOVE\_PLAYER: 플레이어의 이동, 회전한 정보를 보내기 위한 패킷

- CS\_ADD\_MISSILE: 플레이어가 미사일 발사한 사실을 알리기 위한 패킷. 미사일의 위치와 방향은 해당 클라이언트의 worldTransform값으로 결정한다.

- CS\_REMOVE\_MISSILE: 적 미사일이 플레이어와 충돌했을 때 삭제될 미사일id를 보내기 위한 패킷

- CS\_REMOVE\_PLAYER: 플레이어가 죽었을 때 그 사실을 알리기 위한 패킷

\*서버->클라이언트

텍스트이(가) 표시된 사진

자동 생성된 설명

- SC\_WORLD\_DATA: 처음 클라이언트가 접속했을 때, 그 클라이언트에게 id를 부여하고 이미 접속해 있는 플레이어의 정보를 보내주는 패킷. 이 패킷 이후에 그 수에 맞게 아래의 패킷을 추가로 보내준다

- SC\_ADD\_PLAYER: 다른 플레이어의 정보를 보내주기 위한 패킷

- SC\_ADD\_MISSILE: 나를 포함한 다른 플레이어가 발사한 미사일의 정보를 보내주기 위한 패킷. 내가 발사한 미사일이라도 서버에서 다시 이 패킷을 받는다.

- SC\_MOVE\_PLAYER: 해당 플레이어가 이동했을 때 변한 정보를 보내주기 위한 패킷

- SC\_REMOVE\_MISSILE: 어떤 플레이어가 미사일에 충돌했다는 패킷을 보내왔을 때 다른 플레이어들에게 이 사실을 알려주기 위한 패킷

- SC\_REMOVE\_PLAYER: 해당 플레이어가 사망하거나 접속 종료했을 때 이 사실을 알려주기 위한 패킷

[클라이언트 기능별 함수 및 변수]

\* 사용되는 전역 변수:

-SOCKET serverSock; - 서버와 통신하기 위한 소켓

CRITICAL\_SECTION missileCS; - 스레드간 미사일 정보 등을 동기화 해주기위한 임계영역

CRITICAL\_SECTION playerCS; - 스레드간 플레이어 정보 등을 동기화 해주기위한 임계영역

\* 사용되는 함수 (써 있지 않은 함수들은 이미 구현이 된 함수이다.)

1. 서버에 접속한다.

void ConnectToServer();

윈속 초기화, 소켓에 서버 주소, 포트 번호를 입력, connect를 한번에 하는 함수

1. 현재 월드 정보를 Recv해서 반영해준다.

void GameFramework::RecvWorldData();

임시로 버퍼를 생성해 서버 소켓으로 SC\_WORLD\_DATA 패킷을 받아 그 내용을 생성된 게임 월드에 적용한다.

1. 새로 접속한 플레이어는 2초간 무적이 된다.

Player클래스, 쉐이더 수정

처음 접속시 바로 죽지 않도록 플레이어를 미사일 충돌을 받지 않는 상태로 하고 다른 색으로 표시해준다.

1. Recv 전용 쓰레드를 하나 생성한다.

CreateThread() 사용

Recv전용 쓰레드를 만들고 반복문을 돌며 아래에 있는 기능을 계속해서 수행한다.

1. 메시지가 왔는지 지속적으로 확인하고 메시지가 있을 경우 해당 패킷을 읽어 임계영역을 설정 후 클라이언트에 플레이어 혹은 미사일 정보를 수정하고 임계영역을 해제한다.

void Scene::AddMissile(); void Scene::AddEnemy(); void Scene::EnemyMove();

void Scene::RemoveMissile(); void RemovePlayer();

1. 서버에 새 미사일을 추가했다고 Send를 한다.

int SendNewMissile();

서버에게 패킷의 타입만 보내주고, 서버에서는 이 타입을 보낸 클라이언트의 id를 찾아 해당 월드 행렬로 미사일의 방향, 위치를 알아낸다. Send~~()함수들은 send()가 반환하는 값을 그대로 반환한다.

1. 서버에 플레이어가 움직였다고 Send를 한다.

int SendPlayerMove();

플레이어가 움직이고 난 후 position, rotation값을 보내준다.

1. 임계영역 설정 후 플레이어와 미사일 간의 충돌체크를 프레임마다 하고, 충돌 시 HP를 깎고 다시 푼다.

int CheckCollideWithMissile();

충돌체크 후 충돌하지 않았다면 0을, 충돌하였다면 플레이어의 HP를 감소시키고 해당 미사일 id를 반환한다

1. 애니메이션과 렌더링을 각각 임계영역 설정 후 진행한 다음 푼다.

.Animate(), Render() 수정

미사일에 대한 Animate시 임계영역을 걸고 실행한다.

1. 서버에 충돌한 미사일을 삭제하라고 Send를 한다.

int SendMissileRemove(UINT \_missileId);

플레이어와 충돌해 사라져야 할 미사일의 id를 인자로 받아 해당 정보를 서버에게 보내준다.

1. 체력이 없을 경우 서버에 플레이어가 죽었다고 Send를 하고 클라이언트를 종료한다.

int SendPlayerRemove();

클라이언트를 삭제하라는 명령을 서버에게 보내고 해당 클라이언트를 종료한다.

[서버 기능별 함수 및 변수]

// worldTransform대신 pos, rotate, 미사일 클래스 삭제 수정

텍스트이(가) 표시된 사진

자동 생성된 설명

\* 사용되는 전역 변수 :

SESSION – ~~해당 클라이언트의 id값과 이동,회전 정보를 갖고 있는 월드 행렬을 담는 클래스~~

해당 클라이언트의 id값과 이동,회전 정보를 담는 클래스

사용되는 전역 변수

unordered\_map<USHORT, SESSION> clients; - 클라이언트의 정보를 cid값으로 찾기 위한 컨테이너

int cid, mid; - 클라이언트나 미사일이 추가될 때마다 부여 받을 id. 추가될 때 1씩 증가한다.

CRITICAL\_SECTION cs; - 입출력 스레드간 타 공유자원 (전역변수)를 동기화 해주기위한 임계영역

\*사용되는 함수 (써 있지 않은 함수들은 이미 구현이 된 함수이다.)

1. 접속을 받을 준비를 한 후 계속해서 accept를 받고 새 입출력 스레드를 생성한다.

void AcceptClient();

윈속 초기화, 대기 소켓 생성 및 bind와 listen등 초기 설정을 해주고 while문을 돌며 accept요청을 받는다. 클라이언트가 접속할 때 마다 해당 클라이언트와의 입출력을 담당하는 쓰레드를 하나씩 생성해주고, clients에 SESSION을 추가해주고. 미사일 정보를 갱신해주는 쓰레드를 따로 생성한다.

1. 이미 서버에 접속해 있는 모든 클라이언트에게 해당 접속 정보를 Send한다.

bool SendAddPlayer();

방금 접속한 세션의 정보를 이미 접속해 있는 모든 클라이언트에게 보내주고, cid값을 1 증가시킨다. 모든 send가 성공했을 시에 true를 반환한다.

1. 해당 클라이언트에게 현재 월드 정보를 Send하고 서버에 그 정보를 추가한다.

bool SendWorldData();

~~해당 클라이언트를 포함한 clients와 missiles의 모든 정보를 패킷으로 만들어 해당 클라이언트에게만 전송하고, 모든 Send가 성공 시 true를 반환한다.~~> 해당 클라이언트를 제외한 clients의 정보들을 패킷으로 만들어 해당 클라이언트에게만 전송하고, 모든 Send가 성공 시 true를 반환한다.

DWORD WINAPI ProcessIO(LPVOID \_arg)

아래에 서술할 1~4번의 함수를 순차적으로 수행한다. \_arg로 그 클라이언트의 id를 넘긴다.

1. 메시지가 왔는지 계속해서 확인하고, 받은 메시지에 따라 모든 클라이언트에게 해당 정보를 Send한다.

3-1. bool SendAddMissile(USHORT \_cid);

특정 클라이언트가 미사일 발사 시 mid를 1 증가시키고, 그 세션의 월드 행렬 값으로 방향과 위치를 정하고 미사일 id값을 부여한 후, 그 정보를 모두에게 Send후 모두 성공 시 true를 반환한다.

3-2. bool SendMovePlayer(const CS\_MOVE\_PLAYER& \_packet);

특정 클라이언트가 이동 시 clients에 위치,회전 값을 갱신하고 그 정보를 모두에게 Send 후 모두 성공 시 true를 반환한다.

3-3. bool SendRemoveMissile(UINT \_mid);

특정 클라이언트가 어떤 미사일과 충돌 시 충돌된 missile id를 모두에게 Send 후 모두 성공 시 true를 반환한다.

3-4. bool SendRemovePlayer(USHORT \_cid);

특정 클라이언트가 사망하거나 접속 종료 시 모든 클라이언트에게 인자로 받은 소켓의 cid를 Send해주고, clients에서도 그 정보를 삭제한다.

#5. PM

\*팀원역할분담

김시인

-Client

GameFramework::RecvWorldData();

int SendPlayerMove();

int SendPlayerRemove();

-Server

SESSION 클래스 제작

bool SendAddPlayer();

bool SendRemovePlayer(SOCKET \_socket);

김미령

-Client

ConnectToServer();

int SendMissileRemove(UINT \_missileId);

Server

void AcceptClient();

DWORD WINAPI ProcessIO(LPVOID \_arg);

bool SendWorldData();

bool SendRemoveMissile(UINT \_mid);

전수민

-Client

클라이언트 수정

ProcessRecv(LPVOID \_arg);

Animate, Render();

Scene::Add~, Remove~();

int CheckCollideWithMissile();

-Server

bool SendAddMissile(USHORT\_cid);

bool SendMovePlayer(const CS\_MOVE\_PLAYER& \_packet);

|  |  |  |  |
| --- | --- | --- | --- |
| 날짜 | 김미령 | 김시인 | 전수민 |
| 11월  9일 | 클라이언트 전역변수 추가 | class SESSION 및 서버 전역변수 추가 | Enemy 클래스 삭제, Missile 클래스 수정 등 클라이언트 코드 정리 |
| 11월 10일 | AcceptClient() |  |  |
| 11월 11일 | ConnectToServer() |  |  |
| 11월 12일 |  | SendAddPlayer(); | Recv스레드 생성 및ProcessRecv(LPVOID \_arg) |
| 11월 13일 | SendWorldData() |  |
| 11월 14일 | Framework::  RecvWorldData(); | Scene::AddMissile(); Scene::AddEnemy(); Scene::EnemyMove(); |
| 11월 15일 |  |  | Scene::RemoveMissile(); Scene::RemovePlayer(); |
| 11월 16일 | 테스트 및 디버그 | | |
| 11월 17일 |  | SendPlayerRemove();  SendPlayerRemove() 호출하는 부분 추가. |  |
| 11월 18일 | ProcessIO(LPVOID \_arg) |  | Animate(), Render() 동기화 |
| 11월 19일 | SendPlayerRemove(); 및  게임이 종료되는 부분 구현 | SendAddMissile(); |
| 11월 20일 |  |  |
| 11월 21일 |  | CheckCollideWithMissile(); |
| 11월 22일 |  | SendPlayerMove(); |
| 11월 23일 | 테스트 및 디버그 | | |
| 11월 24일 |  | 강제종료시에도 SendRemovePlayer( SOCKET \_socket); 불리도록 구현 | SendMovePlayer(); |
| 11월 25일 |  |  |  |
| 11월 26일 | SendMissileRemove(UINT \_missileId); | SendRemovePlayer(SOCKET \_socket) 객체 삭제부분 구현 | 처음 접속 시 무적 기능 구현  (쉐이더 및 Scene, Player클래스 수정) |
| 11월 27일 |  |  |
| 11월 28일 | bool SendRemoveMissile(UINT \_mid); | bool SendRemovePlayer(SOCKET \_socket); Send부분 구현 |  |
| 11월 29일 |  |  |  |
| 11월  30일 | **최종 버그 확인 및 테스트 진행, 발표자료 필요 시 제작, 최적화 등** | | |
| 12월1일 |
| 12월2일 |
| 12월3일 |
| 12월4일 |
| 12월5일 |
| 12월6일 |
| 12월7일 | 최종 검수 | | |
| 12월8일 | 최종 평가 | | |

* 수요일은 각자가 개발한 내용과 진행일정을 종합해, 검수하는 날
* 12/1 이후 제출 전까지 오류, 버그 수정과 테스트 등 최종 점검을 진행

|  |  |  |  |
| --- | --- | --- | --- |
| Project Progress | | | |
| 날짜 | 김미령 | 김시인 | 전수민 |
| 11월  9일 | 클라이언트 전역변수 추가 | class SESSION 및 서버 전역변수 추가 | Enemy 클래스 삭제, Missile 클래스 수정 등 클라이언트 코드 정리 |
| 11월 10일 | AcceptClient() |  |  |
| 11월 11일 | ConnectToServer() |  |  |
| 11월 12일 |  | SendAddPlayer(); | Recv스레드 생성 및ProcessRecv(LPVOID \_arg) |
| 11월 13일 | SendWorldData() |  |
| 11월 14일 | Framework::  RecvWorldData(); | Scene::AddMissile(); Scene::AddEnemy(); Scene::EnemyMove(); |
| 11월 15일 |  |  | Scene::RemoveMissile(); Scene::RemovePlayer(); |
| 11월 16일 | 테스트 및 디버그 | | |
| 11월 17일 |  | SendPlayerRemove();  SendPlayerRemove() 호출하는 부분 추가. |  |
| 11월 18일 | ProcessIO(LPVOID \_arg) |  | Animate(), Render() 동기화 |
| 11월 19일 | SendPlayerRemove(); 및  게임이 종료되는 부분 구현 | SendAddMissile(); |
| 11월 20일 |  |  |
| 11월 21일 |  | CheckCollideWithMissile(); |
| 11월 22일 |  | SendPlayerMove(); |
| 11월 23일 | 테스트 및 디버그 | | |
| 11월 24일 |  | 강제종료시에도 SendRemovePlayer( SOCKET \_socket); 불리도록 구현 | SendMovePlayer(); |
| 11월 25일 |  |  |  |
| 11월 26일 | SendMissileRemove(UINT \_missileId); | SendRemovePlayer(SOCKET \_socket) 객체 삭제부분 구현 | 처음 접속 시 무적 기능 구현  (쉐이더 및 Scene, Player클래스 수정) |
| 11월 27일 |  |  |
| 11월 28일 | bool SendRemoveMissile(UINT \_mid); | bool SendRemovePlayer(SOCKET \_socket); Send부분 구현 |  |
| 11월 29일 |  |  |  |
| 11월  30일 | **최종 버그 확인 및 테스트 진행, 발표자료 필요 시 제작, 최적화 등** | | |
| 12월1일 |
| 12월2일 |
| 12월3일 |
| 12월4일 |
| 12월5일 |
| 12월6일 |
| 12월7일 | 최종 검수 | | |
| 12월8일 | 최종 평가 | | |

개발하며 변경된 사항들

11/13 패킷내에 오브젝트의 행렬을 보내주던 부분 (64바이트)와 SC\_ADD\_MISSILE내 position, direction을 float3형 position, float4형 rotation으로 통일 하였다. (총 28바이트). 이에 따라 서버의 SESSION클래스에 월드행렬을 담고 있는 것이 아닌, 플레이어의 위치값, 회전값을 따로 저장하게 바꾸었다.

11/14 클라이언트가 접속하여 RecvWorldData가 호출될 때 본인의 위치, 방향 정보도 포함해서 받는 것으로 계획했지만, 이를 받지 않고 SC\_WORLD\_DATA로는 현재 서버에 접속해 있던 클라이언트 수와 그 수에 맞는 SC\_ADD\_PLAYER만 받고, 접속한 클라이언트는 접속해있던 클라이언트 수를 가지고 본인의 클라이언트id를 알아내는 것으로 바꾸었다.

11/17 쓰레드를 나눈 후 SendAddPlayer와 SendWorldData를 보내는 것이 아닌, 메인 쓰레드에서 직접 보내는 것으로 변경 하였다.