네트워크 게임 프로그래밍 제작기획서

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

#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 몇 초간 죽어 있다가 랜덤한 위치에 다시 부활한다.

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

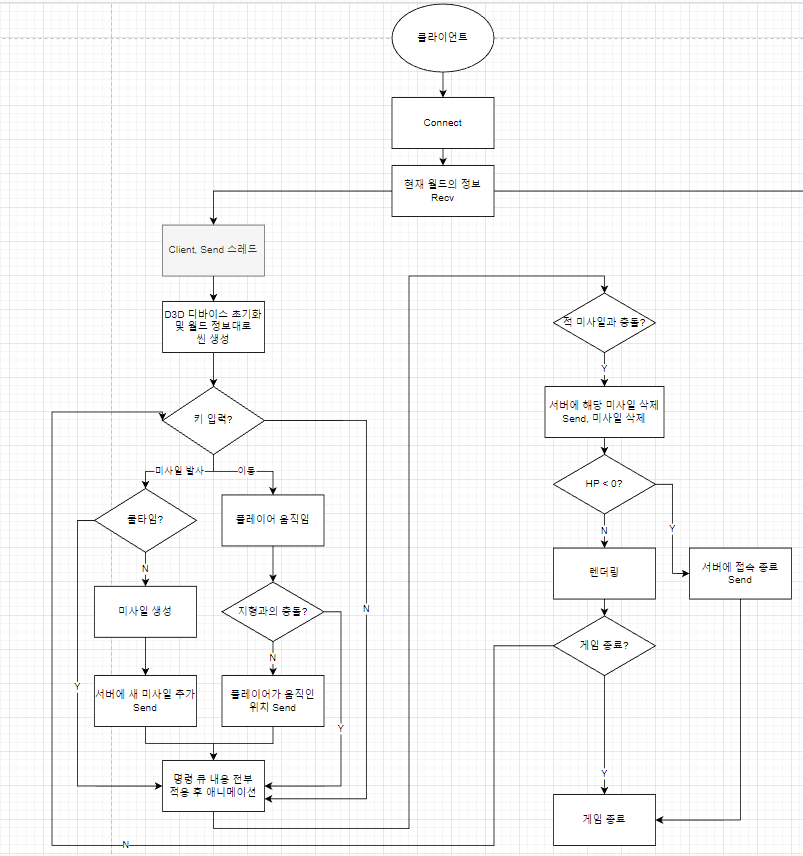
#2. 개발 환경

*   화살이(가) 표시된 사진

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

#3. High Level Design

[클라이언트의 Flow chart]

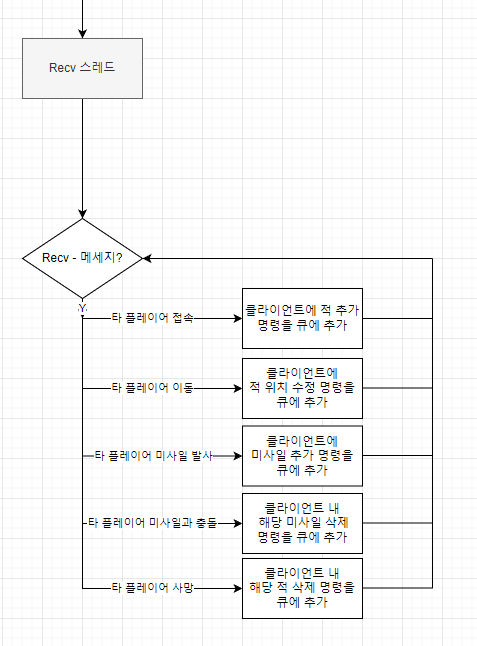


클라이언트는 크게 게임진행, Send를 동시에하는 쓰레드와 Recv하는 쓰레드 두개로 구성.

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.

\* Client, Send 쓰레드 ( main 쓰레드 ) 의 기능

1. 서버에 접속한다.
2. 현재 월드 정보를 Recv해서 반영해준다.
3. Recv 전용 쓰레드를 하나 생성한다.
4. 서버에 새 미사일을 추가했다고 Send를 한다.
5. 서버에 플레이어가 움직였다고 Send를 한다.
6. 플레이어와 미사일 간의 충돌체크를 프레임마다 하고, 충돌시 HP를 깎는다.
7. 명령 큐에 있는 내용을 클라이언트 씬에 적용한다.
8. 서버에 충돌한 미사일을 삭제하라고 Send를 한다.
9. 체력이 없을 경우 서버에 플레이어가 죽었다고 Send를 하고 클라이언트를 종료한다.



\*Recv 쓰레드

1. 메시지가 왔는지 지속적으로 확인하고 메시지가 있을 경우 해당 패킷을 메시지 큐에 추가

[서버의 Flow chart]

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

서버는 새 클라이언트의 접속을 받는 Accept(메인 쓰레드), 접속을 받을 때 마다 생성되는 입출력 쓰레드, 현재 월드의 정보를 계산하고 저장해두는 쓰레드로 나누어진다. 플레이어는 이동했다는 패킷을 받았을 때 바로 서버의 월드 정보를 갱신해주고, 미사일의 움직임은 따로 패킷을 받지 않고 서버에서 따로 움직여 준다.

\* Accept 쓰레드

1. 접속을 받을 준비를 한 후 계속해서 accept를 받고 새 입출력 스레드를 생성한다.
2. 해당 클라이언트에게 초기 위치를 정해 보내준다.

\* 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 Send한다.

\* 미사일 정보 갱신 쓰레드

1. 미사일 방향에 따라 지난 시간에 비례해 미사일 위치를 업데이트 해준다.

#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 인자값으로 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 충돌했을때 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 플레이어와 발사되어있는 미사일들의 정보를 전달하기 위한 패킷. 이 패킷 이후에 그 수에맞게 아래의 패킷들을 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 미사일이여도 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

-queue<void\*> recvCommandQueue; - 서버로부터 받은 패킷을 담는 큐

\* 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. 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 메시지 큐에 추가

DWORD WINAPI ProcessRecv(LPVOID \_arg);

Recv로 1바이트를 읽어 타입을 알아내고, 타입에 맞는 사이즈만큼 다시 Recv를 받아 동적할당한 패킷을 임계영역을 만든 후 큐에 넣고 다시 해제한다.

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 나서의 월드 행렬을 그대로 보내준다.

1. 플레이어와 미사일 간의 충돌체크를 프레임마다 하고, 충돌 시 HP를 깎는다.

int CheckCollideWithMissile();

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

1. 명령 큐에 있는 내용을 클라이언트 scene에 적용한다.

void ExecuteCommandQueue();

while문을 돌며 CommandQueue가 비어있을때 까지 원소를 꺼내 (꺼내는 동안 임계영역 설정) 패킷 타입에 맞게 게임 정보를 갱신해준다. 이 큐 안의 명령은 Recv받은 순서대로 꺼내어져 실행되고, 모두 실행 후 클라이언트를 진행시킨다.

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]

텍스트이(가) 표시된 사진

자동 생성된 설명

\* 사용되는 전역 변수 :

SESSION – 해당 클라이언트의 id값과 이동,회전 정보를 갖고있는 월드 행렬을 담는 클래스

MISSILE – 게임 월드 내의 미사일들의 정보를 담기 위한 클래스

각 멤버를 수정, 참조 하기 위한 Get, Set함수를 만들어준다.

TIMER – 미사일들의 위치를 갱신해줄 때 사용할 경과 시간을 위한 클래스

사용되는 전역 변수

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

list<MISSILE> missiles; - 새로 접속한 클라이언트에게 현재 월드의 미사일을 보내주기 위한 컨테이너

int cid, mid; - 클라이언트나 미사일이 추가될때마다 부여받을 id. 추가될 때 1씩 증가한다.

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

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

void AcceptClient();

윈속 초기화, 대기소켓 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.

DWORD WINAPI ProcessIO(LPVOID \_arg)

아래에 서술할 2~5번의 함수를 순차적으로 수행한다. \_arg로 그 클라이언트의 id를 넘긴다.

1. 해당 클라이언트에게 초기 위치를 정해 보내준다

void SetInitPosition();

missiles, clients내의 월드 행렬의 위치값을 바탕으로 충돌하지 않는 안전한 위치를 계산해 그 값을 방금 접속한 세션의 위치값으로 정한다.

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 성공했을시에 true를 반환한다.

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

bool SendWorldData();

해당 클라이언트를 포함한 clients와 missiles의 모든 정보를 패킷으로 만들어 해당 클라이언트에게만 전송하고, 모든 Send가 성공시 true를 반환한다.

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

5-1. bool SendAddMissile(USHORT \_cid);

특정 클라이언트가 미사일 발사 시 mid를 1 증가시키고, 그 세션의 월드 행렬값으로 방향과 위치를 정해 missiles에 그 미사일을 추가하고 미사일 id값을 부여한 후, 그 정보를 모두에게 Send후 모두 성공시 true를 반환한다.

5-2. bool SendMovePlayer(const CS\_MOVE\_PLAYER& \_packet);

특정 클라이언트가 이동 시 clients에 월드 행렬값을 갱신하고 그 정보를 모두에게 Send 후 모두 성공시 true를 반환한다.

5-3. bool SendRemoveMissile(UINT \_mid);

특정 클라이언트가 어떤 미사일과 충돌 시 충돌 된 missile id를 모두에게 Send 후 모두 성공시 true를 반환한다.

5-4. bool SendRemovePlayer(USHORT \_cid);

특정 클라이언트가 사망하거나 접속 종료시 모든 클라이언트에게 인자로 받은 소켓의 cid를 Send해주고, clients에서도 그 정보를 삭제 한다.

1. 미사일 방향에 따라 지난 시간에 비례해 미사일 위치를 업데이트 해준다.

DWORD WINAPI ProcessMissileUpdate(LPVOID \_arg);

미사일 리스트를 for문을 돌면서, 경과 시간에 비례하여 미사일의 위치를 계속해서 바꾸어준다.

#5. PM

\*팀원역할분담

김시인

-Client

int SendPlayerMove();

int SendPlayerRemove();

-Server

SESSION, MISSILE, TIMER클래스 제작

void SetInitPosition();

bool SendAddPlayer();

bool SendRemovePlayer(SOCKET \_socket);

김미령

-Client

ConnectToServer(); - 김미령

int SendMissileRemove(UINT \_missileId); - 김미령

Server

void AcceptClient();

DWORD WINAPI ProcessIO(LPVOID \_arg);

bool SendWorldData();

bool SendRemoveMissile(UINT \_mid);

전수민

-Client

GameFramework::RecvWorldData();

ProcessRecv(LPVOID \_arg);

void ExecuteCommandQueue();

int CheckCollideWithMissile();

-Server

DWORD WINAPI ProcessMissileUpdate(LPVOID \_arg);

bool SendAddMissile(SOCKET \_socket);

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

\*전체개발일정

김미령

전수민

김시인

: 캘린더 형식으로 개발일정 정리. 블록 색깔로 구분하면 됨.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 월 | 화 | 수 | 목 | 금 | 토 | 일 |
| 10/20~ 11/1  기획 및 기획서 작성 | | 11/ 2  문서 제출일 | 3 | 4  RecvWorldData() 구현  SESSION, MISSILE, TIMER 클래스 제작 | 5  AcceptClinet(), ConnectToServer(), SendWorldData() 구현 | 6 |
| 7 | 8  ProcessIO(LPVOID \_arg)구현  SetInitPosition() 구현 | 9 | 10  SendAddPlayer() 구현 | 11 | 12  ProcessRecv(LPVOID \_arg) 구현 | 13 |
| 14 | 15  SendPlayerMove() 구현 | 16  ExecuteCommandQueue() 구현 | 17 | 18  CheckCollideWithMissile() 구현 | 19 | 20 |
| 21  SendMovePlayer(const CS\_MOVE\_PLAYER& \_packet) 구현 | 22  SendPlayerRemove() 구현 | 23 | 24 | 25 | 26  ProcessMisslieUpdatae(LPVOID \_arg) | 27  SendRemovePlayer(UINT \_missileled) 구현 |
| 28 | 29 | 30  SendAddMissile(SOCKET \_socket) 구현 | 12 / 1 | 2  SendMissileRemove(UINT \_missilled) 구현 | 3 | 4  SendRemoveMissile(UINT \_mid) 구현 |
| 5 | 6 | 7 | 8  최종 발표일 |  |  |  |