**네트워크 게임 프로그래밍**

**Term-Project 기획서**

팀원

**1. 어플리케이션 기획**

- 게임 이름

Tank Fighters

- 게임 장르

멀티 탱크 대전게임

-게임 특징

최대 4인 참여가 가능한 탱크 대전 게임

DirectX 9을 사용

조작법 : W,A,S,D : 이동, 방향키 : 공격

- 게임 방법

최대 4명의 플레이어의 탱크가 전투를 하고 최후에 살아남은 플레이어가 승리하게 된다.

- 게임 방식, 구조

Y

N

Y

Y

N

N

방향키를 눌럿는가

이동

포탄 발사

다른 플레이어가 모두 죽었는가

게임 종료

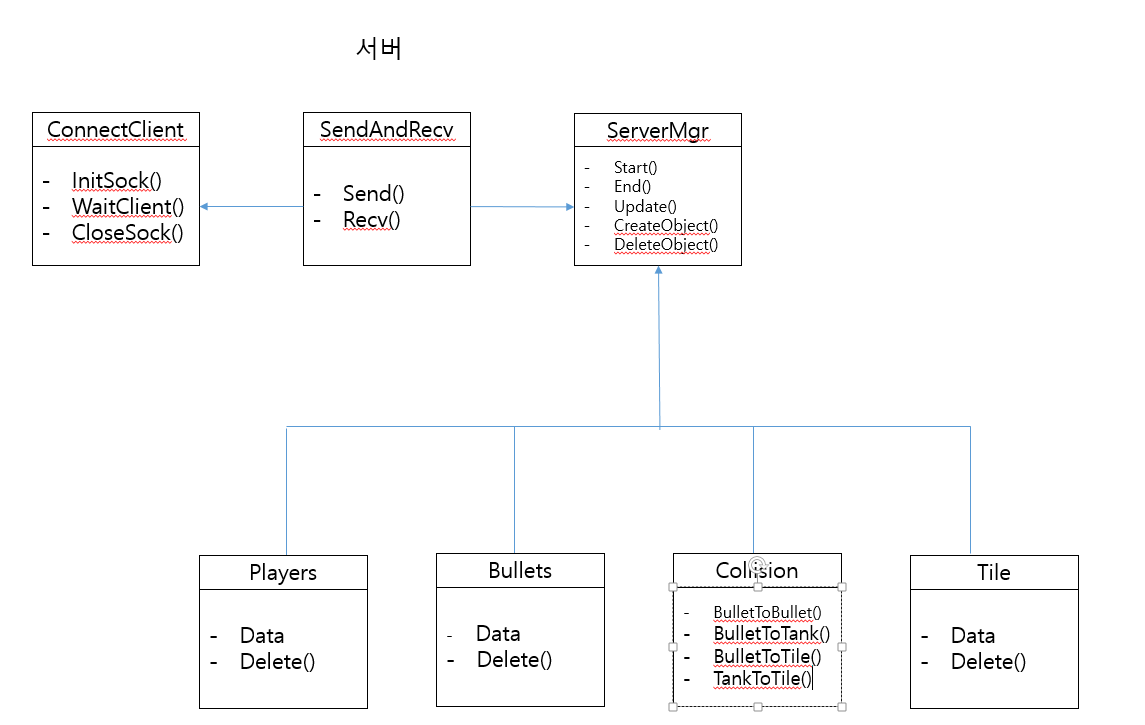
공격키를 눌럿는가

업데이트

게임 시작

**2. HIGH LEVEL DESIGN**

- 서버



- SendThread()

- RecvThread()

Data

* ConnectClient

클라이언트와의 접속을 위해서 InitSock()에서 소켓을 초기화,생성 해주고 WaitClient()에서 서버의 상태를 listen상태로 만든다.

* SendAndRecvData

클라이언트와의 다양한 정보(체력, 좌표, 키값)를 송수신 할 데이터

* ServerMgr

서버매니져로 게임의 시작과 종료, 객체의 생성과 파괴, 게임에서 발생하는 여러 상태들을 Update한다.

- Players

플레이어의 위치, 체력 정보

플레이어의 체력이 0이됬을 때 삭제한다.

- Bullets

포탄의 위치, 방향 정보

포탄이 화면을 벗어났을 경우, 탱크에 충돌했을 경우,타일에 맞았을 경우에 삭제한다.

- Collision

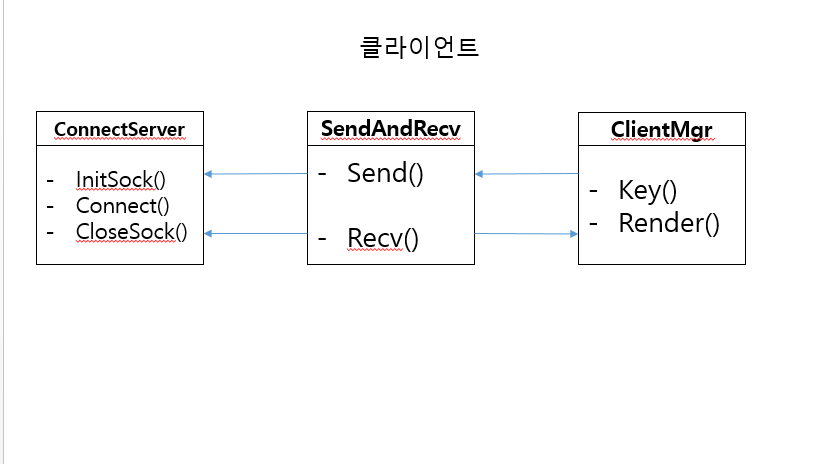
탱크와 포탄과 타일간의 충돌 판단

- Tile

타일 상태의 정보

포탄이 타일에 맞았을 경우에 삭제한다.

- 클라이언트



- SendThread()

- RecvThread()

-KeyInput()

Data

* ConnectServer

InitSock()으로 소켓을 초기화하고, Connect()로 서버와 접속을 시도한다.

* SendAndRecvData

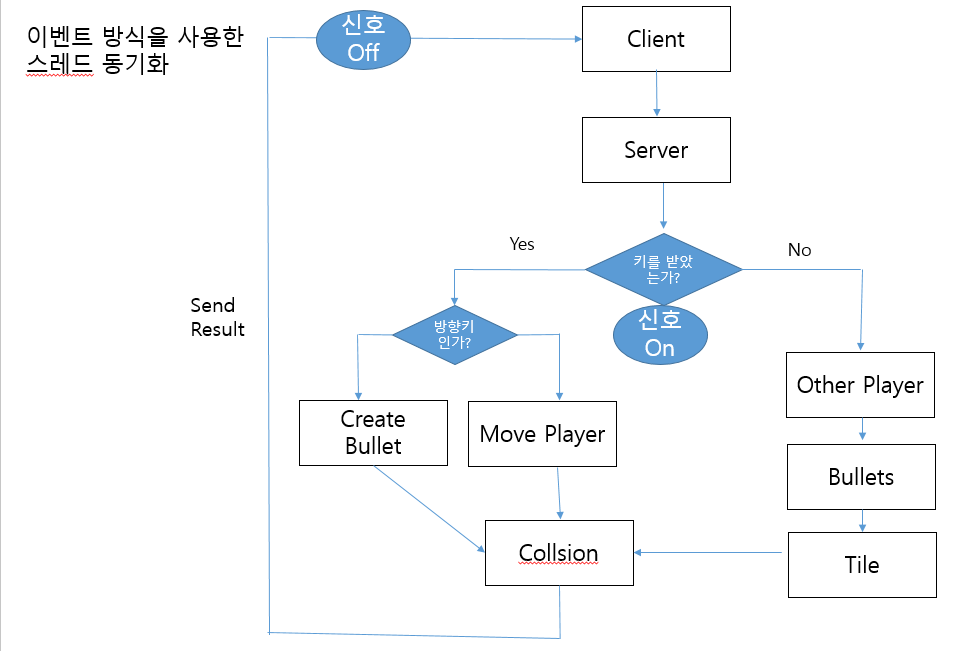
서버와 정보(체력, 좌표, 키값)를 송수신 하기 위한 데이터.

* ClientMgr

플레이어가 입력한 키정보를 전송하는 한편, 서버로부터 전송받은 정보를 Render해준다.

- 스레드 구조

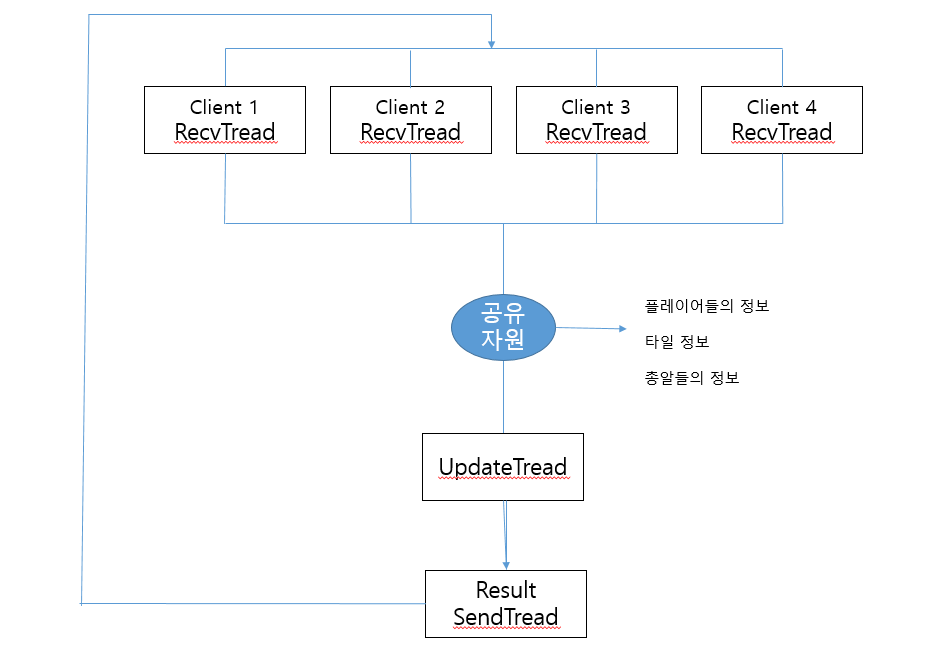
전체 동기화 구조



이벤트방식을 이용하여 스레드를 동기화

Update도중에 클라이언트에서 키를 입력 받으면 이벤트를 신호 상태로 바꾼다. 그후 서버에서 키 값 데이터를 공유자원에 저장을 한다. 이후에 그 키 값을 판단하여 총알, 플레이어, 타일들의 상태를 변환 , 충돌체크 단계를 거친 후에 이벤트를 비 신호 상태로 변환한다.

세부구조



클라이언트에서 RecvTread를 통해 정보를 받는다.

공유자원은 UpdateTread를 통해 Update 후에 최종 결과물을 SendTread로 클라이언트들에게 다시 전달(브로드캐스팅)한다.

**3. LOW LEVEL DESIGN**

- 서버 개요 :

TCP 서버 사용

스레드 동기화 : 이벤트 방식

바이트 정렬 방식 : 리틀 엔디안

전송방식 : 고정길이 + 가변길이

- 서버 함수

ConnectToClient : 서버의 메인

1. Void InitSock(int& retval)

클라이언트의 접속을 위해 소켓을 생성 및 초기화 해준다.

2. Void WaitClient(int& retval)

클라이언트의 접속을 기다린다. 소켓을 listen상태로 만든다.

3. Void CloseSock()

게임이 종료되면 클라이언트와의 접속을 끊고 소켓을 닫는다.

4. DWORD WINAPI RecvThread(LPVOID clientSock)

키 값을 클라이언트에서 받아서 스레드의 공유자원에 저장한다.

이때 이벤트 신호는 비신호 상태로 설정한다.

5. DWORD WINAPI UpdateTread(LPVOID ServerMgr)

ServerMgr클래스에서 충돌과 객체들의 움직인 좌표 값을 받아서 연산을 하거나 객체를 생성한다. 그 후에 SendTread 호출한다.

6. DWORD WINAPI SendThread(LPVOID ServerMgr)

ServerMgr의 데이터들을 브로드캐스팅 하여 모든 클라이언트 들에게 전달한다.

Class ServerMgr : 서버 메니저로 서버의 게임상태와 객체를 관리

7. void ServerMgr::GameStart()

게임상태를 게임 중 으로 바꾼다. -> int GameStatus = start;

8. void ServerMgr::GameEnd()

게임 상태를 게임 종료로 바꾼다. -> int GameStatus = end;

9. void ServerMgr::CreateObject(int Object\_Type)

#define BULLET\_OBJECT 0

#define TANK\_OBJECT 1

#define TILE\_OBJECT 2

오브젝트 타입 값을 받아서 해당 오브젝트를 생성한다.

10. void ServerMgr::DeleteObject(int Object\_type)

오브젝트를 파괴하라는 명령이 들어왔을 경우 조건문으로 해당 오브젝트의 Delete() 함수를 호출해준다.

Class GameObj : 게임오브젝트에 대한 세부 내용

11. void GameObj::Move(int iX, int iY, int client)

탱크 : 입력받은 해당 client의 번호에 해당되는 플레이어의 좌표값을 바꾼다.

포탄 : 속도에 따라 좌표값을 바꾼다.

12. Void GameObj::Delete(int clinet)

탱크 : 해당 플레이어의 체력이 0이되서 죽으면 객체를 지운다.

포탄 : 물체와의 충돌시에 객체를 지운다.

- 클라이언트 함수

1. Void InitSock(int& retval)

서버와의 접속을 위해 소켓을 초기화 해준다.

2. Void Connect(int& retval)

서버와 연결을 한다. RecvThread를 호출한다.

3. DWORD WINAPI RecvThread(LPVOID ClientMgr)

서버로부터 받은 데이터를 클라이언트 메니저에 저장한다.

4. DWORD WINAPI SendThread(LPVOID Key)

키를 입력 할 때 호출 되며, 키값을 서버에 전송한다.

5. void Render()

- ClientMgr데이터를 기반으로 사용자의 화면에 렌더링 해준다.

- 패킷 구조체

Typedef Struct ObjInfo : 오브젝트 정보

{

Int pos\_x;

Int pos\_y;

Int speed;

Int state; // 오브젝트 상태

}ObjInfo

Typedef struct allSorce : 공유 자원

{

Int Id; - 클라이언트 넘버

ObjInfo objInfo; - 오브젝트 정보

Bool InputKey[8] – 입력받은 키 상태값 ( 동시키 입력을 처리하기 위해, true : 누름, false : 안누름)

}

클라이언트 넘버가 0이 되었을 경우 클라이언트가 키를 누르지 않았다고 판단. 4인 일 경우 1,2,3,4 전달

**4. 개발환경**

Windows 10 64 bit

Intel CPU

Visual studio 2017

DirectX 9

**5. 팀원 별 역할 분담**

|  |  |  |  |
| --- | --- | --- | --- |
| 역할  날짜 | 홍승필 | 황서윤 | 유창민 |
| 11/9 | 리소스 수집 | 리소스 수집 | 리소스 수집 |
| 10 | 클라)  기존 게임 프로젝트  2008 -> 2017 리팩토링 | 클라)  InitSocket(), ConnetToServer(),  CloseSocket() | 서버)  InitSocket(),  WaitClient(),  CloseSocket() |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 | 연결확인 및 디버깅 | 연결확인 및 디버깅 | 연결확인 및 디버깅 |
| 16 | 클라)  Player(), Enemy() |  | 서버)  Player(),Enemy() |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 | 서버)  Bullet(), Tile() | 클라)  Bullet(), Tile() |  |
| 20 |  |  |  |
| 21 |  |  |  |
| 22 | 키 상태 전달 테스트 | 키 상태 전달 테스트 | 키 상태 전달 테스트 |
| 23 |  |  |  |
| 24 | 서버) 충돌체크  BulletToTank(),  TankToTile() | 클라)  RecvThread() | 서버) 충돌체크  BulletToBullet(),  BulletToTile() |
| 25 |  |  |  |
| 26 |  |  |  |
| 27 |  |  |  |
| 28 |  | 서버)  SendThread() |  |
| 29 | 클라)  ClientManager() |  | 서버)  ServerManager() |
| 30 |  |  |  |
| 12 /1 |  |  |  |
| 2 | 1차중간 테스트, 디버깅 | 1차 중간 테스트, 디버깅 | 1차 중간 테스트, 디버깅 |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 | 클라)  대기실 씬 구현 | 서버)  스코어, 체력 상태 구현 | 클라)  리소스 교체, 타일 배치 |
| 6 |  |  |  |
| 7 | 2차 중간 테스트, 디버깅 | 2차 중간 테스트, 디버깅 | 2차 중간 테스트, 디버깅 |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 | 클라)  GameResult 씬 구현 | 서버)  GameResult 결과 전달 | 클라)  GameResult 씬 구현 |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 | 3차 중간 테스트, 디버깅 | 3차 중간 테스트, 디버깅 | 3차 중간 테스트, 디버깅 |
| 14 | 클라)  FMOD 사운드 추가 | 클라)  FMOD 사운드 추가 | 클라)  FMOD 사운드 추가 |
| 15 | 실습실 환경 테스트 | 실습실 환경 테스트 | 실습실 환경 테스트 |
| 16 | 최종점검 및 디버깅 | 최종점검 및 디버깅 | 최종점검 및 디버깅 |
| 17 |  |  |  |

일정 확인표

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 |  | 10 |  | 11 |  | 12 |  | 13 |  | 14 |  | 15 |  |
| 16 |  | 17 |  | 18 |  | 19 |  | 20 |  | 21 |  | 22 |  |
| 23 |  | 24 |  | 25 |  | 26 |  | 27 |  | 28 |  | 29 |  |
| 30 |  | 1 |  | 2 |  | 3 |  | 4 |  | 5 |  | 6 |  |
| 7 |  | 8 |  | 9 |  | 10 |  | 11 |  | 12 |  | 13 |  |
| 14 |  | 15 |  | 16 |  | 17 |  |  |  |  |  |  |  |