**네트워크 게임 프로그래밍**

**Term Project**

**2015182004 김하윤**

**2016180011 김연지**

1. **애플리케이션 기획**
   1. **게임 제목**

Metal Slug Shooting

* 1. **게임 장르**

1:1 슈팅게임

* 1. **게임 설명**

Metal Slug라는 횡 스크롤 액션 아케이드 게임 시리즈를 모작하여 1:1 슈팅게임으로 재구성했습니다.

게임을 실행 후 로고가 뜨고 서버에 접속한 순서대로 1P, 2P가 지정됩니다.

로비화면에서는 자신이 1P. 2P인지 확인을 할 수 있고, 서버에 2명이 접속을 하게 된 후 5초를 카운트한 후 게임이 시작됩니다.

플레이 중 헬기가 나타나 맵 중앙에 아이템을 떨어트립니다.

한 게임당 제한시간은 90초입니다.

승리 조건은 상대 플레이어를 죽이고 살아남는 것입니다.

단, 시간이 다 하여 둘 다 살아있는 경우에는 체력이 더 많이 남은 쪽이 승리하게 됩니다.

Player

1. 게임에 로그인 한 순서에 따라 1P, 2P가 자동으로 선택됩니다.
2. 플레이어의 총알 이미지는 1P, 2P 동일합니다.
   * 총알의 데미지:10
   * ~~탄창을 교환할 때 딜레이 발생 약 2초, 그동안은 공격이 불가합니다.~~
   * 초반 지급 탄환은 총알 10발과 탄창 2개 즉, 총알 30개를 지급합니다.

|  |  |  |
| --- | --- | --- |
| 1P | 2P | Bullet |
|  |  |  |

1. 플레이어의 체력:100
   * 체력 회복 아이템을 먹을 시 체력 회복이 가능합니다.

헬기

|  |  |
| --- | --- |
| 이미지 | 설명 |
|  | 게임 시작 후 10초 뒤 ~~30~~ 20초 간격으로 ~~3~~ 4번 출현합니다.  ~~10초에는 수류탄 아이템을 떨굽니다.~~  ~~40초에는 탄창을 충전해주는 아이템을 떨굽니다.~~  ~~마지막으로 70초에는 30또는 70의 피회복을 해주는 아이템이 랜덤으로 떨어트립니다.~~  HP+30, HP+70, 탄창, 수류탄 순서대로 떨어트립니다. |

Item

1. 아이템은 자신이 승리 조건에 도달하기 쉽도록 도와줍니다.
   * 아이템의 획득 여부에 따라 게임의 승패 요인에 영향을 끼칩니다.
2. 게임 플레이 중 헬기가 나타나 맵 중앙에 아이템을 떨굽니다.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | 이미지 | 변경 이미지 | 설명 |
| 체력 회복 |  |  | 먹는 즉시 체력 회복 |
| 수류탄 |  |  | 공격력:~~50~~ 30  포물선으로 이동함 |
| 탄창 충전 | íì°½ ì¼ë¬ì¤í¸ì ëí ì´ë¯¸ì§ ê²ìê²°ê³¼ |  | 1~~~3~~개 랜덤으로 획득 |

맵

|  |  |
| --- | --- |
| 이미지 | 설명 |
|  | 크기 : 1280x720 |

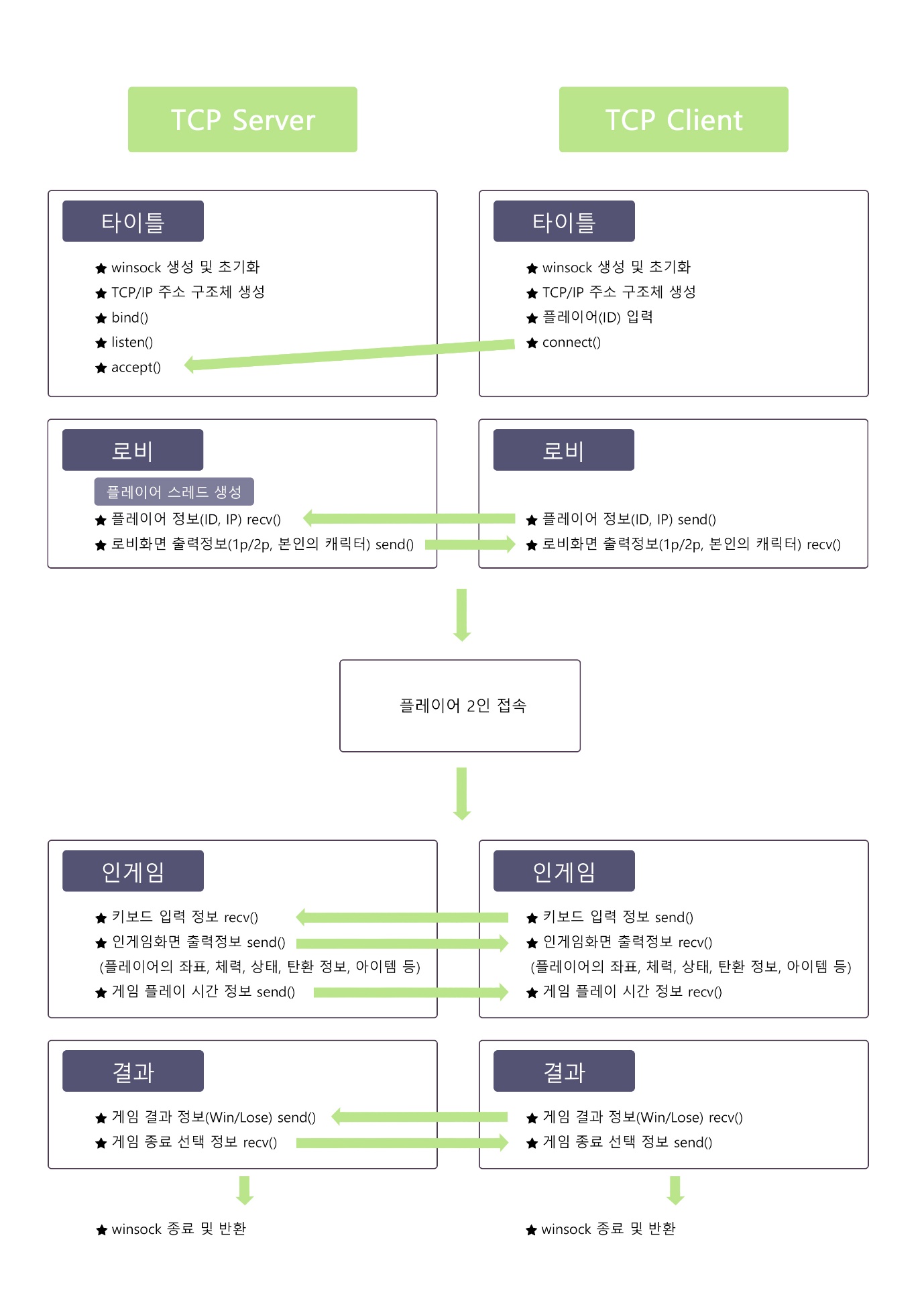
|  |  |  |
| --- | --- | --- |
| 기능 | 키 | 설명 |
| 좌, 우 이동 | ←, → | 플레이어가 좌, 우로 이동 |
| 앉기 | ↓ | 플레이어가 현재 위치에서 앉음 |
| 탄창 교환 | R | 총알을 다 사용했을 경우 탄창을 교환하여 총알을 채움 |
| 총알 발사 | SPACE BAR | 플레이어가 총알을 발사 |
| 수류탄 발사 | Q | 플레이어가 수류탄을 발사 |

* 1. **게임 조작법**
  2. **게임 스크린샷 예**

 [그림1] 로비 화면에서 첫번째로 접속한 클라이언트가 플레이어 1로 선택된 예이다.

[그림2] 인게임 화면에서 플레이어2가 총을 발사하고 있으며, 헬기에서 아이템이 드랍되는 상황이다.

1. **High-Level 디자인**



1P접속시 게임(방) 스레드 생성

* 1. **클라이언트**
     1. 서버에 접속을 시도하고 다른 클라이언트가 접속할 때 까지 ‘WAIT’라는 단어가 화면에 나옵니다.
     2. 모든 플레이어가 접속을 성공했으면 서버로부터 인 게임으로 가기 위한 카운트를 세기위해 카운트 정보를 받습니다.
     3. 총알, 아이템, 플레이어의 정보를 그립니다.
     4. UI를 그립니다.
     5. 키 입력 값을 서버에게 보냅니다.
     6. 지정된 시간(10, 30, 50, 70)이 경과하면 헬기가 나타나 아이템을 떨어트립니다.
     7. 서버로부터 게임의 승패 결과를 받습니다.
  2. **서버**
     1. 접속한 클라이언트의 순서에 맞게 1P, 2P라는 ID를 부여합니다.
     2. 접속한 클라이언트가 2명이 되었을 때 자동으로 ~~5~~ 3초의 카운트를 세는 것을 클라이언트에게 알립니다.
     3. 플레이어의 최초 상태를 클라이언트에게 알려줍니다.
     4. 클라이언트의 키 입력 정보를 받습니다.
     5. 플레이어와 총알의 위치좌표를 갱신합니다.
     6. 각각의 클라이언트에게 게임제한시간(90초부터 0으로 카운트)을 계속해서 보내줍니다.
     7. ~~지정된 시간(10, 40, 70)이 경과하면 헬기가 나타나 아이템을 떨어트립니다.~~
     8. 아이템의 정보를 클라이언트에게 보내줍니다.
     9. 총알, 아이템과 플레이어의 충돌처리를 합니다.
     10. 플레이어 둘 중 한명이라도 죽으면 게임 결과를 클라이언트에게 보낸 후 게임 종료합니다.
     11. 게임제한시간이 지난 후, 플레이어 둘 다 살아 있으면 체력이 더 많은 쪽에게 승리 결과를 보내고, 체력이 더 적은 쪽에는 패배 결과를 보낸 후 게임 종료합니다.

1. **Low-Level 디자인**
   1. 프로토콜
      1. #define



#define LEFT\_KEY 3 // 키보드 방향키 ←

#define RIGHT\_KEY 4 // 키보드 방향키 →

#define DOWN\_KEY 5 // 키보드 방향키 ↓

#define SHOOT\_KEY 6 // 제자리에서 총알발사

#define PLAYERMAX 2

#define MAXCOUNT 10

#define BOOMCOUNT 3

#define DEFAULT\_MID\_HEIGHT 300.f

#define BOOM\_DIST 500.f

#define PLAYER\_SPEED 15.f

#define BULLET\_SPEED 30.f

#define BULLET\_DAMAGE 10.f

#define BOOM\_DAMAGE 30.f

* + 1. enum 열거형

enum GAMESTATE

{

GAME\_READY = 0, // 게임 대기중

GAME\_OK = 1, // 게임 준비완료

GAME\_PLAY = 2, // 게임 플레이상태

GAME\_END = -1 // 게임 종료상태

};

enum GAMERESULT

{

NONE = 0, // 아직 결과 없음

WIN = 1, // 승

LOSE = -1 // 패

};

enum PLAYER\_STATE

{

STAND, // 서 있는 상태

SIT, // 앉아 있는 상태

ATTACK, // 공격하는 상태

DIE // 죽은 상태

};

* + 1. struct 구조체

struct PlayerInfo

{

int num; // 1p인지 2p인지

int x; // 좌표 x

int y; // 좌표 y

int magazinecnt; // 탄창수

int boomcnt; // 수류탄수

int bulletcnt; // 총알수

int hp; // 현재 HP

int dir; // 바라보는 방향 -1:왼 1:오

int state; // 현재상태 -> 애니메이션 출력 때문에...

};

struct BulletInfo

{

~~int num; // 총알의 주인 1:1p 2:2p~~

int x; // 좌표 x

int y; // 좌표 y

~~int damage; // 총알의 데미지~~

int dir; // 총알의 방향

bool shoot; // 발사했는지

};

struct ItemInfo

{

int num; // 아이템 정보 0:HP30회복 1:HP70회복 2:탄창 3:수류탄

int x; // 좌표 x

int y; // 좌표 y

bool enable; // 그려지는 건지

}; // 12byte

struct BoomInfo

{

int x;

int y;

bool shoot;

};

struct HeliInfo

{

int x; // 좌표 x

bool drop; // 아이템 떨어트릴건지

};

struct DATA

{

int num; // 1P, 2P

PlayerInfo player[PLAYERMAX]; // 플레이어 정보

BulletInfo bullet[PLAYERMAX][MAXCOUNT]; // 총알 정보

BoomInfo boom[PLAYERMAX][BOOMCOUNT]; // 수류탄 정보

};

struct ITEM\_DATA

{

ItemInfo Item[4];

};

struct Key\_DATA

{

int num; // 1P, 2P

int key; // key값

};

* 1. class 구조

1. Network // 서버 통신과 관련된 Class
   1. Network
2. GameFramework // 게임의 전반적인 Framework를 관리하는 Class
   1. Timer
   2. Input
   3. Scene
   4. GameObject
   5. Animation
   6. Texture
3. GameScene // 게임 장면에 대한 Class
   1. Lobby
   2. InGame
4. Object // 게임 내의 오브젝트에 대한 Class
   1. Background
   2. Player
   3. Boom & Bullet
   4. Item
      1. Heli
      2. Boom
      3. Magazine
      4. Healing
   5. UI // UI에 대한 Class
      1. HP Bar & Gauge
      2. Timer
      3. Win or Lose
   6. 스레드 함수

~~// 타이머 스레드 함수  
// -> 클라이언트에게 시간을 전달할 때 사용하는 함수입니다.  
DWORD WINAPI Timer(LPVOID arg);~~

// 클라이언트와 데이터 통신 스레드 함수  
~~// -> ReceiveKeyData와 SendData 스레드 함수를 이 함수 내에서 호출합니다.~~// -> Send와 RecvKeyAndDataUpdate 함수를 이용하여 GameState와 Timer, 제한시간, DATA, 헬기정보, item정보, 승패여부 등을 주고받으며 처리합니다.  
DWORD WINAPI ~~ProcessClient~~ ProcessGame (LPVOID arg);

~~// 키 입력 받는 스레드 함수  
// -> 클라이언트로부터 키 값을 받고, 받은 키 값을 토대로 Player와 Bullet의 정보를 갱신해주는 함수입니다.  
DWORD WINAPI ReceiveKeyData(LPVOID arg);~~

~~// 입력받은 키를 토대로해서 DATA구조체를 갱신 후 클라이언트에게 보내는 스레드 함수입니다.  
// -> 클라이언트에게 시간을 전달할 때 사용하는 함수입니다.  
DWORD WINAPI SendData(LPVOID arg);~~

* 1. 동기화

Critical Section(임계영역)사용

* + 동기화가 다른 동기화방식보다 빠르고 특히 아이템을 먹을 때 다른 스레드의 접근을 막아야 하기 때문입니다.
  1. **클라이언트**

1. 송신함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| ~~void SendInputKey(int Key)~~  void CNetwork::Input() | 플레이어가 입력한 키보드 정보를 서버로 전송하기 위한 함수입니다.  입력한 키 정보를 ~~int~~Key\_DATA형 데이터로 서버에 전달합니다.  ~~(ex. Q\_KEY = 1, RIGHT\_KEY = 3 등)~~  로비에서 준비 상태 입력, 게임 플레이관련 키 등 |
| ~~void SendPlayerInfo(PlayerInfo Player)~~ | ~~플레이어의 정보를 담은 구조체를 서버로 전송하는 함수입니다.~~ |
| ~~void SendBulletInfo(BulletInfo bullet)~~ | ~~총알의 정보를 담은 구조체를 서버로 전송하는 함수입니다.~~ |

1. 수신함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| ~~void RecvTimerInfo(int time)~~ | ~~게임 플레이의 제한시간을 카운트하는 정보를 수신하는 함수입니다.~~ |
| ~~void RecvGameResult (int num)~~ | ~~게임의 결과를 수신하는 함수입니다.~~  ~~num 값이 -1이면 패배, 1이면 승리~~ |
| ~~void RecvGameState(int state)~~ | ~~게임의 상태를 수신하는 함수입니다.~~  ~~state = 0(게임 중), state = -1(게임 종료)~~ |
| ~~void RecvItemInfo(int num)~~ | ~~헬기에서 떨어진 아이템의 정보를 받는 함수입니다.~~  ~~num 값 1(회복), 2(수류탄), 3(탄창)~~ |
| ~~void RecvHelicopter(bool heli)~~ | ~~헬기가 그려질 때를 알기 위한 함수입니다.~~  ~~heli 값이 false이면 안 그림, true이면 그림~~ |
| ~~void RecvTime(int time)~~ | ~~경과한 시간을 서버에서 받는 함수입니다.~~ |
| ~~void RecvTimeLimit(int time)~~ | ~~제한 시간을 서버에서 받는 함수입니다.~~ |
| void CNetwork::Update() | 서버로부터 데이터들(게임 상태, 제한시간, 헬기정보, 아이템, 승패 정보)을 받고 업데이트 합니다. |

* 1. **서버** 
     1. 송신관련함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| ~~void SendTimerInfo(int time)~~ | ~~게임 플레이의 제한시간을 카운트하는 정보를 송신하는 함수입니다.~~ |
| ~~void SendGameResult (GAMERESULT result)~~ | ~~게임의 결과를 송신하는 함수입니다.~~  ~~result = -1(패배), result = 1(승리)~~ |
| ~~void SendGameState(int state)~~ | ~~게임의 상태를 송신하는 함수입니다.~~  ~~state = 0(게임 중), state = -1(게임 종료)~~ |
| ~~void SendItemInfo(int num)~~ | ~~헬기에서 떨어트릴 아이템의 정보를 송신할 함수입니다.~~  ~~num 값 1(회복), 2(수류탄), 3(탄창)~~ |
| ~~void SendHelicopter(bool heli)~~ | ~~헬기가 그려질 때를 알기 위한 함수입니다.~~  ~~heli = true 그림, false 안 그림~~ |
| ~~void SendTime(int time)~~ | ~~경과한 시간을 클라이언트로 전송하는 함수입니다.~~ |
| ~~void SendTimeLimit(int time)~~ | ~~제한 시간을 클라이언트로 전송하는 함수입니다.~~ |
| void Send(const char \* buf, int len, string str) | 재정의한 send함수 입니다.  각각의 클라이언트에게 send해주고 retval이 -1(SOKET\_ERROR)일 시, 해당 클라이언트가 종료했음을 알려줍니다. 또한 입력한 str을 console창에서 cout 해줍니다.  Ex) 1P send 승패여부 |

* + 1. 수신관련함수

|  |  |
| --- | --- |
| 이름 | 설명 |
| ~~void RecvInputKey(int Key)~~ void RecvKeyAndDataUpdate() | 플레이어가 입력한 키보드 정보를 recv 함수로 수신 받고 그에 따른 캐릭터 정보, 불렛, 수류탄 정보들을 처리해주는 ~~클라이언트로 수신하기 위한~~ 함수입니다.  ~~입력한 키 정보를 int형 데이터로 서버에서 전달받습니다.~~ |
| ~~void RecvPlayerInfo(PlayerInfo Player)~~ | ~~플레이어의 정보를 서버에서 수신하는 함수입니다.~~  ~~PlayerInfo 구조체를 받습니다.~~ |
| ~~void RecvBulletInfo(BulletInfo bullet)~~ | ~~총알의 정보를 서버에서 수신하는 함수입니다.~~  ~~BulletInfo~~~~구조체를 받습니다.~~ |
| void Recv(char \* buf, int len, string str) | 재정의한 recv함수 입니다.  각각의 클라이언트로부터 recv받고 retval이 -1(SOKET\_ERROR)일 시, 해당 클라이언트가 종료했음을 알려줍니다. 또한 입력한 str을 console창에서 cout 해줍니다.  Ex) 1P recv 승패여부 |

1. **개발환경**
   1. **Visual studio 2017**
   2. **Github, gitsvn**
   3. **포토샵, 그림판, Word**
2. **팀원 별 역할분담**
   1. **클라이언트**

|  |  |  |
| --- | --- | --- |
| 구현 내용 | 김하윤 | 김연지 |
| 게임 UI, 플레이어, 아이템 그래픽 디자인 및 구현 |  | √ |
| 클라이언트 프로토콜 디자인 및 구현 | √ |  |
| class Timer, class Input (타이머, 키 입력)구현 | √ |  |
| class Timer, class Input (타이머, 키 입력)구현 | √ |  |
| class Scene, class SceneManager (장면과 장면을 관리하는 매니저) 구현 | √ |  |
| class Object, class Player (오브젝트와 플레이어) 구현 | √ |  |
| class Bullet 구현 | √ | √ |
| void SendBulletInfo() 구현 | √ | √ |
| class Item (아이템 3가지)구현 |  | √ |
| 기본적인 네트워크 환경 구현 |  | √ |
| 게임 결과 recv함수 구현 |  | √ |
| 아이템 정보 recv함수 구현 |  | √ |
| void SendKeyData() 구현 |  | √ |
| DWORD WINAPI SendData() 구현 |  | √ |

* 1. **서버**

|  |  |  |
| --- | --- | --- |
| 구현 내용 | 김하윤 | 김연지 |
| 기본적인 네트워크 환경 구현 |  | √ |
| DWORD WINAPI Timer(LPVOID arg) 구현 | √ |  |
| DWORD WINAPI ProcessClient(LPVOID arg) 구현 | √ |  |
| 플레이어와 총알, 아이템의 충돌체크 구현 | √ |  |
| 서버 프로토콜 디자인 및 구현 | √ |  |
| DWORD WINAPI RecvKeyData() 구현 |  | √ |
| void RecvData() 구현 |  | √ |
| void RecvBulletInfo() 구현 | √ | √ |
| 총알 정보 관련 함수 구현 | √ | √ |
| 게임 결과 send함수 구현 |  | √ |
| 아이템 정보 send함수 구현 |  | √ |
| 클라이언트의 강제 종료 여부를 체크는 함수 구현 | √ |  |
| 로비 내 플레이어의 상태를 확인하는 함수 구현 | √ | √ |

|  |  |
| --- | --- |
| **김하윤** | **김연지** |

1. **개발일정(원본)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |
| Class Input  구현 | 기본적인  네트워크  환경 구현  (클라이언트) | Class Timer  구현 | 기본적인 네트워크 환경 미 구현 보완 | Timer 스레드 함수 구현 | RecvInputKey() 구현 | Class Scene,  Class SceneManager  구현 | RecvKeyData () 구현 |
|  | |  | |  | |  | |
|  |  | Class Object  구현 |  | Class Object  구현 |  | Class Player  구현 | Timer  UI띄우기 |
|  | |  | |  | |  | |
| Class Player  구현 |  | Player  보완 | RecvData()  구현 |  | Class Item  구현  (HP) |  | Class Item  구현  (탄창) |
|  | |  | |  | |  | |
| Bullet 구현 | | | | | | | |
|  | |  | |  | |  | |
| 총알과  플레이어  충돌 구현 | Class Item  구현  (수류탄) | 총알과  플레이어  충돌 구현 | Class Item  구현  (수류탄) | 총알과  플레이어  충돌 구현 | Class Item  구현  (수류탄) |  |  |
|  | |  | |  | |  | |
| 아이템과  플레이어  충돌 구현 | Item관련  Send, Recv  함수 구현 | 아이템과  플레이어  충돌 구현 | HP UI 구현 | 충돌 오류  체크 | 수류탄 UI 구현 | 충돌 오류  체크 |  |
|  | |  | |  | |  | |
|  | 탄창 수 UI  구현 |  | 총알 수 UI  구현 | 탄창 수  동기화 확인 |  | 총알 수  동기화 확인 |  |
|  | |  | |  | |  | |
| 게임결과 UI  구현 |  |  | 게임 결과 send, recv 함수 구현 | 강제종료  여부  체크 함수 |  |  |  |
|  | |  | |  | |  | |
| 로비구현 | 로비구현 | 로비구현 | 로비구현 | 로비구현 | 로비구현 |  |  |

|  |  |
| --- | --- |
| **김하윤** | **김연지** |

**수정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  | |
| ~~클라이언트 및 서버 프로토콜 디자인 및 구현~~ | ~~기본적인~~  ~~네트워크~~  ~~환경 구현~~ | ~~Class Timer~~  ~~구현~~ | ~~게임 UI,~~  ~~플레이어,~~  ~~아이템,~~  ~~맵 디자인~~ | Timer 스레드 함수 구현 | ~~게임 UI,~~  ~~플레이어,~~  ~~아이템,~~  ~~맵 디자인~~ | ~~Class Scene,~~  ~~Class SceneManager~~  ~~구현~~ | ~~게임 UI,~~  ~~플레이어,~~  ~~아이템,~~  ~~맵 디자인~~ |
|  | |  | |  | |  | |
| ~~Class Input~~  ~~구현~~ |  | ~~Class Object~~  ~~구현~~ | ~~Recv, Send~~  ~~KeyData ()~~  ~~구현~~ | ~~Class Object~~  ~~구현~~ |  | ~~Class Player~~  ~~구현~~ | ~~Timer~~  ~~UI띄우기~~ |
|  | |  | |  | |  | |
| ~~Class Player~~  ~~구현~~ |  | ~~Player~~  ~~보완~~ | ~~RecvData()~~  ~~구현~~ | ~~Player, Key~~  ~~동기화 확인~~ | ~~Class Item~~  ~~구현~~  ~~(HP)~~ | ~~Player, Key~~  ~~동기화 확인~~ | ~~Class Item~~  ~~구현~~  ~~(탄창)~~ |
|  | |  | |  | |  | |
| ~~Bullet 구현~~ | | | | | | | |
|  | |  | |  | |  | |
| ~~총알과~~  ~~플레이어~~  ~~충돌 구현~~ | ~~Class Item~~  ~~구현~~  ~~(수류탄)~~ | ~~총알과~~  ~~플레이어~~  ~~충돌 구현~~ | ~~Class Item~~  ~~구현~~  ~~(수류탄)~~ | ~~총알과~~  ~~플레이어~~  ~~충돌 구현~~ | ~~Class Item~~  ~~구현~~  ~~(수류탄)~~ |  |  |
|  | |  | |  | |  | |
| ~~아이템과~~  ~~플레이어~~  ~~충돌 구현~~ | ~~Item관련~~  ~~Send, Recv~~  ~~함수 구현~~ |  | ~~HP UI 구현~~ |  | ~~수류탄 UI 구현~~ |  |  |
|  | |  | |  | |  | |
|  | ~~탄창 수 UI~~  ~~구현~~ | ~~충돌 오류~~  ~~체크~~ | ~~총알 수 UI~~  ~~구현~~ | ~~탄창 수~~  ~~동기화 확인~~ |  | ~~총알 수~~  ~~동기화 확인~~ |  |
|  | |  | |  | |  | |
|  |  | ~~게임결과 UI~~  ~~구현~~ |  | ~~강제종료~~  ~~여부~~  ~~체크 함수~~ | ~~게임 결과~~  ~~send, recv~~  ~~함수 구현~~ |  |  |
|  | |  | |  | |  | |
| ~~로비구현~~ | ~~로비구현~~ | ~~로비구현~~ | ~~로비구현~~ | ~~로비구현~~ | ~~로비구현~~ |  |  |

1. **세부 개발 일정**

|  |  |  |
| --- | --- | --- |
| 11/5 | 김하윤 | 김연지 |
| 개발 목표 | 클라이언트 및 서버 프로토콜 디자인 및 구현 | 기본적인 네트워크 환경 구현 |
| 결과 | 기획한 프로토콜 구현 | 클라이언트가 서버에 접속하는 지 확인 |

|  |  |  |
| --- | --- | --- |
| 11/6 | 김하윤 | 김연지 |
| 개발 목표 | Class Timer 구현 | 게임 UI, 플레이어, 아이템, 맵 디자인 및 리소스 편집 |
| 결과 | 시험 | 시험 |

|  |  |  |
| --- | --- | --- |
| 11/7 | 김하윤 | 김연지 |
| 개발 목표 | Timer스레드 함수 구현 | 게임 UI, 플레이어, 아이템, 맵 디자인 및 리소스 편집 |
| 결과 | Class CTimer 서버 및 클라이언트에 구현 | 리소스 편집 |

|  |  |  |
| --- | --- | --- |
| 11/8 | 김하윤 | 김연지 |
| 개발 목표 | Class Scene, SceneManager 구현 | 게임 UI, 플레이어, 아이템, 맵 디자인 및 리소스 편집 |
| 결과 | Class Scene, SceneManager, SceneScript 구현 | 시험 |

|  |  |  |
| --- | --- | --- |
| 11/9 | 김하윤 | 김연지 |
| 개발 목표 | Class Input 구현 |  |
| 결과 | Class Input(클라이언트)구현 | 리소스 편집 |

|  |  |  |
| --- | --- | --- |
| 11/10 | 김하윤 | 김연지 |
| 개발 목표 | Class Object 구현 | Recv, Send KeyData 함수 구현 |
| 결과 | Class Object 구현,  Class Scene에서 Object들을 관리하게 끔 수정  클라에서 SendKeyData() 함수 구현 | KeyData를 주고 받기위한 관련 프로토콜 수정 및 추가  void RecvKeyAndDataUpdate(); 함수 구현(서버) |

|  |  |  |
| --- | --- | --- |
| 11/11 | 김하윤 | 김연지 |
| 개발 목표 | Class Object 구현 | Recv, Send KeyData 함수 구현 |
| 결과 | 휴식 | GameState  Send, recv 함수 구현 및 확인 |

|  |  |  |
| --- | --- | --- |
| 11/12 | 김하윤 | 김연지 |
| 개발 목표 | Class Player구현 | 게임 제한시간 UI 구현 |
| 결과 | Class Player 구현, Animation 기능 구현  키 입력 데이터 변경이 되는지 확인 및 수정  (send, recv KeyData() 함수 수정) | Player 리소스 수정 |

|  |  |  |
| --- | --- | --- |
| 11/13 | 김하윤 | 김연지 |
| 개발 목표 | Send, recv PlayerData() 함수 구현 |  |
| 결과 | Send, recv PlayerData() 함수 구현 |  |

|  |  |  |
| --- | --- | --- |
| 11/14 | 김하윤 | 김연지 |
| 개발 목표 | Class Player보완 | Recv Data() 구현 |
| 결과 | 총을 쏘는 모션을 할 때 애니메이션이 끝까지 실행이 안되는 버그를 발견하여 해결 | Send, recv PlayerData를 총알과 플레이어 데이터를 다 들고 있는 DATA구조체의 send, recv로 수정 |

|  |  |  |
| --- | --- | --- |
| 11/15 | 김하윤 | 김연지 |
| 개발 목표 | Player, Key 동기화 확인 | Class Item(HP) 구현 |
| 결과 | Class Network 구현으로 Network::Input()에서는 KeyData를 send하고 Network::Update()에서는 GameState와 Data를 recv함  PlayerData프로토콜에 state(애니메이션 상태)를 추가 | Player 리소스가 수정된 애니메이션과 맞지 않아 리소스 수정 |

|  |  |  |
| --- | --- | --- |
| 11/16 | 김하윤 | 김연지 |
| 개발 목표 | Player, Key 동기화 확인 | Class Item(탄창) 구현 |
| 결과 | 총알과 수류탄 프로토콜 수정 및 보완 | G-star 견학 |

|  |  |  |
| --- | --- | --- |
| 11/17 | 김하윤 | 김연지 |
| 개발 목표 | Bullet 구현 | Bullet 구현 |
| 결과 |  | 클라이언트에서 Bullet 구현 |

|  |  |  |
| --- | --- | --- |
| 11/18 | 김하윤 | 김연지 |
| 개발 목표 | Bullet 구현 | Bullet 구현 |
| 결과 | 클라에서 구현된 Bullet을 토대로 서버에서 구현 | Class Item 구현 |

|  |  |  |
| --- | --- | --- |
| 11/19 | 김하윤 | 김연지 |
| 개발 목표 | Bullet 구현 | Bullet 구현 |
| 결과 | Bullet의 데이터가 서버에서 클라이언트로 제대로 보내는지 확인 및 클라이언트가 Bullet정보를 받아 그 위치에 잘 그려지는지 확인  PlayerInfo내 데이터 동기화 확인 | |

|  |  |  |
| --- | --- | --- |
| 11/20 | 김하윤 | 김연지 |
| 개발 목표 |  |  |
| 결과 | 서버와 클라간의 통신을 초당 30번으로 수정  (서버) Class Timer 사용 X | 인공지능 시험 준비 |

|  |  |  |
| --- | --- | --- |
| 11/21 | 김하윤 | 김연지 |
| 개발 목표 | 총알과 플레이어 충돌 구현 | 수류탄 구현 |
| 결과 | Collision이라는 함수를 만들어 bool값으로 충돌이 됐는지 안됐는지 체크 | Class Item 구현 및 드랍 시키기 |

|  |  |  |
| --- | --- | --- |
| 11/22 | 김하윤 | 김연지 |
| 개발 목표 | 총알과 플레이어 충돌 구현 | Class Item 구현 (수류탄) |
| 결과 | Collision이라는 함수를 만들어 bool값으로 충돌이 됐는지 안됐는지 체크 | Class Item 기반으로 Class Healing 구현 |

|  |  |  |
| --- | --- | --- |
| 11/23 | 김하윤 | 김연지 |
| 개발 목표 | ~~총알과 플레이어 충돌 구현~~  (서버)스레드를 방 하나당 하나씩 생성하는 것으로 변경 | Class Item 구현 (수류탄) |
| 결과 | g\_socket[]로 접속한 클라이언트 소켓을 관리하여 임계영역으로 g\_socket[]에 소켓정보를 담을 때 마다 동기화를 시킴 | 아이템 drop을 위한 Class Heli 구조와 아이템 간의 flow 및 세부내용 재조정 |

|  |  |  |
| --- | --- | --- |
| 11/24 | 김하윤 | 김연지 |
| 개발 목표 | 제한시간 Timer UI 띄우기 |  |
| 결과 |  |  |

|  |  |  |
| --- | --- | --- |
| 11/25 | 김하윤 | 김연지 |
| 개발 목표 | 아이템과 플레이어 충돌 | Item 관련 Send, Recv 함수 구현 |
| 결과 |  | 아픔 |

|  |  |  |
| --- | --- | --- |
| 11/26 | 김하윤 | 김연지 |
| 개발 목표 |  | HP UI 구현 |
| 결과 | 수류탄 구현 및 데이터 송수신 | 다른 수업 과제 |

|  |  |  |
| --- | --- | --- |
| 11/27 | 김하윤 | 김연지 |
| 개발 목표 |  | 수류탄 UI 구현 |
| 결과 | 수류탄 버그 확인(수정 X) | 다른 수업 과제 마무리 |

|  |  |  |
| --- | --- | --- |
| 11/28 | 김하윤 | 김연지 |
| 개발 목표 |  |  |
| 결과 |  |  |

|  |  |  |
| --- | --- | --- |
| 11/29 | 김하윤 | 김연지 |
| 개발 목표 |  | 탄창 수 UI 구현 |
| 결과 |  | 서버로 item drop try 1  서버에 기존 timer 외의 아이템 드랍 시간 (30초 간격)을 세는 timer 추가하여 bool drop을 한 번만 전송하여 클라이언트에서 이를 받는다. 근데 송수신이 제대로 안 됨. |

|  |  |  |
| --- | --- | --- |
| 11/30 | 김하윤 | 김연지 |
| 개발 목표 | 충돌 오류 체크 | 총알 수 UI 구현 |
| 결과 | 플레이어와 총알간 충돌 오류는 없음 | 서버로 Heli 띄우기 try 2  서버에서 보내는 timer에 따라서 bool drop을 전송 시켜서 이를 클라이언트에서 받는다. 근데 송수신이 제대로 안 됨. |

|  |  |  |
| --- | --- | --- |
| 12/1 | 김하윤 | 김연지 |
| 개발 목표 | 탄창 수 동기화 확인 |  |
| 결과 | (UI미구현)탄창 수 동기화 확인 | Class Heli 구현.  일단 헬리콥터를 띄워서 헬리콥터가 중앙에 도착하면 아이템을 드롭하는 일련의 과정이 진행되게 만들었다. |

|  |  |  |
| --- | --- | --- |
| 12/2 | 김하윤 | 김연지 |
| 개발 목표 | 총알 수 동기화 확인 |  |
| 결과 | (UI미구현)총알 수 동기화 확인 |  |

|  |  |  |
| --- | --- | --- |
| 12/3 | 김하윤 | 김연지 |
| 개발 목표 |  |  |
| 결과 | 게임 승패 결과 UI 출력  승패 결과 send, recv함수 구현  <승패여부 조건>   * + - 1. 제한시간 전에 둘 중 hp가 0이되는 경우 -> hp가 0인 플레이어가 패       2. 제한시간이 끝난 후 -> 둘 중 hp가 높은 경우 승 | 헬리콥터&아이템 동작 클라이언트에서 구현해 놓은 것을 서버에서 보내준 Timer에 맞춰 지정한 시간에 클라에서 띄우고, 아이템 드랍 까지 연속 과정 잘 된다. |

|  |  |  |
| --- | --- | --- |
| 12/4 | 김하윤 | 김연지 |
| 개발 목표 | 게임 결과 UI | 게임 결과 send, recv 함수 구현 |
| 결과 | 시험공부 | Lobby리소스 편집 |

|  |  |  |
| --- | --- | --- |
| 12/5 | 김하윤 | 김연지 |
| 개발 목표 | 강제종료 여부 체크 함수 |  |
| 결과 | 수류탄 수정 및 충돌 처리  Lobby화면 구현  아이템 수정 및 충돌 처리  헬기 구현 | Lobby화면에서 자신이 1P인지 2P인지 알기 위해 캐릭터창에 테두리 추가  Ingame화면에서 자신의 캐릭터를 명확하게 표시하기위해 캐릭터 추가 |