**NETWORK GAME PROGRAMMING**

**TERM PROJECT REPORT**

게임공학과 2014180044 허신영

게임공학과 2015182030 이동희

* **목차**

1. **애플리케이션 기획**
2. **게임 설명**
3. **게임 구성 요소**
4. **High Level Design**
5. **Low Level Design**
6. **Client 패킷 구조체 / 매크로&전역변수**
7. **Server 패킷 구조체 / 매크로&전역변수**
8. **Client&Server 공통 매크로**
9. **Client 함수**
10. **Server함수**
11. **프로토콜**
12. **개발 환경 및 세부 기획**
13. **개발 환경**
14. **역할 분담**
15. **개발 일정**
16. **애플리케이션 기획 - ① 게임 설명**

****

* **[ 게임 이름 ]**
* **테일즈위버 (TalseWeaver) / 2D RPG**
* **[ 게임 내용 ]**
* **플레이어들이 협동하여 몬스터와 보스를 잡고, 던전을 클리어한다.**
* **[ 게임 설정 ]**
* **플레이어는 레벨 업을 통해 공격력을 높일 수 있다**
* **플레이어는 상점에서 아이템을 통해 공격력, HP, MP, SP를 높일 수 있다.**
* **몬스터는 플레이어의 공격을 받으면 HP가 소모되고, 0이되면 사망한다.**
* **두 명의 플레이어가 모두 몬스터의 공격을 받아 HP가 0이되면,**

**게임을 처음부터 다시 시작한다.**

* **최종 보스를 잡으면 게임이 클리어된다.**
* **[ 게임 조작 ]**
* **이동 : 키보드 W / A / S / D**

****

* **스킬 : 키보드 F1(Moon) / F2(Skull) / F3(MultiHit)**

****

* **물약 : 키보드 F5(HP) / F6(MP) / F7(SP)**
* **공격 : 기본공격 SPACE / 스킬공격 마우스 R\_BUTTON**
* **종료 : 키보드 ESC**

1. **애플리케이션 기획 - ② 게임 구성 요소**

* **[ 플레이어 ]**
* **플레이어는 HP, MP, SP 세 가지 상태 자원을 가진다. HP는 몬스터의 공격을 받을 때 소모, MP는 스킬을 사용할 때 소모, SP는 달리기 상태일 때 이동 시 소모된다. MP가 스킬 소모 값 보다 작으면 스킬을 사용할 수 없다. SP가 0이되면 플레이어는 자동으로 걷기 상태가 된다.**
* **플레이어는 일정 시간 안에 공격 또는 스킬을 사용하면 콤보 공격을 통한 빠른 공격이 가능해진다.**
* **[ 몬스터 ]**



* **~~Jelly : 비선공 타입 몬스터. 먼저 공격받으면 플레이어를 공격한다.~~**
* **Cow : 선공 타입 몬스터. 플레이어와 5만큼 거리가 가까워지면 공격한다.**
* **~~Ninja : 중간 보스 몬스터.~~**

**~~HP가 60% 이하로 내려가면 플레이어 뒤로 이동하여 공격한다.~~**

* **~~Boris : 최종 보스 몬스터.~~**

**~~HP가 70% 내려가면 스킬 공격을, 40%이하로 내려가면 순간이동을 하며 공격한다.~~**

* **[ 게임 씬 ]**



* **MENU : 게임 시작 버튼을 통해서 게임을 시작할 수 있다.**
* **TOWN : 게임 시작 시 시작하는 마을이다.**
* **STORE : 플레이어가 NPC를 통해 아이템을 구입할 수 있는 상점이다.**
* **FIELD : Jelly 몬스터가 있는 던전이다.**
* **DUNGEON : Cow와 중간 보스인Ninja 몬스터가 있는 던전이다.**
* **BOSS : 최종 보스인 Boris 몬스터가 있는 던전이다.**
* **[ 게임 진행]**

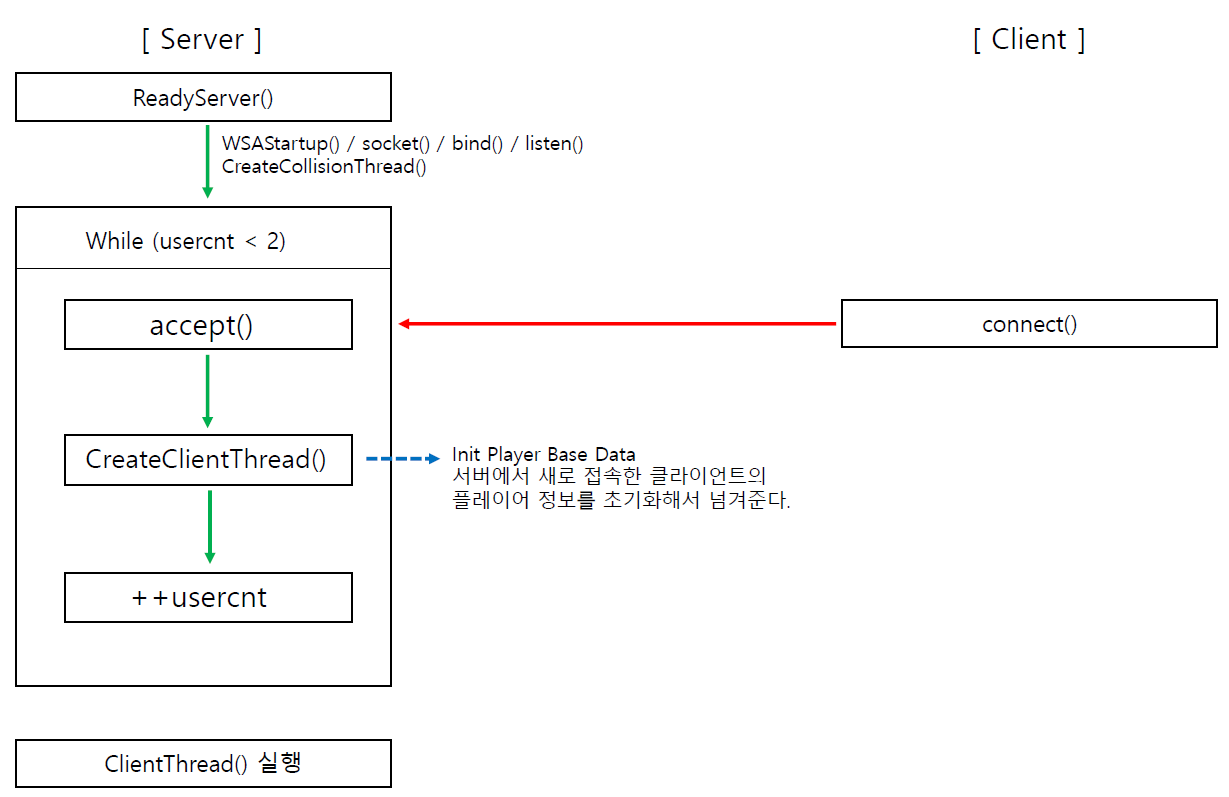
1. **MENU에서 ‘게임 시작’ 버튼을 통해서 게임을 시작한다.**
2. **TOWN씬에서 게임을 시작하며, TOWN에서는 STORE, FIELD, DUNGEON으로 이동할 수 있다.**
3. **~~두 명의 플레이어가 모두 접속해야 던전으로 이동할 수 있다.~~**

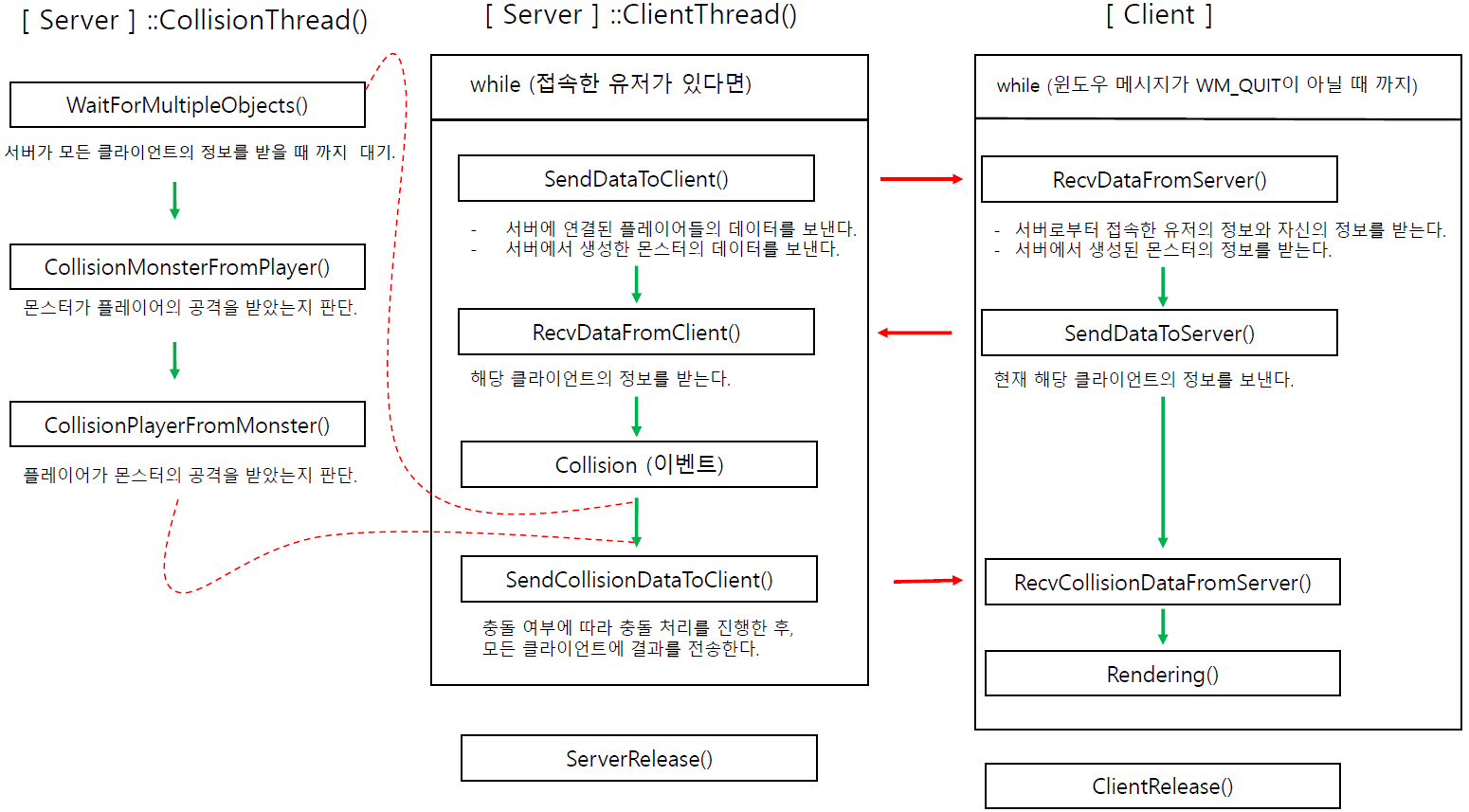
**한 캐릭터만 있어도 던전으로 이동할 수 있다.**

1. **STORE에서 플레이어는 장비를 사고팔 수 있다.**
2. **~~TOWN -> DUNGEON -> BOSS 씬으로 이동하며 몬스터와 보스를 잡고 게임을 클리어한다.~~**

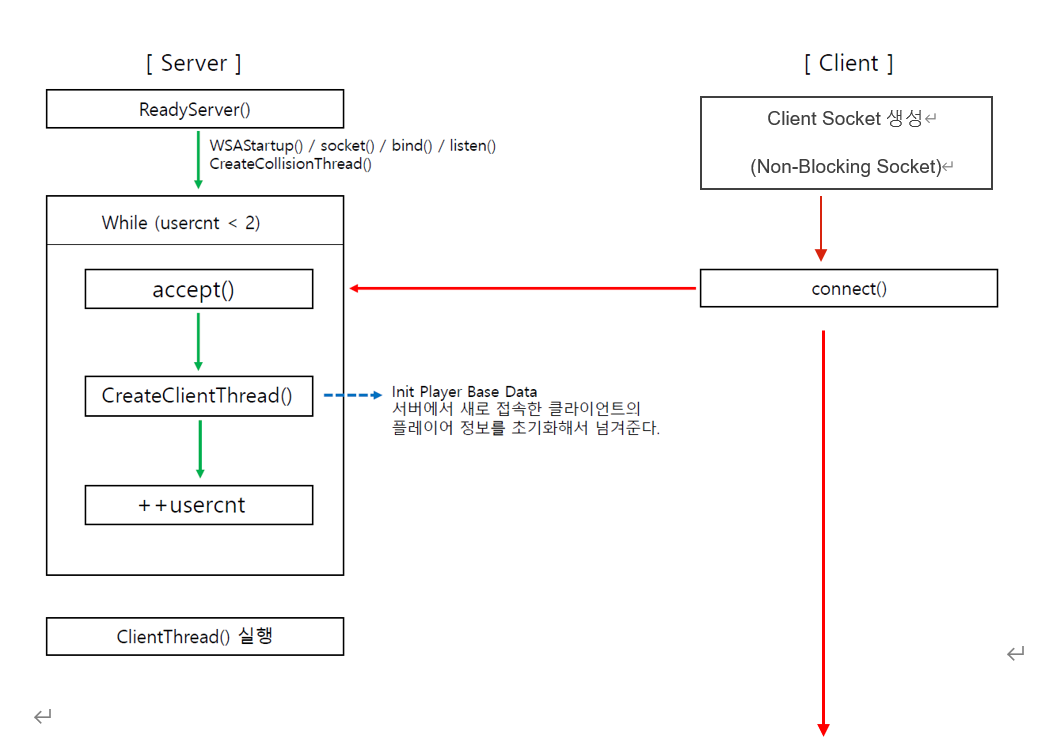
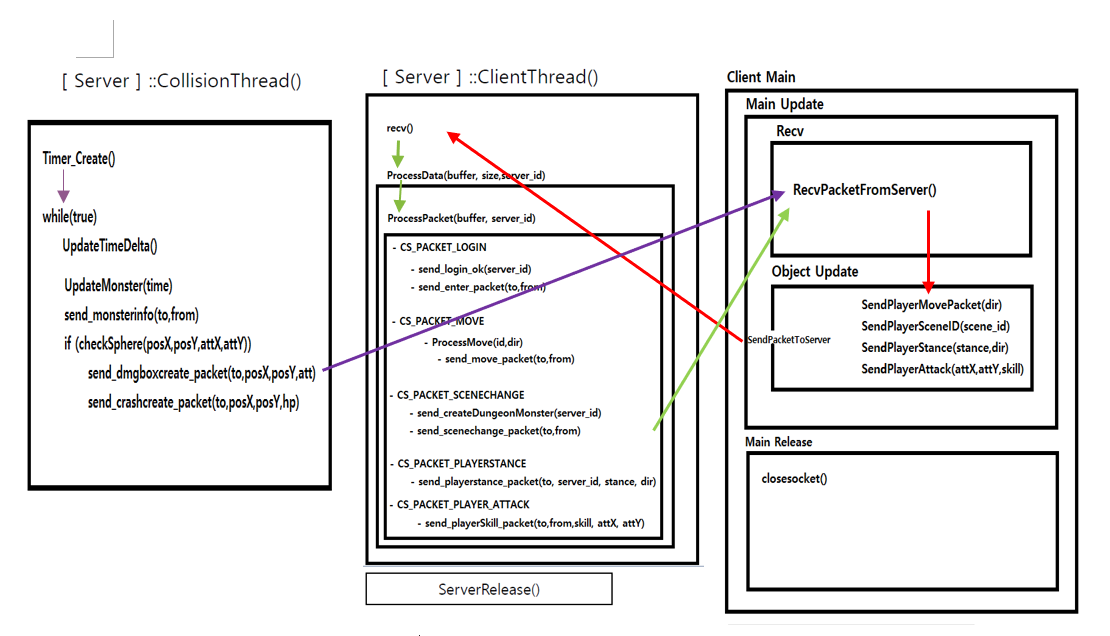
**DUNGEON에서 Cow몬스터를 잡고 게임을 클리어한다.**

1. **High Level Design [ 수정 전 ]**





**[ 수정 후 ]**



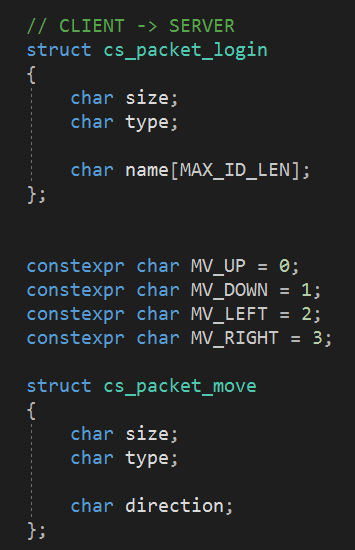
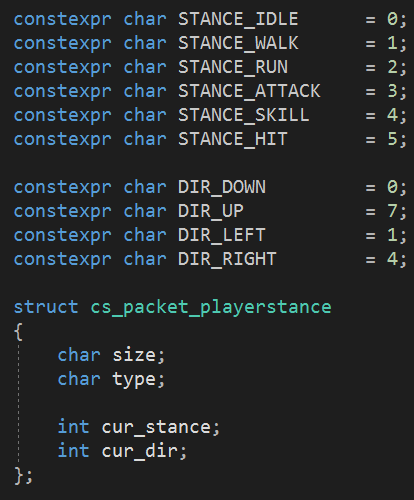
1. **Low Level Design - ① Client 패킷 구조체 / 매크로&전역변수**
2. **클라이언트 전용 패킷**

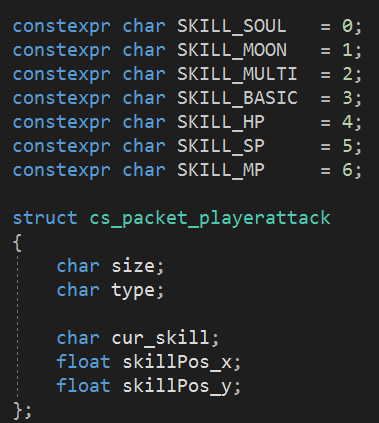
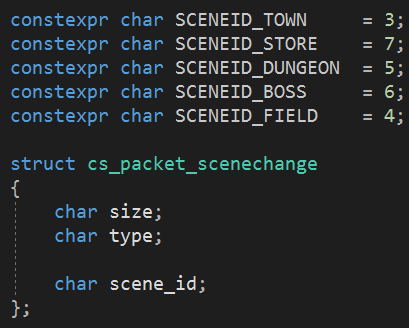
|  |
| --- |
| **~~클라이언트가 서버로부터 받는 패킷입니다.~~** |
| ~~struct cl\_recv~~  ~~{~~  ~~bool is\_connected;~~  ~~float cx;~~  ~~float cy;~~  ~~float x;~~  ~~float y;~~  ~~wstring name;~~  ~~int level;~~  ~~int hp;~~  ~~int mp;~~  ~~int sp;~~  ~~int att;~~  ~~float speed;~~  ~~int exp;~~  ~~STANCE stance;~~  ~~SKILL\_STATE skill;~~  ~~DIR dir;~~  ~~};~~ |

|  |
| --- |
| **~~클라이언트가 서버로 보내는 패킷입니다.~~** |
| ~~struct cl\_send~~  ~~{~~  ~~bool is\_connected;~~  ~~char input;~~  ~~};~~ |

|  |
| --- |
| **~~플레이어와 몬스터의 애니메이션을 재생하기위한 구조체입니다.~~** |
| ~~struct tagFrame~~  ~~{~~  ~~int iFrameStart;~~  ~~int iFrameEnd;~~  ~~int iScene;~~  ~~DWORD dwOldTime;~~  ~~DWORD dwFrameSpd;~~  ~~};~~ |

|  |
| --- |
| **~~충돌박스를 구성하는 구조체입니다.~~** |
| ~~struct tagCollInfo~~  ~~{~~  ~~float fCX;~~  ~~float fCY;~~  ~~float fX;~~  ~~float fY;}~~  ~~;~~ |





유저 씬 전환 패킷.

플레이어 스킬 이팩트 패킷.

cs\_packet\_login : 유저 접속 시, 서버로 ID를 보낸다.

cs\_packet\_move : 플레이어 캐릭터 움직임을 서버로 보낸다.

플레이어 애니메이션 관리용 패킷.

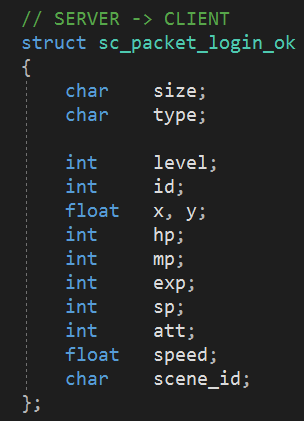
**3. Low Level Design - ② Server 패킷 구조체 / 매크로&전역변수**

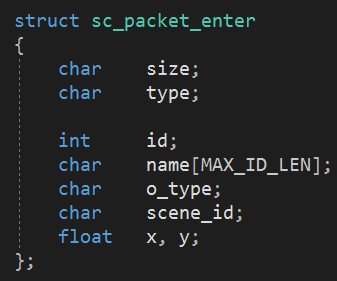
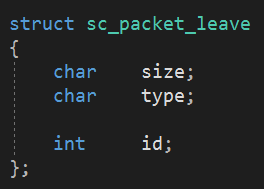
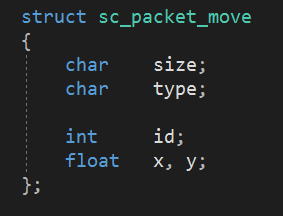
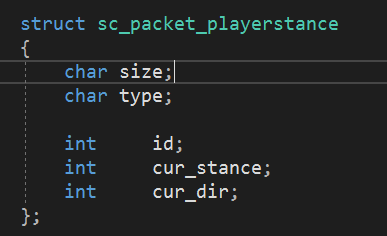
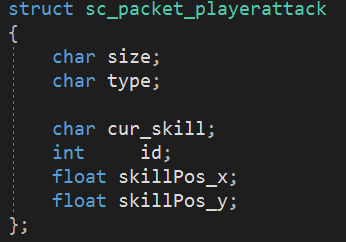
**1) 서버 전용 패킷**

|  |
| --- |
| **~~서버가 클라이언트로부터 받는 패킷입니다.~~** |
| ~~struct sv\_recv~~  ~~{~~  ~~bool is\_connected;~~  ~~char input;~~  ~~};~~ |

|  |
| --- |
| **~~서버가 클라이언트로 보내는 패킷입니다.~~** |
| ~~struct sv\_send~~  ~~{~~  ~~bool is\_connected;~~  ~~float cx;~~  ~~float cy;~~  ~~float x;~~  ~~float y;~~  ~~wstring name;~~  ~~int level;~~  ~~int hp;~~  ~~int mp;~~  ~~int sp;~~  ~~int att;~~  ~~float speed;~~  ~~int exp;~~  ~~STANCE stance;~~  ~~SKILL\_STATE skill;~~  ~~DIR dir;~~  ~~};~~ |

|  |
| --- |
| **~~서버에서 관리하는 클라이언트 객체 구조체~~** |
| ~~struct socket\_info~~  ~~{~~  ~~bool is\_connected;~~  ~~int sock\_number;~~  ~~SOCKET sock;~~  ~~sv\_recv recv\_packet;~~  ~~sv\_send send\_packet;~~  ~~};~~ |





플레이어 스킬 이팩트 생성 패킷.

플레이어 애니메이션 상태 패킷.

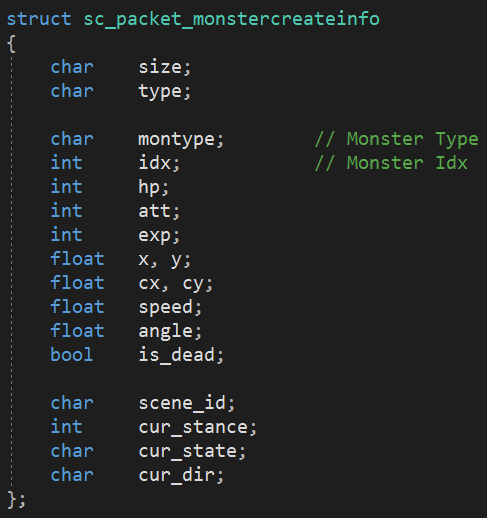
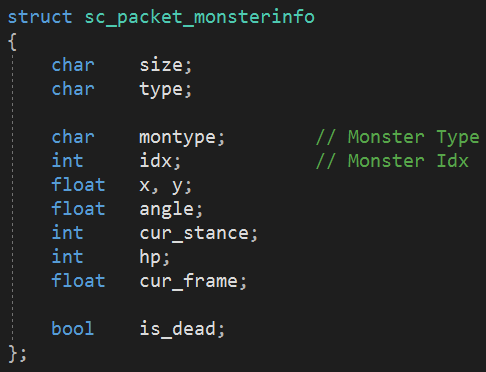
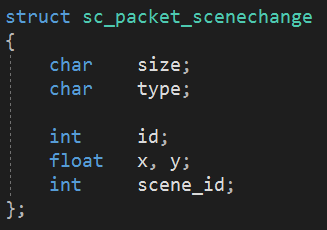
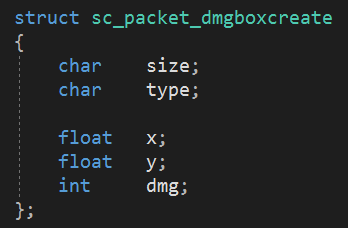
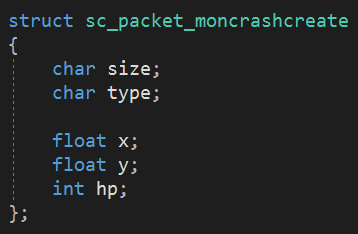
플레이어 이동 패킷.

유저 종료 패킷.

NPC나 유저가 씬으로 접속하는 알림 패킷.

접속한 유저를 확인하고

유저 정보를 클라이언트로 전달하는 패킷.



플레이어 피격 이팩트 생성 패킷.

데미지 폰트 이팩트 생성 패킷.

플레이어 씬 전환 알림 패킷.

서버에 존재하는 몬스터 정보 패킷.

몬스터 생성 정보 패킷.

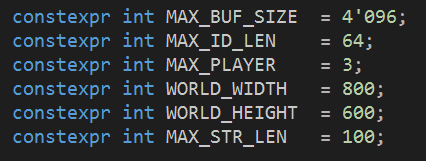
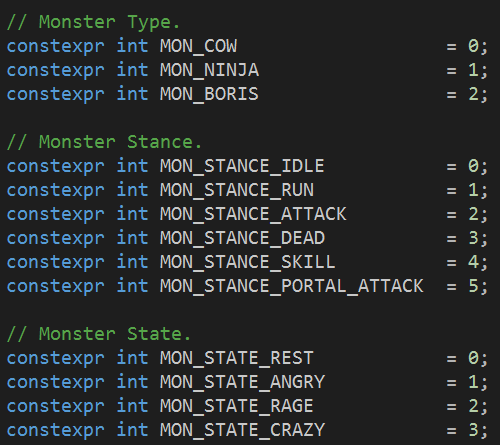
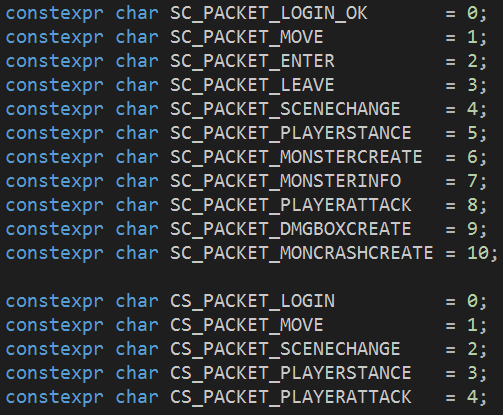
**3. Low Level Design - ③ Client&Server 공통 매크로**

**~~enum STANCE { IDLE, WALK, RUN, ATTACK, SKILL, HIT, END };~~**

**~~enum SKILL\_STATE { SOUL, MOON, MULTI, BASIC, SKILL\_END };~~**

**~~enum DIR { DOWN, LEFT, LEFT\_DOWN, LEFT\_UP,~~**

**~~RIGHT, RIGHT\_DOWN, RIGHT\_UP, UP };~~**



몬스터 타입과 상태정보.

서버 -> 클라이언트 패킷 타입.

클라이언트 -> 서버 패킷 타입.

버퍼 사이즈 및 윈도우 크기.

**3. Low Level Design - ④ Client 함수**

* **~~void ReadyServer()~~**
* **~~소켓 생성 / 서버에 접속 준비를 수행하는 함수이다.~~**
* **~~int RecvDataFromServer()~~**
* **~~서버로부터 접속한 유저의 정보와 자신의 정보를 받는다.~~**
* **~~int SendDataToServer()~~**
* **~~현재 해당 클라이언트의 입력 정보를 보낸다.~~**
* **~~Int RecvCollisionDataFromServer()~~**
* **~~서버로부터 충돌의 결과와 관련된 데이터 및 상태정보를 받는다.~~**

**int RecvPacketFromServer()**

* **서버로부터 패킷을 받는다.**

**void ProcessData(char\* net\_buf, size\_t io\_byte)**

* **TCP의 순차적인 데이터 전송 특성을 이용하여 하나의 패킷을 완성할 때 까지 패킷을 재조립한다.**

**void ProcessPacket(char\* ptr)**

* **서버로부터 받은 패킷 타입에 따라 이벤트를 처리한다.**
* **void Rendering()**
* **씬을 이루는 오브젝트들을 그린다.**
* **Void ClientRelease()**
* **동적할당 오브젝트 소멸 및 소켓반환.**

**void SendPlayerMovePacket(unsigned char dir)**

* **플레이어 움직임 정보를 서버로 보낸다.**

**void SendPlayerSceneID(unsigned char scene\_id)**

* **플레이어의 씬 정보를 서버로 보낸다.**

**void SendPlayerStance(int stance, int dir)**

* **플레이어의 애니메이션 상태 정보를 서버로 보낸다.**

**void SendPlayerAttack(float attX, float attY, unsigned char skill)**

* **플레이어의 스킬 상태 정보를 서버로 보낸다.**

**void SendPacketToServer(void\* packet)**

* **서버로 패킷을 보낸다.**

**3. Low Level Design - ⑤ Server함수**

* **void ReadyServer()**
* **윈속 초기화 / 소켓 생성 / bind() 및 listen()**
* **충돌 스레드를 생성한다.**
* **~~void CreateClientThread()~~**
* **DWORD WINAPI CreateClientThread()**
* **접속한 클라이언트의 통신을 담당하기 위한 스레드를 생성한다.**
* **~~void InitPlayerBaseData()~~**
* **~~서버에서 새로 접속한 클라이언트의 Player 정보를 초기화해서 넘겨준다.~~**
* **~~int SendDataToClient()~~**
* **~~서버에 연결된 플레이어들의 데이터를 보낸다.~~**

**~~서버에서 생성한 몬스터의 데이터를 보낸다.~~**

* **~~int RecvDataFromClient()~~**
* **~~해당 클라이언트로부터 정보를 받는다.~~**
* **~~int SendCollisionDataToClient()~~**
* **~~충돌 여부에 따라 충돌 처리를 진행한 후,~~**

**~~모든 클라이언트에게 결과를 전송한다.~~**

* **~~void ServerRelease()~~**
* **~~소켓 반환 / 윈속 해제~~**
* **~~void CollisionMonsterFromPlayer()~~**
* **~~몬스터가 플레이어의 공격을 받았는지 충돌검사를 수행하는 함수이다.~~**
* **~~void CollisionPlayerFromMonster()~~**
* **~~플레이어가 몬스터의 공격을 받았는지 충돌검사를 수행하는 함수이다.~~**

**DWORD WINAPI CollisionThread(LPVOID arg)**

* **몬스터 Update와 충돌을 관리하는 스레드.**

**int ConnectNewClient(SOCKET ns)**

* **새로 접속한 클라이언트를 연결하는 함수.**

**void Disconnect\_Client (int id)**

* **클라이언트의 접속을 해제하는 함수.**

**void ProcessPacket(char\* ptr, int server\_id)**

* **TCP의 순차적인 데이터 전송 특성을 이용하여 하나의 패킷을 완성할 때 까지 패킷을 재조립한다.**

**void ProcessData(char\* net\_buf, size\_t io\_byte, int server\_id)**

* **클라이언트로 받은 패킷 타입에 따라 이벤트를 처리한다.**

**void send\_packet(int id, void\* p)**

* **클라이언트로 패킷을 보낸다.**

**void send\_login\_ok(int id)**

* **클라이언트의 아이디를 확인하고, 아이디에 맞는 해당 정보를 보낸다.**

**void send\_enter\_packet(int to, int id)**

* **게임 씬에 클라이언트를 입장시키는 함수.**

**void send\_move\_packet(int to, int id)**

* **플레이어의 움직임을 클라이언트로 보내주는 함수.**

**void send\_leave\_packet(int to, int id)**

* **서버 연결 해제를 클라이언트로 보내주는 함수.**

**void send\_playerstance\_packet(int to, int id)**

* **플레이어의 애니메이션 상태정보를 보내주는 함수.**

**void send\_scenechange\_packet(int to, int id)**

* **플레이어의 씬 전환 정보를 클라이언트로 보내주는 함수.**

**void send\_createDungeonMonster(int to)**

* **던전 입장시, 몬스터의 생성 정보를 클라이언트로 보내주는 함수.**

**void send\_monsterinfo(int to, CMonster\* src)**

* **던전에 존재하는 몬스터의 정보를 보내주는 함수.**

**void send\_playerSkill\_packet(int to, int id, char skillname, float attX, float attY)**

* **플레이어의 스킬 상태 정보를 보내주는 함수.**

**void send\_dmgboxcreate\_packet(int id, float x, float y, int dmg)**

* **데미지 이팩트 생성 정보를 클라이언트로 보내주는 함수.**

**void send\_moncrashcreate\_packet(int id, float x, float y, int hp)**

* **플레이어 피격 정보를 보내주는 함수.**

**void ProcessMove(int id, char dir)**

* **플레이어의 이동 정보를 처리하는 함수.**

**bool CheckSphere(const float& dst\_x, const float& dst\_y, const float& dst\_radius, const float& src\_x, const float& src\_y, const float& src\_radius)**

* **충돌 여부를 판단하는 함수.**

**void SaveLoginData(const int& id)**

* **클라이언트의 접속 시간 데이터를 저장하는 함수.**

**void SaveLeaveData(const int& id)**

* **클라이언트의 접속 해제 시간 데이터를 저장하는 함수.**

1. **개발 환경 및 세부 기획 - ① 개발 환경**
2. **컴파일러 : VisualStudio 2019**
3. **개발언어 : C++**
4. **라이브러리 : Win32 API / WinSock2 / fMod**
5. **동작 OS : Windows 10**

**4. 개발 환경 및 세부 기획 - ② 역할 분담**

1. **허신영**

* **~~[서버] CollisionMonsterFromPlayer() 구현.~~**
* **~~[서버] CollisionPlayerFroMonster() 구현.~~**
* **~~[서버] SendCollisionDataToClient() 구현.~~**
* **~~[클라이언트] RecvCollisionDataFromServer() 구현~~**

**[ Client ]**

* **void SendPlayerMovePacket(unsigned char dir)**
* **void SendPlayerSceneID(unsigned char scene\_id)**
* **void SendPlayerStance(int stance, int dir)**

**[ Server ]**

* **DWORD WINAPI ClientThread(LPVOID arg);**
* **void send\_playerstance\_packet(int to, int id, int stance, int dir)**
* **void send\_scenechange\_packet(int to, int id)**
* **void send\_createDungeonMonster(int to**
* **void send\_monsterinfo(int to, CMonster\* src)**
* **void send\_playerSkill\_packet(int to, int id, char skillname, float attX, attY)**
* **void send\_dmgboxcreate\_packet(int id, float x, float y, int dmg);**
* **void send\_moncrashcreate\_packet(int id, float x, float y, int hp);**

1. **이동희**

* **~~[서버] SendDataToClient() 구현.~~**
* **~~[서버] RecvDataFromClient() 구현.~~**
* **~~[클라이언트] RecvDataFroServer() 구현.~~**
* **~~[클라이언트] SendDataToServer() 구현.~~**

**[ Client ]**

* **int RecvPacketFromServer()**
* **void ProcessData(char\* net\_buf, size\_t io\_byte)**
* **void ProcessPacket(char\* ptr)**
* **void SendPacketToServer(void\* packet)**

**[ Server ]**

* **DWORD WINAPI ClientThread(LPVOID arg)**
* **int ConnectNewClient(SOCKET ns)**
* **void Disconnect\_Client(int id)**
* **void ProcessPacket(char\* ptr, int server\_id)**
* **void ProcessData(char\* net\_buf, size\_t io\_byte, int server\_id)**
* **void send\_packet(int id, void\* p)**
* **void send\_login\_ok(int id**
* **void send\_enter\_packet(int to, int id);**
* **void send\_move\_packet(int to, int id);**
* **void send\_leave\_packet(int to, int id);**

**4. 개발 환경 및 세부 기획 - ③ 개발 일정**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **월** | **화** | **수** | **목** | **금** | **토** | **일** | |
|  |  |  |  |  |  |  | **1** | |
| **이동희** |  |  |  |  |  |  | 기획서 작성 마무리 | |
| **허신영** |
|  | **2** | **3** | **4** | **5** | **6** | **7** | **8** | |
| **이동희** | 프로젝트  기획서 검토 및 수정 | 프로젝트  기획서 피드백 | 게임 리소스 수집 | 클라이언트 및 서버 프레임워크 설계 및 제작. | | | | |
| **허신영** |
|  | **9** | **10** | **11** | **12** | **13** | **14** | **15** | |
| **이동희** | 클라이언트 및 서버 프레임워크 설계 및 제작. | 1주차 프로젝트 검토 및 회의 | SendDataToClient() 구현. | RecvDataFromClient 구현.  (데이터 이동 확인.) | 서버에서 Player 캐릭터 이동 및 공격하기. | | | |
| **허신영** | 서버 멀티 스레드 환경 구현. | | 클라이언트 타일 충돌 구현 및 서버 충돌 연구. | | | |
|  | **16** | **17** | **18** | **19** | **20** | **21** | **22** | |
| **이동희** |  | 2주차  프로젝트 검토 및 회의 | RecvDataFromServer() 및SendDataToServer() 구현. (전투) | | | 서버 충돌 스레드에 대한 동기화 연구. | | |
| **허신영** | CollisionMonsterFromPlayer()와CollisionPlayerFroMonster()을 통해 충돌 판단. | | |  | |  |
|  | **23** | **24** | **25** | **26** | **27** | **28** | **29** | |
| **이동희** |  | 3주차  프로젝트 검토 및 회의 | 전투에 따른 몬스터 렌더링 관리 | | | 전투 이펙트 관리. | | |
| **허신영** | SendCollisionDataToClient()와 RecvCollisionDataFromServer()를 통해 충돌 이벤트 구현 | | | 접속한 클라이언트들에게 모두 데미지 폰트 생성. | | |
|  | **30** | **1** |  |  |  |  |  | |
| **이동희** |  | 4주차  프로젝트 검토 및 회의 |  |  |  |  |  | |
| **허신영** |

**[ 1주차 (11.03 ~11.09) PROGRESS REPORT ]**

|  |  |
| --- | --- |
| 허 신 영 | 이 동 희 |
| - Git 저장소 생성.  - [Server] 기본 프레임워크 구조 생성.  1Client – 1Server Model  - [Client] Keybord Input 함수화 하여 재정립. | - [Server] 프로토콜 정의  - [Client] Server와 connect()  - [Client] 추가 리소스 수집. |

**[ 2주차 (11.10 ~11.16) PROGRESS REPORT ]**

|  |  |
| --- | --- |
| 허 신 영 | 이 동 희 |
| [ Server ] Protocol 세분화 및 재정립  [ Client ] 선택한 윈도우만 활성화하는 작업  [ Client] 클라이언트 타일 충돌  [ Client ] 접속한 유저 클래스 생성.  [ Server ] Move Packet을 통한 통신. | [ Client ] Non-Blocking Socket으로 변환.  [ Client ] 패킷 재조립 구현.  [ Server ] 접속 유저 관리 컨테이너 생성  [ Server ] Connert / Enter / Leave 패킷 생성. |

**[ 3주차 (11.17 ~11.23) PROGRESS REPORT ]**

|  |  |
| --- | --- |
| 허 신 영 | 이 동 희 |
| [Client] 씬 아이디가 다른 유저의 객체 렌더링 Off.  [Client] 서버로부터 몬스터의 정보를 받아 던전 씬에서 몬스터 생성. | [ Server ] 몬스터 객체 정보 생성.  [ Server / Client ]  다른 유저의 애니메이션 상태 변경과 애니메이션 재생 구현. |

**[ 4주차 (11.24 ~12.01) PROGRESS REPORT ]**

|  |  |
| --- | --- |
| 허 신 영 | 이 동 희 |
| [ Server ]  Player 공격 – Monster 충돌처리 완료. | [ Server / Client ]  Player 스킬 이팩트 출력 완료. |

**[ 5주차 (12.2 ~12.08) PROGRESS REPORT ]**

|  |  |
| --- | --- |
| 허 신 영 | 이 동 희 |
| [ Server ]  Login/Logout 데이터 파일 저장. | [ Server ]  버그 수정 및 임계영역 관리. |