2024 네트워크 게임 프로그래밍  
추진 계획서

2024 Network Game Programming

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **제출일** | 2024-12-05 | **전공** | 게임공학과 | 게임공학과 | 게임공학과 |
| **과목** | 네트워크 게임 프로그래밍 | **학번** | 2022184015 | 2022180014 | 2022180029 |
| **담당교수님** | 김재경 교수님 | **이름** | 김해님 | 박민지 | 이세영 |

**Contents  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Ⅰ. Aplication Design 03p**

**Ⅱ. High Level Design 08p**

**Ⅲ. Low Level Design 11p**

**Ⅳ. Roles 32p**

**Ⅴ. Schedule 33p**

**Ⅵ. Self Check List 36p**

**--------------------------Ⅰ. Aplication Design--------------------------**

**1. 게임 개요**

• 게임 제목 : Kirby! Don’t overeating!

• 게임 장르 : 횡스크롤 탄막 슈팅 게임

• 게임 소개 : 팀원 중 박민지, 이세영이 윈도우 프로그래밍 과목의 프로젝트로 만든 게임이다. 원래는 1인용 탄막 슈팅 게임으로, 주인공 커비가 적을 처치하고 탄막을 피하는 것이 기본 구조이다. 네트워크 기능을 추가하여 2인용 협동 모드로 확장하였다.

**2. 게임 시스템**

**로그인 기능**  
로그인 시스템을 통해 새로운 아이디로 접속하면 새로운 계정이 생성되며, 프롤로그부터 게임이 시작된다. 기존 계정으로 로그인하는 경우 홈 화면으로 바로 이동하게 된다.



**홈 화면**  
홈 화면에서는 랭킹을 확인하고 장착 아이템 선택, 플레이 모드를 선택할 수 있다.   
플레이 모드는 ~~1인용과~~ 2인용을 지원하며, 2인용 모드 선택 시 친구를 초대할 수 있다.



**플레이 모드**

* **~~1인용 모드~~**~~: 기존의 커비가 등장하여 적을 처치하고, 스스로 생명력과 공격을 관리하며 최대한 오래 살아남아 높은 점수를 얻는 모드이다.~~
* **2인용 협동 모드**: 친구를 초대하여 함께 플레이할 수 있으며, 협동 모드에서는 서로 다른 역할 ( 서포터 / 딜러 )을 선택해야 한다. 협력하며 더 많은 적을 처치하고 높은 점수를 달성하는 것이 목표이다.

**3. 캐릭터 역할 및 스킬**

* **서포터** : 보호막 스킬 ( 일정 시간 보호막이 활성화 되어 탄막을 방어한다. )
* **딜러** : 공격력 2배 스킬 ( 더 강력한 탄막을 발사하여 적을 빠르게 처치한다. )

**4. 게임 진행**

게임에서 적은 무한히 등장하며, 시간이 지날수록 적의 **탄막 속도가 빨라지고 HP도 증가**한다. 플레이어는 아이템과 스킬을 사용해 최대한 많은 적을 처치하고 높은 점수를 기록하는 것을 목표로 한다. 협동 모드에서는 팀의 생명력이 공유되므로 한 명이라도 생명력을 소진하면 게임이 종료된다.

**5. 아이템 시스템**

플레이 전 장착 아이템

|  |  |  |  |
| --- | --- | --- | --- |
| **아이템 이름** | **이미지** | **기능** | **가격** |
| **전설의 포션** |  | 생명력이 하나 추가된 상태로 게임을 시작하게 된다. | 500G (골드) |
| **행운의 보석** |  | 게임 내 획득하는 코인이 두 배로 증가하게 된다. | 1000G (골드) |
| **기합의 링** |  | 스킬 게이지를 최대치로 충전된 상태에서 게임을 시작하게 된다. | 500G (골드) |

게임 중 랜덤 드랍 아이템

|  |  |  |
| --- | --- | --- |
| **아이템 이름** | **이미지** | **기능** |
| **자석** |  | 주변의 코인을 끌어모아 효율적으로 코인을 획득할 수 있게 한다. |
| **듀얼** |  | 탄막이 2배로 나가 공격력이 증가한다. |

**6. 적 시스템**

|  |  |  |
| --- | --- | --- |
| **종류** | **이미지** | **부가 설명** |
| 작은 몬스터 |  | HP 10 + @  ~~코인 드랍~~  80%확률로 코인드랍,20%확률로 아이템 드랍 |
| 중간 몬스터 |  | ~~HP 30 + @~~  HP 50 + @  ~~아이템 랜덤 드랍~~  80%확률로 코인드랍,20%확률로 아이템 드랍 |
| 보스 몬스터 |  | ~~HP 50 + @~~  HP 100 + @  ~~아이템 랜덤 드랍~~  80%확률로 코인드랍,20%확률로 아이템 드랍 |

[ 중간 몬스터 1 + 작은 몬스터 3 ] ~~2~~ 3번 진행 후,  
[ 보스 몬스터 1 + 작은 몬스터 5 ] 1번 진행이 반복 된다.

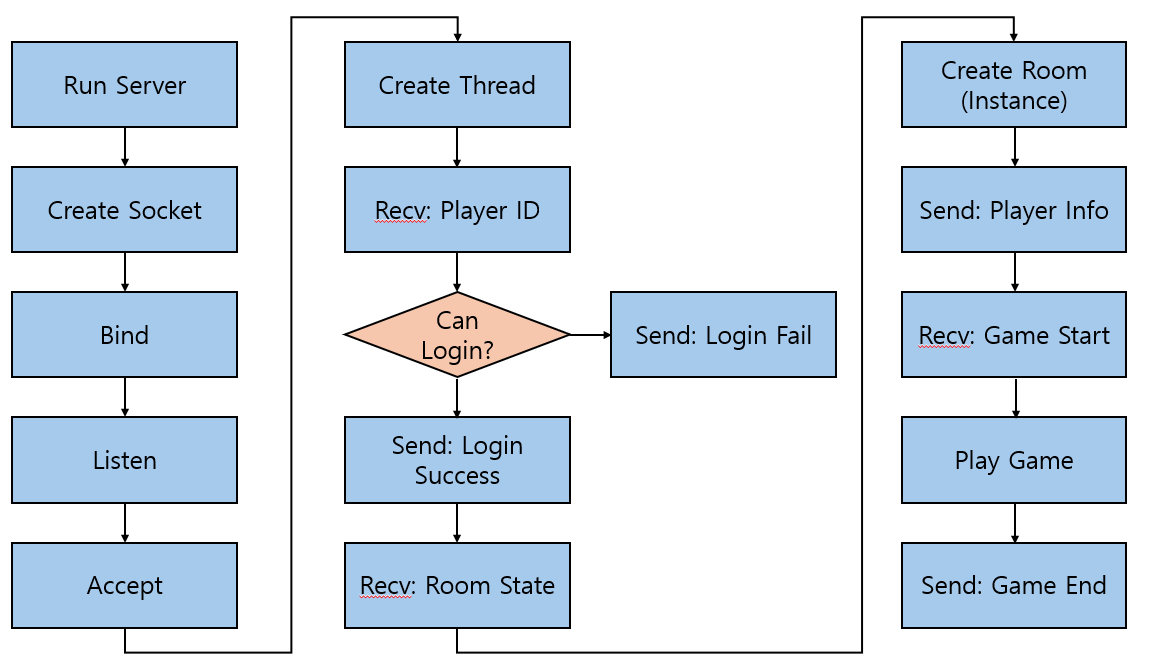
탄막 발사 종류

|  |  |
| --- | --- |
| **종류** | **플레이 화면** |
| 원 발사 |  |
| 일직선 발사 |  |
| 랜덤 발사 |  |
| 회오리 발사 |  |

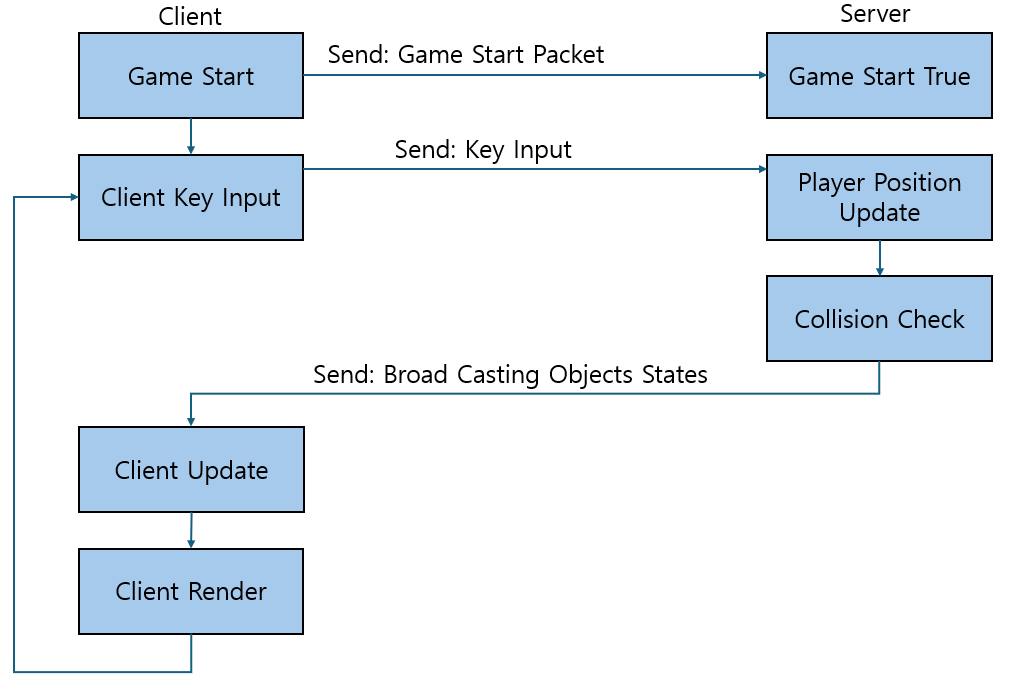
일정 시간마다 4가지 종류 중 하나를 랜덤으로 선택해 발사한다.

**--------------------------Ⅱ. High Level Design --------------------------**

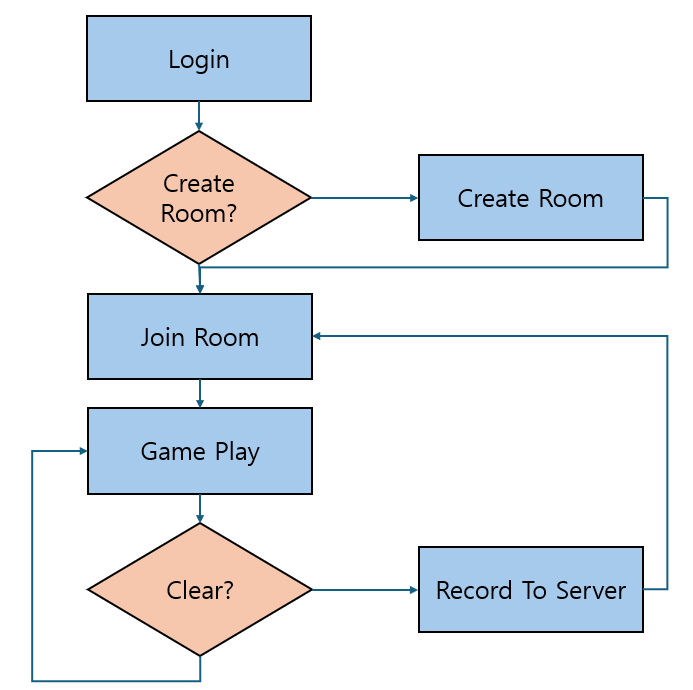
**Server Logic**



**In-Game Logic[[1]](#footnote-1)**

****

**Login Logic**

****

**[클라이언트 로직]**

1. 게임을 실행한다.
2. 아이디를 입력한다.
3. 방을 생성하기 또는 참여하기 중 선택한다. (로그인 성공 패킷을 받은 후)
   * 방에 참여하는 경우, 참여하고자 하는 방의 호스트 플레이어 아이디를 입력한다.
4. 방에 입장하여 캐릭터를 선택한다.
5. 게임을 플레이한다.

**[서버 로직]**

1. 클라이언트의 연결 요청을 수락한다.
2. 클라이언트가 전송한 플레이어 아이디를 확인해 현재 접속 중인지 판별한다.
   * 접속 중인 경우, 이후 접속을 시도한 아이디를 강제 종료한다. (로그인 실패 패킷을 보낸다)
   * 해당 아이디가 접속 중이 아닌 경우, 새로운 플레이어로 초기화하고 스레드를 생성한다. (로그인 성공 패킷을 보낸다)
3. 새로운 방을 만드는 경우 방 객체를 생성해 초기화한다. 기존 방에 참여하는 경우 방 정보를 수정한다.
4. 클라이언트에서 방 설정을 변경할 때마다 서버의 설정도 변경한다.
5. 서버에 랭킹 기록을 관리한다.

**[서버의 스레드 로직]**

1. 클라이언트와의 소켓 통신 스레드
   1. 수신된 패킷을 프로토콜에 따라 해석하고 내용을 확인한다.
   2. 서버에서 처리해야 할 사항을 처리한다. (충돌체크 등)
   3. 처리한 내용을 같은 방의 모든 플레이어에게 브로드캐스트한다.
2. 타이머 스레드
   1. 일정 시간마다 npc ai 로직을 수행한다.
   2. 로직 수행 후 결과 값을 ~~같은 방의 모든 플레이어에게 브로드캐스트한다.~~ 방의 정보에 업데이트한다.

**--------------------------Ⅲ. Low Level Design --------------------------**

|  |  |  |
| --- | --- | --- |
| **Class DataManager : 플레이어 공유 데이터 정보 (client)** | | |
| **struct** | *My\_data\_des* | Char ID[20] // 내 아이디 정보 char otherID[20] // join 아이디 정보 unsigned int high\_score // 나의 최고기록 unsigned int coin // 나의 재화 |
| **struct** | *ranking\_des* | char id[20]; // 다른 플레이어 아이디 정보 unsigned int hs; // 각 플레이어 별 최고기록 |
| **struct** | *item\_data* | RECT rt; // 좌표값 BOOL buy; // 구매 유무 int buy\_cnt; // 구매 & 취소 int price; // 가격 |
|  | | |
| **my\_data\_des** | *my\_data* | 나의 데이터 정보 |
| **ranking\_des** | *rank\_arr[5];* | 랭킹 정보 5순위까지 |
| **item\_data** | *my\_item[3];* | 아이템 구매 정보 |
| **char** | *ROOM\_ID[20]* | 참여하려고 하는 방 아이디 |
| **std::atomic\_bool** | *room\_master* |  |
| **std::atomic\_bool** | *master\_is\_dealer* | 방장이 딜러인지 힐러인지 유무 |
| **int** | *coin\_ani\_frame* | 코인 프레임 |

|  |  |  |
| --- | --- | --- |
| **Class ResourceManager : 리소스들을 선언하고 불러오는 클래스(client)** | | |
| **ResourceManager&** | *getInstance()* |  |
| **void** | *init(HINSTANCE g\_hInst)* | 플레이어 직업 |
| **void** | *destroy()* | 플레이어 id |
| CImage, Sound 등의 리소스들을 로드함. | | |

|  |  |  |
| --- | --- | --- |
| **Class Player : 플레이어 정보 (client)** | | |
| **CImage\*** | *kirby\_pink\_fly*  *kirby\_pink\_hit kirby\_pink\_zombie kirby\_blue\_fly kirby\_blue\_hit kirby\_blue\_zombie* | 이미지 리소스 |
| **short** | *job* | 플레이어 직업 |
| **bool** | *room* | Room 에 들어왔는지, 아닌지를 구분해서 렌더할 용도 |
| **bool** | *inGame* | 게임 플레이 여부 |
| **short** | *x* | 플레이어의 x 좌표 |
| **Unsigned short** | *y* | 플레이어 위치 |
| **bool** | *skill* | Skill 사용 여부 |
| **short** | *skill\_cnt* | Skill 게이지를 그리기 위한 카운트 |
| **~~short~~** | *~~zombie\_cnt~~* | ~~무적 타이머 1 이상 시 무적 타임 적용 30 초과 시 무적 타임 종료~~ |
| **bool** | *die* | 생명력 0 |
| **bool** | *equipments[4]* | 홈 화면에서 사는 장착 아이템 |
| **~~vector <bullet>~~** | *~~bullets~~* | ~~플레이어 탄막~~ |
| **bool** | *who\_is\_me* | 각 클라이언트가 누가 방장인지 참여인지 구분 |
| **SOCKET\*** | *p\_sock* | 소켓 |
|  | | |
| **void** | *render(HDC mdc)* | Player render 함수 |
| **void** | *update();* | Player update 함수 |

|  |  |  |
| --- | --- | --- |
| **Class item : 코인, 듀얼 아이템, 자석 아이템의 부모 클래스** | | |
| **OTYPE** | *type* | 아이템 종류 |
| **CImage\*** | *coin\_bmp dual\_bmp magnet\_bmp* | 이미지 리소스 |
| **unsigned short** | *x* | 아이템 x 좌표 |
| **unsigned short** | *y* | 아이템 y 좌표 |
|  | | |
| **void** | *render(HDC mdc)* | Item render 함수 |

|  |  |  |
| --- | --- | --- |
| **Class enemy** | | |
| **int** | *type* | 0 : 스몰 1~3 : 미들 4 : 빅 |
| **short** | *x* | 적 x 위치 |
| **short** | *y* | 적 y 위치 |
| **short** | *width* | 적의 가로 크기 |
| **short** | *height* | 적의 세로 크기 |
| **short** | *hp* | 적 HP |
| **short** | *cnt* | 탄막 생성 타이머 |
| **short** | *frame* | 리소스가 몇 개의 프레임인지 |
| **short** | *ani* | 현재 프레임 |
| **CImage\*** | *monster[5] monster\_create monster\_die hp\_bar hp\_empty* | 이미지 리소스 |
| **~~vector <enemy\_bullet>~~** | *~~bullets~~* | ~~적 탄막~~ |
|  | | |
| **void** | *render(HDC mdc)* | Enemy render 함수 |
| **void** | *update()* | Enemy update 함수 |

|  |  |  |
| --- | --- | --- |
| **~~Class missile~~** | | |
| **~~short~~** | *~~cnt~~* | ~~미사일 상태 변수 0~20 : 경고/20 : 폭발/44 : 삭제~~ |
| **~~RECT~~** | *~~rt~~* | ~~미사일 xy 좌표, 크기~~ |
| **~~Class Effect[[2]](#footnote-2)~~** | | |
| **~~short~~** | *~~cnt~~* | ~~이펙트 타이머~~ |
| **~~RECT~~** | *~~rt~~* | ~~이펙트 위치, 크기~~ |

|  |  |  |
| --- | --- | --- |
| **Class bullet** | | |
| **CImage\*** | *Kirby\_pink\_bullet Kirby\_blue\_bullet Kirby\_pink\_skill\_bullet Monster\_bullet* | 이미지 리소스 |
| **short** | *job* | 직업 1 : Dealer 2 : Healer |
| **short** | *type* | 탄막 종류 1 : Basic bullet 2 : Skill Bullet |
| **short** | *xy[2]* | 총알의 위치 |
|  | | |
| **void** | *render(HDC mdc)* | Bullet render 함수 |
| **void** | *update()* | Bullet update 함수 |

--------------------패킷 처리 스레드--------------------

|  |  |  |  |
| --- | --- | --- | --- |
| **return** | **calling conventions** | **name** | **desc** |
| void | WINAPI | process\_packet | char\*인수로 들어온 패킷의 종류에 따라 (switch문) 패킷을 처리한다. |

**Packet 종류**• Client To Server

|  |  |
| --- | --- |
| **CS\_LOGIN\_PACKET** | 로그인 시도한 ID를 전송하는 패킷 |
| **CS\_JOIN\_ROOM\_PACKET** | 플레이어가 방을 생성하는지 여부를 전송하는 패킷 |
| **~~CS\_ROOM\_CHANGE\_PACKET~~** | ~~플레이어의 방 설정을 수정하는 패킷[[3]](#footnote-3)~~ |
| **~~CS\_GAME\_START\_PACKET~~** | ~~게임 시작을 알리는 패킷[[4]](#footnote-4)~~ |
| **~~CS\_KEY\_INPUT\_PACKET~~** | ~~플레이어의 키 입력을 전송하는 패킷[[5]](#footnote-5)~~ |
| **CS\_ROOM\_STATE\_PACKET** | 플레이어의 방 설정을 수정하는 패킷 |
| **CS\_MOVE\_PACKET** | 플레이어 마우스 위치 전송 |

• Server To Client

|  |  |
| --- | --- |
| **~~SC\_LOGIN\_SUCCESS\_PACKET~~** | ~~로그인 성공~~ |
| **~~SC\_LOGIN\_FAIL\_PACKET~~** | ~~로그인 실패[[6]](#footnote-6)~~ |
| **~~SC\_PLAYER\_STATE\_CHANGE~~** | ~~HP등 플레이어의 상태 변경[[7]](#footnote-7)~~ |
| **SC\_PLAYER\_MOVE** | 플레이어의 좌표 변경 |
| **SC\_OBJECT\_STATE\_CHANGE** | 오브젝트의 상태 변경 |
| **SC\_RANKING\_UPDATE** | 랭킹 점수 변경 |
| **SC\_LOGIN\_RESULT** | 로그인 성공 실패 정보 반환 |

~~--------------------데이터 처리 스레드--------------------~~

~~각 스레드 내에서 해당작업을 하는 여러 함수를 실행한다.~~

~~멀티스레드 환경에서 병렬적으로 실행된다.~~

-> 스레드가 아닌 이벤트 큐 방식으로 데이터 처리 (하단 기재)

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI player\_update\_thread (LPVOID param) : 플레이어 정보 업데이트 스레드~~** | | | |
| ~~return~~ | ~~name~~ | ~~Virtual~~ | ~~desc~~ |
| ~~void~~ | ~~player\_update~~ | ~~No~~ | ~~플레이어의 좌표, 남은 하트, 점수, 스킬상태, 무적상태를 갱신~~ |

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI collision\_cheak\_thread (LPVOID param) : 충돌 검사 스레드~~** | | | |
| ~~return~~ | ~~name~~ | ~~Virtual~~ | ~~desc~~ |
| ~~void~~ | ~~collision\_enemy~~ | ~~No~~ | ~~적과 내 탄막을 충돌하는 함수~~ |
| ~~void~~ | ~~collision\_player~~ | ~~No~~ | ~~player와 적탄막을 충돌체크하는 함수: 플레이어 둘 중 한명이라 도 탄막을 맞았다면 섕명이 줄어든다~~ |
| ~~void~~ | ~~collision\_missile~~ | ~~No~~ | ~~미사일과 플레이어를 충돌체크하는 함수 플레이어 둘 중 한명이 라도 탄막을 맞았다면 섕명이 줄어든다~~ |

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI enemy\_update\_thread (LPVOID param) : 적 객체 업데이트 스레드~~** | | | |
| ~~return~~ | ~~name~~ | ~~Virtual~~ | ~~desc~~ |
| ~~void~~ | ~~enemy\_move~~ | ~~No~~ | ~~적을 움직여 적의 위치를 갱신하는 함수~~ |
| ~~void~~ | ~~enemy\_create\_bullet~~ | ~~No~~ | ~~적이 총알을 만드는 함수~~ |
| ~~void~~ | ~~die\_enemy~~ | ~~No~~ | ~~적이 죽었는지 (HP가 0이 되었는지)확인하는 함수, 죽은 뒤의 상호 작용 (적 객체 삭제 , 아이템 드롭 등) 도 처리한다.~~ |

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI item\_update\_thread (LPVOID param) : 아이템 객체 업데이트 스레드~~** | | | |
| **~~return~~** | **~~name~~** | **~~Virtual~~** | **~~desc~~** |
| ~~short~~ | ~~get\_type~~ | ~~Yes~~ | ~~서버에서 클라이언트로 아이템 정보를 전달할 때 본인이 코인인지 듀얼인지 자석인지 반환하는 함수. 이를 통해 클라이언트로 타입을 구분해 보낼 수 있다. 각 아이템마다 오버라이딩해 다른 반환값을 냄~~ |
| ~~void~~ | ~~item\_get~~ | ~~Yes~~ | ~~코인, 듀얼아이템, 자석아이템을 먹었는지 확인하는 함수. 각 아 이템마다 오버라이딩 한다~~ |
| ~~void~~ | ~~item\_move~~ | ~~No~~ | ~~코인, 듀얼아이템, 자석아이템을 움직이는 함수. 셋의 이동방식은 똑 같으므로 오버라이딩 하지않는다.~~ |
| ~~void~~ | ~~item\_delete~~ | ~~No~~ | ~~코인, 듀얼아이템, 자석아이템이 화면 밖으로 사라졌는지 확인하는 함수.~~ |

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI bullet\_update\_thread (LPVOID param): 총알정보를 업데이트 하는 스레드~~** | | | |
| **~~return~~** | **~~name~~** | **~~Virtual~~** | **~~desc~~** |
| ~~void~~ | ~~bullet\_update~~ | ~~No~~ | ~~내 총알을 이동시키고 만약 그 총알이 화면을 벗어났을 시 그 객 체를 삭제하는 함수.~~ |
| ~~void~~ | ~~enemy\_bullet\_update~~ | ~~No~~ | ~~적의 총알을 이동시키고 만약 그 총알이 화면을 벗어났을 시 그 객체를 삭제하는 함수.~~ |
| ~~void~~ | ~~missile\_update~~ | ~~No~~ | ~~미사일들을 이동시키고 어떤 미사일이 화면을 벗어나면 그 미사 일을 삭제하는 함수.~~ |

|  |  |  |  |
| --- | --- | --- | --- |
| **~~DWORD WINAPI effect\_update\_thread (LPVOID param) : 이펙트 객체 업데이트 스레드~~** | | | |
| **~~return~~** | **~~name~~** | **~~Virtual~~** | **~~desc~~** |
| ~~void~~ | ~~effect\_update~~ | ~~No~~ | ~~이펙트들의 프레임을 업데이트하고 애니메이션이 끝난 이펙트는 삭제하는 함수~~ |

**------------------------------------서버 데이터 처리 이벤트------------------------------------**

ai\_thread()에 의해 순차적으로 각 이벤트들이 작업을 수행한다.

|  |  |  |
| --- | --- | --- |
| 이름 | 실행주기 (ms) | 작업 내용 |
| FIRE\_PLAYER\_BULLET | 500 | 총알을 생성한다. (1P/2P),(스킬여부),(듀얼아이템 여부)를 모두 고려해 총알을 생성한다. |
| MOVE\_PLAYER\_BULLET | 100 | 총알,아이템이 가진 이동함수를 호출해 총알과 아이템을 움직인다.  움직였을 때 화면 밖을 나간 객체를 지운다.  HP가 0이 된 적을 지운다.  충돌처리를 수행한다. |
| FIRE\_ENEMY\_BULLET | 1000 | 적의 총알을 생성한다. 각 적마다 다른 공격유형을 가진다. |
| AI\_MOVE | 100 | 적의 좌표를 이동시킨다. 적은 랜덤한 경로로 움직인다. |
| SKILL\_END | X(반복돼서 실행되지 않음) | 플레이어가 스킬을 사용한 뒤 3초후에 실행되어 플레이어의 스킬 발동 여부 스위치를 끄는 역할을 한다. |
| CREATE\_SET | 1000 | 화면내의 적을 모두 죽였는지 검사한 후 만약 모든 적이 죽었다면 새로운 적 세트를 생성한다. |
| ENABLE\_COLLISION | X(반복돼서 실행되지 않음) | 플레이어가 적의 총알에 맞고 잠시 동안 무적타임이 되고 난 후 3초뒤에 실행되어 플레이어의 무적 타임 여부 스위치를 끄는  역할을 한다. |
| CANCEL\_DUAL | X(반복돼서 실행되지 않음) | 플레이어가 듀얼 아이템을 먹은 뒤 5초뒤에 실행되어 플레이어의 듀얼 아이템 효과를 중지시킨다. |
| CANCEL\_MAGNET | X(반복돼서 실행되지 않음) | 플레이어가 자석 아이템을 먹은 뒤 5초뒤에 실행되어 플레이어의 자석 아이템 효과를 중지시킨다. |

--------------------클라이언트에서 인게임 도중 실행되는 스레드--------------------

|  |  |  |  |
| --- | --- | --- | --- |
| **DWORD WINAPI Client\_update\_thread (LPVOID param) : 클라이언트가 update하는 함수를 일괄실행하는 스레드-> 기존 있던 함수들을 모두 Client\_network\_thread로 이동**  Client\_update\_thread을 애니메이션을 위한 프레임 관리 작업에 이용. | | | |
| **return** | **name** | **Virtual** | **desc** |
| ~~int~~ | ~~send\_player\_input~~ | ~~No~~ | ~~송신의 결과 반환.(오류시 -1, 성공시 보낸 총 바이트크기) 서버 에게 플레이어의 y좌표를 보낸다~~ |
| ~~int~~ | ~~recv\_player\_data~~ | ~~No~~ | ~~수신의 결과 반환.(오류시 -1, 성공시 받은 총 바이트크기) 플레이 어들의 패킷구조체를 받아와 클라이언트 변수에 저장 (플레이어의 탄막정보 포함)~~ |
| ~~int~~ | ~~recv\_enemy\_data~~ | ~~No~~ | ~~수신의 결과 반환.(오류시 -1, 성공시 받은 총 바이트크기) 적의 패 킷구조체를 받아와 클라이언트 변수에 저장 (적의 탄막정보 포함)~~ |
| ~~int~~ | ~~recv\_missile\_data~~ | ~~No~~ | ~~수신의 결과 반환.(오류시 -1, 성공시 받은 총 바이트크기) 미사일 의 패킷구조체를 받아와 클라이언트 변수에 저장~~ |
| ~~int~~ | ~~recv\_item\_data~~ | ~~No~~ | ~~수신의 결과 반환.(오류시 -1, 성공시 받은 총 바이트크기) 아이템의 패킷구조체를 받아와 클라이언트 변수에 저장~~ |
| ~~int~~ | ~~recv\_effect\_data~~ | ~~No~~ | ~~수신의 결과 반환.(오류시 -1, 성공시 받은 총 바이트크기) 이펙트 의 패킷구조체를 받아와 클라이언트 변수에 저장~~ |
| 플레이어, 적, 총알, 배경, 아이템 애니메이션 프레임 업데이트 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **DWORD WINAPI Client\_network\_thread (LPVOID param) : 클라이언트가 서버와 송수신 하는 함수를 일괄실행하는 스레드** | | | |
| **return** | **name** | argu | **desc** |
| int | send\_player\_input | unsingned short y | 플레이어가 마우스를 움직일때마 y를 보내는 것이 아닌 플레이어의 최종 y를 기억하고 있다가 30FPS 주기로 서버에게 플레이어의 y를 송신한다.실패시 -1, 성공시 1을 리턴한다 |
| int | recv\_process | No | 서버로 부터 패킷을 수신하고, 수신한 패킷의 타입에 따라, 각 타입에 맞는 패킷처리 함수를 유연하게 실행한다. 패킷의 타입이 SC\_PLAYER\_MOVE라면 handle\_player\_data를, SC\_OBJECT\_MOVE라면 handle\_object\_data를 실행한다. |
| void | handle\_player\_data | const uint8\_t\* packetData | Recv\_process에서 수신한 누적버퍼(vector)의 data()를 넘겨받아 수신된 플레이어 정보를 클라이언트의 플레이어 정보에 복사해 플레이어정보를 업데이트한다. |
| void | handle\_object\_data | const uint8\_t\* packetData | Recv\_process에서 수신한 누적버퍼(vector)의 data()를 넘겨받아 수신된 오브젝트 정보를 클라이언트의 임시 오브젝트 정보에 복사하고, 복사가 끝나면 클라이언트의 오브젝트 정보와 임시 오브젝트의 정보를 swap한다. |

|  |  |  |  |
| --- | --- | --- | --- |
| **DWORD WINAPI Client\_render\_thread (LPVOID param) : 클라이언트가 render하는 함수를 일괄실행하는 스레드** | | | |
| **return** | **name** | **Virtual** | **desc** |
| void | Background\_draw  ->UI\_render | No->  HDC mdc | 서버에서 정보를 받아올 필요가 없는 정보들(배경, 기본 템플릿 )을 그리는 함수 -> 배경과 ui 의 레이어를 고려해 배경그리기와 ui그리기를 분리 후 순서 조정 |
| ~~void~~ | ~~player\_draw~~  ->별도의 함수로 분리하지 않음. P1,p2에 바로 접근 후 draw | ~~No~~ | ~~플레이어들을 그리는 함수~~ |
| ~~void~~ | ~~enemy\_bullet\_draw~~  -> 별도의 함수로 분리하지 않음. 적총알과 플레이어 총알을 함께 그리도록 수정 | ~~No~~ | ~~플레이어 탄막을 그리는 함수~~ |
| ~~void~~ | ~~enemy\_draw~~  ->별도의 함수로 분리하지 않음. for문으로 적 객체들에 바로 접근 후 draw | ~~No~~ | ~~적을 그리는 함수~~ |
| ~~void~~ | ~~enemy\_bullet\_draw~~  -> 별도의 함수로 분리하지 않음. 적총알과 플레이어 총알을 함께 그리도록 수정 | ~~No~~ | ~~적 탄막을 그리는 함수~~ |
| ~~void~~ | ~~missile\_draw~~ | ~~No~~ | ~~미사일을 그리는 함수~~ |
| void | item\_draw | No  ->HDC mdc | 아이템 ( 코인, 듀얼, 자석 )을 그리는 함수 |
| ~~void~~ | ~~effect\_draw~~ | ~~No~~ | ~~이펙트를 그리는 함수~~ |

--------------------서버에서 사용하는 클래스--------------------

|  |  |  |
| --- | --- | --- |
| **Class Room : 플레이어가 지금 참여 중인 방 정보** | | |
| **Player\*** | *p1* | 1번 플레이어 |
| **Player\*** | *p2* | 2번 플레이어 |
| **~~string~~** | *~~host\_id~~* | ~~룸 ID[[8]](#footnote-8)~~ |
| **std::string** | *Dealer\_id* | 딜러 ID |
| **bool** | *isPlaying* | 룸 플레이 여부 |
| **unsigned int** | *score* | 점수 |
| **short** | *heart* | 생명력 |
| **vector <Player\_bullet>** | *P\_bullets* | 탄막 |
| **vector <effect>** | *effects* | 충돌 이펙트 |
| **vector <item>** | *items* | 드랍 아이템 |
| **vector <enemy>** | *Enemys* | 적 객체 |
| **vector<Enemy\_bullet >** | *E\_bullets* | 적 탄막 |

|  |  |  |
| --- | --- | --- |
| **Class Player : 플레이어 정보 (server)** | | |
| **SOCKET** | *Sock* | 소켓 |
| **string** | *id* | 플레이어 id |
| **bool** | *inGame* | 게임 플레이 여부 |
| **unsigned short** | *y* | 플레이어 위치 |
| **unsigned int** | *high\_score* | 최고점수 |
| **unsigned short** | *coin* | 현재 소지한 코인 |
| **bool** | *isSkill* | 현재 스킬을 사용 중인지 |
| **short** | *Zombie\_cnt* | 1일 때 무적 |
| **Unsigned short** | *skillCount* | 스킬이 얼마나 채워졌는지 |
| **Bool** | *Magnet\_eat* | 자석(아이템)을 먹었는가? |
| **bool** | *Dual\_eat* | 듀얼(아이템)을 먹었는가? |

|  |  |  |
| --- | --- | --- |
| **Class Enemy\_Bullet (server)** | | |
| **~~Point~~** | *~~speed~~* | ~~탄막 속도[[9]](#footnote-9)~~ |
| **~~Point~~** | *~~xy~~* | ~~탄막 xy 좌표[[10]](#footnote-10)~~ |
| **POINT** | *dir* | 탄막이 날라가는 방향 |
| **unsigned short** | *x* | 탄막 x좌표 |
| **unsigned short** | *y* | 탄막 y좌표 |

|  |  |  |
| --- | --- | --- |
| **Class Player\_Bullet** | | |
| **~~Short~~ OTYPE(enum)** | *type* | 특수 탄막, 일반 탄막 구분 |
| **~~Point~~** | *~~xy~~* | ~~탄막 xy 좌표[[11]](#footnote-11)~~ |
| **Unsigned short** | *x* | X좌표 (default=DEFXPOS) |
| **Unsigned short** | *y* | Y 좌표 |

|  |  |  |
| --- | --- | --- |
| **Class enemy** | | |
| **~~Short~~ OTYPE(enum)** | *type* | 0 : 잡몹 1~3 : 중간보스 4 : 최종보스 |
| **~~RECT~~** | *~~rt~~* | ~~적 위치, 크기[[12]](#footnote-12)~~ |
| **unsigned short** | *hp* | 적 HP |
| **~~short~~** | *~~cnt~~* | ~~탄막 생성 타이머[[13]](#footnote-13)~~ |
| **~~Short~~** | *~~frame~~* | ~~프레임~~ |
| **~~short~~** | *~~attack\_type~~* | ~~탄막 발사 종류~~ |
| **~~vector <enemy\_bullet>~~** | *~~bullets~~* | ~~적 탄막[[14]](#footnote-14)~~ |
| **unsigned short** | *x* | 탄막 x좌표 |
| **Unsigned short** | *y* | 탄막 y좌표 |
| **short** | *attack\_type* | 공격 유형 |
| **short** | *bullet\_create\_cnt* | 총알 생성 개수 |
| **unsigned short** | *width* | 탄막 충돌체 크기 |
| **unsigned short** | *height* | 탄막 충돌체 크기 |

|  |  |  |
| --- | --- | --- |
| **Class item : ~~코인, 듀얼 아이템, 자석 아이템의 부모 클래스[[15]](#footnote-15)~~** | | |
| **~~Point~~** | *~~xy~~* | ~~아이템 xy 좌표[[16]](#footnote-16)~~ |
| **OTYPE** | *type* | 아이템 타입 |
| **unsigned short** | *x* | 아이템 x좌표 |
| **unsigned short** | *y* | 아이템 y좌표 |

--------------------서버에서 사용하는 스레드[[17]](#footnote-17)--------------------

서버는 크게 4종류의 스레드를 가지고 있습니다.

소켓 통신 함수는 return type을 bool 형식으로 통일하였습니다.

|  |  |  |  |
| --- | --- | --- | --- |
| int client\_thread (SOCKET s) : 클라이언트와의 통신 스레드, 아래는 송수신 관련 함수입니다. | | | |
| **return** | name | **Virtual** | **desc** |
| bool | send\_login\_packet | No | 인자로 SOCKET과 CS\_LOGIN\_PACKET\*를 받습니다. 클라이언트에게 로그인 정보를 송신하는 함수입니다. |
| bool | send\_top\_high\_scores | No | 인자로 SOCKET을 받습니다. 클라이언트에게 서버에 기록된 데이터 중 점수 상위 5개의 정보를 송신하는 함수입니다. |
| bool | process\_packet | No | 인자로 char\*, SOCKET, std::string을 받습니다. recv한 버퍼 정보를 바탕으로 어떤 패킷인지 확인해 처리하고 필요한 송신까지 수행하는 함수입니다. |
| bool | send\_player\_move\_packet | No | 인자로 SOCKET, std::string을 받습니다. string값 ID를 가진 플레이어의 위치와 게임 플레이(Room)정보를 송신하는 함수입니다. |
| bool | send\_room\_change\_packet | No | 인자로 SOCKET과 std::string을 받습니다. 클라이언트가 Room에 접속했을 때 Room에 대한 정보를 송신하는 함수입니다. |
| bool | send\_room\_join\_fail\_packet | No | 인자로 SOCKET과 std::string을 받습니다. 위 함수와 똑같이 SC\_ROOM\_CHANGE\_PACKET을 송신하지만 세팅 값을 다르게 담아서 Room 접속 실패를 나타냅니다. |
| bool | send\_object\_move\_packet | No | 인자로 SOCKET과 std::string을 받습니다. SC\_OBJECT\_MOVE\_PACKET을 송신하며 현재 룸 안에 존재하는 모든 오브젝트들의 위치와 상태를 송신합니다. |

|  |  |  |  |
| --- | --- | --- | --- |
| void timer\_thread() : ai등의 시간이 있는 동작을 관리하는 스레드입니다. | | | |
| **return** | name | **Virtual** | **desc** |
| - | - | - | 함수를 따로 사용하지 않습니다. |

|  |  |  |  |
| --- | --- | --- | --- |
| void ai\_thread() : 이벤트를 실행하는 스레드입니다. 위의 타이머 스레드와 통신합니다. | | | |
| **return** | name | **Virtual** | **desc** |
| void | push\_evt\_queue | No | TASK\_TYPE, int, std::string을 인자로 받습니다. 이벤트 타이머를 관리하는 evt\_queue에 이벤트를 넣는 함수입니다. |

|  |  |  |  |
| --- | --- | --- | --- |
| int main() : 기본 세팅과 네트워크 세팅, accept를 수행하는 스레드입니다. 아래는 파일입출력 관련 함수를 작성하였습니다(이 스레드에서만 실행되는 함수는 아닙니다). | | | |
| **return** | name | **Virtual** | **desc** |
| bool | read\_player\_info | No | data.txt 파일에서 플레이어 id, 최고점수, 코인을 읽어 players컨테이너에 저장합니다. |
| bool | add\_new\_player | No | 인자로 std::string, unsigned int, unsigned int를 받습니다. 새로운 아이디의 플레이어가 접속했을 때 파일에 기록합니다. |
| bool | save\_all\_player\_info | No | 유저의 최고점수 등 정보를 data.txt파일에 수정해 적습니다. |

--------------------서버에서 사용하는 컨테이너--------------------

|  |  |  |
| --- | --- | --- |
| **std::unordered\_map**  **<std::string, Room\*>** | *roomInfo* | Room 정보를 관리 |
| **std::unordered\_map**  **<std::string, Player>** | *players* | Player(클라이언트) 정보를 관리 |
| **concurrency::concurrent\_priority\_queue**  **<EVENT>** | *evt\_queue* | Event 시간을 관리 |
| **concurrency::concurrent\_priority\_queue**  **<EVENT>** | *task\_queue* | Ai Thread에서 처리할 이벤트 관리 |

**--------------------------Ⅳ. Roles --------------------------**

|  |  |  |
| --- | --- | --- |
| **김해님** | **박민지** | **이세영** |
| 패킷 프로토콜 정의  패킷 송수신 서버to클라 함수 만들기  파일 입출력 | 패킷 송수진 클라to서버 함수 만들기  로비 UI 제작  리팩토링 | 패킷 송수진 클라to서버 함수 만들기  AI 처리 로직 만들기  리팩토링 |

**--------------------------Ⅴ. Schedule --------------------------**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**--------------------------Ⅵ . Self Check List --------------------------**

|  |  |
| --- | --- |
| **Server** | |
| O | **멀티스레드 구현** (각 클라이언트마다 스레드 할당) |
| O | **패킷 프로토콜 정의** |
| O | **유저 데이터 관리** (파일 입출력을 통해 모든 유저 정보 저장,불러오기) |
| O | **타이틀 씬 데이터 처리** ( SC\_LOGIN\_RESULT\_PACKET 패킷 송신, SC\_RANKING\_PACKET패킷 송신CS\_LOGIN\_PACKET패킷 수신) |
| O | **로비 씬 데이터 처리** (CS\_JOIN\_ROOM\_PACKET패킷 수신, 방 생성/ 방참가 구분, 방참가 예외처리) |
| O | **룸 씬 데이터 처리** (SC\_ROOM\_CHANGE\_PACKET패킷 송신, 직업 변경, 시작 신호 CS\_ROOM\_STATE\_PACKET패킷 수신 후 방 정보 업데이트) |
| O | **플레이 데이터 처리** (객체 생성/이동/삭제 , 객체 충돌처리, , CS\_MOVE\_PACKET패킷 수신, SC\_PLAYER\_MOVE\_PACKET패킷 송신, SC\_OBJECT\_MOVE\_PACKET패킷 송신) |
| **Client** | |
| O | **멀티스레드 구현** (Render, Update, Network 스레드 분리) |
| O | **타이틀 씬 구현** ( 타이틀 씬 Render, CS\_LOGIN\_PACKET패킷 송신, SC\_LOGIN\_RESULT\_PACKET 패킷 수신, SC\_RANKING\_PACKET패킷 수신) |
| O | **로비 씬 구현** (로비 씬 Render, 장비 구매, CS\_JOIN\_ROOM\_PACKET패킷 송신, 방 생성/ 방참가 구분, 방참가 예외처리) |
| O | **룸 씬 구현** (룸 씬 Render, 직업 변경, 시작 신호 CS\_ROOM\_STATE\_PACKET패킷 송신, SC\_ROOM\_CHANGE\_PACKET패킷 수신후 방 정보 업데이트) |
| O | **플레이 씬 구현** (플레이 씬 Render, CS\_MOVE\_PACKET패킷 송신, SC\_PLAYER\_MOVE\_PACKET패킷 수신, SC\_OBJECT\_MOVE\_PACKET패킷 수신) |

1. Object States Send를 패킷 2개로 쪼개서 보내도록 수정하였습니다. Player 정보와 Object 정보를 따로 보냅니다. [↑](#footnote-ref-1)
2. 이펙트 삭제. [↑](#footnote-ref-2)
3. 이름 변경 [↑](#footnote-ref-3)
4. CS\_ROOM\_STATE\_PACKET으로 묶음. [↑](#footnote-ref-4)
5. CS\_MOVE\_PACKET으로 묶음 [↑](#footnote-ref-5)
6. LOGIN\_RESULT\_PACKET으로 묶음. [↑](#footnote-ref-6)
7. SC\_PLAYER\_MOVE으로 묶음. [↑](#footnote-ref-7)
8. 룸 ID를 P1의 아이디로 사용하도록 수정 [↑](#footnote-ref-8)
9. 속도 통일 [↑](#footnote-ref-9)
10. x와 y값 분리 [↑](#footnote-ref-10)
11. X와 y좌표를 나눠서 저장하도록 수정. [↑](#footnote-ref-11)
12. x, y, width, height로 분리 [↑](#footnote-ref-12)
13. 타이머 스레드에서 처리 [↑](#footnote-ref-13)
14. 클래스 분할하며 위치가 바뀜. [↑](#footnote-ref-14)
15. Enum OTYPE으로 선언 [↑](#footnote-ref-15)
16. xy좌표 분리 [↑](#footnote-ref-16)
17. 아예 새로 작성하는 부분입니다.. [↑](#footnote-ref-17)