**네트워크 게임 프로그래밍  
Term Project 추진 계획서**

2017182028 윤혜림

2017182045 황신필

1. 애플리케이션 기획

1-1 개발 환경

1-2 네트워크 기능

1-3 게임 소개

1-4 게임 플레이

1. High-Level 디자인
2. Low-Level 디자인

3-1 서버 디자인

3-2 클라이언트 디자인

3-3 객체 모델링

3-4 패킷 모델링

1. 팀원 별 역할분담
2. 개발 일정

**1. 애플리케이션 기획**

**1-1 개발 환경**

* Window 10
* Visual Studio 2019
* Window Socket API
* Direct2D
* GitHub

**1-2 네트워크 구현**

* 유저 채팅 기능
* 실시간 멀티플레이(2~4명)

**1-3 게임 소개**

이름 : 붐렉트(BombRect)

장르 : 아케이드 대전 게임

플레이어 수 : 2 ~ 4인

플랫폼 : Window PC 게임

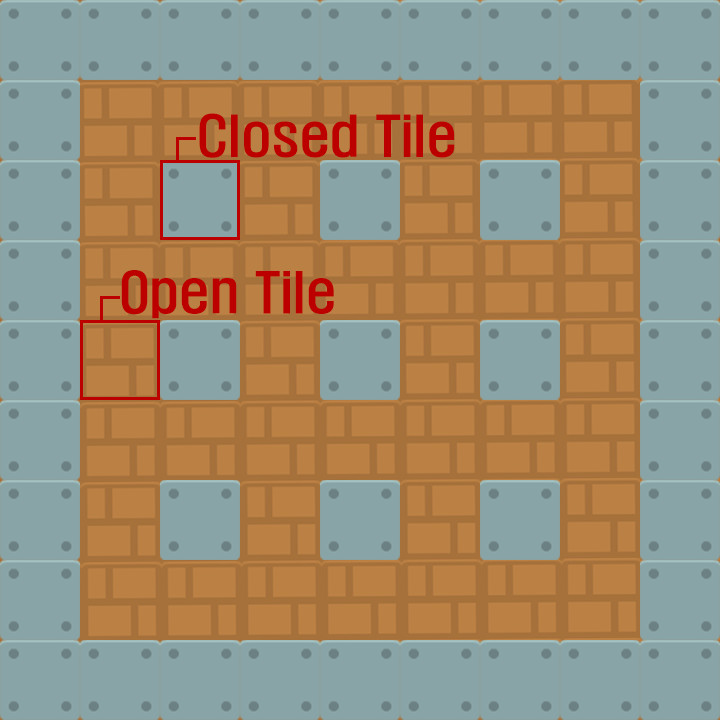
벤치마킹 게임 : 봄버맨(Bomberman)



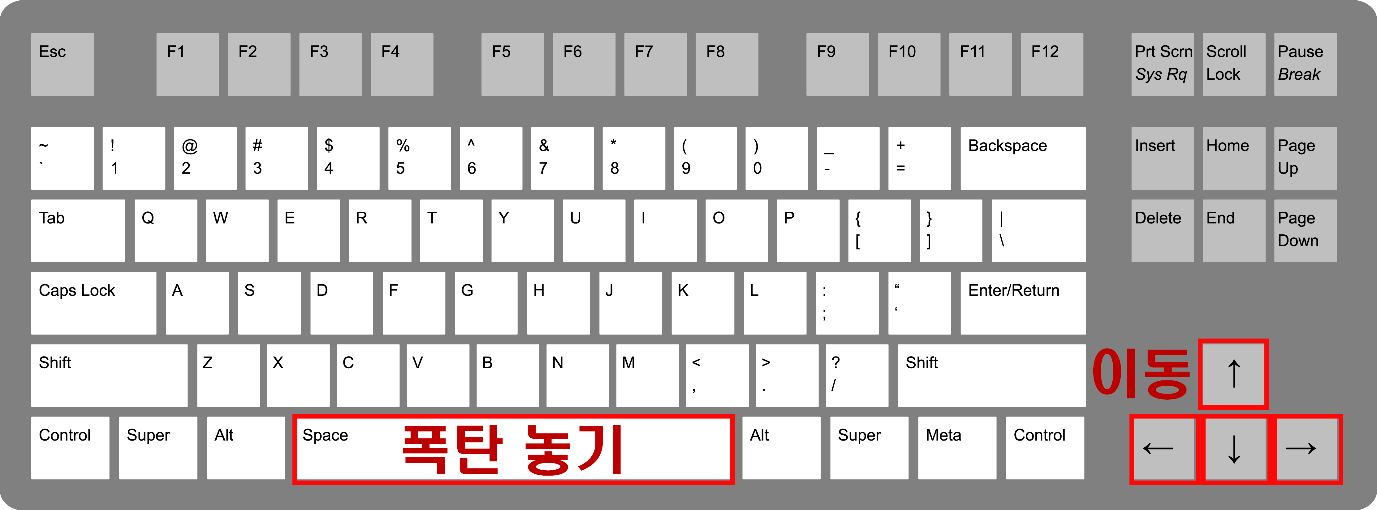
[ 출처 - <https://namu.wiki/w/%EB%B4%84%EB%B2%84%EB%A7%A8%20%EC%8B%9C%EB%A6%AC%EC%A6%88> ]

**1-4 게임 플레이**

* **맵**



* **조작키**



* **플레이어**

플레이어는 최대 3개의 폭탄을 동시에 맵에 놓을 수 있다.  
폭탄으로부터 생성되는 데미지 영역에 3번 충돌하면 패배한다.

* **폭탄**

플레이어가 둔 폭탄은 일정 시간후에 터진다.  
폭탄이 터진 후 데미지 영역이 생성된다.

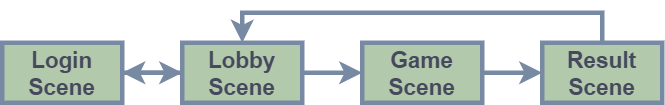
****

* **데미지 영역**

폭탄이 터진 위치에서 상하좌우로 연속된 Open Tile이  
데미지 영역의 범위가 된다.  
데미지 박스는 일정 시간후에 사라진다.

**2. High-Level 디자인**

**Scene 흐름도**



* **Login Scene**

유저가 사용할 닉네임을 설정하고, IP 주소를 입력해 서버에 접속을 시도한다.

* **Lobby Scene**

접속된 유저를 보여주고, 채팅 기능을 통해 소통할 수 있다. 모든 유저가 Ready 사인을 보내면 게임을 시작한다.

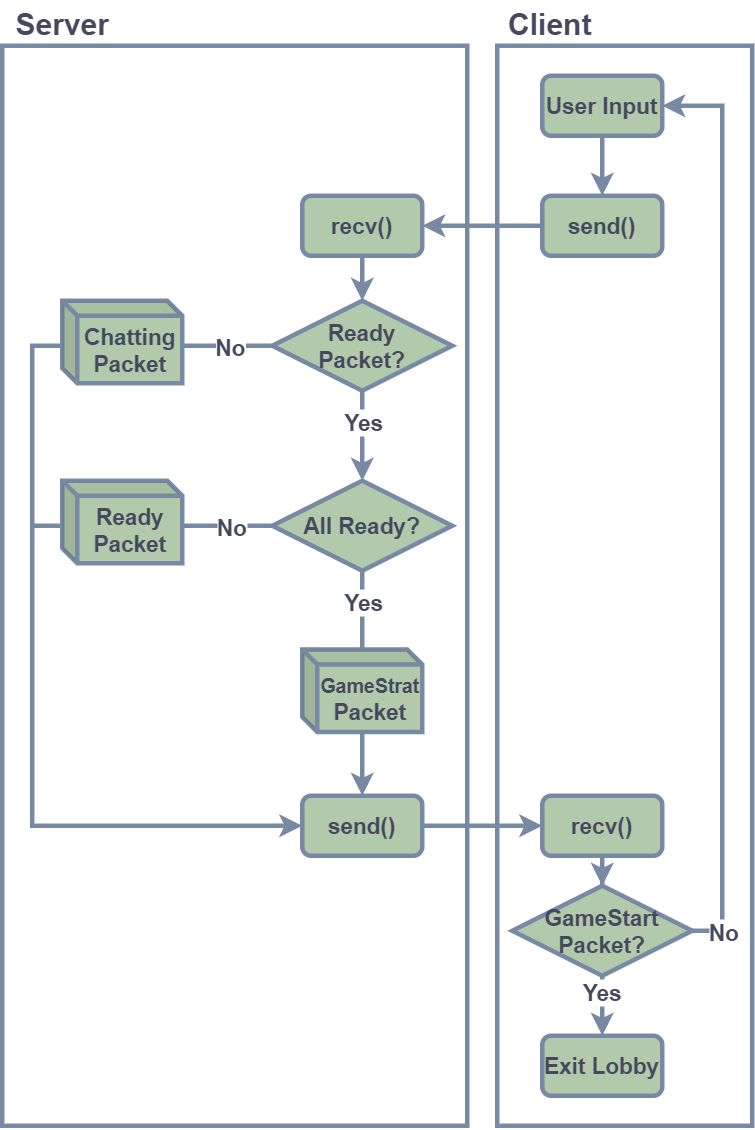
* **Game Scene**

게임을 진행한다. 패배했지만 아직 게임이 끝나지 않은 경우, 관전 모드로 전환되어 게임이 끝날 때까지 기다린다. 생존한 유저가 한 명이라면 게임을 종료한다.

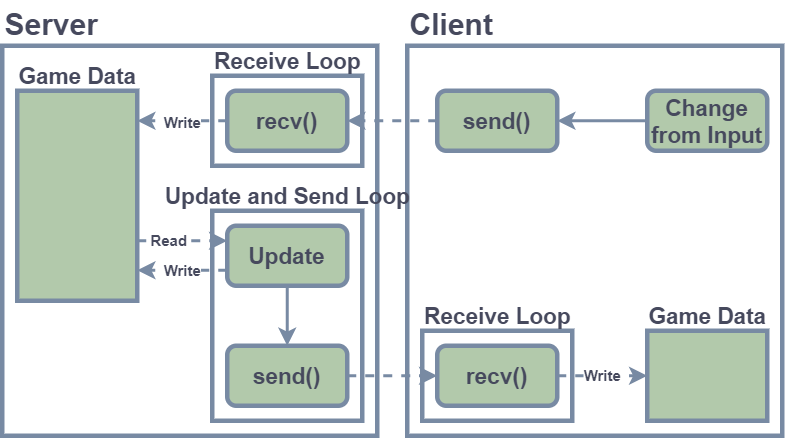
* **Result Scene**

오래 생존한 순서에 따라 순위를 정해 보여준다. 일정 시간이 지난 후 Lobby Scene으로 전환된다.

**Lobby Scene Flow Chart**



**Game Scene Data Flow**



[+] 11.07 ---------------------------------------------------------------------------------------

**Server Thread Design**

**1. client thread**

하나의 클라이언트와 연결을 담당하는 thread이다.  
 서버에는 연결된 클라이언트 수만큼 client thread가 존재한다.  
 Game Scene이 아닌 경우 클라이언트와 데이터를 주고받고,  
 Game Scene인 경우 클라이언트로부터 데이터를 받는 역할을 한다.

**2. update and send thread**

Game Scene 시작되면 생성되고, 끝나면 소멸한다.  
 Game World를 업데이트하고 모든 클라이언트에게 업데이트된 데이터를  
 모든 클라이언트에게 전송한다.

**Client Thread Design**

**1. server thread**

서버와 연결을 담당하는 thread이다.  
 서버로부터 데이터를 받는 역할을 한다.

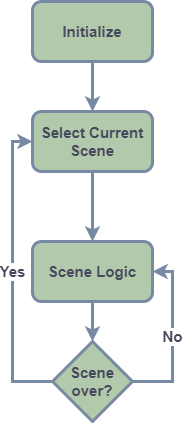
**2. window message thread**

window로부터 오는 메시지를 기반으로 Rendering과 사용자 입력 처리를 담당한다.

-------------------------------------------------------------------------------------------------

**3. Low-Level 디자인**

**3-1 서버 디자인**



**1. Initialize**

서버가 생성될 때 초기 값을 정한다.

**2. Check Scene**

Scene의 정보를 받아와 대응되는 Scene 함수를 호출한다.

**3. Scene Loop**

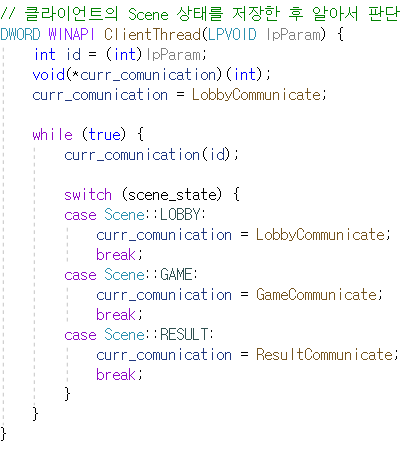
Scene의 Logic을 반복한다.

**4. Scene Over**

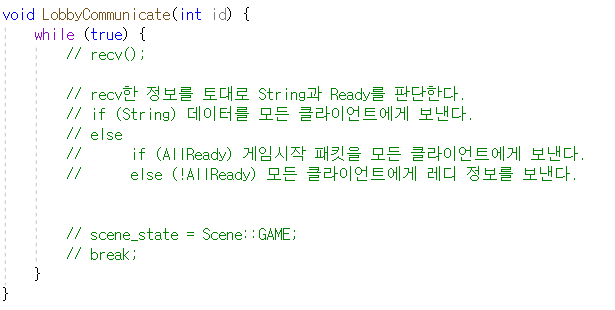
Scene의 Loop 탈출 여부를 판단한다. Scene이 종료되면, Scene의 값을 바꾼다.

**함수 설계**

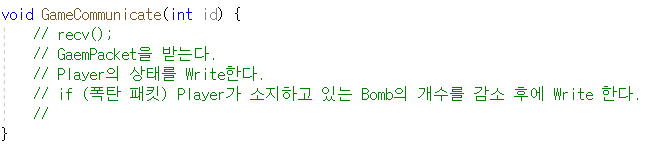
1. ClientThread



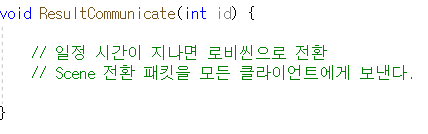
2. LobbyCommunicate



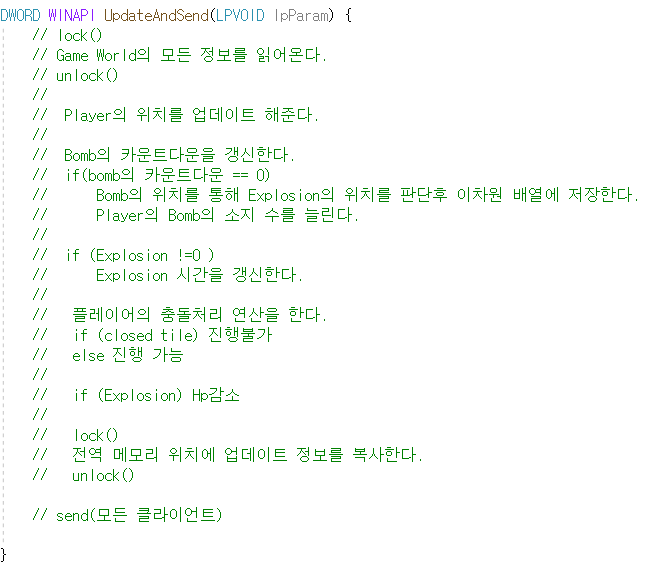
3. GameCommunicate



4. ResultCommunicate



5. UpdateAndSend



1. ClientThread: Scene을 판단해 호출한다.

2. LobbyCommunicate: 로비의 패킷 처리 Logic을 담당한다. 로비는 Chatting과 Ready 패킷을 판단하고 처리한다. 모두가 Ready 상태가 되면 게임시작 패킷을 보내고 리턴한다.

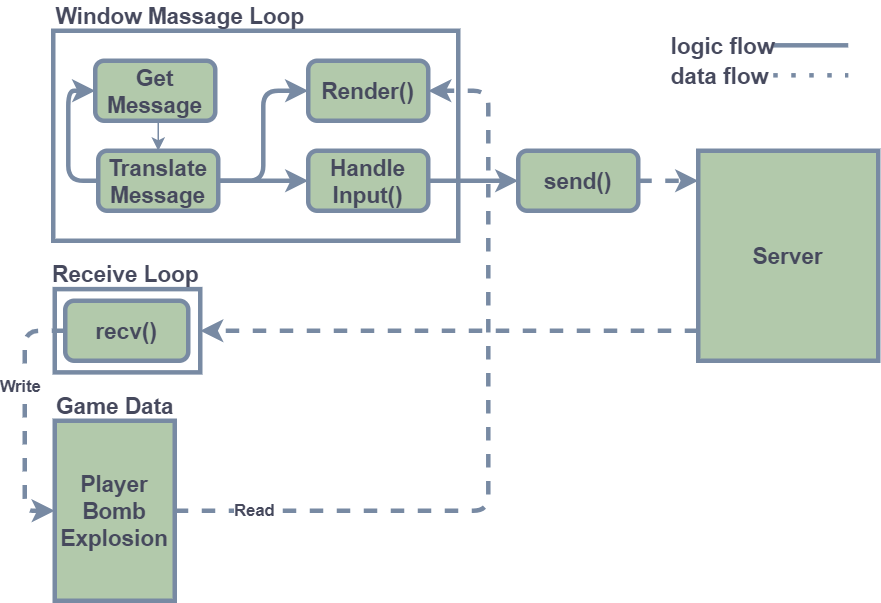
3. GameCommunicate: 게임안에서 플레이어의 상태, 폭탄의 상태를 Write 한다.

4. ResultCommunicate: 결과창을 보여준다. 결과창의 Scene 정보를 가지고 있는다. 결과창에서 로비로 돌아가는 타이머를 갱신하다. 타이머가 0이되면 Scene 전환 패킷을 보낸다.

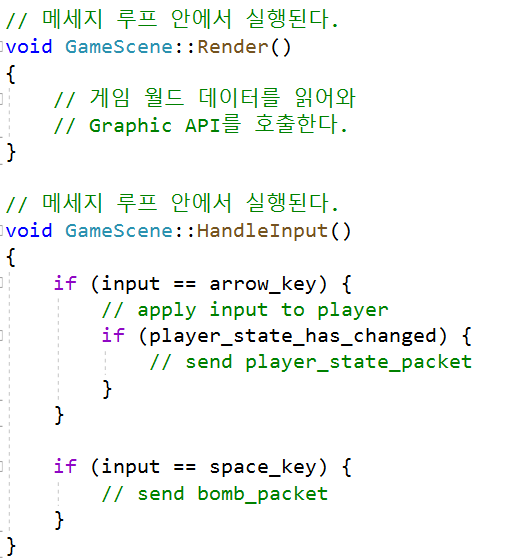
5. UpdateAndSend: 최신의 Game World 정보를 읽어와 계산하고, 정보를 갱신한 후 모든 클라이언트한테 보내준다. 전역 데이터에 접근할 때 동기화 기법을 사용한다.

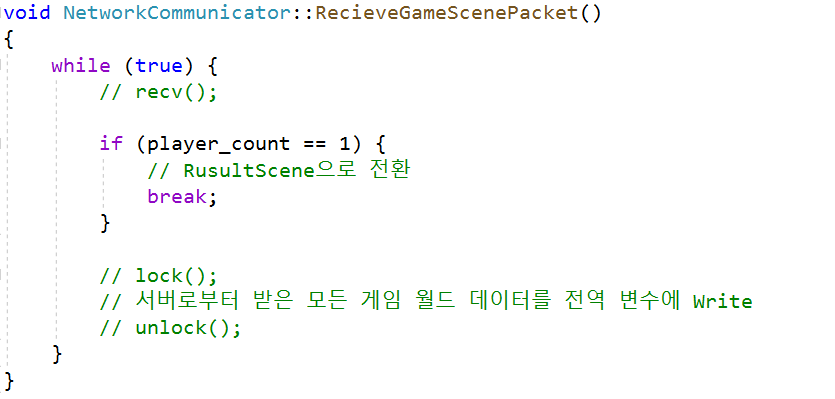
**3-2 클라이언트 디자인**

**Game Scene Flow**

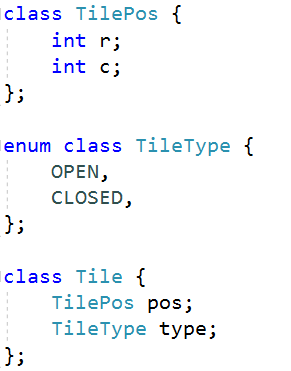
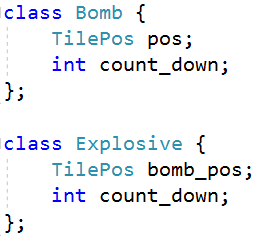
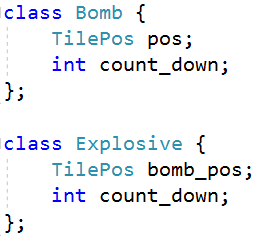
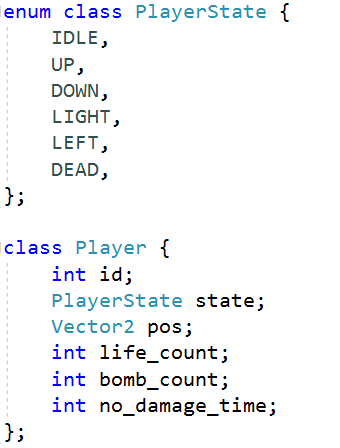


**함수 설계**

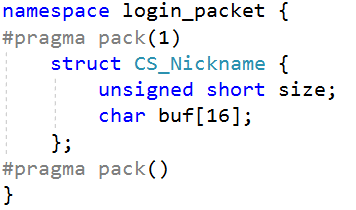


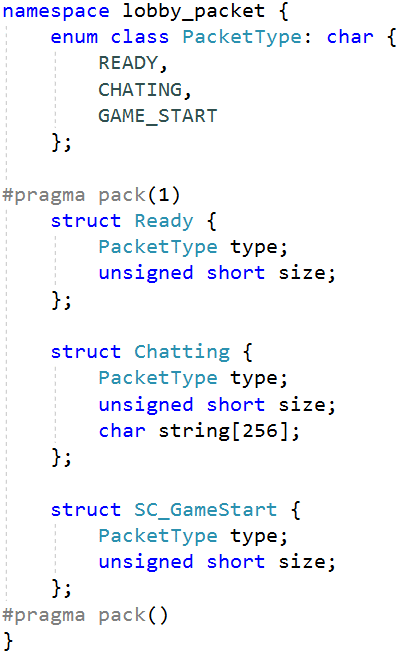


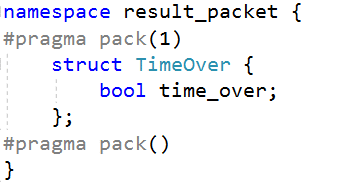
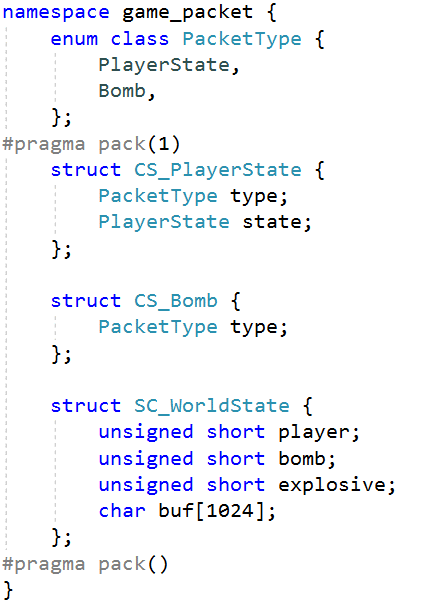
**3-3 객체 모델링**



**3-4 패킷 모델링**







**4. 팀원 별 역할분담**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 윤혜림 | 황신필 |
| 서버 설계 | | O | O |
| 클라이언트 설계 | | O | X |
| 기획서 작성 | | O | O |
| 객체 모델링 | | O | O |
| 패킷 모델링 | | O | O |
| Server | GameCommunication | X | O |
| LobbyCommunication | X | O |
| ResultCommunication | X | O |
| UpdateAndSend | X | O |
| Client | GameFramework | O | X |
| Scene | O | X |
| Renderer | O | X |

**5. 개발 일정**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11월 첫째 주 | 11/02(월) | 11/03(화) | 11/04(수) | 11/05(목) | 11/06(금) | 11/07(토) | 11/08(일) |
| 윤혜림 |  |  |  |  | 설계 및  모델링 | Framework  및 Scene  구현 |  |
| 황신필 |  |  | 설계 및  모델링 |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11월 둘째 주 | 11/09(월) | 11/10(화) | 11/11(수) | 11/12(목) | 11/13(금) | 11/14(토) | 11/15(일) |
| 윤혜림 | ~~Renderer구현~~  Network Communicator 구현 |  | ~~Renderer구현~~  Network Communicator 구현 |  |  | Renderer구현 | Renderer구현 |
| 황신필 | 패킷 전달 구현 | 패킷 전달 구현 | 패킷 전달 테스트 |  | 패킷 전달 테스트 및 수정 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11월  셋째 주 | 11/16(월) | 11/17(화) | 11/18(수) | 11/19(목) | 11/20(금) | 11/21(토) | 11/22(일) |
| 윤혜림 | Renderer구현 |  | ~~Chatting 클라이언트구현~~  Renderer구현 |  | ~~Chatting 클라이언트구현~~  Renderer구현 | Login  Scene  구현 |  |
| 황신필 | 충돌 체크  함수  prototype  제작 | 충돌 체크  함수  prototype  제작 | chatting  서버 제작 |  | chatting  서버 제작 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11월  넷째 주 | 11/23(월) | 11/24(화) | 11/25(수) | 11/26(목) | 11/27(금) | 11/28(토) | 11/29(일) |
| 윤혜림 | Lobby  Scene  구현 | Game  Scene  구현 |  |  |  | Game  Scene  구현 | Game  Scene  구현 |
| 황신필 | Update  구현 | Update  구현 | 동기화 최적화 |  | 동기화 최적화 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 12월  첫째 주 | 11/30(월) | 12/01(화) | 12/02(수) | 12/03(목) | 12/04(금) | 12/05(토) | 12/06(일) |
| 윤혜림 | Result  Scene  구현 |  |  |  | 최적화 및  디버그 | 최적화 및 디버그 | 최적화 및 디버그 |
| 황신필 | 동기화 최적화 |  |  |  | 최적화 및 디버그 | 최적화 및 디버그 | 최적화 및 디버그 |