The Binding of Isaac

(Multi-Player Version)

2017184026 이용석

2018180018 박선윤

**목차**

1. 애플리케이션 기획
2. 게임 기본 정보 ------------------------------------------------------------------ 3
3. 캐릭터 소개 ---------------------------------------------------------------------- 4
4. 게임 플레이 방법 --------------------------------------------------------------- 4
5. 조작 방법 ------------------------------------------------------------------------ 5
6. High-Level Design
7. Game Flow-Chart ---------------------------------------------------------------- 6
8. Login Network ------------------------------------------------------------------- 8
9. Lobby Network ----------------------------------------------------------------- 10
10. InGame Network ---------------------------------------------------------------- 12
11. Low-Level Design
12. Define --------------------------------------------------------------------------- 14
13. Struct & Class ------------------------------------------------------------------- 15
14. Protocol ------------------------------------------------------------------------- 18
15. Function ------------------------------------------------------------------------- 21
16. 팀원 별 역할분담 --------------------------------------------------------------- 23
17. 개발일정 -------------------------------------------------------------------------- 24
18. 개발환경 -------------------------------------------------------------------------- 27
19. 애플리케이션 기획
20. 게임 기본 정보



* 게임 제목 : The Binding of Isaac
* 게임 스토리

: 주인공은 어머니와 단 둘이 살고 있는 아이작. 성실한 기독교 광신자인 어머니는 어느 날 환청을 듣게 되고, 신의 뜻으로 여긴 어머니에 의해 아이작은 학대와 감금에 시달리고 급기야 죽을 처지에 놓인다. 칼을 들고 자신을 죽이려는 어머니를 피해 지하로 도망친 아이작은 지옥 같은 던전에 발을 들이게 되고, 게임이 시작된다.

* 게임 컨셉 : **기존의 싱글 플레이 게임을 멀티 플레이로 재구성한 협동 게임**
* **플레이 방식 : 2인 협동 멀티 플레이 게임**
* 게임 소개 : 맵 중앙에는 보스 몬스터가 위치해 있다. 각 플레이어는 힘을 합쳐 맵에 위치한 보스 몬스터를 쓰러 뜨려야 한다. 몬스터의 공격을 피해 끝까지 살아남아야 한다.

1. 캐릭터 소개



이름 : 아이작 (Isaac)

 체력 : 

무기 : (눈물)

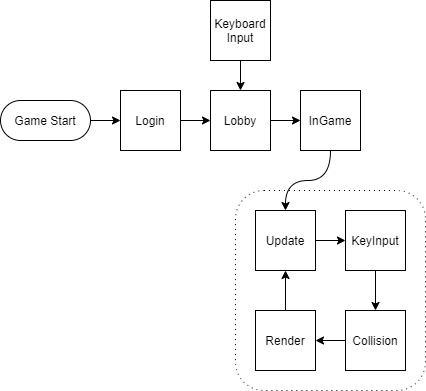
1. 플레이 방법
2. 게임에 접속한다. (서버와 연결)
3. 아이디 입력으로 로그인한다.
4. 팀원의 입장을 기다린다.
5. 모든 팀원이 입장했으면 상단에 위치한 문으로 들어간다.
6. 방향키 조작으로 이동하며 몬스터를 공격한다.
7. 몬스터의 hp가 0이 되면 게임 종료, 플레이어의 승리

플레이어의 hp가 0이 되면 게임 종료, 플레이어의 패배

1. 조작법

|  |  |
| --- | --- |
| W | 위쪽으로 이동 |
| A | 왼쪽으로 이동 |
| S | 오른쪽으로 이동 |
| D | 아래쪽으로 이동 |
| ↑ | 상단 공격 |
| ← | 왼쪽 공격 |
| → | 오른쪽 공격 |
| ↓ | 하단 공격 |

1. High-Level Design
   1. Game Flow-Chart



* + - Login

클라이언트가 서버에 접속한 후, 각 클라이언트에서 사용할 닉네임을 서버에 보낸다. 서버는 각 클라이언트에게 닉네임을 받아 저장한다.

클라이언트가 connect 요청을 하면 서버가 accept로 수락한다. accept 할 때 연결된 클라이언트의 수를 확인하여 최대 2개의 클라이언트만 연결할 수 있도록 조절한다.

* + - Lobby

로그인을 통해 로비로 입장한 각 클라이언트(플레이어)가 키보드 입력으로 이동할 때마다 클라이언트(플레이어)의 변화된 위치를 서버에 전송한다. 서버에서는 전달받은 위치를 다른 클라이언트(플레이어)에게 전송해준다.

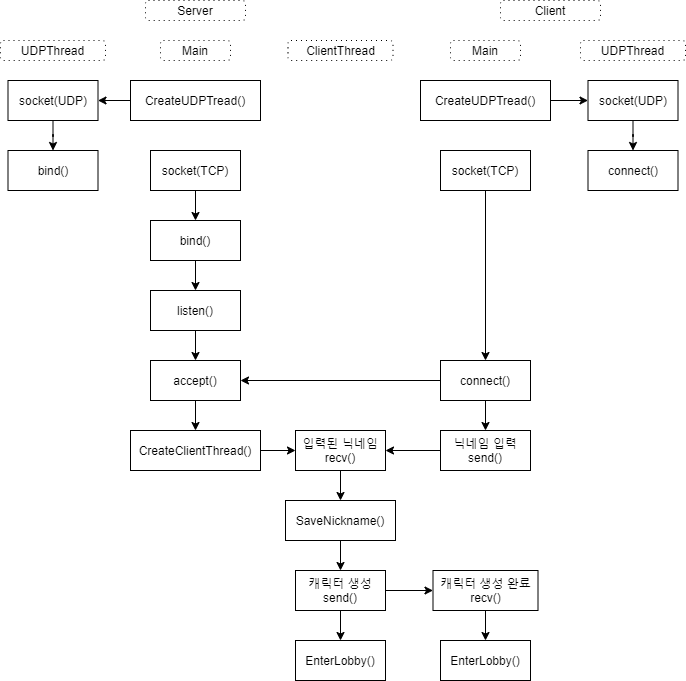
서버에 접속한 모든 클라이언트(플레이어)는 게임을 시작하기 위해 준비 구역으로 이동한다. 모든 클라이언트(플레이어)가 준비 구역으로 이동하면 게임이 시작된다.

* + - InGame

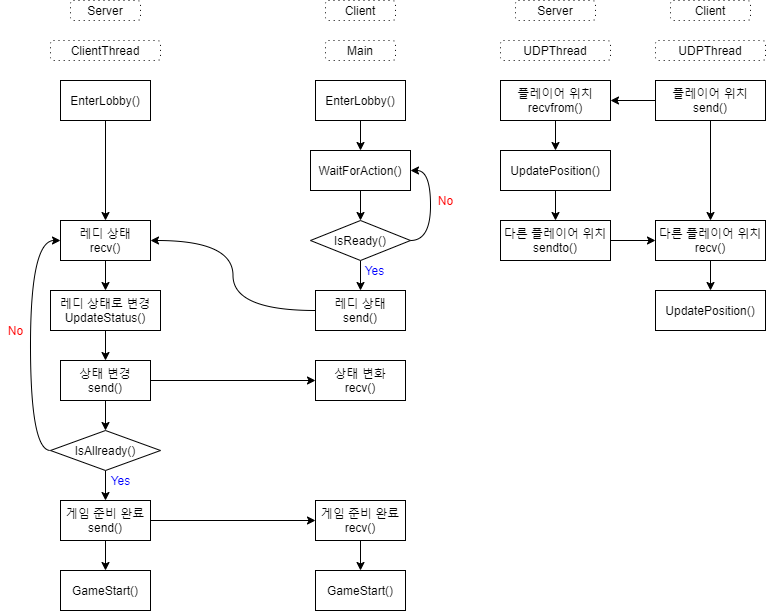
키보드 입력으로 클라이언트(플레이어)의 위치가 변화하면 해당 위치를 서버에 전송한다. 서버에서는 전달받은 위치를 다른 클라이언트(플레이어)에게 전송한다.

캐릭터, 몬스터, 공격 스킬 등의 오브젝트 정보를 실시간으로 업데이트 하고, 업데이트를 기반으로 화면에 렌더링한다.

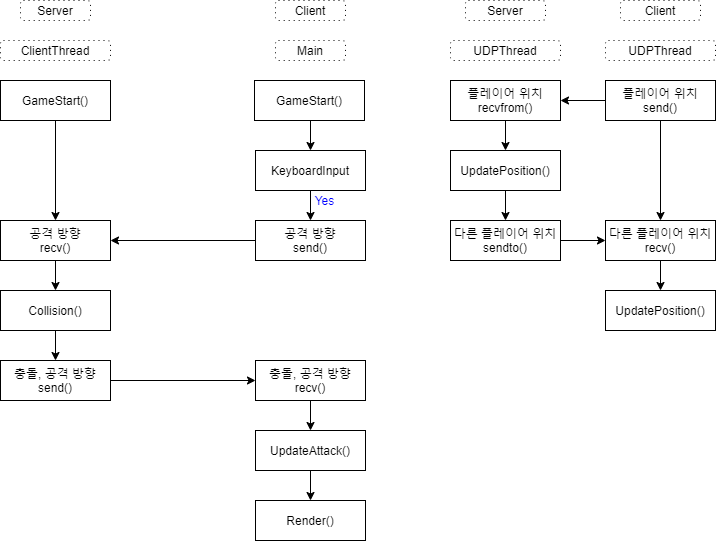
* 1. Login Network



* + - 클라이언트가 서버에 접속 요청을 하면 서버에서 요청을 확인한 후 수락한다. 서버에서 요청 수락으로 연결이 되면 각 클라이언트마다 전용 스레드를 생성한다.
    - 클라이언트에서 게임 내에서 사용할 닉네임을 입력한다. 입력한 닉네임을 서버에 전송하여 저장한다.
    - 닉네임을 성공적으로 저장했다면 서버는 클라이언트에게 캐릭터 생성이 완료되었다는 신호와 함께 Lobby로 입장할 수 있다는 정보를 전달한다.
    - 클라이언트가 성공 신호를 전달받으면 Lobby Network로 넘어간다.
  1. Lobby Network

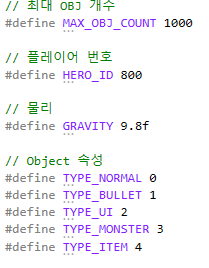


* + - 클라이언트는 키보드 입력으로 로비 맵 내에서 자유롭게 이동할 수 있다.
    - 각 클라이언트(플레이어)의 위치가 이동할 때마다 변화된 위치를 서버에 전송해 다른 클라이언트에게도 해당 정보를 전달하여 변화된 위치를 알려줄 수 있도록 한다.
    - 각 클라이언트는 서버에게 위치 정보를 받아서 위치를 업데이트하고 업데이트 내용을 기반으로 새롭게 변화된 위치를 렌더링한다.
    - 클라이언트가 준비 구역으로 이동하면 해당 클라이언트는 Ready 상태로 변화한다. 상단 UI에서 각 클라이언트의 현재 상태를 확인할 수 있다.
    - 클라이언트의 상태가 변화하면 변화된 상태를 서버에게 전송하고, 서버는 각 클라이언트의 상태를 수집하여 관리한다. 클라이언트의 상태를 관리할 때 닉네임도 함께 저장하여 ‘닉네임-현재상태’의 형태로 관리할 수 있도록 한다.
    - 모든 클라이언트가 Ready 상태라면 접속한 모든 클라이언트에게 게임을 시작해도 된다는 정보를 전달한다.
  1. InGame Network

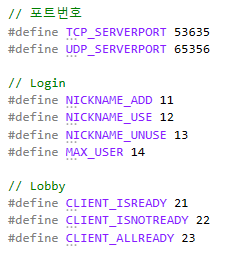
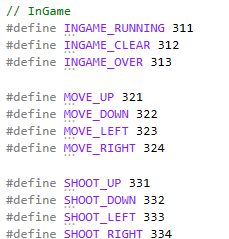


* + - 클라이언트는 키보드 입력으로 로비 맵 내에서 자유롭게 이동할 수 있다.
    - 각 클라이언트(플레이어)의 위치가 이동할 때마다 변화된 위치를 서버에 전송해 다른 클라이언트에게도 해당 정보를 전달하여 변화된 위치를 알려줄 수 있도록 한다.
    - 각 클라이언트는 서버에게 위치 정보를 받아서 위치를 업데이트하고 업데이트 내용을 기반으로 새롭게 변화된 위치를 렌더링한다.
    - 클라이언트가 키보드 입력으로 몬스터를 향해 공격하면 공격 스킬의 위치 정보 또한 서버로 보낸다. 서버로 전송된 스킬 정보는 다른 클라이언트에게도 전달하여 업데이트 후 렌더링할 수 있도록 한다.
    - 이 과정에서 서버는 공격 스킬이 몬스터에게 닿았을 경우 각 클라이언트에게 충돌 정보를 전송해준다. 충돌 시 몬스터의 hp가 감소하고 hp 업데이트로 변화된 내용을 렌더링한다.
    - 몬스터가 공격하는 정보 또한 서버에서 각 클라이언트로 전송하여 몬스터의 공격 스킬 위치를 변화 시 마다 업데이트 하여 새롭게 렌더링할 수 있도록 한다.
    - 몬스터의 공격이 플레이어에게 닿는 경우 또한 서버에서 충돌 정보를 확인한다. 위치 정보를 서버에서 처리하면서 함께 충돌 처리를 하여 몬스터의 공격 스킬과 플레이어가 맞닿았을 경우 해당 정보를 해당하는 클라이언트에게 전송해준다. 충돌한 플레이어의 hp가 업데이트 되어 렌더링할 수 있도록 한다.

1. Low-Level Design
2. Define

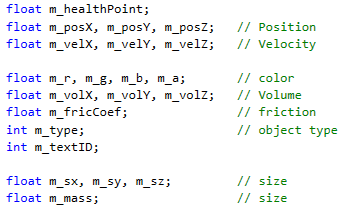


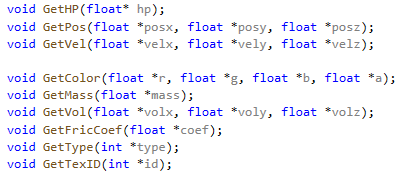
클라이언트 사용 define 상수

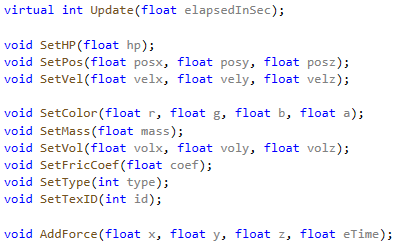


서버 & 클라이언트 공동 사용 define 상수

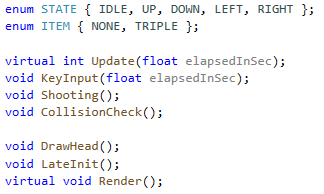
1. Struct / Class
   * + Obj Class



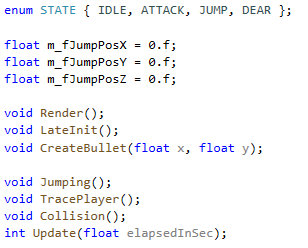




* + - Player Class



* + - Boss Class



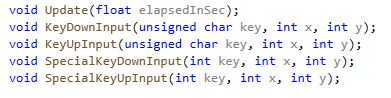
* + - Bullet Class

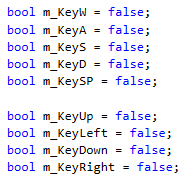


* + - HP Class

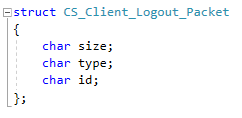


* + - ScnMgr(SceneManager) Class

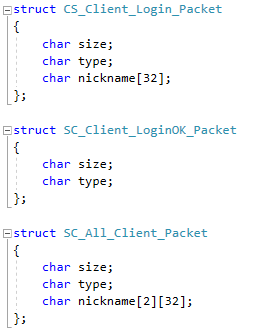


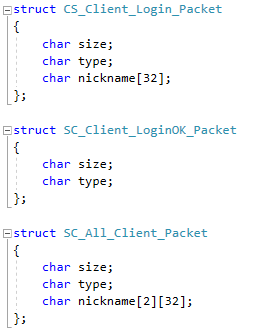


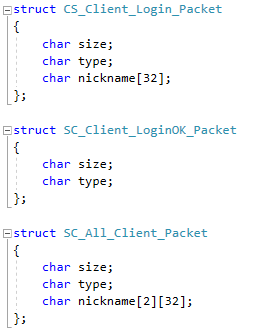
1. Protocol
   * + Login & Lobby

 접속 종료한 클라이언트를 서버에게 알리는 패킷

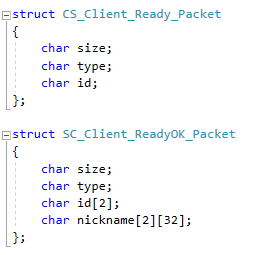
* + - Login

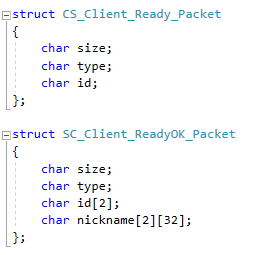
 클라이언트가 서버에 로그인함을 알리는 패킷

 서버에 정상적으로 로그인했음을 알리는 패킷

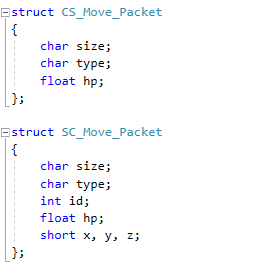
 로그인한 모든 클라이언트의 닉네임을 보내주는 패킷

* + - Lobby

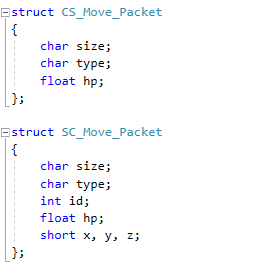
 클라이언트의 레디 상태를 서버에게 알리는 패킷

 모든 클라이언트의 상태를 보내주는 패킷

* + - Lobby & InGame

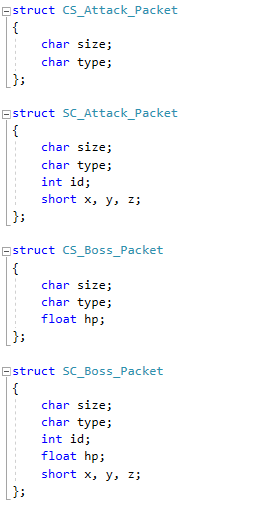
 클라이언트의 키보드 입력에 따른 움직임을 감지하여

클라이언트의 위치 정보를 보내주는 패킷

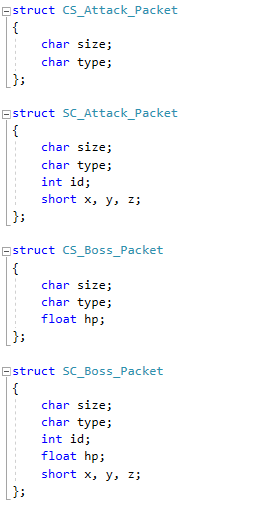
 전달받은 클라이언트 위치 정보를 업데이트하고

충돌 처리하여 새로운 위치 정보를 알려주는 패킷

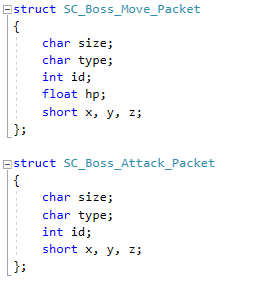
* + - InGame

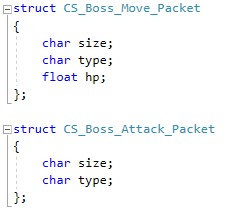
 클라이언트의 키보드 입력에 따른 공격 스킬 위치를

서버로 보내주는 패킷

 전달받은 클라이언트의 스킬 정보를 업데이트하고

충돌 처리하여 새로운 위치 정보를 알려주는 패킷





보스의 움직임을 감지하여 위치 및 전달받은 정보를 업데이트하고 충돌 처리하여

스킬 정보를 서버로 보내주는 패킷 새로운 정보를 클라이언트에게 알려주는 패킷

1. Function
   * + Client

(Login) void CreateNickName() : 닉네임 입력

(Login) void SendNickName() : 입력한 닉네임을 서버에 전송

(Login) void AddNickname() : 다른 클라이언트의 닉네임을 추가

(Login) void EnterLobby() : 로그인이 완료되면 로비로 입장

(Lobby) void SendIsReady() : Reday 상태를 서버로 전송

(Lobby) void SendIsNotReady() : UnReady 상태를 서버로 전송

(Lobby) void ReadyOK() : 모든 플레이어의 Ready 상태를 확인

(Lobby) void GameStart() : 모두 Reday 상태이면 게임 시작

(InGame) void SendAttack() : 공격에 대한 정보를 서버에 전송

(InGame) void SendMove() : 이동에 대한 정보를 서버에 전송

* + - Server

(Login) void SaveNickName() : 전송된 닉네임을 저장

(Login) void SendNickName() : 저장한 닉네임을 전송

(Lobby) void UpdateStatus() : 클라이언트의 상태를 ‘Ready’로 변경

(Lobby) void SendAllStatus() : 모든 클라이언트의 현재 상태를 전송

(Lobby) void IsAllReady() : 모든 클라이언트가 Ready 상태임을 전송

(InGame) void BulletRoute() : 공격 스킬의 거리를 계산

(InGame) void Collision() : 위치와 스킬의 충돌체크 후 정보 전송

(InGame) void UpdatePosition() : 객체의 위치를 업데이트

(Thread) DWORD WINAPI ProcessClient()

: 클라이언트 접속 시에 생성되는 스레드 함수

1. 팀원 간 역할분담

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | 내용 | 상세 설명 | 담당 |
| **Client** | **Collision** | 충돌 체크 알고리즘 수정 | 이용석 |
| **Player Class** | 내용 수정 | 박선윤 |
| **Boss class** | 내용 수정 | 이용석 |
| **Hp Class** | 내용 수정 | 박선윤 |
| **Object Class** | 내용 수정 | 이용석 |
| **Global Header** | Define 추가 | 박선윤 |
| **Login** | UI 제작 | 박선윤 |
| **Network** | 서버와 통신을 위한 Class | 이용석 |
|  | | | |
| **Server** | **Server Framework** | 서버 프레임워크 제작 | 이용석박선윤 |
| **Protocol** | 프로토콜 구현 | 이용석박선윤 |
| **Synchronization** | Mutex 사용 | 이용석 |
| **Login** | 닉네임 입력 구현 | 박선윤 |
| **Lobby** | 로비 통신을 위한 함수 구현 | 이용석 |
| **InGame** | 인게임 함수 구현 | 박선윤 |

1. 개발 일정

|  |  |  |  |
| --- | --- | --- | --- |
| 날짜 | 이용석 | 박선윤 | |
| 10/27 화 | 기획서 작성 | | |
| 28 수 |
| 29 목 |
| 30 금 |
| 31 토 |
| 11/1 일 |
| 2 월 | 기획서 정리 | | |
| 3 화 | 기획서 피드백 | | |
| 4 수 |  | | |
| 5 목 | 서버 프레임워크 제작 | | |
| 6 금 |
| 7 토 |
| 8 일 |
| 9 월 |  | | |
| 10 화 | Protocol 구현 | | |
| 11 수 | (Login) void SaveNickName() | (Login) void SendNickName() | |
| 12 목 | 1차 점검 | | |
| 13 금 | Boss Class 수정 | Hp Class 수정 | |
| 14 토 |  | | |
| 15 일 | Collision 함수 수정 | Player Class 수정 | |
| 16 월 | (Login) void AddNickname()  (Login) void EnterLobby() | (Login) void CreateNickName()  (Login) void SendNickName() | |
| 17 화 |  | | |
| 18 수 | Network Class 구현  (Lobby) void SendIsReady()  (Lobby) void SendIsNotReady() | Login Network 구현  (Lobby) void ReadyOK()  (Lobby) void GameStart() | |
| 19 목 | 2차 점검 | | |
| 20 금 | Lobby Network 구현  (Lobby) void UpdateStatus() | Login Network 구현  (Lobby) void SendAllStatus()  (Lobby) void IsAllReady( | |
| 21 토 |
| 22 일 |  | | |
| 23 월 | 동기화 | InGame Network 구현 | |
| 24 화 | (InGame) void SendAttack() | |
| 25 수 |  | (InGame) void SendMove() | |
| 26 목 |  | | |
| 27 금 | 3차 점검 | | |
| 28 토 | (InGame) void BulletRoute() | |  |
| 29 일 |  | | (InGame) void UpdatePosition() |
| 30 월 | (InGame) void Collision() | |  |
| 12/1 화 | 오류 수정 및 추가 구현 작업 | | |
| 2 수 |  |  | |
| 3 목 | 최적화 | | |
| 4 금 |  |  | |
| 5 토 | 최종 점검 | | |
| 6 일 |
| 7 월 | 피드백 | | |
| 8 화 | 프로젝트 제출 | | |

1. 개발 환경
   1. OS : Windows 10
   2. IDE : Visual Studio 2019
   3. API : OpenGL / Windows Socket API(WSA)
   4. 개발 언어 : C / C++