Network Game Programming

Term Project 추진 계획서

2012182039 정홍래

2013182043 최아도

2013182003 곽범식

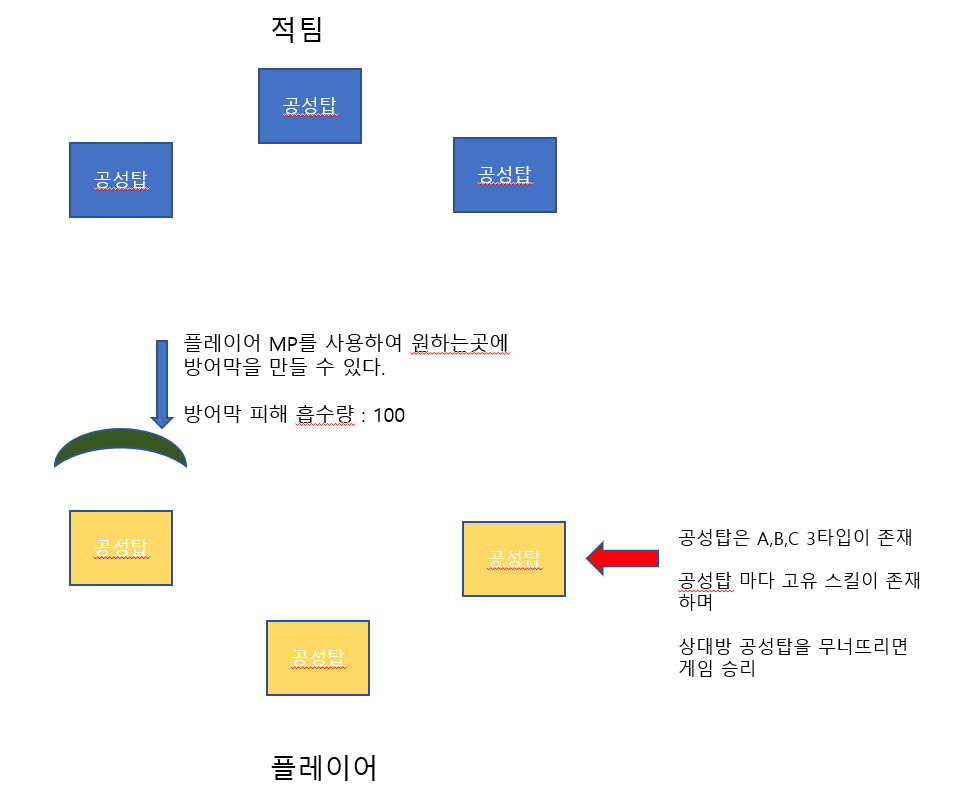
목차

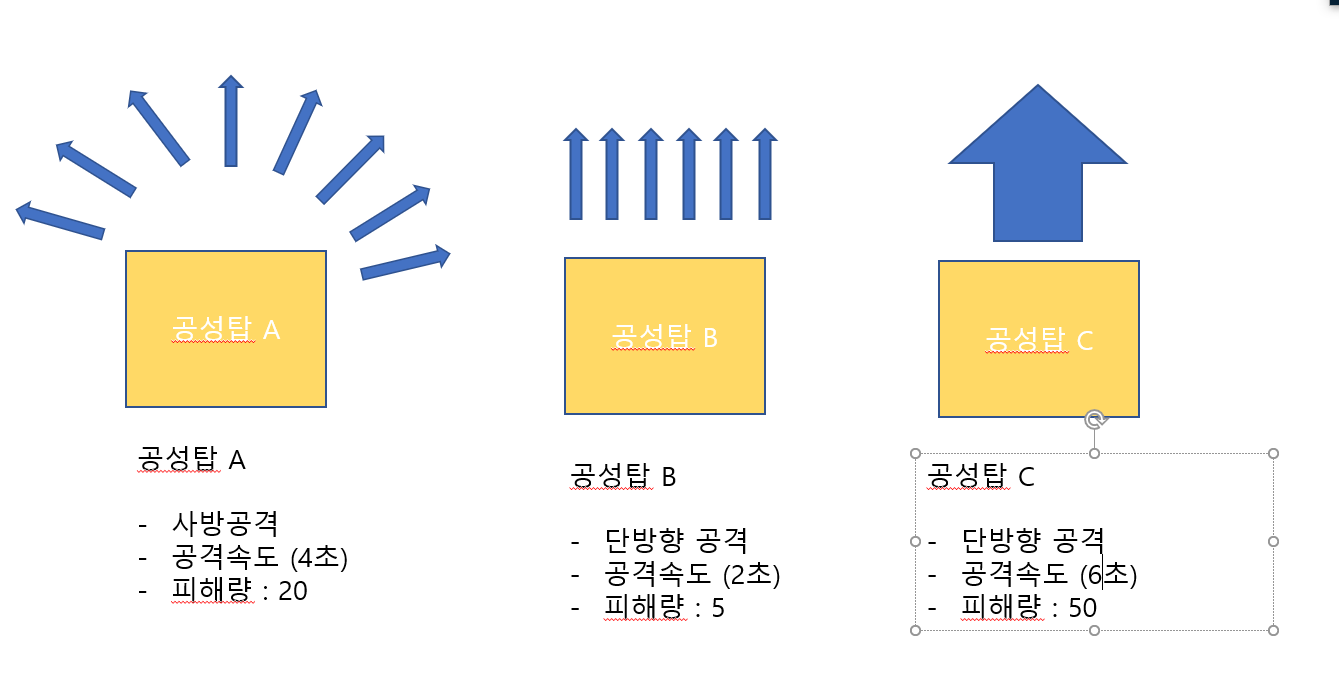
1. **게임 소개**
   1. 게임 제목2
   2. 게임 장르3
   3. 게임 설명2
   4. 게임 조작법3
   5. 게임 흐름 예상도2
2. **요구 사항 분석4**
   1. 기능 요구사항5
3. **High-level 디자인**
   1. 클라이언트 흐름도2
   2. 서버 흐름도3
   3. 스레드 동기화 방식2
4. **Low-level 디자인**
   1. 클라이언트 구조체, 함수2
   2. 서버 구조체, 함수3
5. **팀원 역할분담**
   1. 정홍래2
   2. 최아도3
   3. 곽범식2
6. **개발 환경**
7. **개발 일정**
8. **게임 소개**
   1. 게임 제목 - Dot War
   2. 게임 장르 - RTS
   3. 게임 설명

RTS장르의 게임으로 상대방 진형을 깨 부수어 승리를 거두는 방식이다. 플레이어는 자신의 진형에 3가지 종류의 공성탑을 지을 수 있으며 총 3개 지을 수 있다.

게임 시작 전에는 상대방의 공성탑 배치를 보지 못하지만, 게임이 시작하면 상대방 공성탑 배치를 확인하게 되며 게임을 진행한다.

상대방 공성탑 배치를 유의하며 자신의 공성탑에 적절한 스킬을 사용하여 상대방 모든 공성탑을 무너뜨리면 게임은 승리한다.





\*공성탑의 공격속도, 피해량, 체력(아직 미정)은 게임 난이도 조절하면서 수정할 가능성이 존재

* 1. 게임 조작법

<게임 시작 전>

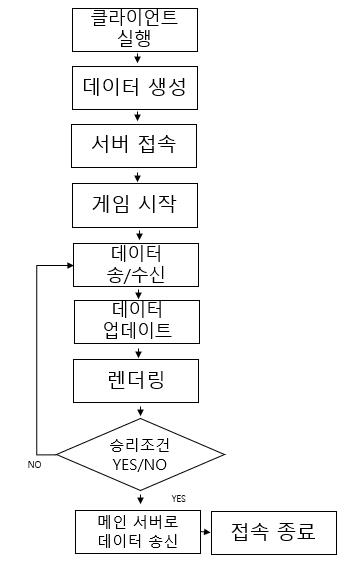
Space : 공성탑 배치

A키 : 공성탑 고르기

<게임 시작 후>

마우스 클릭 : 방어막 설치

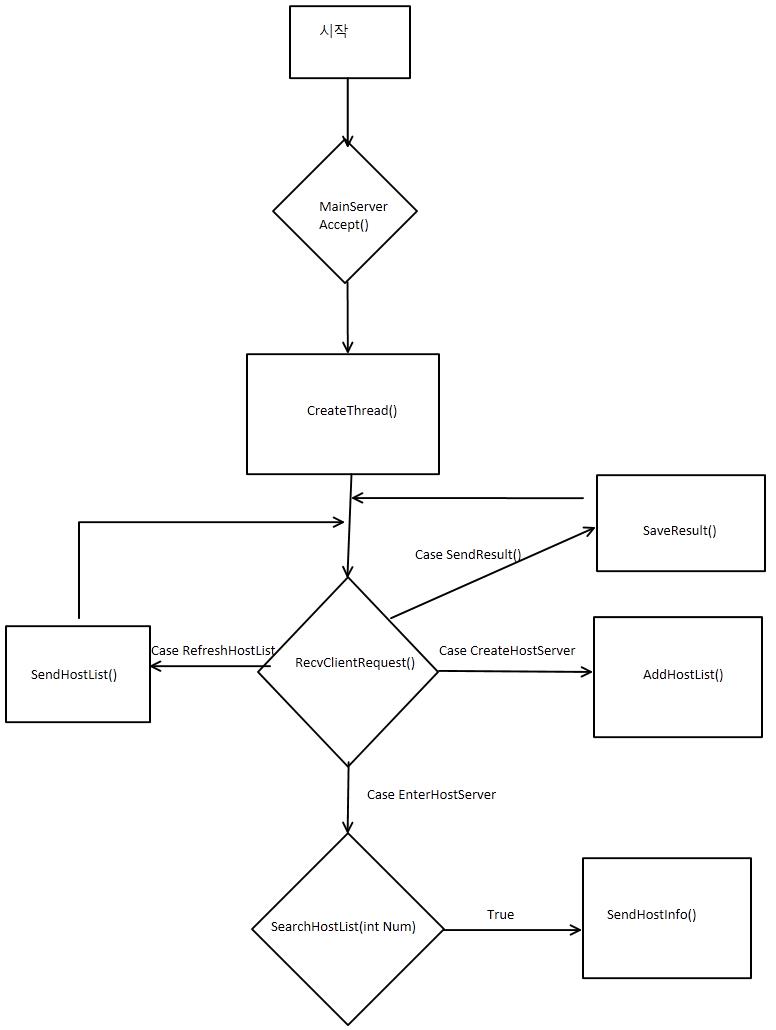
* 1. 게임 흐름 예상도



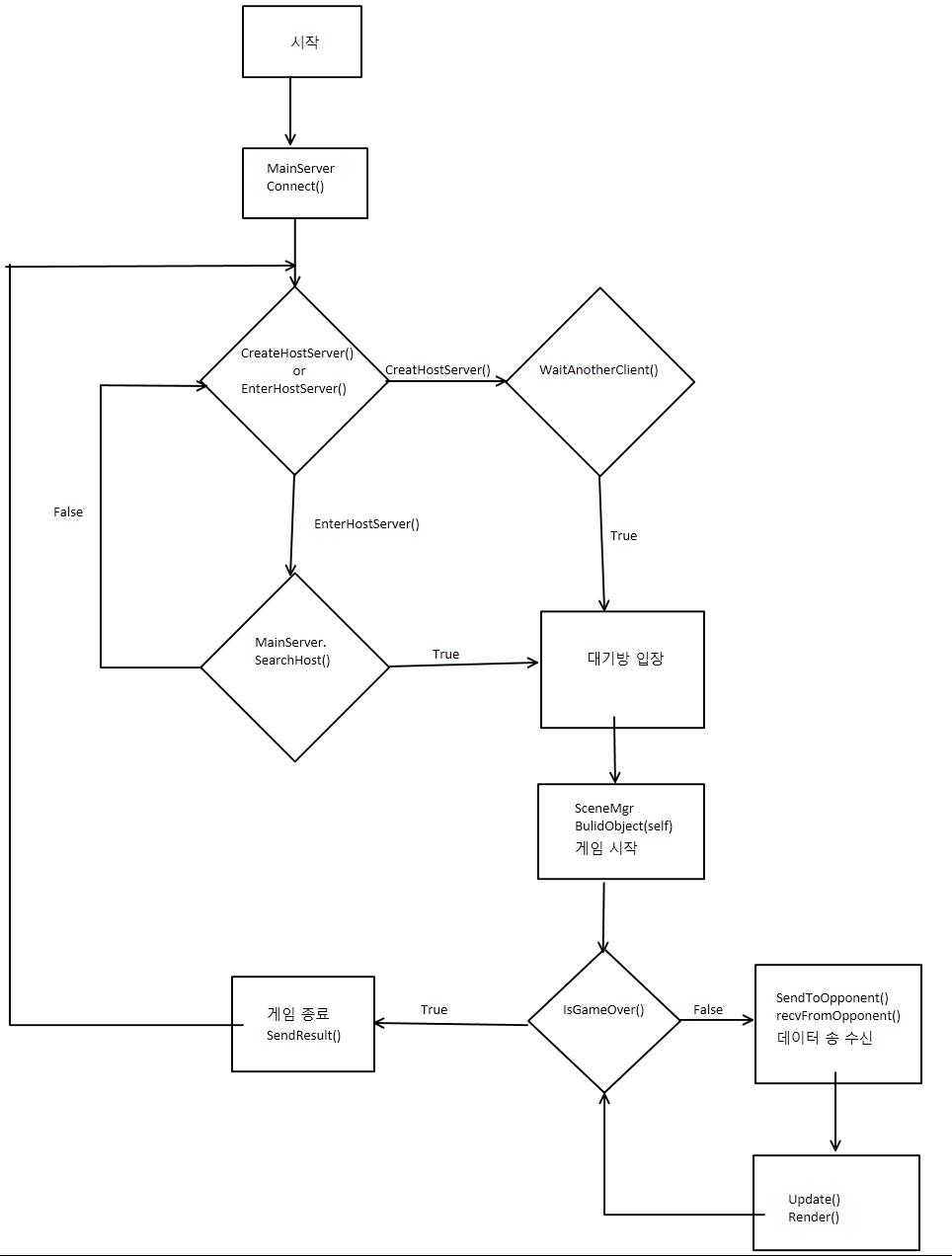
1. **요구 사항 분석**
   1. 기능 요구 사항

* 네트워크 프로토콜은 TCP 서버를 사용한다.
* 서버는 n개의 클라이언트를 수용한다.
* 최대 개수가 정해진 대기방을 생성할 수 있다. (생성시 클라이언트가 간이 서버를 생성 : Host)
  + 대기방을 만들면 만든 대기방 정보 ( 방장의 IP주소, 포트번호 등 서버에 전송)
* 만들어진 대기방에 다른 클라이언트가 접속시 대기방 입장(connect)
  + 서버에서 만들어진 방의 목록을 받아와서 IP, 포트번호 로 연결
* 게임이 종료 될 경우 결과 서버에 저장 ( 승 / 패 )
  + But 연결이 강제로 끊겼을 경우, 끊어진 쪽의 패배로 기록할 수 있어야함.
* 서버는 멀티스레드를 이용한 이벤트 처리로 동기화 과정을 거친다.
  + 동시에 대기방에 입장하려는 클라이언트 간의 충돌이 일어날 수 있다.
* 방장은 상대편의 데이터를 recv 한 후 update 한 결과물을 다시 send한다.
  + 참가자는 받은 데이터를 기반으로 렌더링을 하며, 업데이트 하는 데이터를 보낸다.

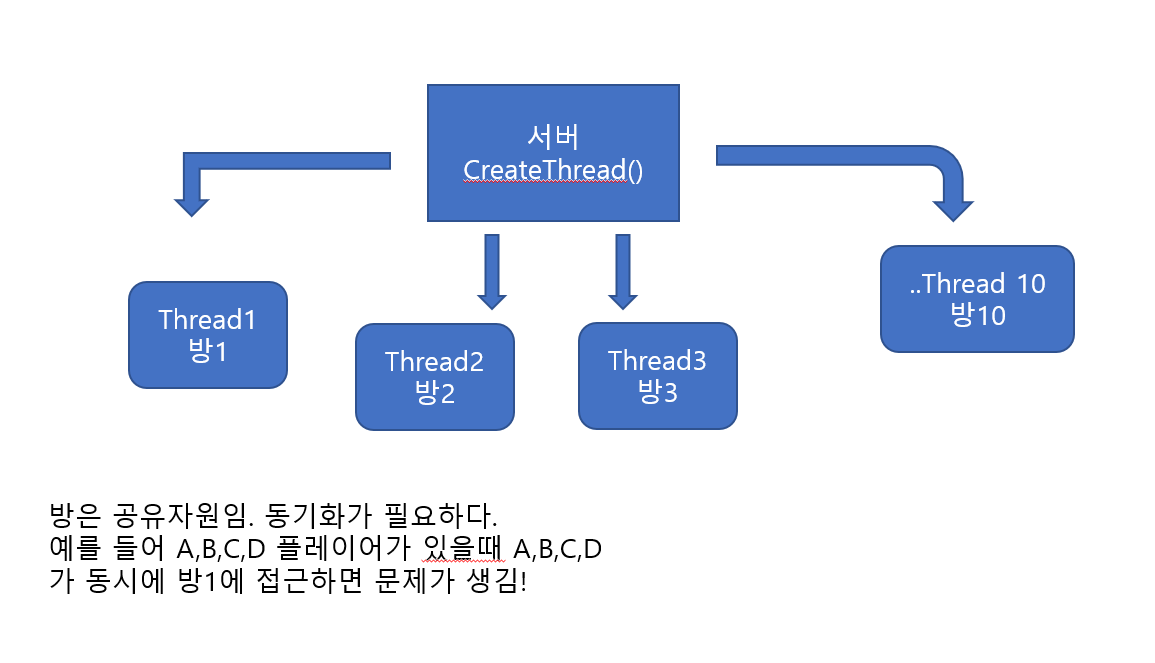
1. **High-level 디자인**
   1. 서버 흐름도



* 1. 클라이언트 흐름도



* 1. 스레드 동기화 방식

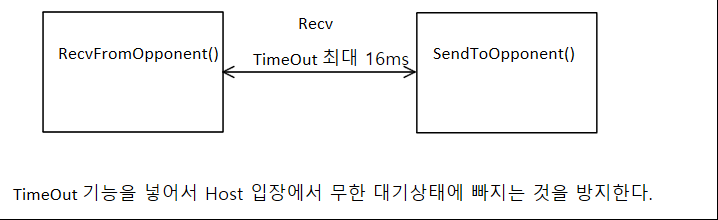


그 외에도 동시에 HostServer를 생성할 때

HostList 배열의 같은 위치에 값을 저장하게 될 수도 있기 때문에

임계영역을 사용한다.

* 1. 클라이언트 통신에서의 데드락 발생 가능성



* + - Host가 죽어버리면 같이 죽기 때문에 주로 접속자의 데이터 관련

1. **Low-level 디자인**
   1. 클라이언트 구조체, 함수

* 클라이언트 구조체

typedef float3 { float x, float y, float z }float3, vector3, position;

투사체의 속도, 객체의 위치 정보 등을 저장

Typedef float4 { float x, float y, float z }float4, color;

객체의 R,G,B,A 값을 저장

* 클라이언트 함수

Bool MainServerConnect(SOCKET s)

s소켓으로 메인 서버에 연결시도 => 성공 시 True를 반환

Bool CreateHostServer(SOCKET s, char \*name)

S 소켓을 사용하여 새로운 1:1통신을 지원해주는 서버를 만들어주며, 그 서버는

name이라는 이름을 가진 대기방을 만든다 => 성공 시 True를 반환 뒤 씬 전환

Bool EnterHostServer(int number)

만들어진 number번의 대기방에 참가를 시도한다 => 성공 시 True를 반환한 후 씬 전환

실패요인 : HostServer의 정원인 2명이 가득 차있는 상태, HostServer가 사라진 상태에서 실패 한뒤 False를 반환

Void WaitAnotherClient()

또 다른 클라이언트의 접속 요청을 기다린다.

Bool RefreshHostList()

만들어진 대기방 목록을 받아온다 char\* 형의 대기방 이름과, int 형태의 번호를 받는다.

=> 성공 시 True를 반환

Void SendToOpponent(SOCKET s)

s소켓을 통해 상대방에게 자신의 변경사항을 전달한다.

Void RecvFromOpponent(SOCKET s)

s소켓을 통해 상대방으로부터 변경사항을 전달 받는다.

Bool SendResult()

게임의 결과를 메인 서버에 전달한다.

Bool IsChangeClient()

클라이언트의 변경사항을 체크한다. 변경사항이 존재하면 True를 리턴

* 1. 서버 구조체, 함수
* 구조체

Host { SOCKADDR\_IN ip, char name[20], int number }

생성된 호스트 서버를 저장해 주기 위한 구조체

Player { SOCKADDR\_IN ip, int win, int lose }

클라이언트의 전적을 저장해주는 구조체

* 서버 함수

DWORD WINAPI RecvClientRequest(LPVOID sock)

멀티 쓰레드를 지원하기 위한 쓰레드 함수로 클라이언트 소켓을 인자로 받는다.

Recv를 할때에 받아오는 Case값으로 요청에 맞는 함수를 호출해 준다.

Case SendResult: SaveResult()

Case CreateHostServer(): AddHostList

Case EnterHostServer(): SearchHostList() -> SendHostInfo()

Case RefreshHostList(): SendHostList()

Void SaveResult(bool result)

result 값이 True면 Host에게 1승 참가자에게 1패를 누적한다.

Host AddHostList(char \*name)

HostServer를 생성한다는 요청을 받으면 대기실의 이름을 받아서 Host의 Ip와 비어있는 번호를 할당해서 구조체를 만든 후 리턴해준다.

Return 값 = 새로 생긴 HostServer 정보가 담긴 구조체

Bool SearchHostList(int num)

Num 번호의 Host를 검색해서 만약 존재한다면 True를 리턴해준다.

void SendHostInfo(int num)

특정 방 번호를 가진 HostServer의 Ip,포트번호를 Send해준다.

Void SendHostList()

서버에 등록된 Host의 번호, 이름을 send()해준다.

1. **팀원 역할분담**
   1. 정홍래 ( PM )

* 메인 서버의 Class 구성
* 클라이언트 제작(Object 생성관련)
* 주로 메인 서버 데이터 송 수신 관리
  1. 최아도
* 클라이언트 제작(충돌처리, 렌더러, Object관련)
* 주로 Host 서버 데이터 송 수신 관리
  1. 곽범식
* 클라이언트 제작(대기방, 씬 관련)
* 주로 클라이언트 내부의 데이터 송 수신 관리

1. **개발 환경**

* Visual Studio 2017 c, c++, openGL
* Github

1. **개발 일정**

|  |  |  |  |
| --- | --- | --- | --- |
| **날짜** | **정홍래** | **최아도** | **곽범식** |
| **11월 9일** | 개발 환경 통합 | | |
| **11월 10일** | 게임 프레임워크 제작 | | |
| **11월 11일** | 게임 프레임워크 제작 | | |
| **11월 12일** | 클래스(Building,Enemy Scene) 구현 | | |
| **11월 13일** | 클래스(Building,Enemy Scene) 구현 | | |
| **11월 14일** | 데이터 송수신 상태 확인(간단한 정보 보내 보면서 테스트) | | |
| **11월 15일** | 충돌 처리 | | 건물 편집화면 제작 |
| **11월 16일** | 실제 보낼 데이터 정보 전송 | 충돌처리 미흡한 부분 없는지  없으면 실제 보낼 데이터 정보 전송 | 승리 조건 구현 |
| **11월 17일** | 구조체 오퍼레이터 및 Const값 점검  오류 확인 (계속 테스트) | | |
| **11월 18일** | 코드 통합 및 클라이언트 완성, 중간 점검 | | |
| **11월 19일** |  | | |
| **11월 20일** | 서버 – 기획한 내용을 구체화 하기위한 회의  버그 수정 및 서버 제작에 앞서 문법 정리 | | |
| **11월 21일** | MainServerAccept() | MainServerConnect() | CreateHostServer() |
| **11월 22일** |
| **11월 23일** | RecvClientRequest() | SearchHostServer() | WaitAnotherClient() |
| **11월 24일** |
| **11월 25일** |  | | |
| **11월 26일** | SendHostList() | SendHostInfo() | AddHostList() |
| **11월 27일** |
| **11월 28일** | 내용 통합 및 중간 점검 | | |
| **11월 29일** |
| **11월 30일** |  | | |
| **12월 1일** |  | | |
| **12월 2일** | SendToOpponent() | RecvFromOpponent() | SaveResult() |
| **12월 3일** |
| **12월 4일** |  | | |
| **12월 5일** | 코드 통합 및 디버깅 | | |
| **12월 6일** |
| **12월 7일** | 쓰레드 동기화 작업 | | |
| **12월 8일** | 쓰레드 동기화 작업 | | |
| **12월 9일** | 테스트 및 버그 수정 | | |
| **12월 10일** | 미흡한 부분 보완 | | |
| **12월 11일** |
| **12월 12일** | 최종 테스트 | | |
| **12월 13일** | 오류 및 버그 수정 | | |
| **12월 14일** |
| **12월 15일** |
| **12월 16일** |  | | |
| **12월 17일** |  | | |
| **12월 18** | **최종 프로젝트 발표** | | |