네트워크 게임 프로그래밍  
 Term Project

**-알케미스트-**

2011182007 게임공학과 김영서

20131560 소프트웨어 공학과 민웅기

2013156037 소프트웨어 공학과 정낙현

2017.11.8

목차

애플리케이션 기획

개요

ui/ux 기획

요구사항 분석

High Level 디자인

전체 Flowchart

Client측면 Flowchart

Server측면 Flowchart

쓰래드 동기화

Low Level디자인

데이터 통신을 위한 구조체

Client 함수

Server함수

팀원 역할 분담

개발 환경

개발 일정

## 애플리케이션 기획

### 개요

게임 제목:

알케미스트

장르:

탄막슈팅 디펜스게임

컨셉:

사방의 랜덤 한 방향에서 적들이 출현하며 PC(Player Character)는 탄막을 발사하여 적을 제거한다.

이때 적을 처치하는게 늦어 필드에 일정 수치 이상의 적들이 소환되면 패배로 정한다.

High Score기록을 목표로 하는 게임이다.

규칙:

플레이어는 키보드를 조작하여 캐릭터조작을 제어할 수 있다.

적은 초기에 1초마다 랜덤한 방향의 외곽에서 생성되며 시간이 지날수록 생성주기가 빨라진다.

적이 필드에 100개 이상 존재하면 게임오버가 되고 High Score가 기록된다.

멀티플레이시 접속자수에 비례하여 몬스터 생성주기가 더 빨라진다.

멀티플레이시 모든 유저가 라이프를 잃으면 GameOver처리되며 팀 단위로 Leader보드에 기록된다.

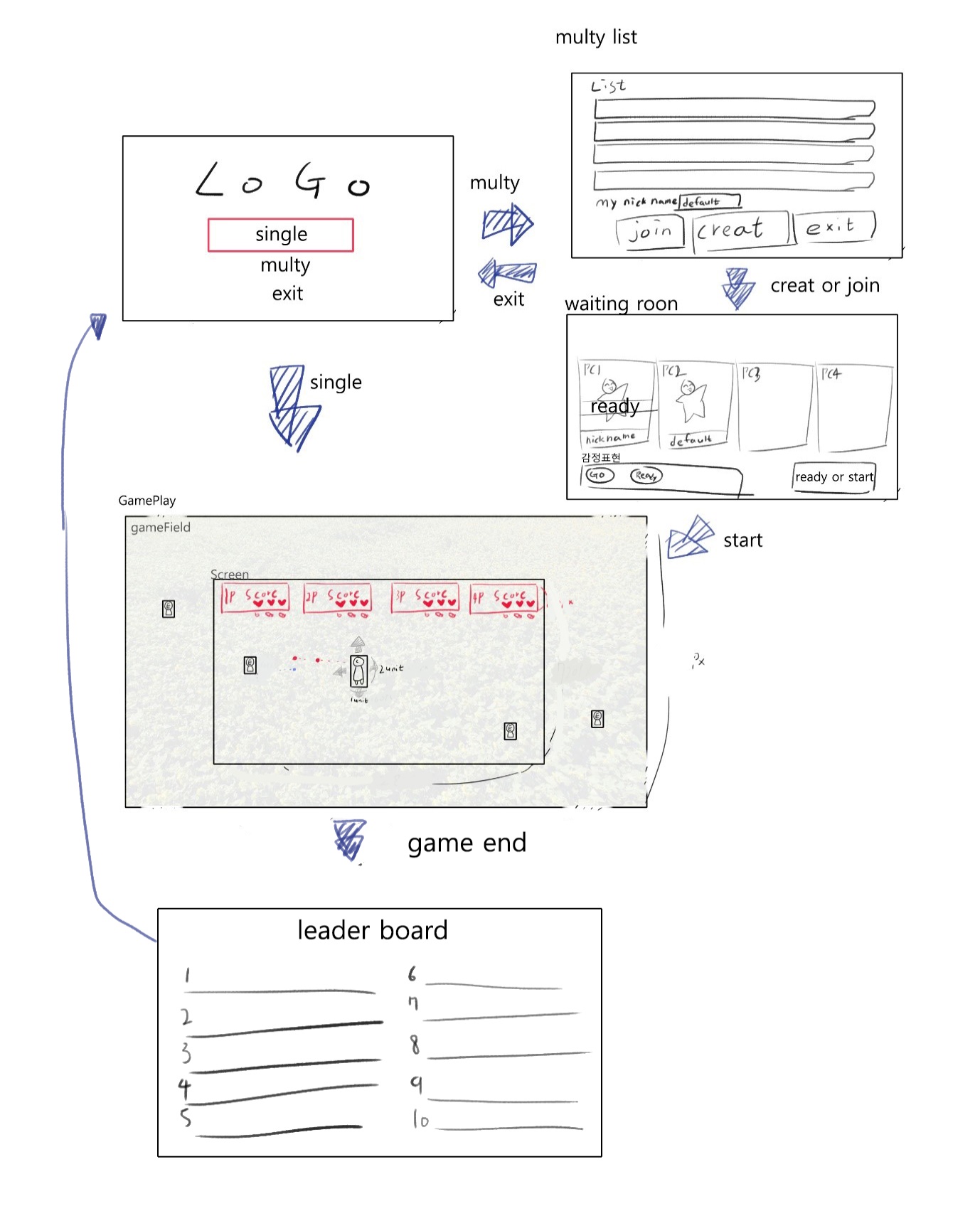
게임의 조작법:

상하좌우 방향키를 이용하여 캐릭터를 상하좌우로 이동시킬 수 있다.

z를 눌러 총알을 발사할 수 있다.

x를 눌러 폭탄은 사용할 수 있다.

### Ui/ Ux 기획서:



해상도: 1600\*900 (16:9 비율)

1unit: 1280 / 16\*1.5

필드크기: 해상도 \*2

캐릭터: 1\*2 unit

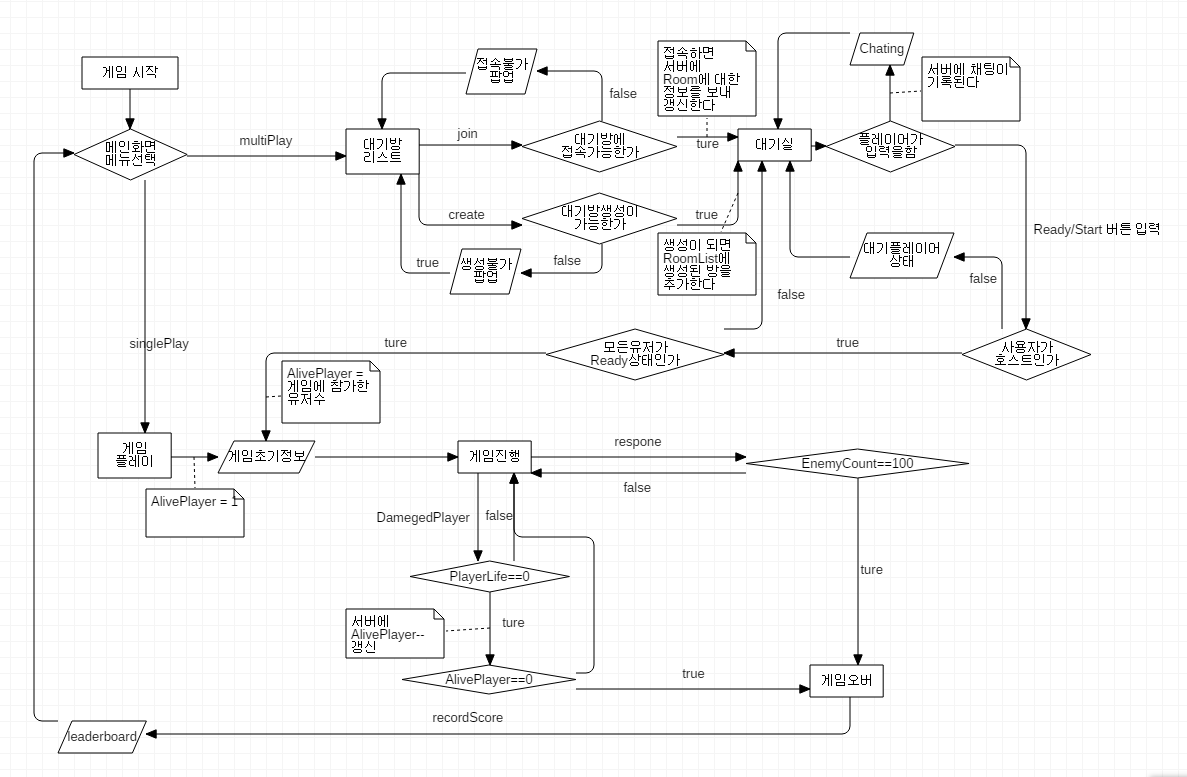
적: (2/3)\*1unit

### 요구사항 분석

* 통신 프로토콜은 경우에 따라 TCP와 UDP를 선택적으로 사용한다.
* 수용인원 : ~~4개의 방에 4명씩 총 16명~~ => 1개방 3명
* 서버의 IP주소와 포트번호는 고정되어 있다고 가정하여 서버IP와 포트번호를 로컬에 text파일을 읽어서 이용한다. = 테스트 용이
* 싱글플레이는 data를 client에서 모두 처리하지만 멀티플레이에서는 data를 서버로 옮겨 처리한후 같은 방의 player에게 보내준다.
* ~~teamScore는 멀티 쓰레드에서 임계영역 방식을 사용하여 동기화한다.~~teamScore산정방식 변경으로 인해 임계영역 사용 X.
* client와 server간 통신에서 프레임 주기에 대해 동기화 하기위해 이벤트를 사용한다.  
  fps주기중 사용자가 눈치 못 채는 최대한 낮은 주기

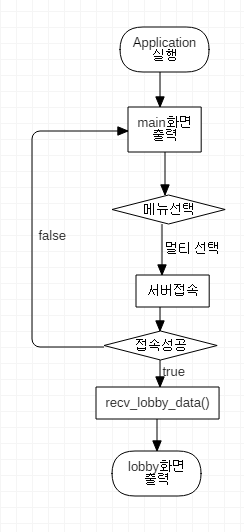
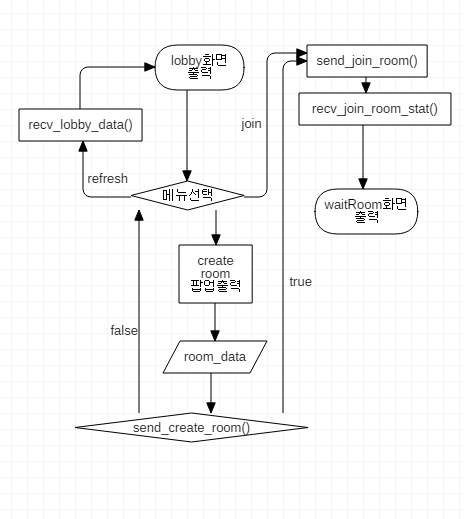
## High-level 디자인

### 전체 Flowchart

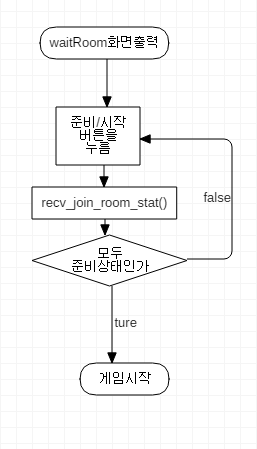
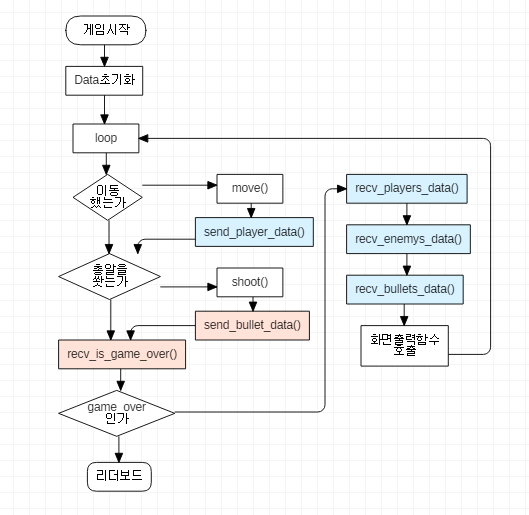


### client측면 Flowchart

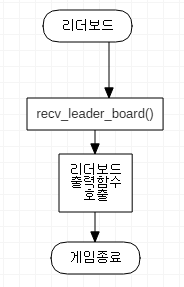
Application lobby

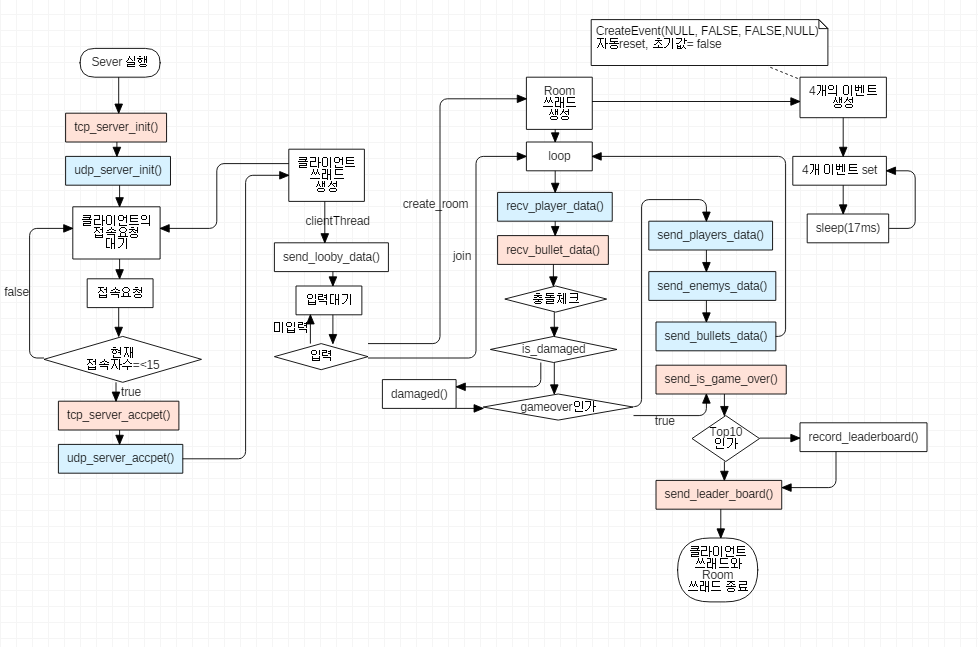
waitRoom gameStart

leaderBoard

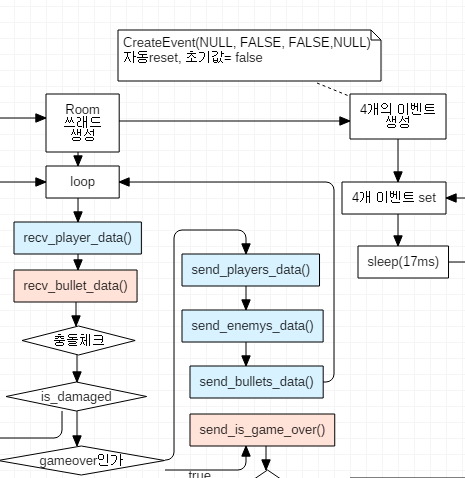


### server측면 Flowchart



### 쓰레드 동기화

쓰레드 이벤트



클라이언트에서 1초에 60번 주기로 데이터를 보내므로 서버도 1초에 60번 데이터를 받도록 해줘야 한다.

loop문에 wait\*를 걸어주면 한바퀴 돌때마다 자동reset때문에 멈추게 된다.

17ms마다 이벤트를 set함으로서 17ms마다 1번씩 loop문을 돌 수 있게 되어 1초에 60번에 데이터를 받게 된다.

플레이어가 감지하지 못한다면 주기를 늦춰도 된다.

## Low-level 디자인

### 데이터 통신을 위한 구조체

struct Point

|  |  |  |  |
| --- | --- | --- | --- |
| type | 이름 | 설명 | 범위 |
| int | pos\_x | x좌표 값 | Filed x축 넓이 |
| int | pos\_y | y좌표 값 | Filed y축 넓이 |

struct PlayerData

|  |  |  |  |
| --- | --- | --- | --- |
| type | 이름 | 설명 | 범위 |
| string | player\_name | Player의 닉네임 |  |
| byte | player\_number | 서로 다른 User가 조작하는 Player를 구분하기 위한 넘버링 | 1~16 |
| struct Point | player\_pos | Player의 포지션 | - |
| byte | direction | Player가 바라보는 방향  12시 방향부터 시계방향으로 4방위 | 1~4 |
| byte | life | Player의 남은 목숨 수 | 0~3 |
| byte | bomb | Player의 남은 폭탄 수 | 0~3 |
| bool | is\_damaged | 일정시간내에 피해를 받았는지 여부 | true, false |
| unsigned long | player\_score | 플레이어가 습득한 점수 | unsigned long |

struct EnemyData

|  |  |  |  |
| --- | --- | --- | --- |
| type | 이름 | 설명 | 범위 |
| struct Point | enemy\_pos | Enemy의 포지션 | - |
| byte | direction | Enemy가 바라보는 방향  12시 방향부터 시계방향으로 4방위 | 1~4 |

struct Bullet

|  |  |  |  |
| --- | --- | --- | --- |
| type | 이름 | 설명 | 범위 |
| struct Point | start\_pos | 발사된 위치 | 필드내 |
| byte | direction | 총알이 향하는 방향 | 1~16 |
| byte | speed | 총알이 향하는 speed | 1~10 |
| float | shot\_time | 총알을 발사한 시간 |  |
| byte | shoter | 총알을 발사한 개체  0: enemy 1~ :player\_number |  |

struct RoomData

|  |  |  |  |
| --- | --- | --- | --- |
| type | 이름 | 설명 | 범위 |
| byte | room\_number | 방에 설정된 번호 | 1~4 |
| byte | host\_number | 방의 Leader인 Player의 player\_number | player\_number |
| String | room\_name | 방의 이름 | 30자 내 |
| byte | full\_Player | 방에 설정된 최대 인원 | 0~4 |
| List | players\_data | 현재 방에 접속한 Player  방에 접속한 수, name, |  |
| bool | is\_started | 게임 중 여부 | true, false |
| byte | ready\_player | 준비중인 플레이어 수  2진수 비트연산으로 0000 왼쪽부터 1p ~ 4p | <1111 |

List players\_data PlayerData의 집합

List enemy\_data EnemyData의 집합

List bullets\_data BulletData의 집합

List lobby\_data RoomData의 집합

Client 디자인

### Client 함수

GameSys Class

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 | 설명 | 인자 | 반환값 |
| game | | | |
| void move()  사용자로부터 key입력을 받아 Player를 움직인다.  void shoot()  사용자로부터 key입력을 받아 bullet을 발사한다.  void ui\_state\_draw (List players\_data)  서버로부터 받은 게임에 참가한 Player들의 UI창을 출력한다.  Name, Life, Boom, Score  void players\_draw (List players\_data)  서버로부터 받은 Player들을 출력한다.  position, direction, is\_damaged  void enemys\_draw (List enemys\_data)  서버로부터 받은 Enemy들을 출력한다.  position, direction  void bullets\_draw (List bullets\_data)  서버로 받은 Bullet들을 위치를 계산하여 출력한다.  start\_pos, shoot\_time, direction, speed, shooter | | | |

TcpController Class

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 | 설명 | 인자 | 반환값 |
| main | | | |
| void server\_tcp\_connect (string server\_ip, int port);  WS 초기화, 소켓 생성, 접속시도 | | | |
| lobby | | | |
| bytes recv\_lobby\_data(int buf\_size)  서버로부터 로비를 구성하기위한 RoomData들의 정보를 받는다.  void send\_create\_room(byte host\_number, string room\_name, byte full\_player, List players\_data)  방을 만들면 RoomData를 만들어서 서버에 보낸다.  byte send\_join\_room(byte room\_number)  인자로 받은 방을 서버에 접속 요청한다.  성공결과 반환, 성공 시1, full방이어서 실패 2, 이미 시작해서 실패 3 | | | |
| wait\_room | | | |
| bytes recv\_room\_data(int buf\_size)  서버로부터 현재 입장한 방의 정보를 받아온다.  void send\_ready\_state()  ready여부 변경을 서버에 보낸다.  void send\_emotion\_chat(byte emotion)  로비에서 감정표현 채팅을 보낸다.  bytes recv\_emotion\_chat()  서버로부터 로비에서의 감정표현 채팅을 수신한다. | | | |
| game | | | |
| void send\_bullet\_data(BulletData bullet)  총알을 발사하면 BulletData정보를 서버에 보낸다.  bytes recv\_is\_game\_over()  게임이 종료되었는가 여부를 서버로부터 수신한다.  void send\_player\_data(PlayerData player\_data)  프레임마다 PlayerData를 서버에 보낸다.  byte recv\_players\_data()  프레임마다 PlayersData를 서버로부터 받아온다.  bytes recv\_enemys\_data()  프레임마다 EnemysData를 서버로부터 받아온다.  bytes recv\_bullets\_data()  프레임마다 BulletsData를 서버로부터 받아온다. | | | |
| leader\_board | | | |
| bytes recv\_leader\_board()  서버로부터 LeaderBoardData를 가져온다. | | | |

### Server 함수

ServerSys Class

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 | 설명 | 인자 | 반환값 |
| game | | | |
| Point get\_bullet\_pos(BulletData)  BulletData를 통해 총알의 좌표를 반한다.  bool collision\_check(List players\_data, List enemys\_data, List bullets\_data)  Bullet과 Enemy혹은 Player와의 충돌을 체크한다.  void record\_learder\_board(List players\_data, float time)  랭킹을 기록한다. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 | 설명 | 인자 | 반환값 |
| thread | | | |
| void client\_thread()  클라이언트가 접속을 요청할 때마다 연결해준다.  void room\_thread()  호스트 클라이언트가 방을 만들 때 마다 동기화 기능이 있는 쓰레드를 만들어 준다. | | | |

TcpController Class

|  |  |  |  |
| --- | --- | --- | --- |
| 함수 | 설명 | 인자 | 반환값 |
| main | | | |
| void tcp\_server\_init()  WS 초기화, 소켓 생성, 소켓주소 할당  void tcp\_client\_accept()  클라이언트와 연결 대기 & 연결 이후 스레드 생성 | | | |
| lobby | | | |
| void send\_lobby\_data(List RoomData)  로비에 있는 방들의 RoomData들을 클라이언트에 보낸다.  bool recv\_create\_room()  클라이언트로부터 생성할 방에 대한 정보를 받는다.  void send\_room\_data(RoomData)  접속을 요청 받았을 때 요청한 플레이어에게 접속하려는 방의 정보를 알려준다 | | | |
| wait\_room | | | |
| bytes recv\_ready()  클라이언트로부터 준비 변경여부를 받는다.  bytes recv\_emotion\_chat()  클라이언트로부터 플레이어들의 감정표현 채팅을 수신한다.  void send\_emotion\_chat(byte emotion chat)  클라이언트에 서버에 수신한 채팅메시지를 보낸다 | | | |
| game | | | |
| bytes recv\_bullet\_data()  클라이언트로부터 BulletData를 받아온다.  void send\_is\_game\_over()  GameOver일 때 클라이언트에 게임오버신호를 보낸다.  bytes recv\_player\_data()  프레임마다 클라이언트로부터 PlayerData를 받아온다.  void send\_players\_data(List PlayerDataList)  프레임마다 모든 클라이언트로부터 받은 PlayerDataList를 보낸다.  void send\_enemys\_data(List EnemyDataList)  프레임마다 모든 클라이언트로부터 받은 EnemyDataList를 보낸다.  void send\_bullets\_data(List BulletDataList)  프레임마다 모든 클라이언트로부터 받은 BulletDataList를 보낸다. | | | |
| leader\_board | | | |
| void send\_leader\_board(List LeaderBoard)  게임이 끝난 후 클라이언트에게 leader board를 보낸다. | | | |

## 팀원 별 역할분담

|  |  |
| --- | --- |
| 김영서 | 클라이언트 개발,  멀티 게임 로직구현  클라이언트tcpController gamepart 구현  서버 tcpController gamepart 구현 |
| 민웅기 | 리소스 제작,  lobby구현,  waitRoom구현  서버 lobby, waitRoom구현 |
| 정낙현 | 클라이언트 기획서에 맞게 리팩토링,  Thread 구현,  leaderBoard구현,  이벤트구현 |

## 개발 환경

Python 3.6.3

pycham

pico2d

GitHub

StarUML

## 개발 일정

공백의 칸은 가장 최근작업의 연장

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 11/13월 | | 11/14화 | | 11/15수 | | 11/16목 | | 11/17금 | | 11/18토 | | 11/19일 |
| 김영서 | client개발 | |  | | client개발 | | client개발 | | client개발 | |  | | 클라 합치기 및 완성 |
| 민웅기 | 리소스제작 | | 리소스제작 | |  | |  | | 리소스제작 | | 리소스제작 | |
| 정낙현 |  | |  | | client 리팩토링 | | client 리팩토링 | | client 리팩토링 | | client 리팩토링 | |
|  | 11/20월 | | 11/21화 | | 11/22수 | | 11/23목 | | 11/24금 | | 11/25토 | | 11/26일 |
| 김영서 |  | |  | | send\_enemys\_  data | | recv\_enemy\_  data | | send\_bullets\_  data | |  | |  |
| 민웅기 | send\_create\_  room | | send\_join\_  room | |  | |  | | send\_ready\_  state | | send\_room\_  data | |  |
| 정낙현 | tcp\_server\_  init | |  | |  | | server\_tcp\_  connect | | tcp\_client\_  accept | |  | | send\_is\_game\_  over |
|  | 11/27월 | | 11/28화 | | 11/29수 | | 11/30목 | | 12/1금 | | 12/2토 | | 12/3일 |
| 김영서 | recv\_player\_  data | |  | | recv\_bullets\_  data | | send\_player\_  data | | recv\_player\_  data | |  | |  |
| 민웅기 | send\_lobby\_  data | | recv\_create\_  room | |  | | recv\_lobby\_  data | | recv\_room\_  data | | recv\_ready | |  |
| 정낙현 | client\_thread | |  | |  | | recv\_is\_game  over | | send\_bullet\_  data | |  | | recv\_bullet\_  data |
|  | 12/4월 | 12/5화 | | 12/6수 | | 12/7목 | | 12/8금 | | 12/9토 | | 12/10일 | |
| 김영서 | send\_players  data |  | | game 로직  수정 | |  | |  | | 종합코드  QA | | 중간 완성  각각 결산물 merge | |
| 민웅기 | send\_  emotion\_  chat\_client | send\_  emotion\_  chat\_server | | recv\_emoition\_  chat\_server | |  | | recv\_emotion\_  chat\_client | | 종합코드  QA | |
| 정낙현 | send\_leader\_  board | recv\_leader\_  boad | | room\_thread | | 시스템 로직 수정 | |  | | 종합코드  QA | |
|  | 12/11월 | 12/12화 | | 12/13수 | | 12/14목 | | 12/15금 | | 12/16토 | | 12/17일 | |
| 김영서 | 중간결산물 테스트 플레이  버그수집 | | | 디버그 및  텀 프로젝트 보고 문서작업 | | | | | | | | 수고  하셧습니다. | |
| 민웅기 |
| 정낙현 |
|  | 12/18월 |  | |  | |  | |  | |  | |  | |
| 김영서 | 최종 제출일 |  | |  | |  | |  | |  | |  | |
| 민웅기 |  | |  | |  | |  | |  | |  | |
| 정낙현 |  | |  | |  | |  | |  | |  | |