네트워크 게임 프로그래밍

Term Project 기획서

2014180015 게임공학과 김정현

2014182043 게임공학과 차종원

2016184017 엔터테인먼트 컴퓨팅 박하연

목차

■ 어플리케이션 기획

[게임 전체적 모습]

• 게임 소개

OpenGL 위에 2D 스프라이트 리소스를 사용하여 만든 게임이다.

플레이어 2명이 일정 시간마다 무작위 위치에 생성되는 아이템을 먹어 상대방을 공격하는 게임이다.
게임 시간이 끝나거나 게임 시간 안에 생명이 0이 된 플레이어가 발생하면 게임이 종료된다.

• 게임 오브젝트

- 플레이어: 
- 아이템: 
- 총알: 

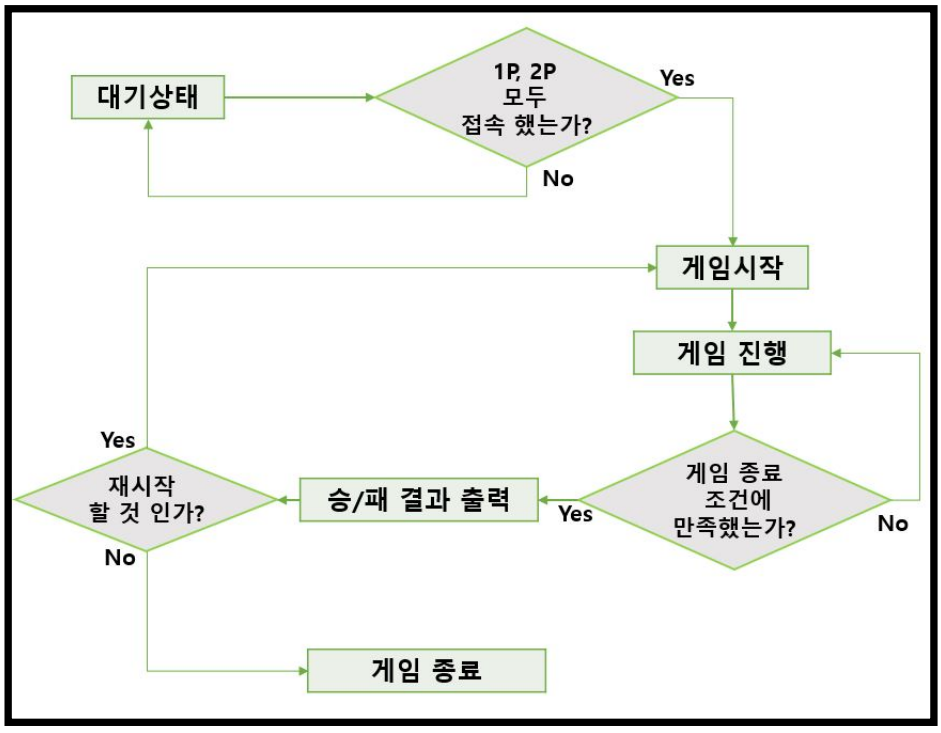
• 조작 방법

- 캐릭터 이동 (W/A/S/D) : 캐릭터 좌, 우, 상, 하, 대각선 이동

• 게임 설명

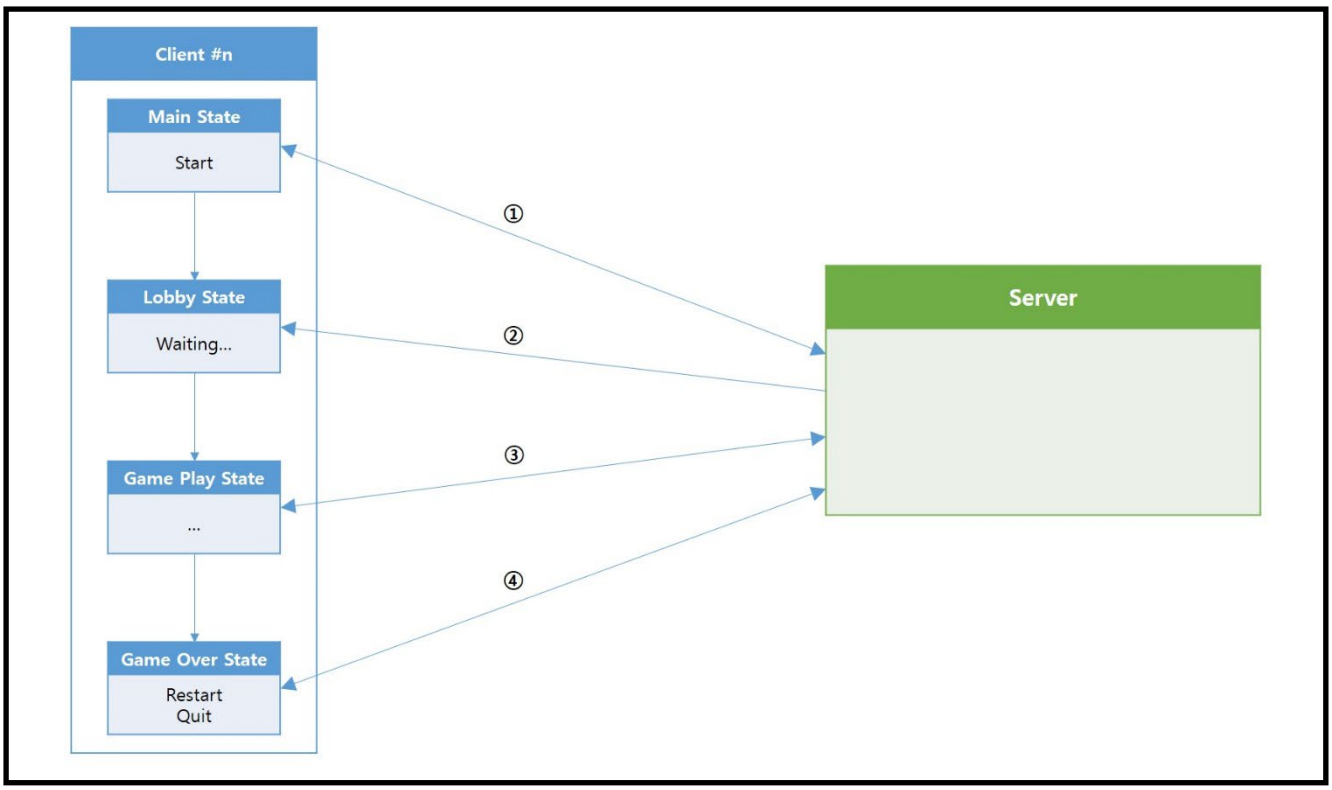
- 이 게임은 2명이 함께 플레이하는 게임이다.
- 캐릭터를 상, 하, 좌, 우로 이동시키며 아이템을 먹거나 상대방의 아이템을 피해야 한다.
- 아이템은 특정 시간마다 맵 안 임의의 위치에 생성된다.
- 아이템을 먹으면 현재 상대방의 위치를 향해 아이템이 발사된다.
- 게임의 제한 시간은 120초, 생명은 5로 시작한다.
- 플레이어와 총알이 충돌하면 플레이어의 생명이 1 감소한다.
- 제한 시간 내에 상대 플레이어의 생명을 0으로 만들면 승리한다.
- 제한 시간 동안 생명이 0이 된 플레이어가 없으면 남은 생명으로 승패를 판단한다.

• Flow Chart



■ High Level Design

• Flow Chart



[State 별 개괄적 통신 플로우 차트]

① Main State – Server 통신

클라이언트는 플레이어가 게임을 실행하여 WASD중 하나의 키를 입력하면 그 키 정보를 서버에게 전송한다. 그 후 서버로부터 변경된 자신의 State 정보를 수신한다.

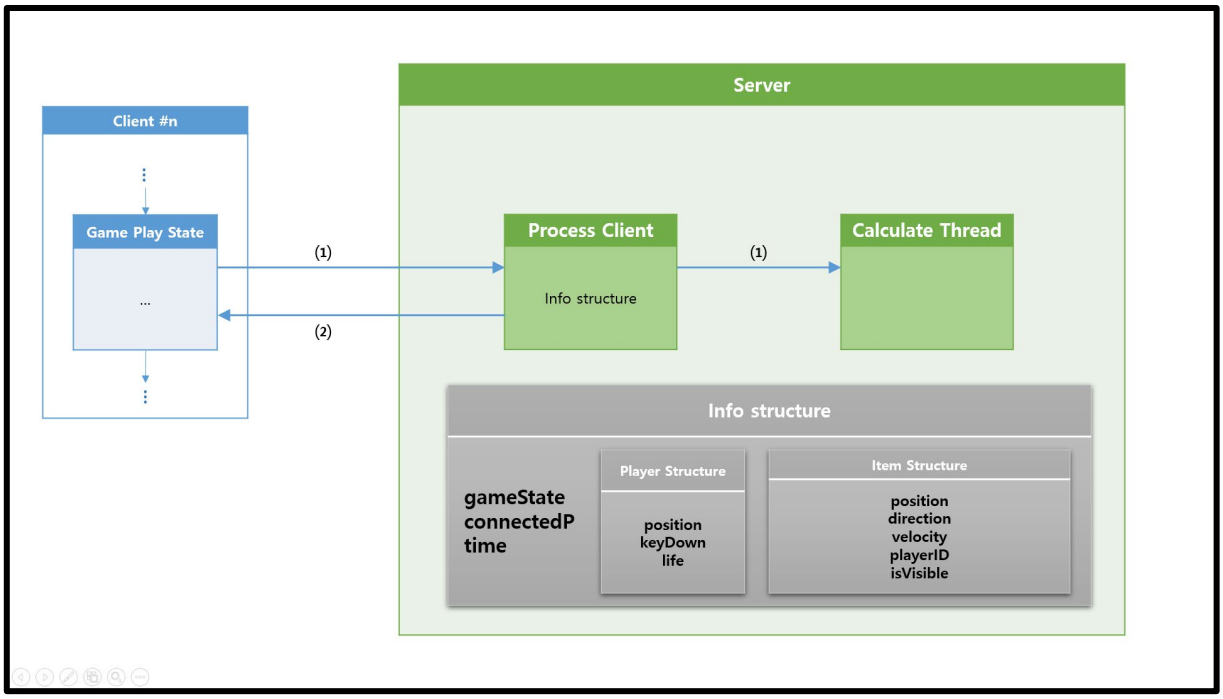
서버는 받은 키 정보를 토대로 해당 클라이언트의 State 정보를 Lobby State로 변경하여 그 정보를 클라이언트에게 재전송한 후 서버 접속자의 수를 늘린다.

② Lobby State – Server 통신

클라이언트는 서버에게 데이터 전송을 하지 않고, 서버로부터 변경된 State 정보를 받을 때까지 계속 대기한다.

서버는 서버 접속자의 수가 총 2명이라면 각 클라이언트의 State 정보를 Game Play State로 변경하고 그 정보를 클라이언트에게 전송한다.

③ Game Play State – Server 통신



[Game Play State – Server 통신 플로우 차트]

(1) 매 프레임 마다 클라이언트로부터 Process Client 스레드로 정보를 받아와 Info structure의 정보를 갱신한다.

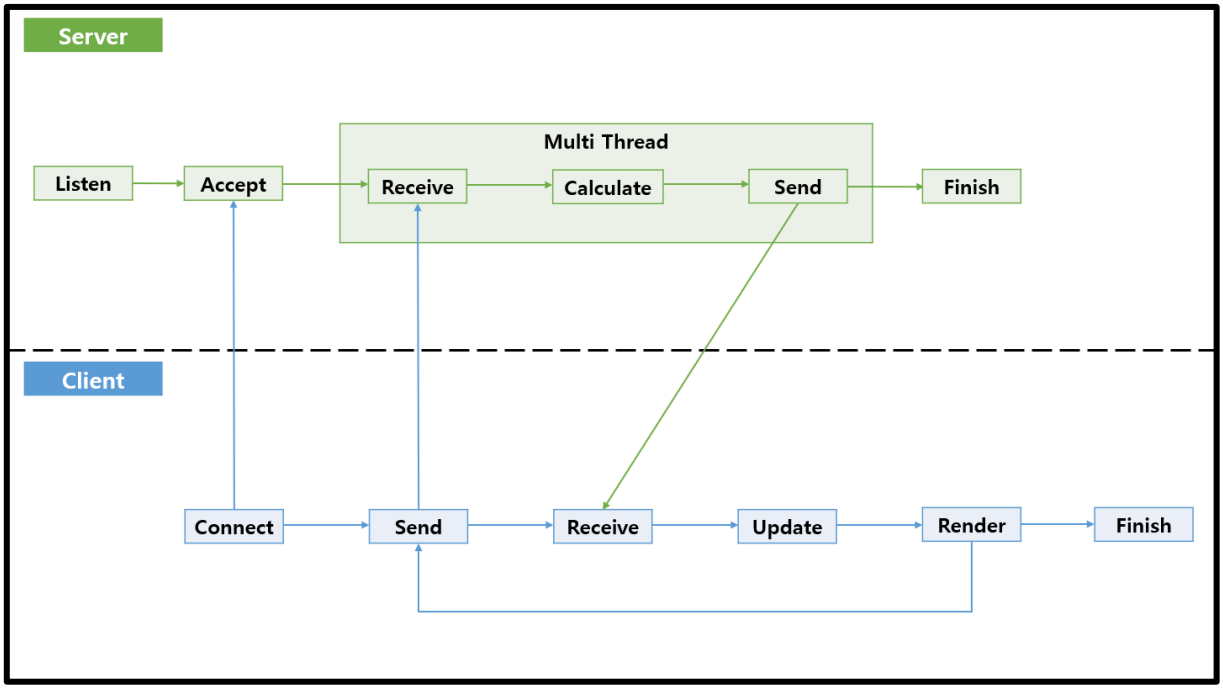
(2) 갱신한 정보를 가지고 게임에 필요한 연산(오브젝트 위치 이동, 충돌 체크, 시간 계산 등)을 처리한 후 그 정보를 다시 Process Client에서 해당 클라이언트에게 재전송한다.

④ Game Over State – Server 통신

클라이언트는 플레이어가 재시작을 위해 "A" 키를 입력하면 그 키 정보를 서버에게 전송한 후 서버로부터 변경된 자신의 State 정보를 수신한다.
게임 종료를 위해 "D" 키를 누르면 그 키 정보를 서버에게 전송한 후 통신을 중단한다.

서버는 받은 키 정보를 토대로 재시작을 원하는 플레이어의 수가 총 2명이라면, 각 클라이언트의 State 정보를 Game Play State로 변경하고 그 정보를 클라이언트에게 재전송한다.
재시작을 원하는 플레이어의 수가 총 2명이 아니라면, 게임을 종료한 클라이언트와는 통신을 중단하고 재시작을 원하는 클라이언트의 State 정보를 Lobby State로 변경하여 재전송한다.

- **Server – Client 통신**



[Client – Server 통신 플로우 차트]

- 순서

1. **서버**는 윈속 초기화, 소켓 생성, bind 를 완료한 후 **listen** 을 한다.
2. **클라이언트**는 서버에 **connect** 요청을 한다.
3. **서버**는 connect 요청이 들어오면 **accept** 하여 ProcessClient 스레드를 생성한다.
4. **클라이언트**는 서버에게 캐릭터, 아이템의 위치정보, 생명, 키 입력 정보를 **send** 한다.
5. **서버**는 클라이언트에게서 데이터를 **receive** 받고 갱신될 위치, 충돌체크 등을 계산한다
6. 계산이 완료되면 갱신된 데이터를 클라이언트에게 **send** 한다
7. **클라이언트**는 서버로부터 갱신된 데이터를 **receive** 받는다.
8. 갱신된 데이터를 **update** 시킨다
9. 데이터를 이용하여 **render** 한다
10. 게임 중에 종료조건을 만족하면 **서버**가 state 를 GAME_OVER 로 변경한 뒤 **send** 한다
11. **클라이언트**는 state 가 FINISH 가 되면 결과를 출력하고 종료 또는 다시 시작한다.

■ Low Level Design

• Server – Client Protocol

◆ 윈속 버전: 2.2

◆ 프로토콜: TCP/IP

◆ 구조체

```
typedef struct Vec {
        float x; float y;
};


// Client -> Server
typedef struct CtoSPacket {
        Vec pos;
        bool keyDown[4];
        short life;
};

// Server -> Client
typedef struct StoCPacket {
        Vec p1Pos;
        Vec p2Pos;

        Vec itemPos[100];
        short characterID[100];
        bool isVisible[100];

        DWORD time;

        short life;
        short gameState;

};
```

- **Server**

◆ 구조체

```
typedef struct Info {
        short connectedP;       // 연결된 플레이어의 수
        DWORD gameTime;         // 게임 시간

        Player p [2];           // 플레이어 구조체
        ItemObj i[100];         // 아이템 구조체
};

typedef struct Player {
        short gameState;        // 게임 상태를 나타내는 변수
        Vec pos;                // 플레이어 위치
        bool keyDown[4];        // 클라이언트 키 입력 배열
        short life;             // 플레이어 생명
};

typedef struct ItemObj {
        Vec pos;                // 아이템 위치
        Vec direction;          // 아이템 발사 방향
        float velocity;         // 아이템 속도
        short playerID;         // 아이템을 먹은 플레이어의 아이디
        bool isVisible;         // 화면 표시 여부
};
```

◆ 함수

◆ void RecvFromClient(LPVOID arg, short PlayerID)

설명: 클라이언트로부터 패킷을 받는 함수

recvn() 함수를 사용하여 고정 크기의 데이터를 받아 서버의 Info 구조체를 갱신해준다.

인자: 스레드 함수의 인자로 받은 arg를 해당 함수의 첫 번째 인자로,

플레이어의 아이디를 두 번째 인자로 받는다.

◆ void SendToClient(LPVOID arg, short PlayerID)

설명: 클라이언트로 패킷을 보내는 함수.

calcThread() 함수에서 연산이 완료된 (갱신된)데이터를 클라이언트로 보내준다.

인자: 스레드 함수의 인자로 받은 arg를 해당 함수의 첫 번째 인자로,

플레이어의 아이디를 두 번째 인자로 받는다.

◆ DWORD WINAPI ProcessClient (LPVOID arg)

설명: 클라이언트로부터 데이터를 send, receive 하여 관리하는 스레드.
HANDLE hUpdateEvt 이벤트 객체를 WaitForSingleObject() 함수로 신호 체크하여 UpdateThread() 함수와의 동기화를 한다.
플레이어를 구분할 수 있게 플레이어 ID를 부여한다.

◆ DWORD WINAPI CalculateThread(LPVOID arg)

설명: 좌표 갱신, 충돌 체크(아이템-플레이어, 총알-플레이어), 게임 종료조건(시간, 생명)을 검사하고 계산하는 함수

HANDLE hRecvEvt 이벤트 객체를 WaitForSingleObject() 함수로 신호 체크하여 ProcessClient() 함수와의 동기화를 한다.

- **Client**

  ◆ 구조체

```
typedef struct Info {
        short gameState;        // 게임 상태를 나타내는 변수
        DWORD gameTime;         // 게임 시간

        Player p [2];           // 플레이어 구조체
        ItemObj i[100];         // 아이템 구조체
};

typedef struct Player {
        Vec pos;                // 플레이어 위치
        short life;             // 플레이어 생명
};

typedef struct ItemObj {
        Vec pos;                // 아이템 위치
        bool isVisible;         // 화면 표시 여부
};
```

  ◆ 함수

◆ void SendToServer(SOCKET sock)

설명: 서버에 데이터를 전송하는 함수.
서버에 CtoSPacket 구조체를 보낸다.

인자: 서버의 정보를 담은 소켓 구조체를 인자로 받는다.

◆ void RecvFromServer(SOCKET sock)

설명: 서버에서 데이터를 받아오는 함수
서버로부터 Info 구조체를 받아온다.

인자: 서버의 정보를 담은 소켓 구조체를 인자로 받는다.

■ 역할 분담

◆ 김정현: 서버 프레임워크 구현, 서버 내의 클라이언트 처리 스레드 함수 개발

서버 측의 송신 함수 개발, 서버 내의 연산 처리 스레드 함수 구현(좌표 계산)

◆ 차종원: 클라이언트 프레임워크 구현, 클라이언트 측의 송신 함수 개발

서버 내의 연산 처리 스레드 함수 구현(충돌 체크 및 탄성 구현)

◆ 박하연: 클라이언트 리소스 제작 및 수정, 클라이언트 측의 수신 함수 개발,

서버 내의 클라이언트 처리 스레드 함수 개발, 서버 측의 수신 함수 개발

서버 내의 연산 처리 스레드 함수 구현(종료 조건)

■ 개발 환경

◆ IDE : Visual Studio 2017

◆ 운영체제 : Windows10 64bit

◆ Lib : ws2_32, OpenGL

◆ VCS : GitHub (https://github.com/natsnatsmon/AENG-GYUNG)

◆ 언어 : C++

■ 개발 일정

◆ 1주차

	화	수	목	금	토	일
	10/23	10/24	10/25	10/26	10/27	10/28
박하연	계획서 초안 논의			계획서 회의	High Level 서버-클라 순서도 작성	Low Level 작성
차종원	계획서 초안 논의			계획서 회의	어플리케이션 기획 작성	Low Level 작성
김정현	계획서 초안 논의			계획서 회의	High Level 플로우 차트 작성	Low Level 작성

◆ 2주차

	월	화	수	목	금	토	일
	10/29	10/30	10/31	11/1	11/2	11/3	11/4
김정현	개발 일정 작성, 계획서 마무리	계획서 심사		서버 프레임워크 구현			팀 회의 및 검토
차종원				클라이언트 프레임워크 구현 (1차)		클라이언트 프레임워크 구현(2차)	
박하연					리소스 수정 및 제작		

◆ 3주차

<table>
<tr><th></th><th>월</th><th>화</th><th>수</th><th>목</th><th>금</th><th>토</th><th>일</th></tr>
<tr><th></th><th>11/5</th><th>11/6</th><th>11/7</th><th>11/8</th><th>11/9</th><th>11/10</th><th>11/11</th></tr>
<tr><td>김정현</td><td colspan="2">UpdatePosition() 구현(1차)</td><td></td><td>개인별 구현
검토 및 보완</td><td>UpdatePosition()
RecvFromClient()
테스트 및 보완</td><td></td><td rowspan="3">3주차
구현 내용
통합
테스트
및 보완</td></tr>
<tr><td>차종원</td><td>SendToServer()
구현(1차)</td><td rowspan="2">SendToServer()
RecvFromClient()
통신 테스트
및 보완
(1차)</td><td></td><td>SendToServer()
구현(2차)</td><td></td><td rowspan="2">SendToServer()
RecvFromClient()
통신 테스트
및 보완
(2차)</td></tr>
<tr><td>박하연</td><td>RecvFromClient()
구현(1차)</td><td></td><td></td><td>UpdatePosition()
RecvFromClient()
테스트 및 보완</td></tr>
</table>

◆ 4주차

<table>
<tr><th></th><th>월</th><th>화</th><th>수</th><th>목</th><th>금</th><th>토</th><th>일</th></tr>
<tr><th></th><th>11/12</th><th>11/13</th><th>11/14</th><th>11/15</th><th>11/16</th><th>11/17</th><th>11/18</th></tr>
<tr><td>김정현</td><td colspan="2">SendToClient()
구현 (1차)</td><td></td><td rowspan="2">UpdatePosition()
CollisionCheck()
테스트 및 보완</td><td>SendToClient()
RecvFromServer()
통신 테스트
및 보완(1차)</td><td></td><td rowspan="3">4주차
구현 내용
통합
테스트
및 보완</td></tr>
<tr><td>차종원</td><td colspan="2">CollisionCheck()
구현(1차)</td><td></td><td></td><td>개인별 구현
검토 및 보완</td></tr>
<tr><td>박하연</td><td>RecvFromServer()
구현 (1차)</td><td>개인별 구현
검토
및 보완</td><td></td><td></td><td>SendToClient()
RecvFromServer()
통신 테스트
및 보완(1차)</td><td>RecvFromServer()
구현 (2차)</td></tr>
</table>

◆ 5주차

	월	화	수	목	금	토	일
	11/19	11/20	11/21	11/22	11/23	11/24	11/25
김정현	hRecvEvt 생성, Process Client() 내 배치	hUpdateEvt 생성, CalcThread() 내 배치		개인별 구현 검토 및 보완	소켓옵션 설정 및 테스트		5주차 구현 내용 통합 테스트 및 보완
차종원	충돌 시 탄성력 계산			탄성력 적용		개인별 구현 검토 및 보완	
박하연	TimeCheck() 구현	개인별 구현 검토 및 보완			LifeCheck() 구현	GameEndCheck() 구현	

◆ 6주차

	월	화	수	목	금	토	일
	11/26	11/27	11/28	11/29	11/30	12/1	12/2
김정현	UpdatePosition() 내 탄성력 적용 추가	탄성력 적용한 UpdatePosition() 구현 및 테스트		개인별 구현 검토 및 보완	SendToClient() RecvFromServer() 통신 테스트 및 보완(2차)		6주차 구현 내용 통합 테스트 및 보완
차종원	SendToServer() RecvFromClient() 통신 테스트 및 보완 (3차)			최종 수정		개인별 구현 검토 및 보완	
박하연		개인별 구현 검토 및 보완			SendToClient() RecvFromServer() 통신 테스트 및 보완(2차)	최종 수정	

◆ 7주차

	월	화	수	목	금	토	일
	12/3	12/4	12/5	12/6	12/7	12/8	12/9
김정현	통합 연동 테스트 및 코드 최적화			개인별 구현 검토 및 보완	최종 테스트 및 미비점 보완		
차종원							
박하연							

◆ 8주차

	월	화
	12/10	12/11
김정현	최종 심사 준비	최종 심사
차종원		최종 심사
박하연		최종 심사