

# 네트워크 게임 프로그래밍 프로젝트 추진 계획서

## -Raising Sunfish-

분반	02반
지도교수	김재경 교수님
학과	게임공학과
학번 / 이름	2018182024 이동현 2017180036 장형택 2019182049 손희수

# 목차

## 1. 애플리케이션 기획

- 게임 소개
- 스크린샷
- 조작법

## 2. High-level 디자인

## 3. Low-level 디자인

## 4. 팀원 별 역할분담

## 5. 개발환경

## 6. 개발일정

## 애플리케이션 기획

### - 게임 소개

원제작자 및 과목 : 손희수, 윈도우프로그래밍

## 제목 : **Raising Sunfish**(개복치 키우기)

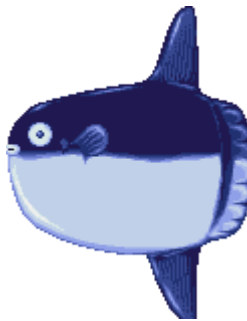
장르 : 경쟁

게임 방식 :




3명의 플레이어가 각자 자신의 개복치를 키우기 위해 먹이를 먹고 상대방의 개복치를 견제해가며 제한 시간 내에 가장 높은 점수를 내기 위해 경쟁하는 게임입니다.

## 등장요소

### - 플레이어

이미지	
설명	<p>- 상하좌우, <b>대각선으로</b> 움직일 수 있다. 먹이를 먹으면 덩치를 키울 수 있다. 덩치가 커지면 속도가 느려진다.</p> <p><del>상대 플레이어의 진로를 몸으로 막을 수 있다. 장애물에 충돌할 시 사망하며 일정 시간 후 부활한다.</del></p>

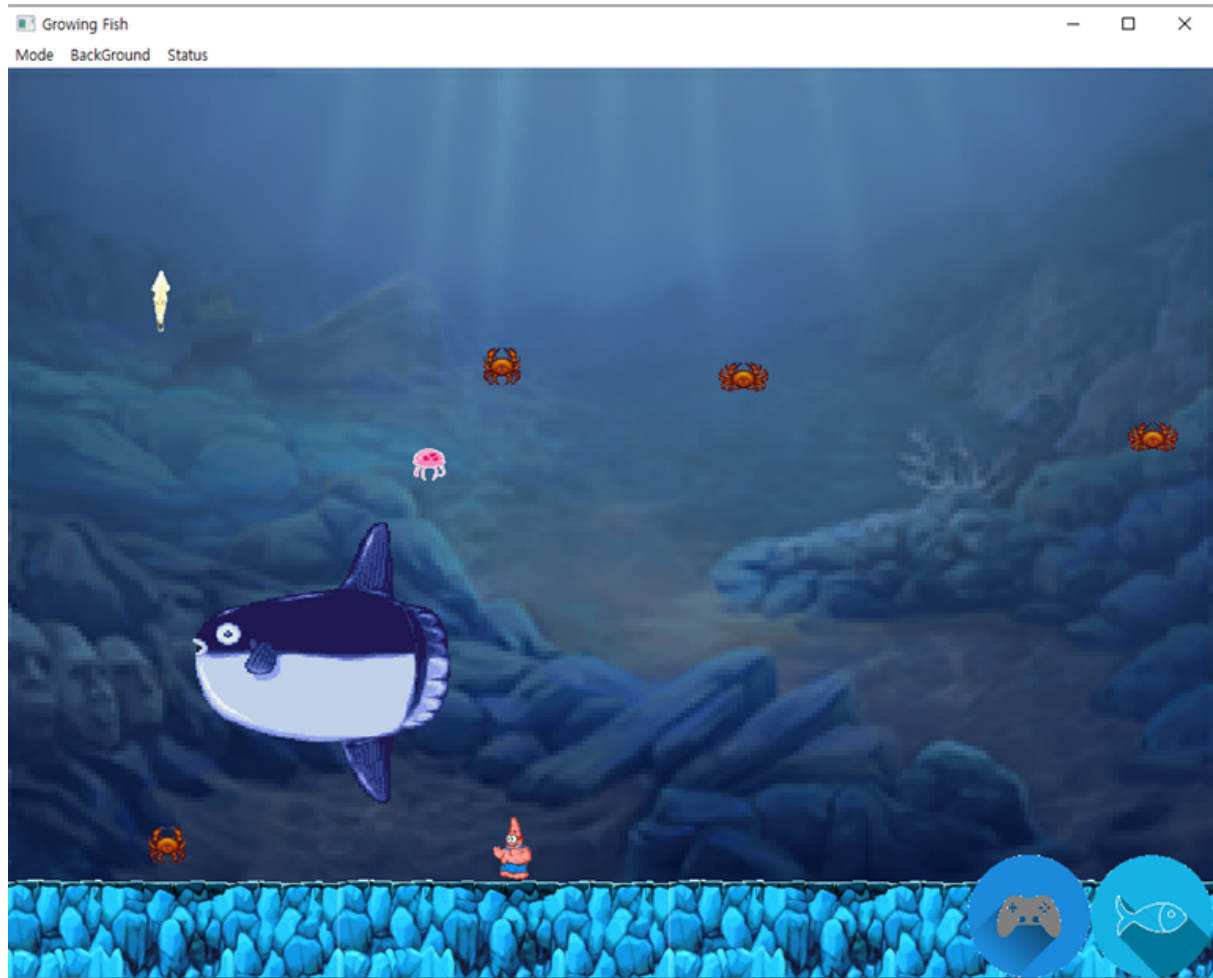
- 장애물

	장애물 1	장애물 2	장애물 3
이미지			
설명	<p>일정 주기로 맵에 생성되며,          개복치에 닿으면 개복치가 끌려가며 사망한다.          플레이어가 마우스 클릭으로 파훼할 수 있다.          1~50 1~30 사이의 랜덤한 체력을 가진다.(클릭당 1감소)          사망 시 10점의 점수를 잃는다.</p>		

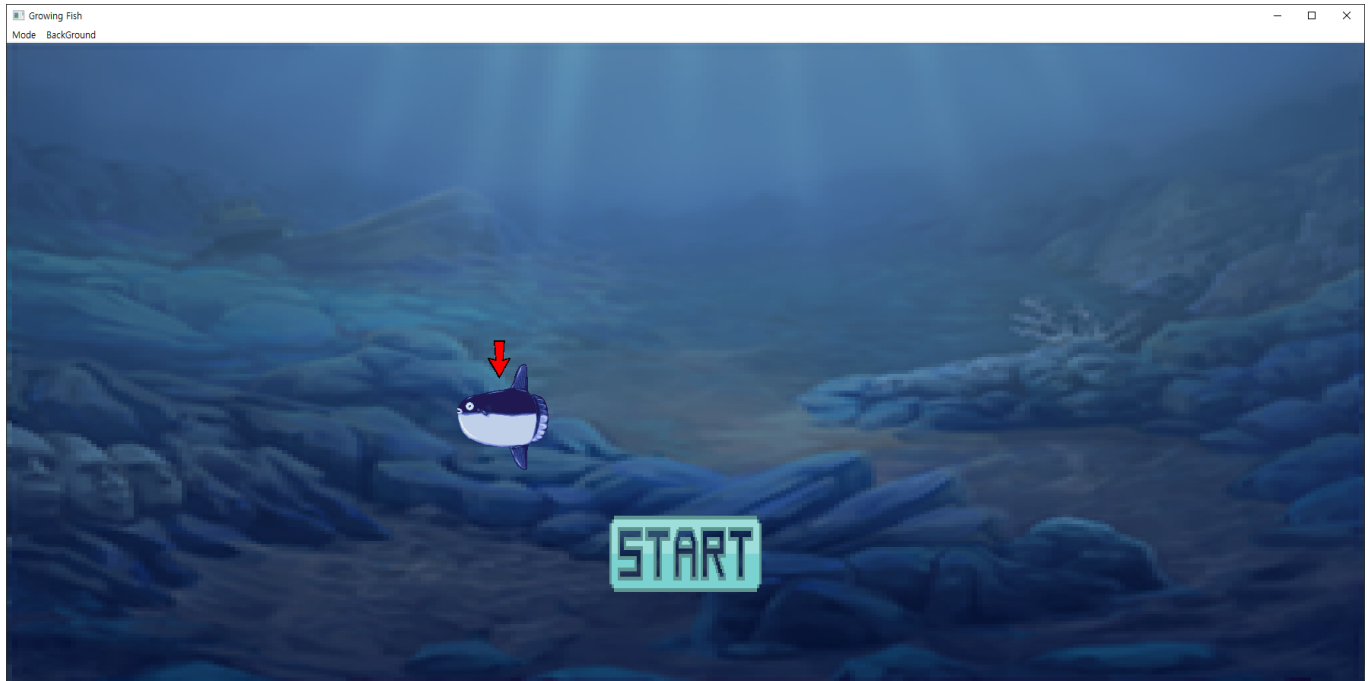
- 먹이

	먹이 1	먹이 2	먹이 3
이미지			
점수	1	2	3
설명	<p>일정 주기로 맵에 생성되며,          개복치가 닿으면 개복치의 덩치가 커지고,          먹이를 먹은 플레이어는 점수를 획득한다.          먹이에 따라 획득하는 점수에 차이가 있다.</p>		

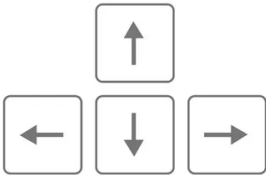
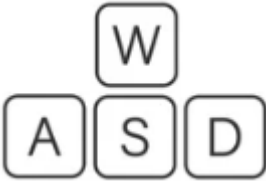

## ~~플레이 스크린샷(변경 전)~~



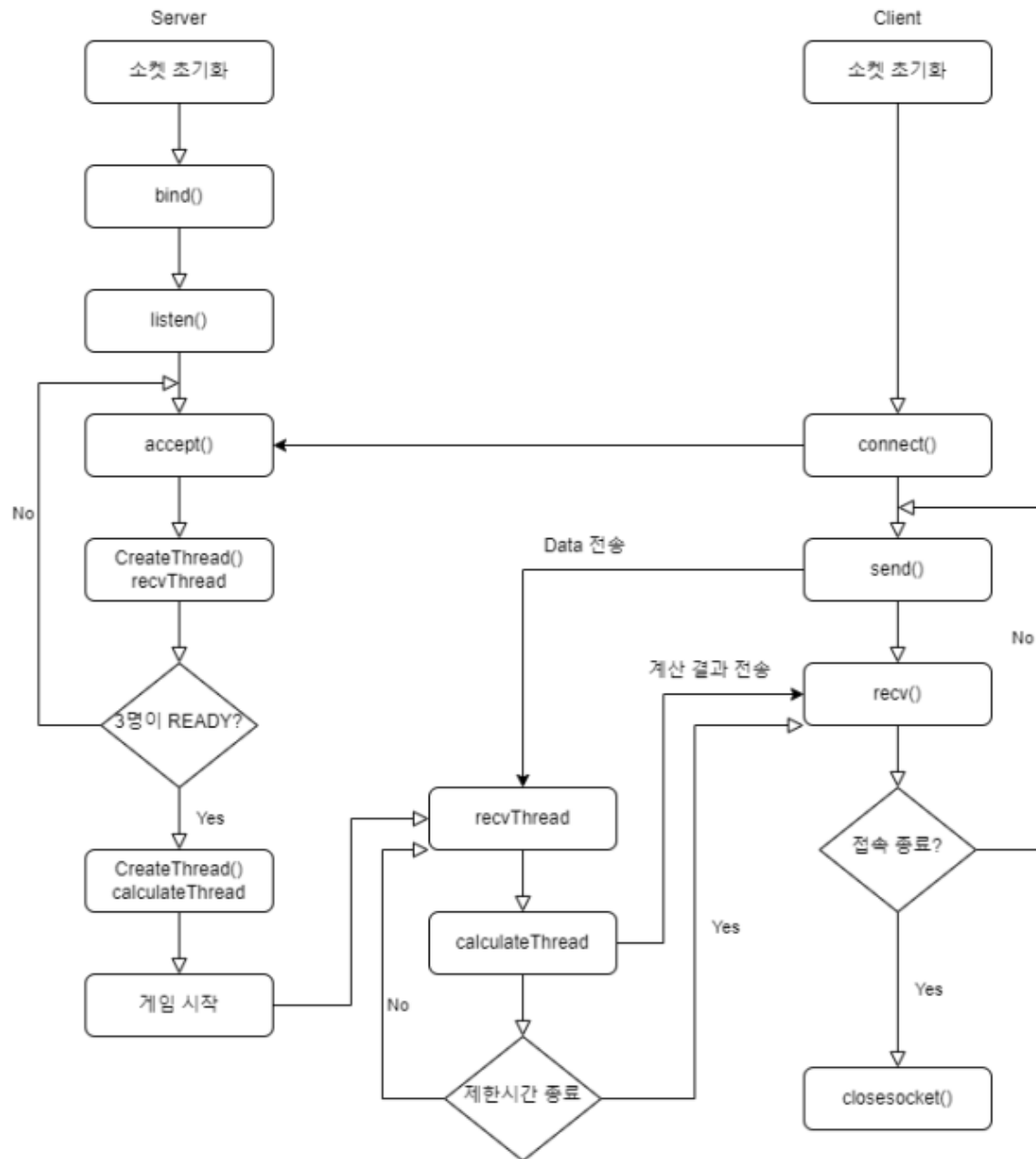
## 플레이 스크린샷(변경 후)



조작법

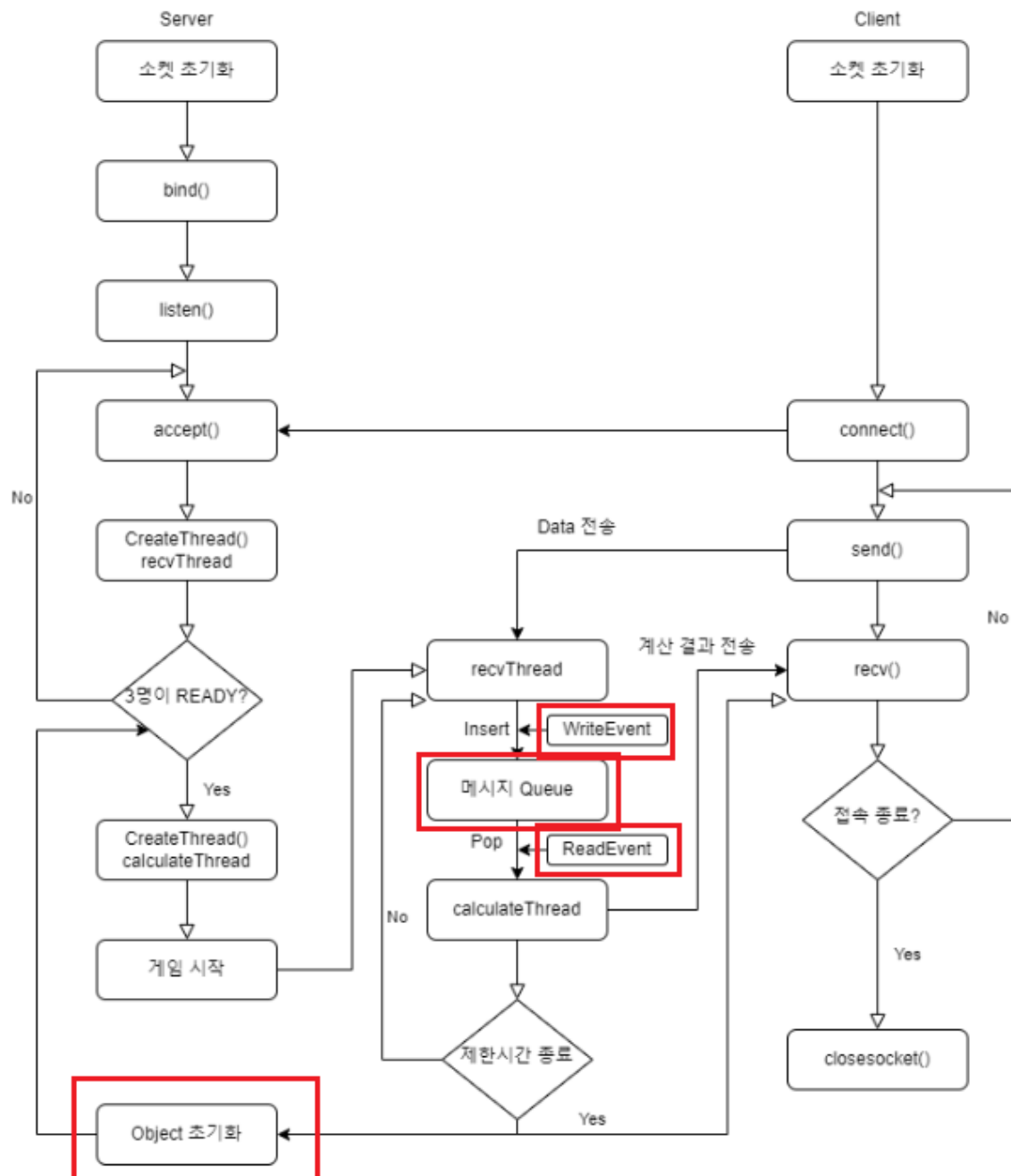
<div>변경 전</div> <div></div> <div>변경 후</div> <div></div>	<div><del>상, 하, 좌, 우</del> 방향키</div> <div>WASD 키</div>	<div>플레이어를</div> <div>상, 하, 좌, 우 <b>및 대각선</b> 방향으로 이동시킨다.</div> <div>개복치의 크기가 클 수록 이동 속도가 느려진다.</div>
<div></div>	<div>마우스 좌 클릭</div>	<div>장애물을 파훼한다.</div>

## High-level 디자인(변경 전)





## High-level 디자인(변경 후)



## Client

1. 서버와 통신할 소켓을 생성한 후 서버와 **connect** 한다.
2. 서버와 연결이 되면 **ID**를 부여받는다.
3. **Lobby Scene**에서 준비상태로 바꾼다.
4. 게임이 시작되면 서버로 부터 시작 메시지를 받는다.
5. 키를 입력하면 해당 키에 해당하는 이벤트 값을 서버로 전송한다.
6. 서버에서 연산한 데이터(플레이어 위치, 먹이 위치, 게임결과 등)를 **recv**한다.
7. 받은 데이터를 이용해 **Render**한다.
8. 게임이 종료되는 메시지를 받을때 까지 반복한다.
9. 사용자가 접속을 종료하면 게임이 종료된다.

## Server

1. **listen**상태에서 클라이언트의 **connect** 요청이 들어오면 **accept** 후 클라이언트에게 **ID**를 부여한후, **recv** 스레드를 생성한다.
2. 3개의 클라이언트가 모두 **ready**상태가 되면 **Calculate** 스레드를 생성한 후 게임의 초기상태를 클라이언트에게 전송한다.
3. **recv** 스레드에서 클라이언트의 입력 메시지를 받고 공유자원 **class** 를 생성한다. **queue** ~~우선순위 queue~~에 삽입한다.
4. ~~**Calculate** 스레드에서 공유자원 **class queue** ~~우선순위 queue~~를 처리하고 갱신된 정보를 클라이언트에게 전송한다.~~  
(이때 **recv** 스레드와 **Calculate** 스레드 간 임계영역 ~~이벤트 기법~~을 이용하여 플레이어 이동, 총알 이동, 이벤트 처리, 충돌 검사를 할때 다른 클라이언트의 데이터가 들어가지 않도록 한다.)  
**recv** 스레드, **caculate** 스레드는 동시에 쓰기동작을 하는 공유자원에만 임계영역을 적용하고 그외에는 각각 처리한다  
(**recv** 스레드는 패킷을 처리하고, **caculate** 스레드는 오브젝트의 실시간 처리를 담당한다)
5. 게임이 종료되면 클라이언트에게 게임결과를 전송하고 이후 클라이언트에게 **Lobby Scene**으로 다시 돌아가도록 명령한다.
6. 다시 3개의 클라이언트가 **ready**상태가 되면 게임을 다시 시작한다.

## Low-level 디자인

```
enum PacketType {  
    SC_CHANGE_DIRECTION,  
    CS_CHANGE_DIRECTION,  
    CS_LBUTTONDOWNCLICK,  
    SC_PLAYER_DEAD,  
    CS_PLAYER_READY,  
    SC_GAME_START,  
    SC_GAME_OVER,  
    SC_ACCEPT_PACKET,  
    SC_ADD_PLAYER,  
    SC_COLLISION,  
    SC_CREATE_FOOD,  
    SC_ERASE_FOOD,  
}
```

```

        SC_CREATE_OBSTACLE,
        SC_ERASE_OBSTACLE,
        CS_LOGIN,
        SC_LOGIN_OK,
        SC_LEAVE_PLAYER,
        CS_DISCONNECT,
        SC_UPDATE_OBSTACLE,
        SC_UPDATE_PLAYER_WH,
        CS_INTERPOLATION,
        SC_INTERPOLATION,
        SC_CAUGHT
};

```

**process\_packet(client id, packet)**

```

struct position
{
    short x;
    short y;
}
enum ObjectType {
    NET,
    HOOK,
    SHARK,
    CRAB,
    SQUID,
    JELLYFISH
};

```

```

SC_MOVE_PACKET { id를 가진 플레이어의 좌표를 클라이언트에 보냄
    unsigned char size;
    char type;
    position* pos;
}

```

```

struct SC_CHANGE_DIRECTION_PACKET {
    char type;
    unsigned char dir;
    int id;
    int speed;
};

```

```

CS_MOVE_PACKET { 이동할 방향을 서버에 보냄
    unsigned char size;
    char type;
}

```

```
char dir;  
}
```

```

struct CS_CHANGE_DIRECTION_PACKET {
    char type;
    unsigned char dir;
};

```

```

CS_CLICK_PACKET{
    unsigned char size;
    char type;
    position* pos;
    POINT point;
}

```

- 클릭한 좌표를 서버에 보냄

```

SC_DEAD_PACKET{
    unsigned char size;
    char type;
    short id;
    short x, y;
    int score;
}

```

- 해당 id를 가진 플레이어가 죽었다고 보냄

```

SC_READY_PACKET{
    unsigned char size;
    char type;
    short id;
}

```

- 로비에서 준비되었음을 서버로 보냄

```

SC_COLLISION_PACKET{
    unsigned char size;
    char type;
    int* scores;
}


```

~~- 충돌에 따른 변동 점수를 클라이언트에 보냄~~

```

SC_GAME_OVER_PACKET{
    unsigned char size;
    char type;
    int scores[3];
    int id;
}

```

- 게임이 종료되었음을 클라이언트에 보냄

```

SC_LEAVE_PLAYER_PACKET {
    char type;
    int id;
}

```

```

CS_DISCONNECT_PACKET {
    char type;
}

```

```

CS_LOGIN_PACKET{
    char type;
}

```

```

}
SC_LOGIN_OK_PACKET{
    char type;
    int id;
}
SC_ADD_PLAYER_PACKET{
    char type;
    int id;
}

object
{
    short type;    - 오브젝트 구분용 (먹이1, 먹이2, 장애물1, 장애물2..)
    position* pos;
}

object_info_calculate {                // 객체 충돌 및 계산용 서버 구조체
    object_info object_info;
    bool is_active = false;
    short width, height;
    int i_hook = 0;
    int y_hook;
    int life = -1;
    int dir = -1;
    int o_speed;
    std::mutex life_lock;
};

SC_OBJECT_PACKET{                오브젝트들의 타입, 위치를 클라이언트에 보냄
    unsigned char size;
    char type;
    object* objects;
}

SC_CREATE_OBJCET_PACKET {
    char type;
    int index;
    object_info object;
    unsigned char dir;
    short col_x, col_y;
};

SC_ERASE_OBJECT_PACKET {
    char type;
    int index;
    char object_type = -1;

```

```
};
```

```
SC_UPDATE_OBJECT_PACKET {  
    char type;  
    object_info oi;  
};
```

```
SC_LEAVE_PLAYER_PACKET {  
    char type;  
    int id;  
};
```

```
CS_DISCONNECT_PACKET {  
    char type;  
};
```

```
CS_INTERPOLATION_PACKET {  
    char type;  
    short x, y;  
};
```

```
SC_INTERPOLATION_PACKET {  
    char type;  
    int id;  
    short x, y;  
};
```

```
SC_CAUGHT_PACKET {  
    char type;  
    int id;  
    short x, y;  
};
```

## 공용

<code>void overload_packet_process (char* buf, int packet_size, int&amp; remain_packet)</code>	여러 패킷을 받았을 경우 반복해서 처리하도록 하는 함수
--	--------------------------------

## 클라이언트

<code>void send_Ready(char id)</code>	플레이어가 준비신호를 서버에게 전송
<code>void send_Event(SOCK sock, char buf)</code>	플레이어의 정보를 서버로 전송
<code>void Recv_Data(Sock sock)</code>	서버의 정보를 플레이어가 수신
<code>void Process_Data(char *buf)</code>	받은 패킷 데이터를 관리
<code>void Process_Packet(char* packet_buf)</code>	받은 패킷 데이터를 처리
<code>DWORD WINAPI NetworkThread(LPVOID arg)</code>	

## 서버

<code>Thread recv_thread(LPVOID id)</code>	서버가 클라이언트에서 정보를 수신하는 스레드
<code>Thread calculate_thread(LPVOID id)</code>	게임의 진행에 필요한 연산을 수행, 클라이언트에게 갱신된 데이터를 전송
<code>void SendReadyOkPacket()</code>	3명의 플레이어에게서 준비상태를 수신하면 게임의 초기 상태 정보와 게임시작 이벤트를 전송
<code>void SendGameResultPacket(char winner_id)</code>	제한시간이 종료되면 결과와 게임 종료 이벤트를 담은 패킷 <b>SC_GAME_OVER_PACKET</b> 을 클라이언트에게 전송
<code>void SendPlayerPosPacket(position* pos)</code>	갱신된 플레이어 위치를 전송
<code>void SendObjectsPosPakeet(object* objects)</code>	갱신된 장애물, 먹이의 위치를 전송
<code>void SendDeadPlayerPacket(short player_id)</code>	죽은 플레이어에 대한 정보를 전송
<code>void SendCollisionPakeet(short* scores)</code>	먹이, 장애물 충돌 시 변동된 스코어 값을



	전송
class client{}	
void client::send_add_player(int id)	
void client::send_erase_object(object_info_calculate& oic)	
void client::send_update_object(object_info_calculate& oic)	
void client::send_update_object(client& cl)	
void client::ReSpawn()	
void makeFood()	
void makeObstacle()	
void updateObjects()	
void progress_Collision_po(client& client, object_info_calculate& oic)	
void progress_Collision_mo(object_info_calculate& oic)	
void collision()	
MovePlayer()	
void disconnect(int c_id)	

## 팀원 별 역할분담

손희수	이동현	장형택
기획서 작성	기획서 작성	기획서 작성
서버 베이스 구현	서버 베이스 구현	서버 베이스 구현
(클라) send_Ready 구현	recv_thread 구현	calculate_thread 구현
(서버) SendPlayerPosPacket 구현	(클라)Recv_Data 구현	<del>SendObjectsPosPakeet</del> 구현
(서버) SendGameResultPacket 구현	SendReadyOkPacket 구현	<del>SendCollisionPakeet</del> 구현
Game Scene	<del>SendDeadPlayerPacket</del> 구현 ReSapwn, SendDead 구현	(클라)send_Event 구현
Lobby Scene		(클라)Process_Packet 구현


개발환경 - Visual Studio C++, Window Socket API, Git(GitHub, Github Desktop, SourceTree)

# 개발일정 (일별/개인별 계획 수립, 달력 형태로 작성)

	일	월	화	수	목	금	토		
					11/3	4	5		
희수							서바 베이스 구현 및 서바-클라 이언트간 연결 확인 제안서 수정	서바 베이스 구현 및 서바-클라 이언트간 연결 확인 제안서 수정	개인 학습
동현									개인 학습
형택									개인 학습
	6	7	8	9	10	11	12		
희수	개인 학습	Game Scene 수정 제안서 수정	Game Scene 수정 제안서 수정	개인 학습	Lobby Scene 제작	Lobby Scene 제작	개인 학습		
동현	개인 학습	recv_thre ad 패킷 구분 제안서 수정	recv_thre ad 패킷별 처리 제안서 수정	개인 학습	recv_thre ad 패킷별 처리 및 queue에 저장 서버 기반 작성	recv_thre ad 스레드 동기화, 멀티스레 드화 서버 기반 작성	개인 학습		
형택	개인 학습	calculate_ thread 내부 배열 구분 및 클라이언 트 계산 함수 옮기기 제안서 수정	calculate_ thread 내부 계산 함수 수정(플래 이어 위치) 제안서 수정	calculate_ thread 내부 계산 함수 수정(플래 이어 상대변경)	calculate_ thread 내부 계산 함수 수정(먹이 위치 변경) calculate_ thread 내부 배열 구분 및 클라이언 트 계산 함수 옮기기	calculate_ thread 내부 계산 함수 수정(먹이 위치 랜덤 생성 로직 설계)	개인 학습		

	13	14	15	16	17	18	19
희수	개인 학습	SendPlayerPosPacket 플레이어의 위치 전송 GameScene 수정	SendPlayerPosPacket 완성 GameScene 수정	개인 학습	게임 결과창 구현, sendReady 준비상태 확인 후 게임시작	SendGameResultPacket 게임 결과 전송 sendReady 준비상태 전송	개인 학습 SendPlayerPosPacket 플레이어의 위치 전송
동현	개인 학습	recv_thread 스레드 동기화, 멀티스레드화 recv_thread 작성	(클라)Recv_Data 패킷 구분 (클라)패킷 수신 NetworkThread 작성	개인 학습	(클라)Recv_Data 패킷 별 처리 함수로 연결 개인 학습	개인 학습 ReadyOk 패킷 전송, 3명에서 게임 시작 구현	개인 학습
형택	개인 학습	calculate_thread 내부 계산 함수 수정(장애물 위치 변경 먹이 위치 랜덤 생성 로직 설계)	SC_OBJECT_PACKET 선언 및 적용 계산용 구조체 선언 및 먹이 생성 시 수납	개인 학습	SC_OBJECT_PACKET을 활용한 SendObjectsPosPacket 구현 계산용 구조체에 장애물 생성 시 수납	SendObjectsPosPacket 구현 장애물 이동 구현, 장애물 충돌범위 적용	개인 학습
	20	21	22	23	24	25	26
희수	개인 학습	SendGameResultPacket 게임 결과창에 띄우기 SendPlayerPosPacket 완성	SendGameResultPacket 결과 확인 루버 이동 SendGameResultPacket 게임 결과창	개인 학습	개인 학습	sendReady 준비상태 전송 SendGameResultPacket 게임 결과 플레이어에 전송	개인 학습
동현	개인 학습	SendDeadPlayerPacket 사망한 플레이어 id 전송 각 클라로 add_player 패킷 전송, 여러 패킷이 한번에 올 경우의 처리	SendDeadPlayerPacket 사망한 플레이어 클라이언트 처리 게임 시작 시 랜덤 좌표 spawn,	개인 학습	SendReadyOk 클라이언트 별 ready 확인 구현 다른 플레이어 출력 및	개인 학습	개인 학습

			클라이언트 플레이어 움직임 수정		이동 처리, 좌표 전송 오류 수정		
형택	개인 학습	<del>SendObjectPosPacket</del> 구현 객체 간 충돌 검사 함수 선언 및 적용, 로그 확인	<del>send_Event</del> 내부 이벤트 구분 함수 선언 및 구현 계산용 구조체 관련 버그 수정	개인 학습	<del>send_Event</del> 내부 이벤트 구분 함수 구현 CS_LBUT TONCLICK 마우스-장 애물 간 충돌처리	<del>send_Event</del> 내부 이벤트 처리 및 수신한 패킷 내용 서버에 적용 (클라) 먹이 및 장애물 렌더링 관련 처리	개인 학습 장애물-플 레이어, 먹이-플레 이어 간 충돌 처리 및 서버-클라 이언트 간 정보 공유

	27	28	29	30	12/1	2	3
희수	개인 학습	<del>send_Ready</del> 준비상태 확인 후 게임시작 SendGameResultPacket 결과 확인 후 로비화면 이동	개인 학습 게임 결과창 구현	개인 학습	버그 픽스, 세부사항 검토	버그 픽스, 세부사항 검토	개인 학습
동현	개인 학습	<del>SendReadyOk</del> ready 확인 및 클라이언트 로 상대 전송 클라이언트 종료시 알려주는 SC_LEAVE_PLAYER_PACKET 정의, 처리	<del>SendReadyOk</del> ready 게임 추가 시작 상대 구현 클라이언트 에서 게임 종료를 서버로 알리는 패킷 추가, 처리	개인 학습			개인 학습
형택	개인 학습	<del>SendCollisionPacket</del> 에서 보낼 배열 선언 및 내부 인자 정리 장애물-플 레이어, 먹이-플레 이어 간	<del>SendCollisionPacket</del> 에서 보낼 배열 구분 및 처리 구현 장애물-플 레이어, 먹이-플레 이어 간 충돌 처리	<del>SendCollisionPacket</del> 에서 보낼 배열 구분 및 처리 구현 장애물-플 레이어, 먹이-플레 이어 간 충돌 처리			개인 학습

		충돌 처리 및 서버-클라 이언트 간 정보 공유	및 서버-클라 이언트 간 정보 공유	및 서버-클라 이언트 간 정보 공유			
	4	5	6	7	8	9	10
희수	개인 학습	버그 픽스, 세부사항 검토	버그 픽스, 세부사항 검토	버그 픽스, 세부사항 검토	버그 픽스, 세부사항 검토	패킷 전송 제한, 패킷 처리 검토	패킷 전송 제한, 패킷 처리 검토
동현	개인 학습						
형택	개인 학습						
	11	12	1주차 - 기획 2주차 - 클라이언트 멀티플레이 대응 관련 구현 및 서버 베이스 구현 3주차 - 서버 제작 4주차 - 서버 send ,calculate 스레드 구현 클라이언트 send recv 구현 5주차 - 동기화 작업 및 테스트 6주차 - 오류수정 및 세부사항 검토 7주차 - 패킷 처리 검토 및 최종 검수				
희수	패킷 전송 제한, 패킷 처리 검토	최종 검수					
동현							
형택							