네트워크 게임 프로그래밍
Term Project 추진 계획서

2019180009 김선빈
2020182019 안현우
2022182039 최정민

목차

1. 애플리케이션 기획

1 - 1 게임이름

- 땅따먹기

1 - 2 게임출처

- 2019180009 김선빈 컴퓨터 그래픽스 텀 프로젝트

1 - 3 게임특징

- 3D 백뷰 게임으로 제한시간내에 차지한 땅이 많은 플레이어가 이기는 게임

- 곳곳에 아이템을 먹어 특정 효과 부여

- 상대 플레이어를 공격하여 부정적 효과를 부여

1 - 4 인게임 요소

- 플레이어
  - 총알
- 아이템
- 기타
  - 맵
  - 점수

1 - 5 멀티게임 특징

- 플레이어가 지나가는 큐브가 해당 플레이어 색깔로 변경되며 제한 시간내에 더 많은 큐브를 먹는 플레이어가 승리

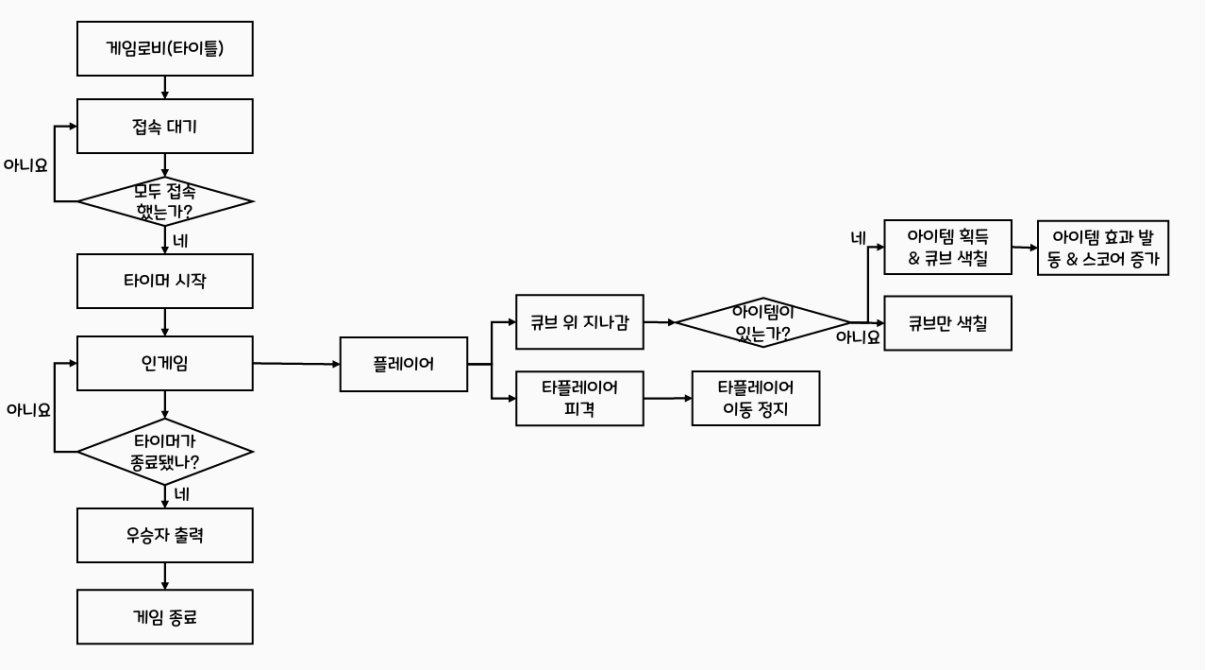
- 만약, 차지한 땅에 다른 플레이어가 지나간다면 해당 플레이어 땅으로 변경됨

- 총을 쏴서 다른 플레이어를 맞추면 맞은 플레이어의 움직임이 멈춰짐

- 맵에 출현되는 아이템은 서로 공유되기 때문에 한 플레이어가 먹으면 사라짐

- 게임 종료 후 어떤 플레이어가 이겼는지 출력함

1 - 6 전체 게임 흐름도



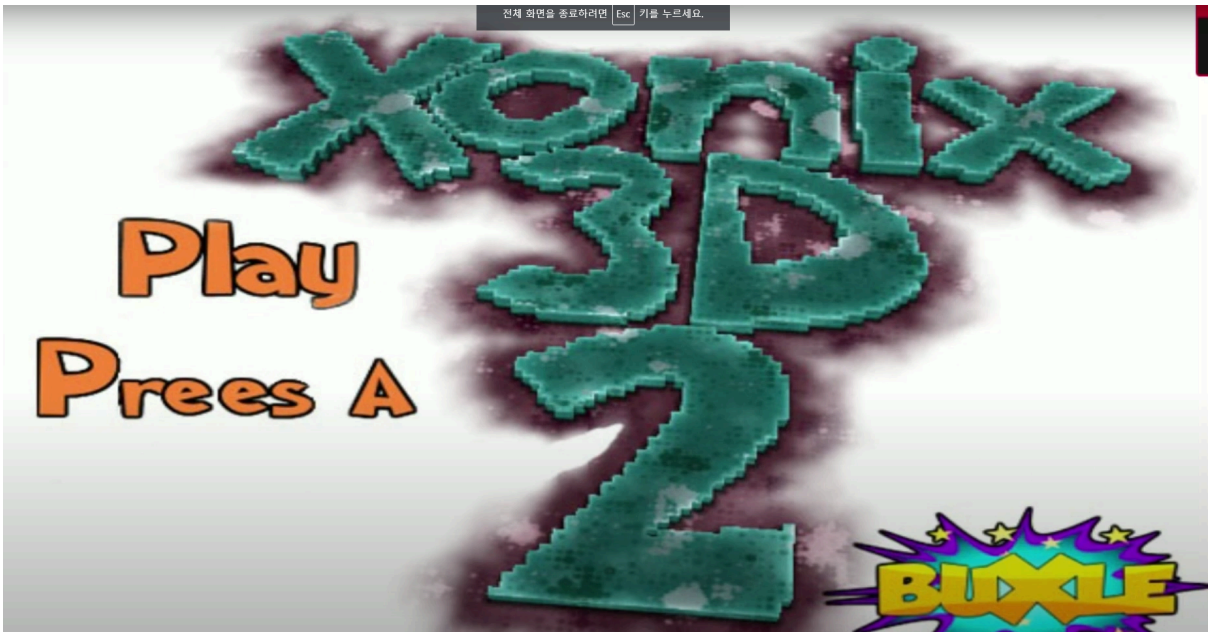
1 - 7 게임 조작키

이동 키 - 전후좌우 이동

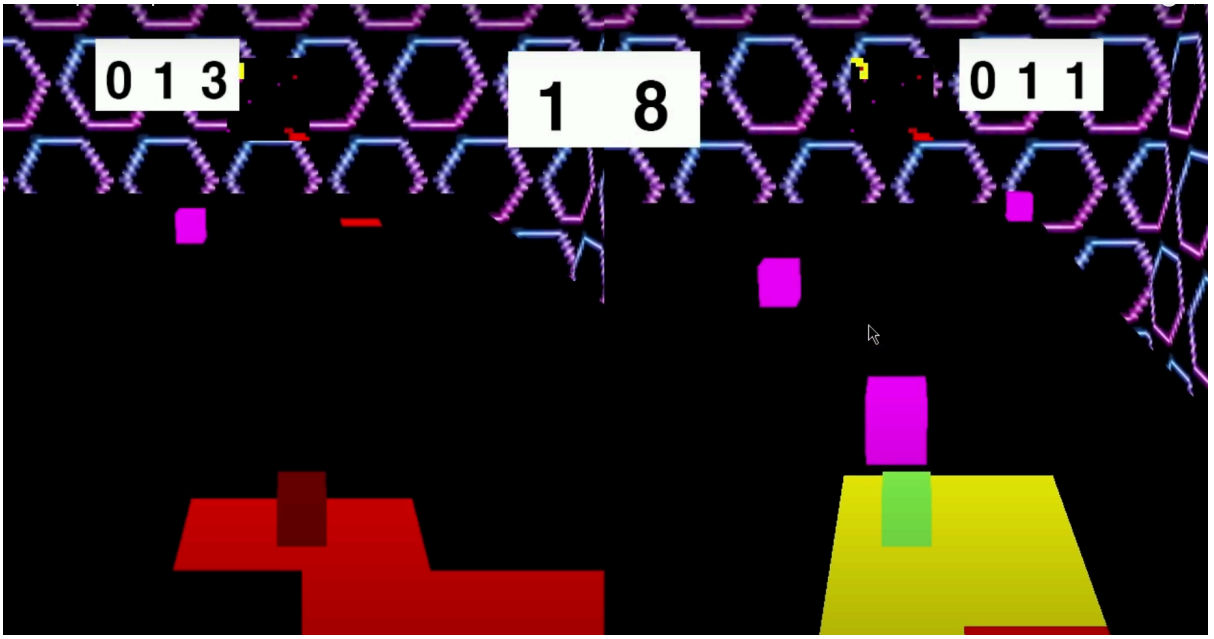
A키 - 공격키

Q,E키 - 카메라 회전 키

1 - 8 게임 조작키

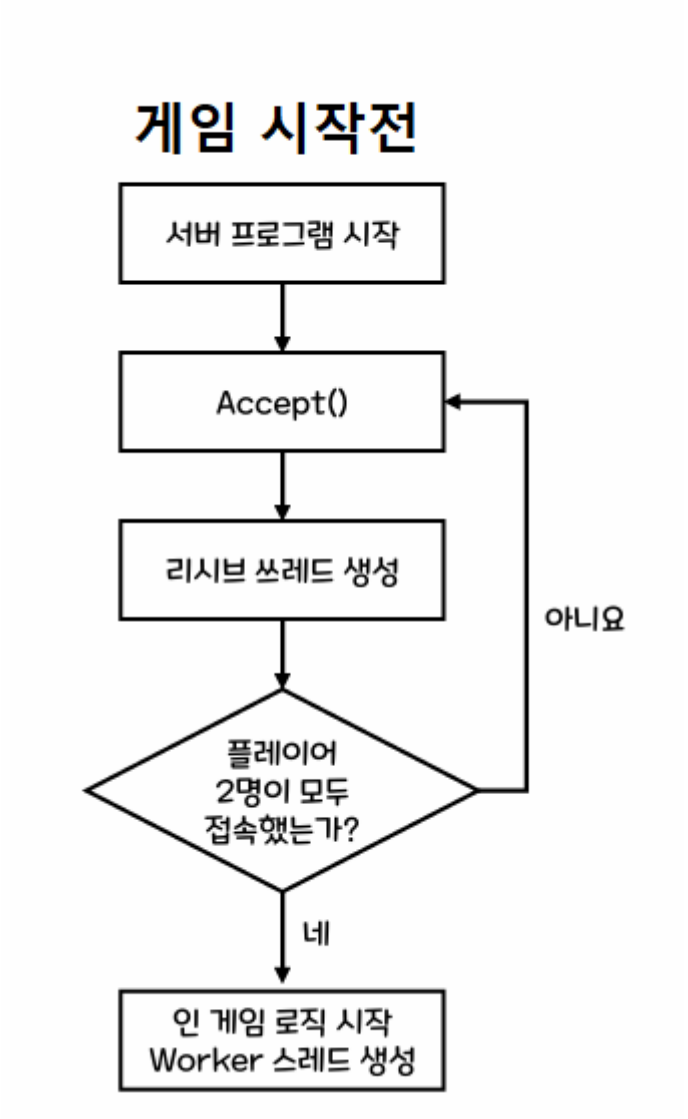


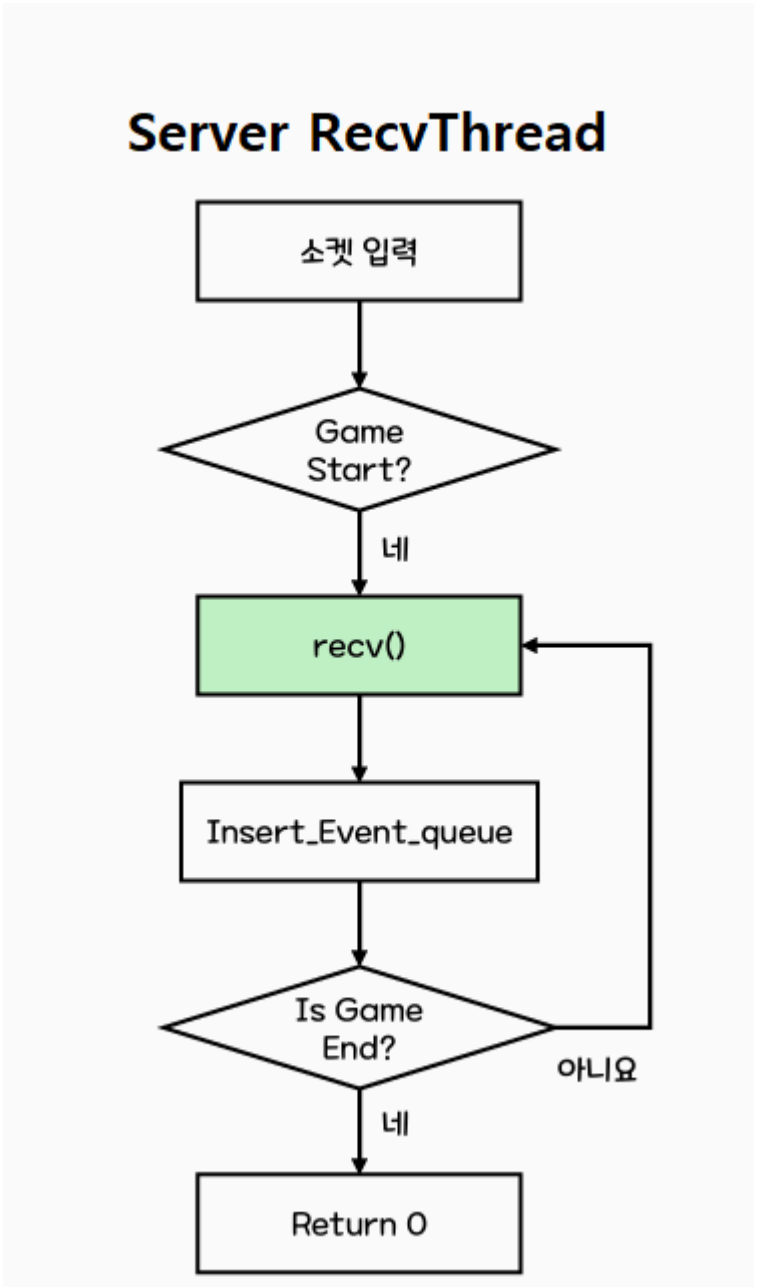
- 현재 로비 화면**(**추후 변경 예정**)**

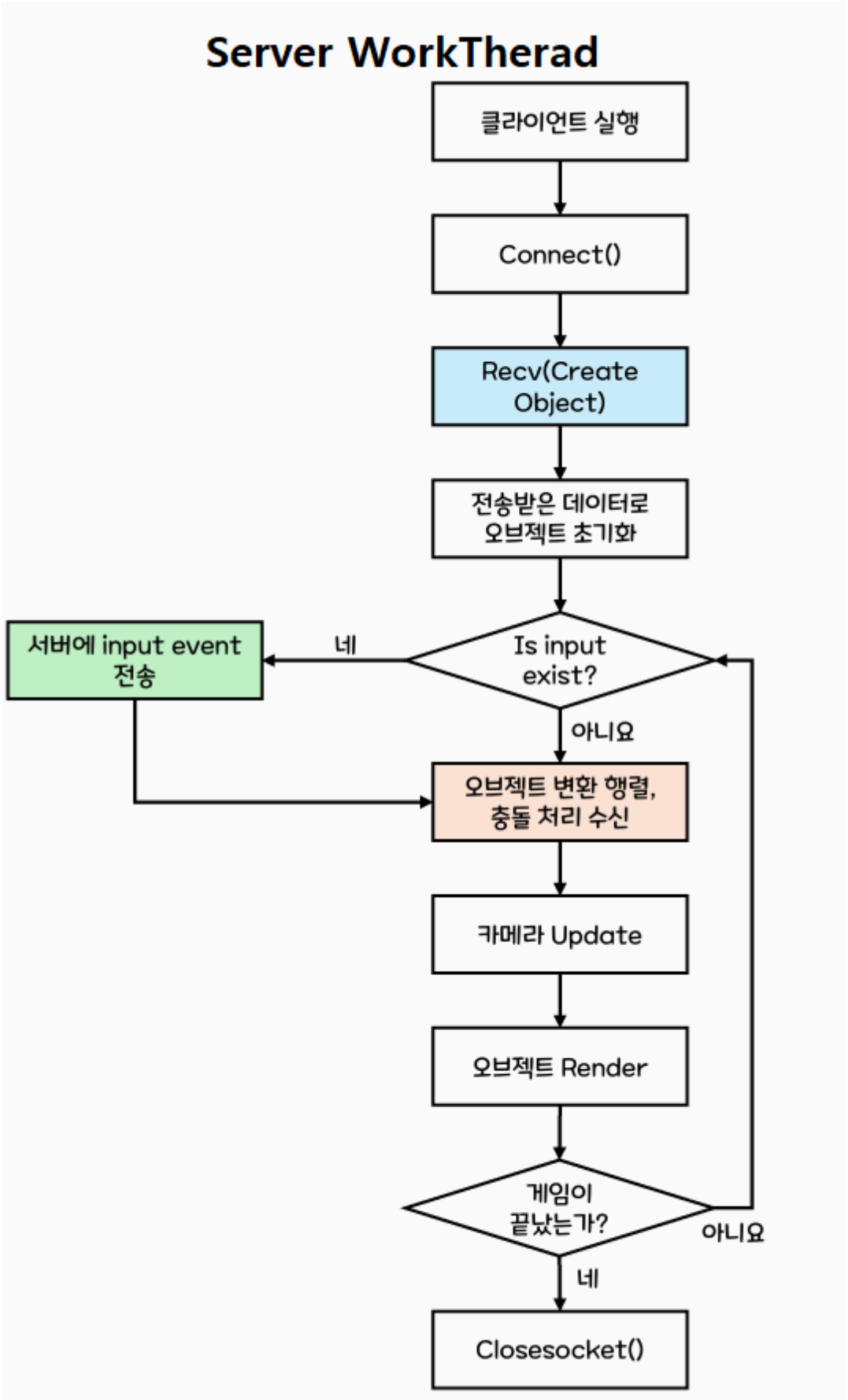


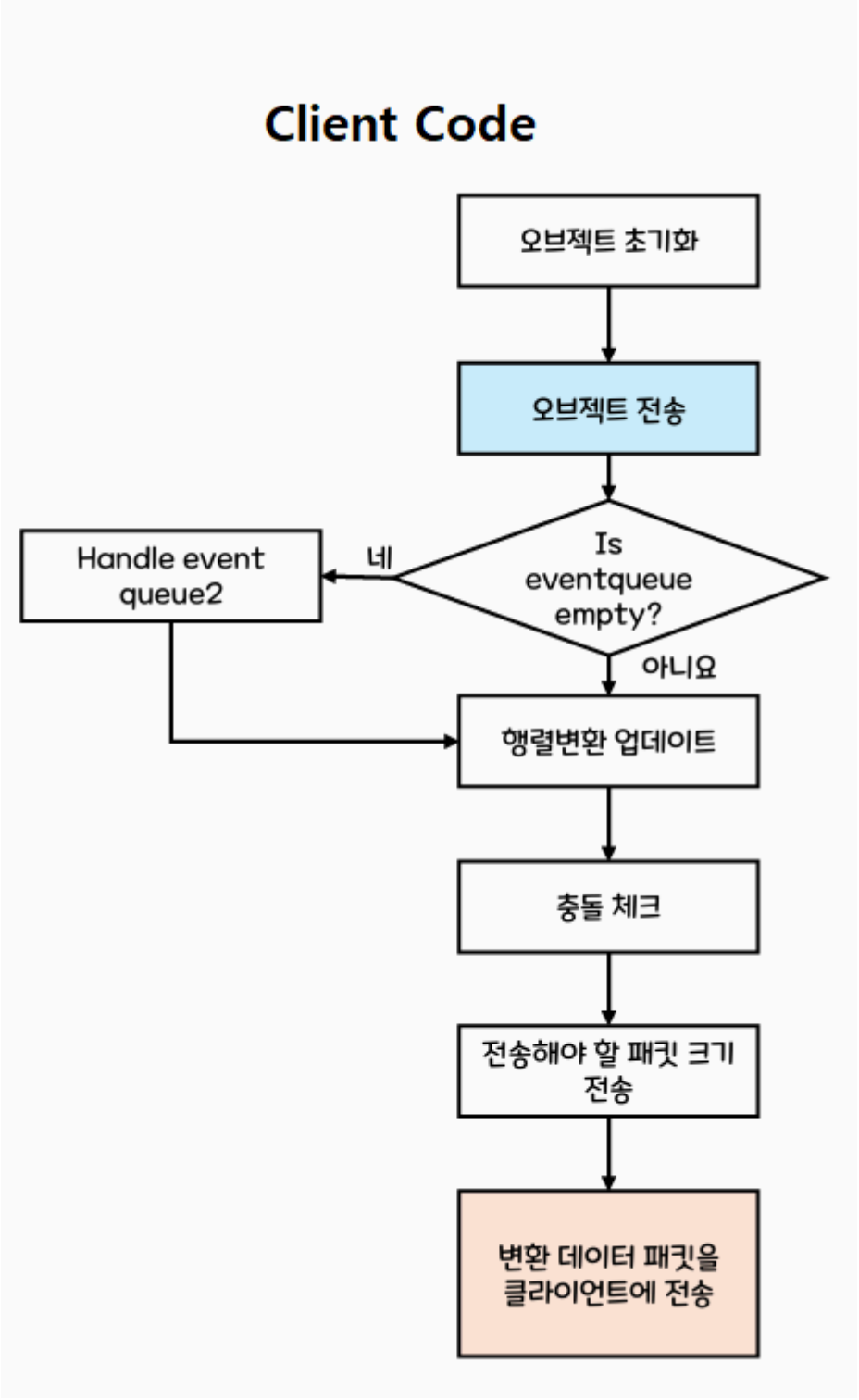
- 현재 인게임 화면**(**추후 플레이어 한명만 보이게 만들 예정**)**

2. High Level Design




Server RecvThread
소켓 입력
Game Start?
네
recv()
Insert_Event_queue
Is Game End?
아니요
네
Return 0


Server WorkTherad
클라이언트 실행
Connect()
Recv(Create Object)
전송받은 데이터로 오브젝트 초기화
Is input exist?
네
서버에 input event 전송
아니요
오브젝트 변환 행렬, 충돌 처리 수신
카메라 Update
오브젝트 Render
게임이 끝났는가?
아니요
네
Closesocket()


Client Code
오브젝트 초기화
오브젝트 전송
Is eventqueue empty?
네
Handle event queue2
아니요
행렬변환 업데이트
충돌 체크
전송해야 할 패킷 크기 전송
변환 데이터 패킷을 클라이언트에 전송

3. Low Level Design

3 - 1 공통 헤더파일

```
typedef enum {
   KEY_UP    = 0x01,  // UP 키
   KEY_DOWN  = 0x02,  // DOWN 키
   KEY_LEFT  = 0x04,  // LEFT 키
   KEY_RIGHT = 0x08,  // RIGHT 키
   KEY_Q     = 0x10,  // Q 키 (0x016 -> 0x10)
   KEY_E     = 0x20,  // E 키 (0x032 -> 0x20)
   KEY_A     = 0x40   // A 키 (0x064 -> 0x40)
} KeyInput;

typedef struct playerInput {
  byte p_index    // - 플레이어 1인지 2인지 구분(1 or 2)
  byte input_key    // - 1 ~ 7 입력된 키 정보
} playerInput;


typedef enum {
   PACKET_MOVE_PLAYER       = 0x01, // Move_Player
   PACKET_CREATE_BULLET     = 0x02, // Create_bullet
   PACKET_CREATE_ITEM       = 0x03, // Create_Item
   PACKET_DELETE_ITEM       = 0x04, // Delete_Item
   PACKET_DELETE_BULLET     = 0x05, // Delete_bullet
   PACKET_CHANGE_FLOOR      = 0x06, // Change_floor
   PACKET_MOVE_BULLET       = 0x07, // Move_bullet
   PACKET_UPDATE_TIMER      = 0x08, // Update_timer
   PACKET_UPDATE_SCORE      = 0x09  // Update_score
} PacketType;
```

다음은 무슨 변환이 일어났는지 **Send, Recv**하기 위한 구조체

```
struct Parent_packet{
	byte pakcet_type;
}

struct Move_Player : public Parent_packet{
	byte player_index;
	플레이어 정보(위치 값) -> struct 으로 만들어야됨.
}

struct Create_bullet : public Parent_packet{
	byte player_index;
	byte index;
}

struct Create_Item : public Parent_packet{
	byte index;
	item 정보(item 색깔, 좌표) -> struct으로 만들어야됨.
}

struct Delete_Item : public Parent_packet{
	byte index;
}

struct Delete_bullet : public Parent_packet{
	byte player_index;
	byte index;
}


struct Change_floor : public Parent_packet{
	byte index;
	박스 정보(박스 색깔)
}

struct Move_bullet : public Parent_packet{
	byte player_index;
	byte index;
	총알 정보(위치)
}

struct Update_timer : public Parent_packet{
	short timer;
}

struct Update_score : public Parnet_pakcet{
	short my_score
	short enermy_score;}
```

3 - 2 EventQueue

```
class EventQueue {
public:
    void addEvent(const std::function<void()>& func) {
        queue.push(func);
    }

    void executeAll() {
        while (!queue.empty()) {
            auto event = queue.front();
            event();
            queue.pop();
        }
    }

private:
    std::queue<std::function<void()>> queue;
};
```

위 **EventQueue** 클래스를 사용하여
Player, Item 같은 **class** 내부 함수 또는 함수를 **EventQueue**에
std::bind 사용하여 **Queue** 삽입시키고
EventQueue에서 꺼낼 때 바로 해당하는 함수가 불러지게 만들
예정입니다.

ex)

```
Player player1;
Player player2;

eventQueue.addEvent(std::bind(&Player::move, &player1));
eventQueue.addEvent(std::bind(&Player::move, &player2));

eventQueue.executeAll();
```

3 - 3 Worker_Thread

Worker_thread
{
Reset_Object(); // 오브젝트 초기화(여기서 시간도 초기화시킴)
send(오브젝트**)** // 각각 클라이언트로 생성한 오브젝트 전송

while(true){

Timer Check 작동(아이템 생성 이벤트, 게임 종료 체크, 프레임 사이에 지나간 체크,스타트시간으로부터 지나간 체크, 플레이어 정지 시간 체크)

// 임계 영역 진입

EventQueue.executeAll();

// 임계 영역 탈출

행렬 변환

충돌체크(발생하는 이벤트들 이벤트큐에 넣어줌)

전송할 패킷리스트 개수 전송(고정크기)

for(int i = 0 ; i < 패킷 개수 **; i++){**
send(클라이언트 **1,**패킷**)**
send(클라이언트 **2,**패킷**)**
}
}
}

3 - 4 Client

recv(오브젝트**)**

```
while(true){
	if(key_change){
		send(playerInput,sizeof(playerInput));
		key_change = false;
	}

	Parent_pakcet packet;

	패킷개수를 고정크기로 recv

	for(int i = 0 ; i < 패킷 개수 ; i++)
	{
		recv(client_socket, &packet, sizeof(packet), 0)
		handlePacket(packet)
	}

	카메라 업데이트

	오브젝트 렌더
}
```

```
void handlePacket(const Parent_packet& packet){
	switch (packet.packet_type){
		case move_player:
			const Move_Player& move_packet = static_cast<const
Move_Player&>(packet);
			해당하는 처리
		case create_bullet:
			const Create_bullet& Create_bpacket = static_cast<const
Create_bullet&>(packet);
			해당하는 처리
			.
			.
	}
}
```

key_change에 경우 윈도우에서 **Key_down, Key_Up** 이벤트가 불러질 때
true로 바뀌어 **input**이 됐는지 안됐는지 체크합니다**.**

4. 개발 일정

2024년 11월

일요일	월요일	화요일	수요일	목요일	금요일	토요일
27 1. 김선빈 2. 최정민 3. 안현우	28	29	30	31	1 기획 검수	2 개인 공부
3 개인 공부	4 Client 플레이어 클래스 수정 바닥(맵) 클래스 수정 아이템 클래스 수정	5 시험공부 시험공부 시험공부	6 State 클래스 구현 Timer 클래스 수정 Client 총알 클래스 구현	7 졸작기획 회의	8 stay 클래스 구현 score 클래스 수정 Struct 구조체, enum선언	9 start 클래스 구현 피드백 및 부족한 점 채우기
10 개인 공부	11 시험공부 시험공부 시험공부	12 시험공부 시험공부 시험공부	13 게임 구동 확인	14 졸작기획 회의	15 G스타 클라 Send 구현 서버 recv_Thread 구현	16 G스타 피드백 및 부족한 점 채우기
17 G스타 개인 공부	18 클라이언트 Send와 서버 recv_Thread를 통해 네트워크 구동 확인	19 Worker_Thread 뼈대 (Reset Object, Object Send) 구현 클라 recv_Thread 뼈대 구현 Move_Player packet에 사용할 함수 구현	20 Timer Check함수 구현 Player Key Event 구현(상하좌우, 카메라 회전) Create_bullet packet에 사용할 함수 구현	21 졸작기획 회의	22 행렬 변환, 충돌체크 구현 Player Key Event(A키) Create_Item packet에 사용할 함수 구현	23 피드백 및 부족한 점 채우기
24 개인 공부	25 전송할 패킷리스트와 패킷 send 구현 Timer Event(아이템 생성)구현 Delete_Item packet에 사용할 함수 구현	26 블록과 플레이어 충돌 시 Event 구현 Timer Event(플레이어 정지) 구현 Delete_bullet packet에 사용할 함수 구현	27 플레이어와 총알 충돌 시 Event 구현 Change_floor packet에 사용할 함수 구현 Move_bullet packet에 사용할 함수 구현	28 졸작기획 회의	29 End state 구현 및 End Event 구현 Update_Timer에 사용할 함수 구현 Update_score에 사용할 함수 구현	30 피드백 및 부족한 점 채우기

2024년 12월

일요일	월요일	화요일	수요일	목요일	금요일	토요일
1 개인 공부	2 서버와 클라 이용하여 디버깅	3 부족한 점 수정	4 부족한 점 수정	5 졸작기획 회의	6 최종 점검 및 정리	7 개인 공부
8 개인 공부	9 최종 보고서 작성	10 최종 보고서 검토 및 제출	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4