서버 포트폴리오

WITCH OF TIME

- 제작 목적: 언리얼 엔진을 이용한 3D게임 및 서버 제작 경험
- 개발 기간: 2020년 12월 부터~ 2021년 8월 까지 약 8개월
- 사용도구: UNREAL ENGINE 4.25.4, VISUAL STUDIO 2019
- 프로토콜: TCP/IP -> IOCP
- 개발 언어: C++
- 개발 인원: 3인
- 담당 업무: 기본적인 서버 시스템 구현

  블록 배치, 삭제, 커맨드, 로드 • 캐릭터 이동 등의 기본적인 서버 연동 및 클라이언트 적용

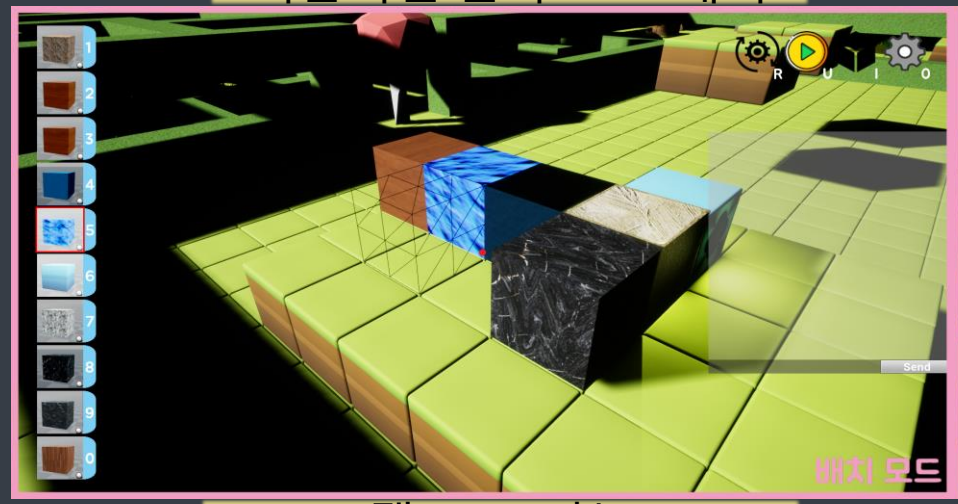
  간단한 채팅 기능 및 UI 구현,

어려웠던 점: 언리얼을 처음 다뤄보아 제공기능을 잘 사용하지 못한 경우가 있었으나 언리얼을 잘 다루는 팀원의 도움으로 기본적인 언리얼 기능을 습득.
서버를 배워나가면서 진행한 프로젝트라 서버 구현에 어려움이 있었지만 지도교수님 조언, 팀원간 협업을 통하여 혼자 해결하지못한 문제를 해결.

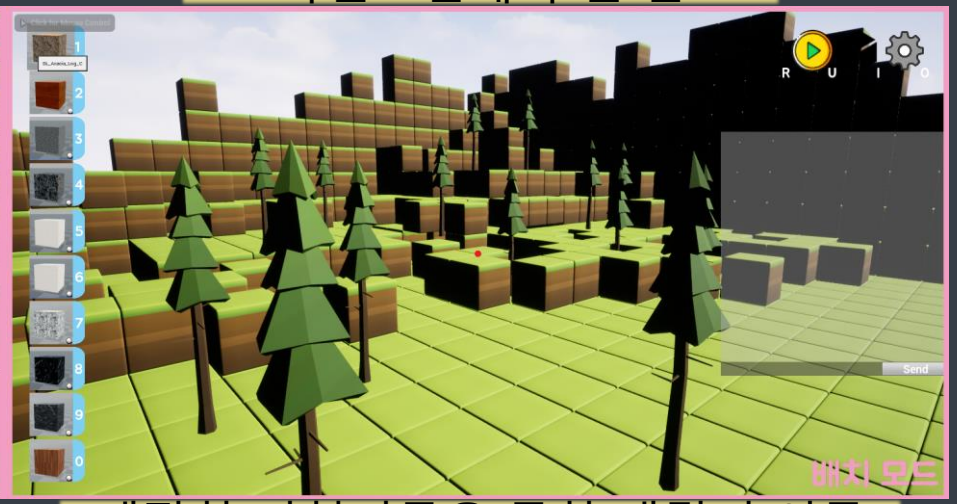
GITHUB 주소: https://github.com/yimgunho/Witch_of_Time

WITCH OF TIME - 게임 스크린샷

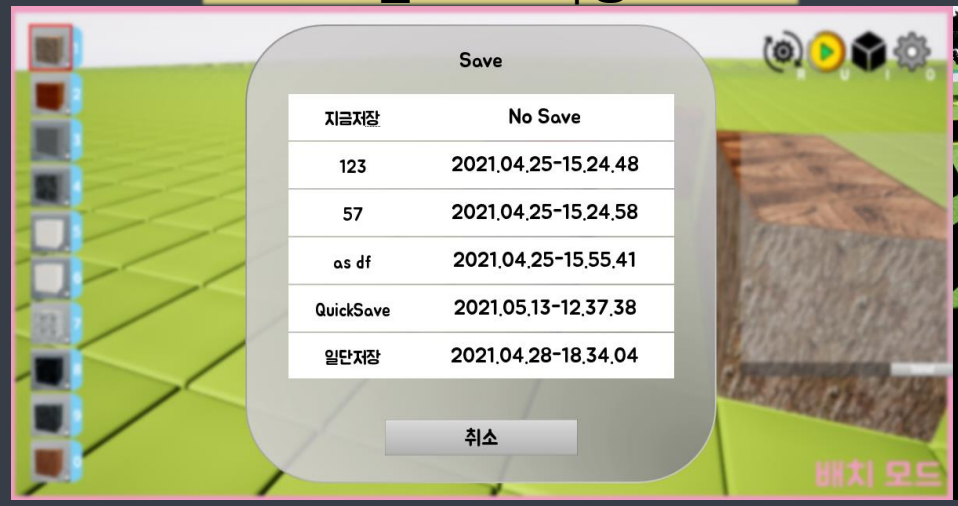
기본적인 블록 요소 배치



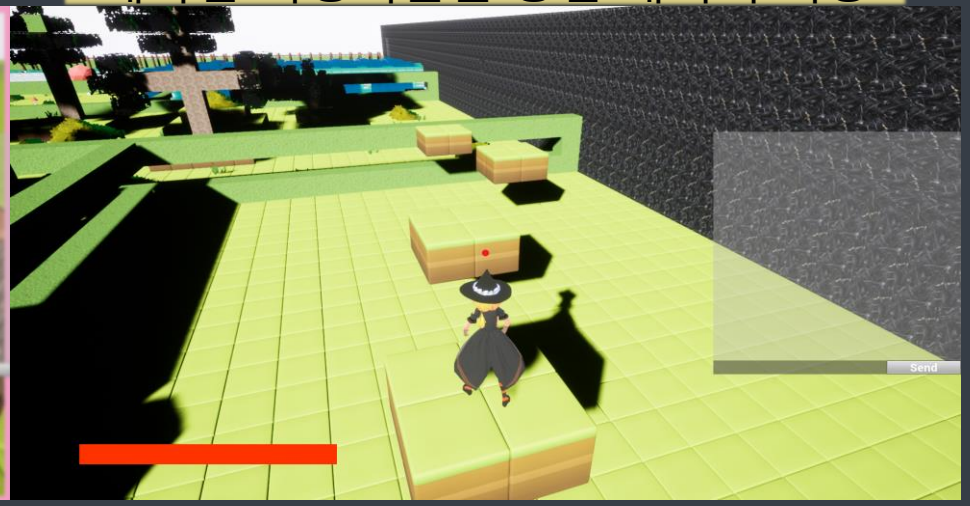
자연스럽게 꾸민 맵

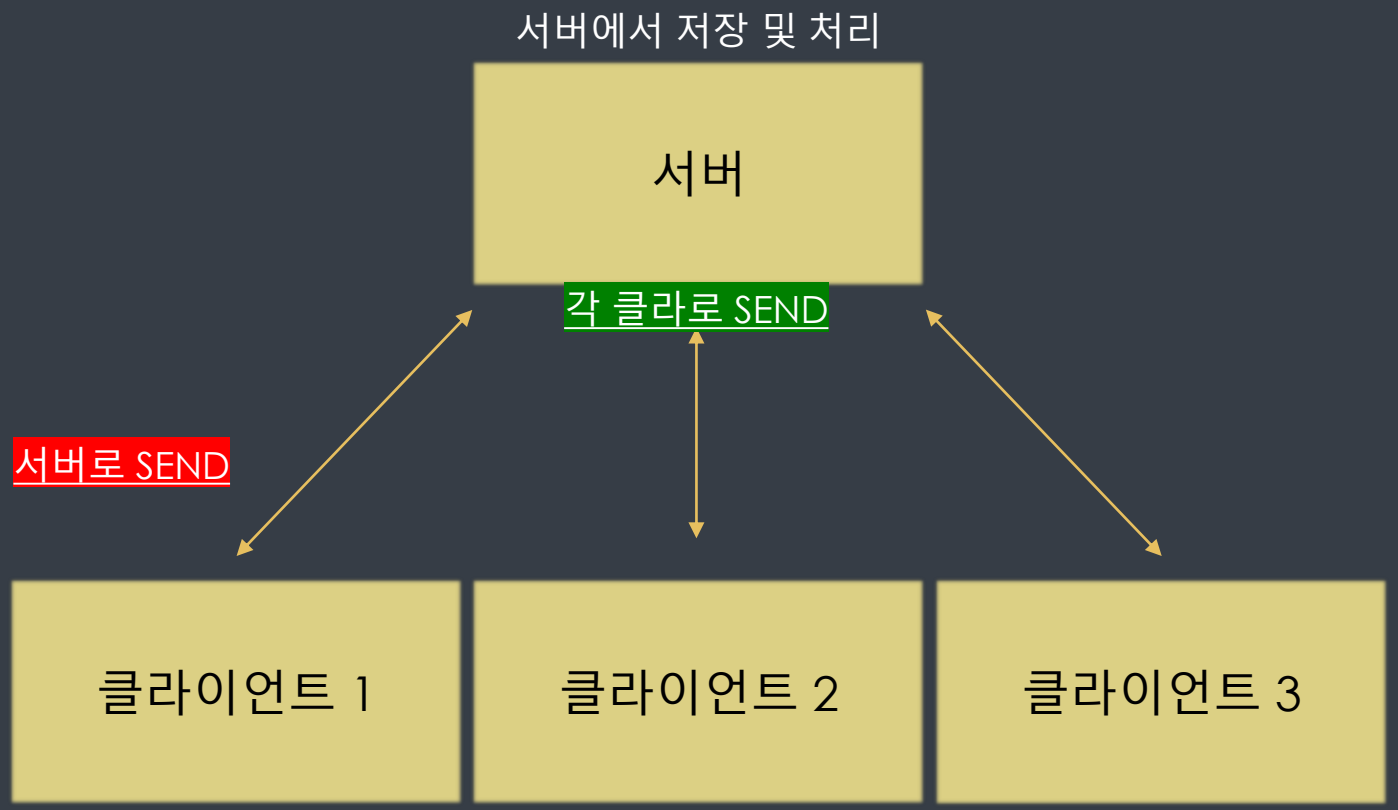


맵 로드 기능



제작한 지형지물을 통한 캐릭터 이동





클라이언트에서 서버에 SEND
서버에서 받은 정보를 저장 및 처리
서버에서 모든 클라이언트에 처리한 정보를 SEND

예시 - 블록

```
void Aclient::send_block_packet(int blockindex,
    float block_pos_x,
    float block_pos_y,
    float block_pos_z)
{
    BlockPacket blockpacket;
    blockpacket.blockindex = blockindex;
    blockpacket.blocklocation_x = block_pos_x;
    blockpacket.blocklocation_y = block_pos_y;
    blockpacket.blocklocation_z = block_pos_z;

    send_packet(&blockpacket);
}
```

클라이언트 SEND

```
case BLOCK:
{
    auto cast = reinterpret_cast<BlockPacket*>(buffer);

    BlockPacket blocklistpacket;
    blocklistpacket.blockindex = cast->blockindex;
    if (blocklistpacket.blockindex == 76)
    {
        start_x = cast->blocklocation_x;
        start_y = cast->blocklocation_y;
        start_z = cast->blocklocation_z;

        blocklistpacket.blocklocation_x = cast->blocklocation_x;
        blocklistpacket.blocklocation_y = cast->blocklocation_y;
        blocklistpacket.blocklocation_z = cast->blocklocation_z;

        Broadcast_Packet(&blocklistpacket);

        break;
    }

    int blockid = get_new_block_id();
    blocklistpacket.block_id = blockid;

    blocklistpacket.blocklocation_x = cast->blocklocation_x;
    blocklistpacket.blocklocation_y = cast->blocklocation_y;
    blocklistpacket.blocklocation_z = cast->blocklocation_z;

    objects[blockid].block_index = cast->blockindex;
    objects[blockid].x = cast->blocklocation_x;
    objects[blockid].y = cast->blocklocation_y;
    objects[blockid].z = cast->blocklocation_z;

    if (cast->blockindex == 68 || cast->blockindex == 75)
    {
        // 몬스터 블럭일 경우
        objects[blockid].hp = 100;
        monster_block_id[blockid] = cast->blockindex;
    }

    Broadcast_Packet(&blocklistpacket);
}
```

서버에서 처리 후 SEND

```
switch (ex_over->m_op)
{
case OP_RECV: {
    unsigned char* packet_ptr = ex_over->m_packetbuf;
    int num_data = num_bytes + m_prev_size;
    int packet_size = packet_ptr[0];

    while (num_data >= packet_size)
    {
        process_packet(key, packet_ptr);
        num_data -= packet_size;
        packet_ptr += packet_size;
        if (0 >= num_data)  break;
        packet_size = packet_ptr[0];
    }
    m_prev_size = num_data;
    if (num_data != 0)  memcpy(ex_over->m_packetbuf, packet_ptr, num_data);

    do_recv(key);
}
```

```
switch (p_buf[4])
{
case BLOCK:
{
    auto cast = reinterpret_cast<BlockPacket*>(p_buf);

    spawn_block(cast->blockindex,
        cast->block_id,
        cast->blocklocation_x,
        cast->blocklocation_y,
        cast->blocklocation_z);
}
```

클라이언트 RECV 및 적용

클라이언트에서 서버에 SEND
서버에서 받은 정보를 저장 및 처리
서버에서 모든 클라이언트에 처리한 정보를 SEND