이브와

SW개발/HW제작 설계서

프로젝트 명 : haptic을 활용한 메타버스 실감교육 플랫폼 '메타블록'

2022. 11. 01

수행 단계별 주요 산출물

단계	산출물	일반 ·모바일 APP ·Web 등	응용 소프트웨어 ·빅데이터 ·인공지능 ·블록체인 등	응용 하드웨어 ·IoT ·로봇 ·드론 등
환경 분석	시장/기술 환경 분석서	△	△	△
	설문조사 결과서	△	△	△
	인터뷰 결과서	△	△	△
요구사항 분석	요구사항 정의서	○	○	○
	유즈케이스 정의서	▲	▲	▲
아키텍처 설계	서비스 구성도(시스템 구성도)	○	○	○
	서비스 흐름도(데이터 흐름도)	△	○	△
	UI/UX 정의서	△	△	△
	하드웨어/센서 구성도	-	-	○
기능 설계	메뉴 구성도	○	○	○
	화면 설계서	○	○	△
	엔티티 관계도	○	○	△
	기능 처리도(기능 흐름도)	○	○	○
	알고리즘 명세서/설명서	△	○	○
	데이터 수집처리 정의서	-	○	-
	하드웨어 설계도	-	-	○
개발 / 구현	프로그램 목록	○	○	○
	테이블 정의서	○	○	△
	핵심 소스코드	○	○	○

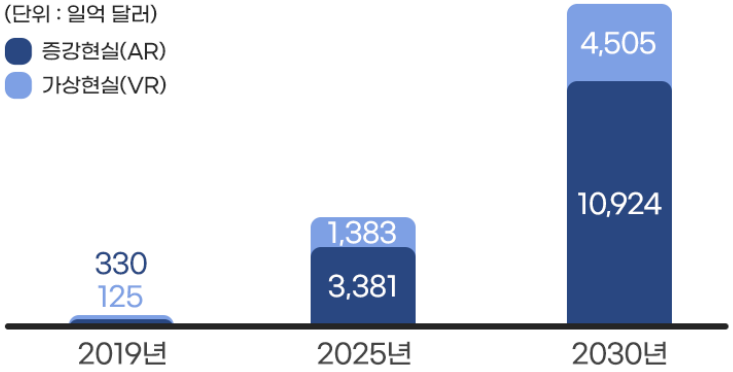
※ ○ 필수, △ 선택

| 시장/기술 동향 분석

코로나 장기화로 '비대면', '집콕'이 일상화되고 다양한 메타버스 플랫폼이 등장하면서 MZ세대를 중심으로 메타버스에 대한 관심이 증폭되고 있다. 급성장한 메타버스 적용 범위는 게임, 생활·소통 서비스를 넘어 전 산업과 사회 분야로 확산·적용되어 그 영향력이 확대되고 있다. 또 사회적 화두로 떠오른 메타버스를 마케팅에 활용하는 기업들이 늘면서 관련 버즈량도 급증하고 있는 것으로 분석됐다.

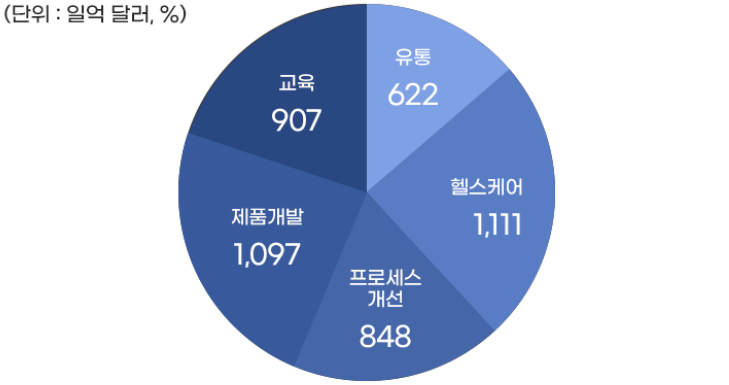
| 시장/기술 동향 분석

메타버스 시장전망



자료 : PwC, 하나금융연구소

2025년 메타버스 주요 응용시장 전망



자료 : PwC, 하나금융연구소

PwC, 하나금융 연구소의 메타버스 시장전망 자료에 따르면 메타버스(AR+VR)의 시장은 급속도로 성장할 예정이다. 2019년 455억 달러에 불과했던 메타버스 시장규모는 2030년에 2019년 시장규모보다 약 330배 증가한 15,429억 달러일 것으로 예측된다. 또한 2025년 메타버스 주요 응용시장 전망은 제품개발, 헬스케어, 교육등이 있다. 이 중에서 우리팀의 프로젝트 'haptic을 활용한 메타버스 실감교육 플랫폼 '메타블록''와 연관된 교육분야는 전체에서 약 20%를 차지하고 있다.

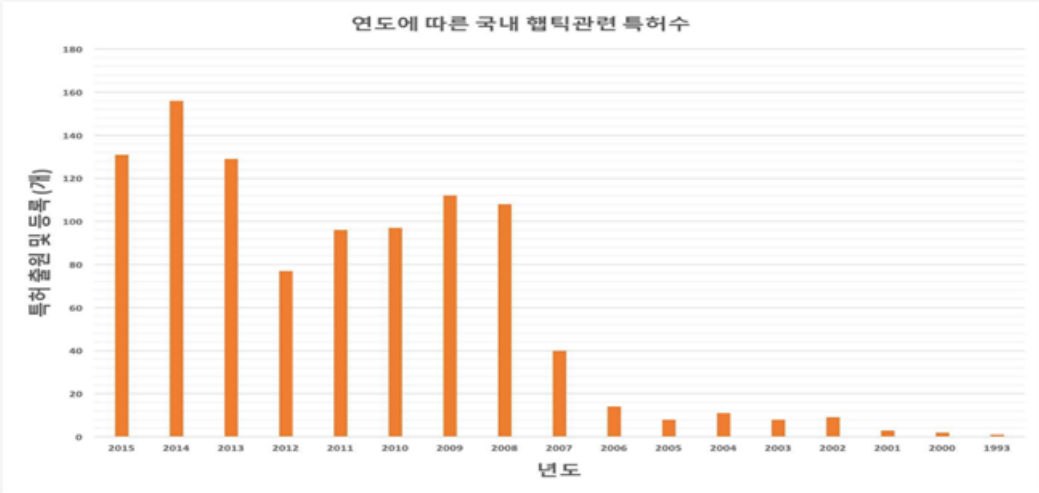
햅틱 시장 현황과 전망

햅틱 기술은 사용자의 인터페이스로 촉각적 경험이나 피드백을 통합하는 기술로 정의될 수 있으며 진동, 움직임 또는 다른 힘을 통한 촉각 감각을 만들어낼 수 있는 기술이다. 그러므로 햅틱은 사용자에게 힘, 진동 또는 움직임이 전달되도록 함으로써 사용자가 촉감을 느끼도록 할 수 있다. 글로벌 햅틱 기술 시장의 성장 동력으로 소비자 기기에서의 도입확대, 의료 산업에서 햅틱 기술의 채택 증가, 자동차 안전 응용 분야에서 햅틱 기술의 채택 증가 등이 있다.

(단위 : 억원)

년도	(2017 년) 현재년도	(2020 년) 개발 종료후 1년	(2022 년) 개발 종료후 3년
세계 시장 규모	99,110	153,670	215,050
한국 시장 규모	----------	----------	----------

위의 수치는 Market and Market Report 2016에 실린 그림 2의 그래프를 참고하였다. 국내 시장 규모는 관련 자료의 부족으로 명시하지 못하였다. 햅틱스 관련 시장은 특히 높은 연평균 성장률(16.2%)을 보여주고 있다.



위의 자료는 국가 전자과학검색엔진에서 추출하였다. 1993년 첫 특허를 기점으로, 2007년 스마트폰의 보급을 기점으로 햅틱 관련 특허의 수가 급격하게 증가하였다. 이후 5년간 연간 100여개 정도 규모에서 천천히 감소하는 추세를 띄었다가 2013년 이후 최근 다시 특허 수가 증가하는 추세를 보인다. 햅틱 관련 국내 특허를 국제특허분류(International Patent Classification; IPC)로 분류해 보면, G06F 코드의 특허가 전체의 70%에 육박한다. 이 코드는 컴퓨터 및 계산과 관련된 전자관련 디지털 신호 처리 관련 코드로서 햅틱 관련 특허에서는 입출력 장치 관련 특허들이 대다수로 파악된다.

BlockStack3D

BlockStack3D는 국내 충청남도 서산시에서 만들어진 3D 블록쌓기 게임이다. 안드로이드 앱을 통해 플레이되는 게임으로써 사용자가 원하는 방식대로 블록을 쌓을 수 있다. 사용자가 플레이 할 때마다 블록의 색깔과 형태가 다르게 구성됨으로써 제품의 완성도를 높이고 사용자의 미학적 만족도를 높였다. 블록을 빈틈없이, 높게 쌓을수록 높은 점수를 받으며 쌓은 블록의 높이를 측정해준다. 블럭쌓기에 실패하면 게임오버 표시가 뜨며 게임이 종료된다

Fun Games For Free의 Block Craft 3D : Building Game

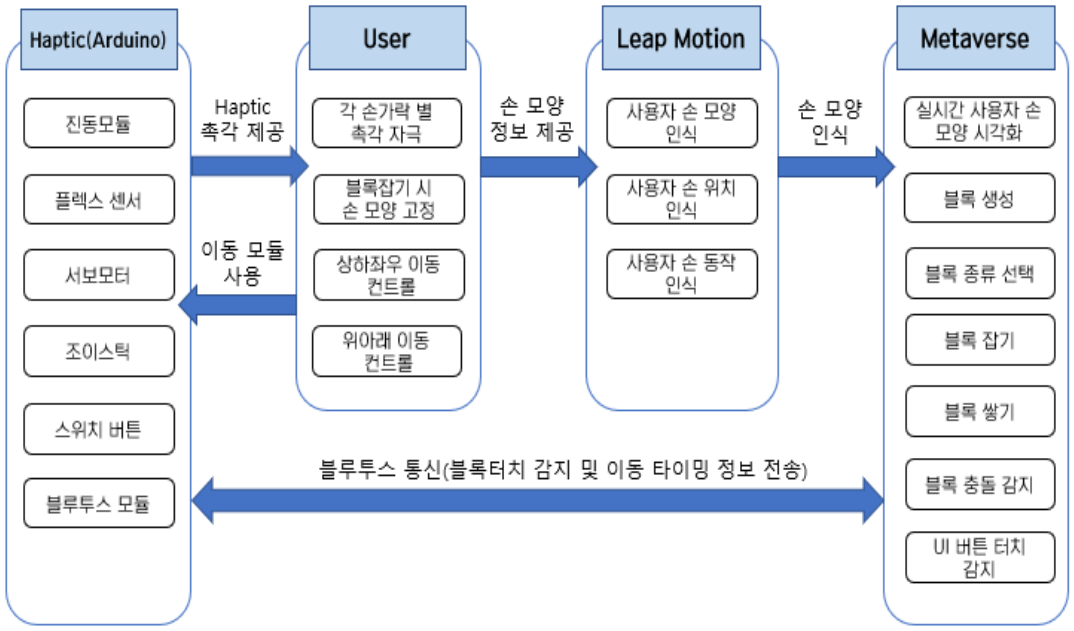
Block Craft 3D : Building Game은 Fun Games For Free에서 제공하는 3D 블록 게임이다. 블록 게임에 건축을 한다는 점에서 마인크래프트와 비슷하지만 그래픽이 다르고 시스템에 차이가 있다. 채굴 시스템이 없으며 블록을 원하는대로 꺼낼 수 있지만 날 수는 없다. 이 외에 멀티플레이를 지원하며 선물 뽑기, 다른 사람의 마을을 방문하는 등의 기능이 있다. 그리고 개, 고양이, 코끼리 등 애완동물을 입양하여 키우거나 세계를 탐험할 수 있다.

| 요구사항 정의서

구분	기능	설명
S/W	사용자 손 모양 시각화	사용자의 손을 Leap Motion을 이용하여 인식한 후 사용자의 손 모양을 메타버스 상에 시각화한다.
	블록 터치 감지	사용자가 손을 이용해 블록을 터치하는 것을 감지한다.
	UI 보이기/숨기기	사용자가 손가락을 모두 펼친 상태인 왼손을 뒤집으면 방향키 UI, 오른손을 뒤집으면 블록 색상 팔레트가 사용자 눈에 보이게 된다. 다시 원래대로 손을 뒤집으면 UI요소가 숨겨진다.
	UI 터치 감지	사용자가 게임시작 버튼, 게임방법 버튼, 방향키 버튼 등의 UI 버튼을 검지손가락으로 터치하는 것을 감지한다.
	블록 생성	하늘에서 블록이 떨어지며 사용자의 시야 안 랜덤한 위치로 생성된다.
	블록 쌓기	블록이 생성되면 사용자가 손을 이용하여 블록을 집어 올려서 쌓을 수 있다.
	블록 색상 선택	블록 색깔 선택 팔레트로 원하는 블록의 색상을 터치하면 블록 생성이 지정된다.

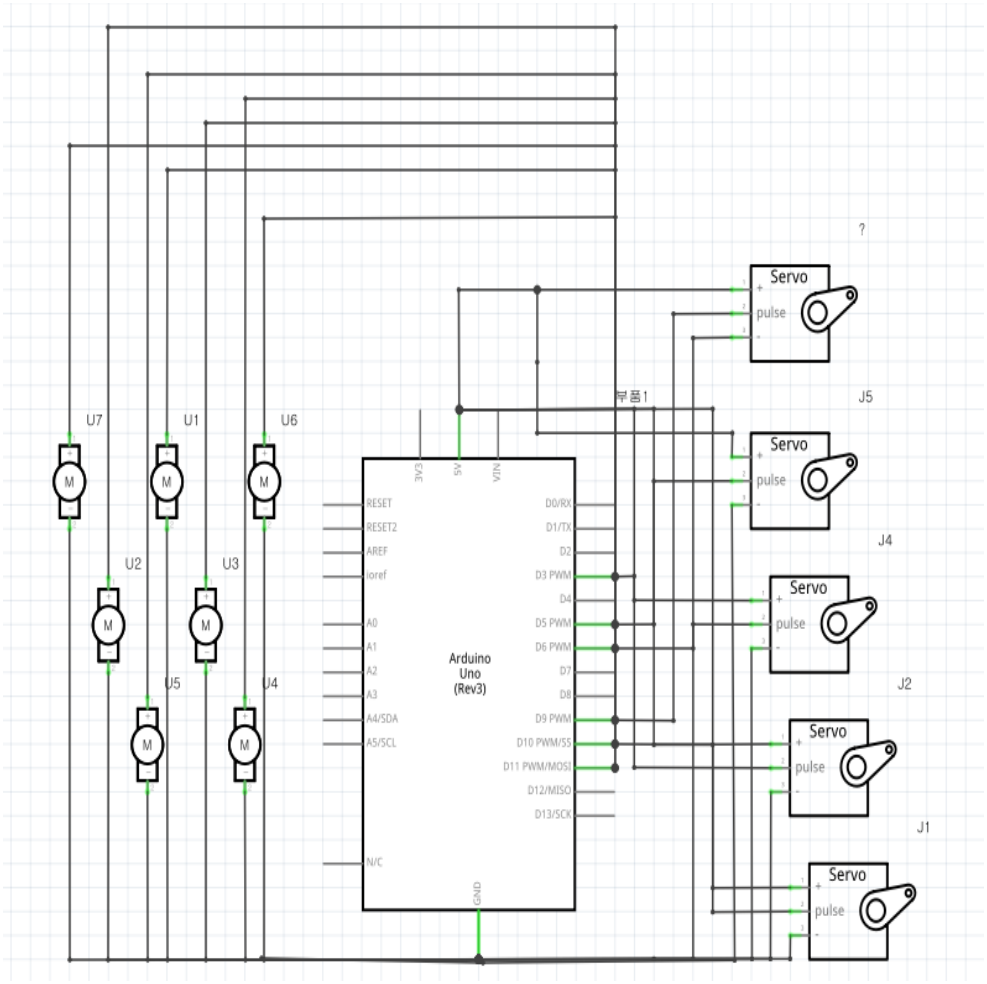
구분	기능	설명
H/W	사용자의 손 모양 인식	사용자의 손 모양(어떤 손가락을 접고,펼쳤는지 등)을 립모션을 활용하여 인식한다.
	손 위치 인식	사용자의 손의 위치를 립모션을 활용하여 인식한다.
	손목/손바닥 촉감 자극	메타버스 상에서 사용자가 블록을 잡거나 놓을 때, 블록을 없앨 때 진동센서를 이용하여 각 손가락별로 진동을 준다.
	블록 잡기 촉감	메타버스 상에서 사용자가 블록을 잡으면 플렉서 센서와 서보모터를 이용해 손 모양을 고정시켜 실제로 블록을 잡은 듯한 느낌을 준다.
	손가락 촉감 자극	메타버스 상에서 사용자가 UI를 터치하였을 때와 블록과 관련 행동을 할 때 진동센서를 이용하여 손가락에 진동을 준다.
	화면 이동	스위치 버튼과 조이스틱을 이용하여 메타버스 화면을 자유롭게 컨트롤하며 이동할 수 있다.

| 서비스 구성도 - 서비스 시나리오



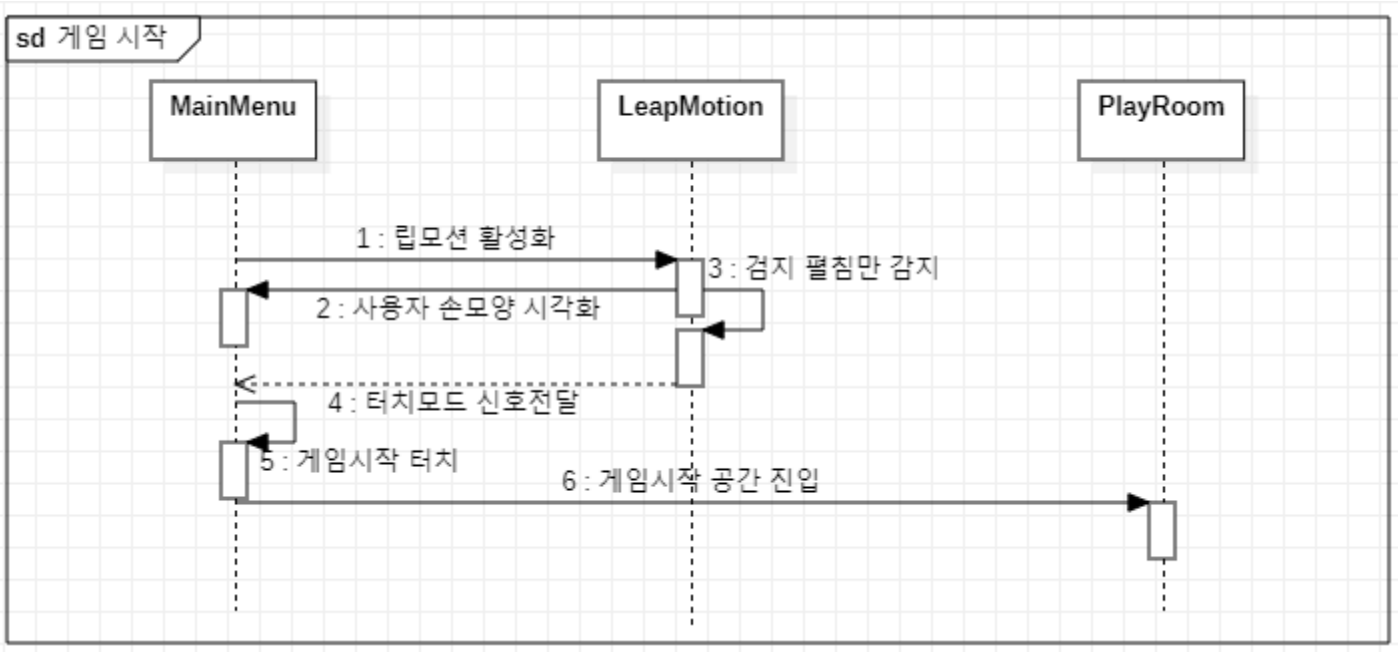
본 시스템은 크게 Arduino(Haptic), User, Leap Motion, Metaverse로 구분된다. 사용자가 손 모양 정보를 제공하면 Leap Motion은 사용자의 손모양을 인식한다. 이때 손의 모양뿐만 아니라 손의 위치, 동작도 함께 인식한다. Leap Motion을 통해 인식된 손모양 정보를 Metaverse 상에서 시각화한다. Metaverse는 블록생성, 블록 종류선택, 블록 잡기 및 쌓기, 블록 충돌 및 UI버튼 터치 감지 등의 다양한 기능이 존재한다. 해당 기능들을 통해 사용자는 시각적으로 "메타 블록"을 즐길 수 있다. 사용자가 Metaverse상에서 블록 혹은 UI버튼을 터치한 경우 블루투스 통신을 통해 Haptic device(Arduino)에 타이밍 정보를 전송한다. Haptic은 타이밍에 맞게 진동모듈을 통해 사용자에게 각 손가락 별 진동 촉각을 제공한다. 블록을 잡은 경우 서보모터를 통해 사용자 손 모양을 고정시킨다. 진동모듈, 플렉스센서, 서보모터를 종합적으로 이용하여 사용자에게 상황에 따른 손 촉감(Haptic 기능)을 자극시키고 실제로 블록을 잡은 듯한 감각을 제공한다. 사용자가 이동모듈(조이스틱 및 스위치 버튼) 사용 시 Metaverse에 이동 타이밍 정보를 전송한다. Metaverse는 타이밍에 맞게 화면을 이동시킨다.

| 하드웨어/센서 구성도

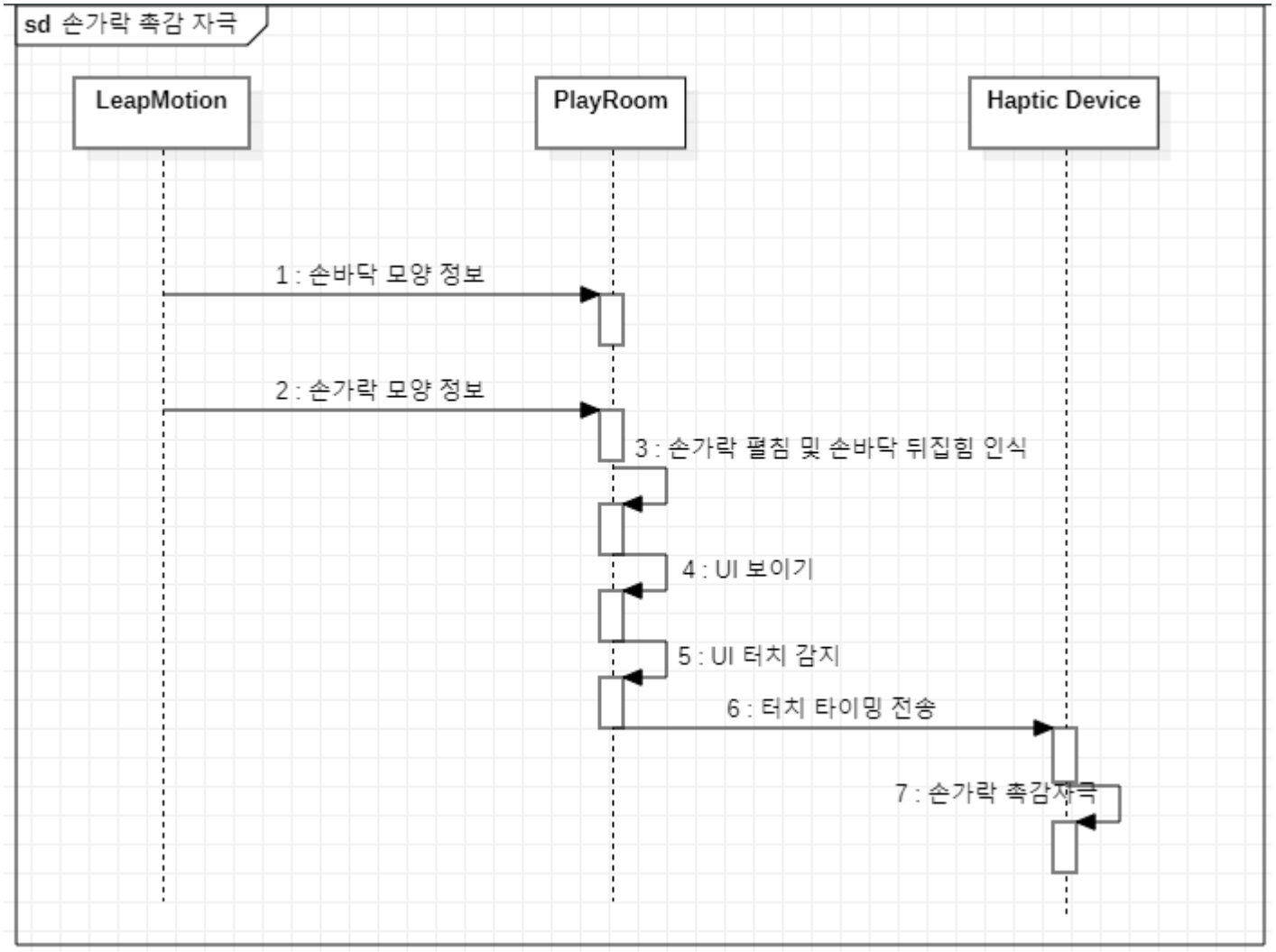


센서 종류	연결 핀	설명
서보모터1	GND	아두이노의 GND에 연결
	VCC	아두이노의 5V에 연결
	IN	아두이노 9번 핀에 연결
서보모터2	GND	아두이노의 GND에 연결
	VCC	아두이노의 5V에 연결
	IN	아두이노 10번핀에 연결
서보모터3	VCC	아두이노의 5V에 연결
	GND	아두이노의 GND에 연결
	IN1	아두이노의 6번 핀에 연결
서보모터4	VCC	아두이노 5V에 연결
	GND	아두이노 GND에 연결
	IN	아두이노 3번 핀에 연결
서보모터5	GND	아두이노의 GND에 연결
	VCC	아두이노의 5V에 연결
	IN	아두이노의 5번 핀에 연결
진동모듈1	GND	아두이노의 GND에 연결
	IN	아두이노 9번 핀에 연결
진동모듈2	GND	아두이노의 GND에 연결
	IN	아두이노 10번 핀에 연결
진동모듈3	GND	아두이노의 GND에 연결
	IN	아두이노 6번 핀에 연결
진동모듈4	GND	아두이노의 GND에 연결
	IN	아두이노의 3번핀에 연결
진동모듈5	GND	아두이노의 GND에 연결
	IN	아두이노의 5번핀에 연결
진동모듈6	GND	아두이노의 GND에 연결
	IN	아두이노의 11번핀에 연결

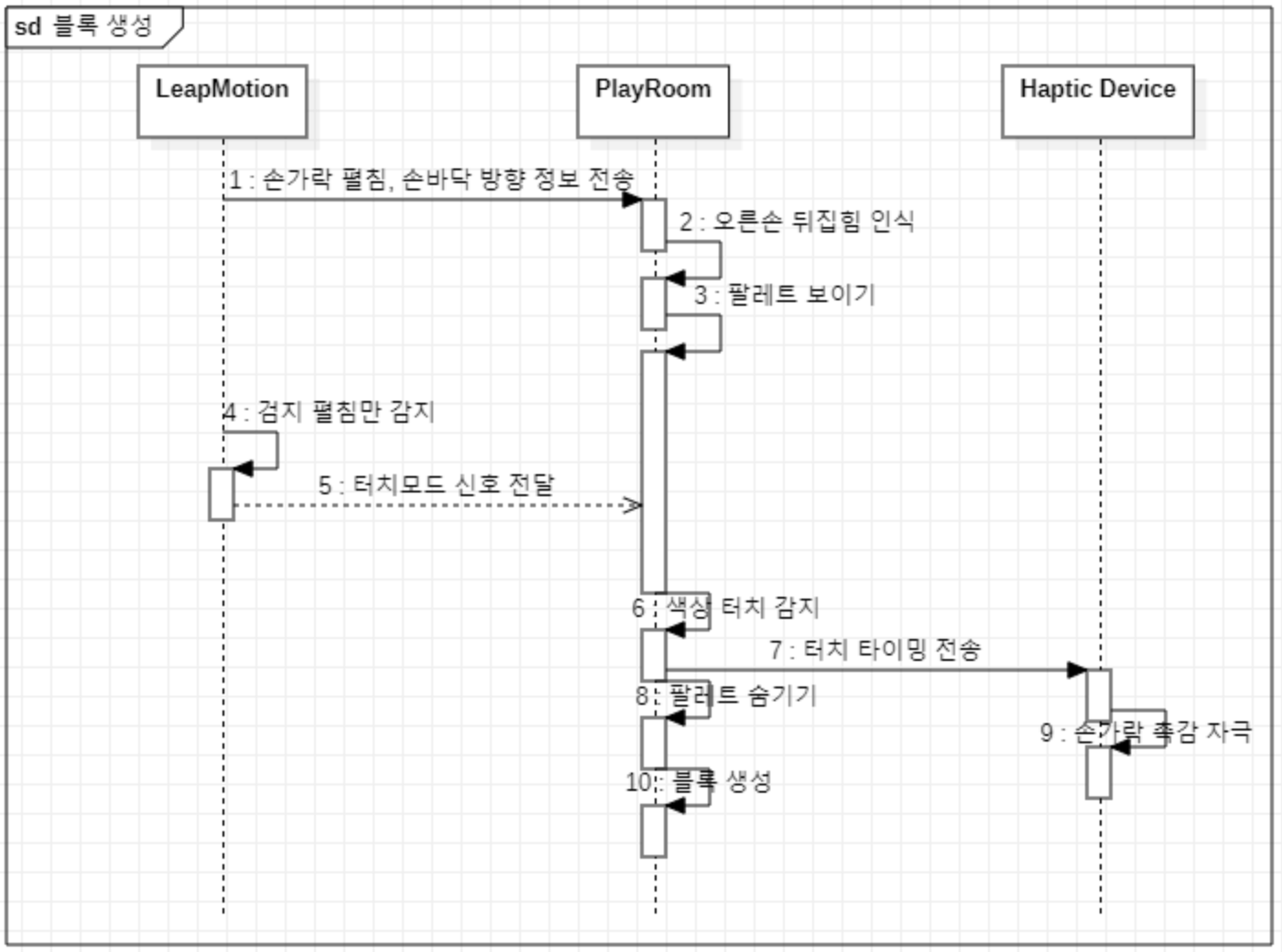
| 기능 처리도(기능 흐름도)


sd 게임 시작
MainMenu
LeapMotion
PlayRoom
1 : 립모션 활성화
3 : 검지 펼침만 감지
2 : 사용자 손모양 시각화
4 : 터치모드 신호전달
5 : 게임시작 터치
6 : 게임시작 공간 진입

| 기능 처리도(기능 흐름도)


sd 손가락 촉감 자극
LeapMotion
PlayRoom
Haptic Device
1 : 손바닥 모양 정보
2 : 손가락 모양 정보
3 : 손가락 펼침 및 손바닥 뒤집힘 인식
4 : UI 보이기
5 : UI 터치 감지
6 : 터치 타이밍 전송
7 : 손가락 촉감자극

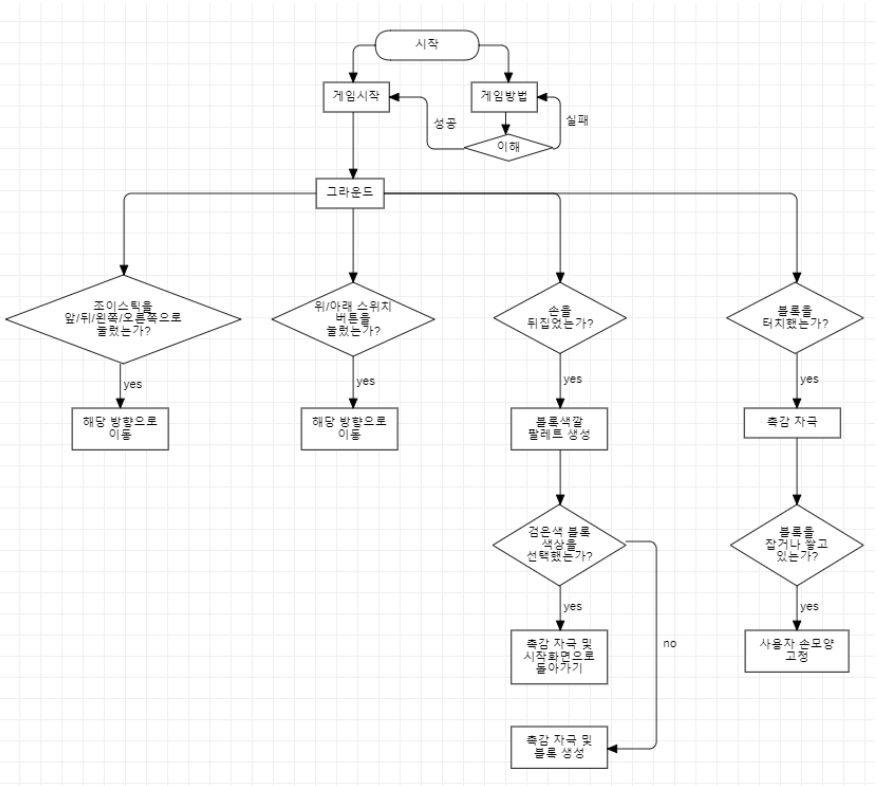
| 기능 처리도(기능 흐름도)


sd 블록 생성
LeapMotion
PlayRoom
Haptic Device
1 : 손가락 펼침, 손바닥 방향 정보 전송
2 : 오른손 뒤집힘 인식
3 : 팔레트 보이기
4 : 검지 펼침만 감지
5 : 터치모드 신호 전달
6 : 색상 터치 감지
7 : 터치 타이밍 전송
8 : 팔레트 숨기기
9 : 손가락 촉감 자극
10 : 블록 생성

| 메뉴 구성도


첫 화면
실시간 사용자 손 모양 시각화
게임 설명
게임 시작
블록 생성
사용자 촉감자극
블록 종류 선택
사용자 촉감자극
블록 잡기 및 쌓기
사용자 손 모양 고정
블록 충돌
사용자 촉감자극
UI 버튼 터치
사용자 촉감자극

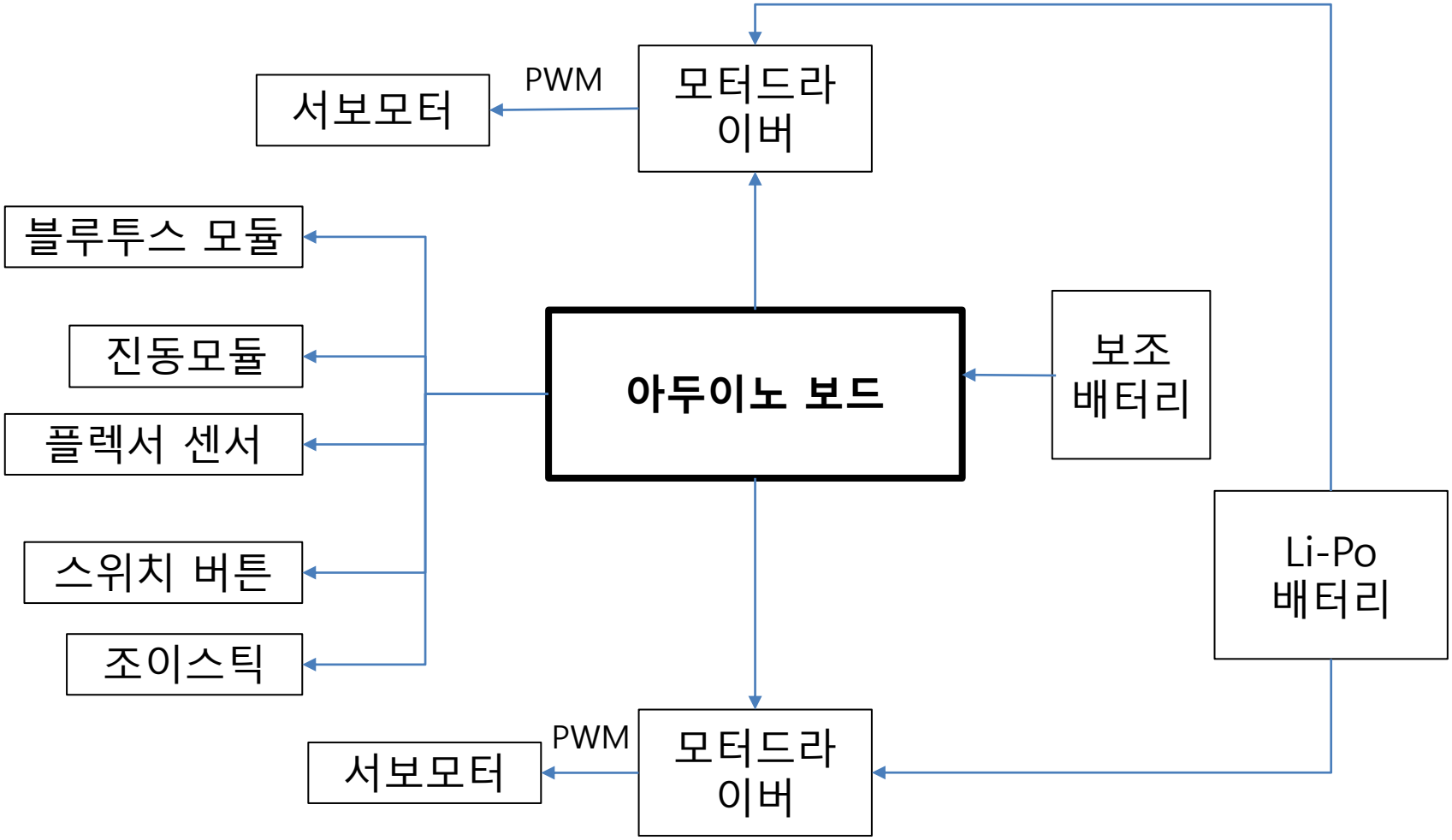
| 알고리즘 명세서



알고리즘 시나리오:

프로그램을 시작하면 게임 시작과 게임 방법으로 나뉜다. 사용자가 게임 방법을 이해하면 게임 시작 모드로 전환된다. 게임을 시작하면 그라운드가 시작된다. 사용자는 그라운드에서 화면이동 및 블록 관련 기능을 사용할 수 있다. 조이스틱을 이용해 앞/뒤/왼쪽/오른쪽 방향으로 화면 이동이 가능하며 스위치 버튼으로는 위/아래 이동이 가능하다. 사용자가 손을 뒤집으면 블록색깔을 선택할 수 있는 팔레트가 나타난다. 만약 팔레트 가운데에 위치한 검은색 블록을 선택하면 햅틱장갑의 진동모듈을 통해 사용자에게 촉감 자극이 가고 시작화면으로 돌아간다. 검은색을 제외한 나머지 색상을 선택하면 촉감 자극 및 블록이 생성이 진행된다. 사용자가 블록을 터치하면 촉감 자극이 가며 이 상태에서 블록을 잡거나 쌓는다면 햅틱장갑의 서보모터를 통해 사용자의 손모양이 고정된다.

| 하드웨어 설계도


서보모터
PWM
모터드라이버
블루투스 모듈
진동모듈
플렉서 센서
스위치 버튼
조이스틱
아두이노 보드
보조 배터리
Li-Po 배터리
서보모터
PWM
모터드라이버

| 프로그램 - 목록

기능 분류	기능번호	기능 명
MED	MED-01	게임 방법 설명
GAM	GAM-02-01	사용자 손 모양 시각화
	GAM-02-02	블록 터치 감지
	GAM-02-03	UI 보이기/숨기기
	GAM-02-04	UI 터치 감지
	GAM-02-05	블록 생성
	GAM-02-06	블록 쌓기
	GAM-02-07	블록 색상 선택
	GAM-03-01	사용자의 손 모양 인식
	GAM-03-02	손 위치 인식
	GAM-03-03	블록 터치 시 손가락 촉감 자극
	GAM-03-04	블록 잡기 시 손모양 고정
	GAM-03-05	화면 위아래 컨트롤
	GAM-03-06	화면 앞뒤좌우 컨트롤

햅틱디바이스 아두이노 핵심 소스코드

```ino
#include <Servo.h>
#include <SoftwareSerial.h>
//엄지-검지-중지-약지-새끼
#define Vibe1 8
#define Vibe2 5
#define Vibe3 4
#define Vibe4 7
#define Vibe5 6
Servo servo[3];
const byte servoPin[3]={9,10,11};
int flexpin1=A0;
//int flexpin2=A1;
//int flexpin3=A2;

const int rxPin = 2;
const int txPin = 3;

SoftwareSerial BT(rxPin, txPin);

void setup() {

  Serial.begin(9600);
  BT.begin(9600);
  servo[0].attach(servoPin[0]);
  servo[1].attach(servoPin[1]);
  servo[2].attach(servoPin[2]);
  pinMode(Vibe1, OUTPUT);
  pinMode(Vibe2, OUTPUT);
  pinMode(Vibe3, OUTPUT);
  pinMode(Vibe4, OUTPUT);
  pinMode(Vibe5, OUTPUT);
}

void loop() {

  int flexVal;
  flexVal=analogRead(flexpin1);

  if(flexVal>=350){
    servo[0].write(90);
    servo[1].write(90);
    servo[2].write(90);
    BT.println("G");
```

| 핵심소스코드(1)

```
    Serial.print("sensor: "); Serial.print(flexVal);
    Serial.println("    grabbing the block");
  }

  if(flexVal<=250){
    servo[0].write(0);
    servo[1].write(0);
    servo[2].write(0);
    delay(20);
    Serial.print("sensor: "); Serial.print(flexVal);
    Serial.println("    normal state");
  }

    if (BT.available())
   {

    char cmd = BT.read();
    //Serial.print(cmd);

    if(cmd=='1'){
      Serial.println("tumb finger viberate feedback");
      analogWrite(Vibe1, 255);
      delay(1500);
      analogWrite(Vibe1, 0);
      delay(500);
    }
    if (cmd == '2') {
      Serial.println("index finger viberate feedback");
      analogWrite(Vibe2, 255);
      delay(1500);
      analogWrite(Vibe2, 0);
      delay(500);
    }
    if (cmd == '3') {
      Serial.println("middle finger viberate feedback");
      analogWrite(Vibe3, 255);
      delay(1500);
      analogWrite(Vibe3, 0);
      delay(500);
    }
    if (cmd == '4') {
      Serial.println("ring finger viberate feedback");
      analogWrite(Vibe4, 255);
      delay(1500);
      analogWrite(Vibe4, 0);
      delay(500);
    }
    if (cmd == '5') {
      Serial.println("pinky finger viberate feedback");
      analogWrite(Vibe5, 255);
      delay(1500);
      analogWrite(Vibe5, 0);
      delay(500);
    }

}
```

| 핵심소스코드(2)

Class: PlayerBluetoothManager
플레이어 이동 컨트롤러HW와 자동 블루투스 연결
및 메시지 수신 소스코드

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;
using ArduinoBluetoothAPI;

Unity 스크립트 | 0 참조
public class playBluetoothManager : MonoBehaviour
{

    // Use this for initialization
    static BluetoothHelper bluetoothHelper;
    string deviceName="Haptic";

    string received_message;
    // static Boolean reconnect = false;

    [SerializeField]
    public GameObject player;
    private float currentSpeed=0.1f;


    // Start is called before the first frame update
    Unity 메시지 | 0 참조
    void Start()
    {

        try
        {
            BluetoothHelper.BLE = false;
            bluetoothHelper = BluetoothHelper.GetInstance(deviceName);
            bluetoothHelper.OnConnected += OnConnected;
            bluetoothHelper.OnConnectionFailed += OnConnectionFailed;
            bluetoothHelper.OnDataReceived += OnMessageReceived; //read the data
```

```
if (received_message.Contains("U"))
{
    player.transform.Translate(Vector3.up * currentSpeed / 30);
}

if (received_message.Contains("D"))
{
    player.transform.Translate(Vector3.down * currentSpeed / 30);
}


if (received_message.Contains("a"))
    player.transform.Translate(Vector3.left * currentSpeed / 50);

if (received_message.Contains("d"))
    player.transform.Translate(Vector3.right * currentSpeed / 50);

if (received_message.Contains("w"))
    player.transform.Translate(Vector3.forward * currentSpeed / 50);

if (received_message.Contains("s"))
    player.transform.Translate(Vector3.back * currentSpeed / 50);
```

| 핵심소스코드(2)

Class: gloveBluetoothManager
햅틱장갑HW와 자동 블루투스 연결
및 메시지 송신 소스코드

```
Unity 스크립트(자산 참조 1개)|참조 5개
public class gloveBluetoothManager : MonoBehaviour
{

    // Use this for initialization
    static BluetoothHelper bluetoothHelper;
    string deviceName = "PASTA";

    string received_message;
    static string send_message;
    static Boolean reconnect = false;


    참조 1개
    void OnConnected()
    {
        try
        {
            bluetoothHelper.StartListening();
            Debug.Log("glove Bluetooth Connected");

            if (reconnect == true)
            {
                bluetoothHelper.SendData(send_message);
                reconnect = false;
            }
        }
        catch (Exception ex)
        {
            Debug.Log(ex.Message);
        }


    }

    참조 5개
    public static void sendData(string str)
    {
        //Debug.Log("sendfunc call");
        send_message = str;
        if (bluetoothHelper.isConnected())
        {
            bluetoothHelper.SendData(str);
        }
        else
        {
            reconnect = true;
            bluetoothHelper.Connect(); // tries to connect
        }

    }

    참조 1개
    void OnMessageReceived()
    {
        received_message = bluetoothHelper.Read();
        //Debug.Log(received_message);
        if (received_message.Contains("G"))
        {
            Debug.Log("Grapping Block Motion");
        }

    }

    참조 1개
    void OnConnected()
```

한이음 ▶

Class: CollisionDetector
사용자 손에 충돌 감지 지점 설정과 블록과 충돌할 시 피드백 클래스

이브와

핵심소스코드(2)

```
Unity 스크립트(자산 참조 0개) | 참조 4개
public class CollisionDetector : MonoBehaviour
{

    public int fingerNum = 0;
    public string hand = "L";
    private bool check = true;
    public static bool collsionStay = false; //손가락 충돌중인 상태인지 표시 (충돌하면서 주먹을 쥐기 시작한다 -> 블록을 잡음)

    // public static bool[] collisionFinger = { false, false, false, false, false };

    Unity 메시지 | 참조 0개
    void OnCollisionEnter(Collision c)
    {

        // Debug.Log(hand + "손" + fingerNum + "손가락 터치");
        // 각 손가락이 무언가에 충돌할때 불러지는 함수

        if (c.gameObject.CompareTag("block") && check)
        {

            collsionStay = true;
            //주먹을 쥔 상태에서 블록과 충돌중 -> 이미 블록을 잡은 상태
            if (ExtendedFingerInfo.fistGesture == true)
            {
                Debug.Log("holding the block and moving");
            }

            if (hand == "L" && fin
            {
                check = false;
```

```
            }
            else if (c.gameObject.CompareTag("block"))
            {
                collsionStay = true;
            }
            else
                collsionStay = false;

        }

        참조 5개
        IEnumerator WaitForIt()
        {
```

```
        }
        else if (hand == "L" && fingerNum == 3 && check)
        {
            check = false;

            Debug.Log("middle finger touch");

            gloveBluetoothManager.sendData(fingerNum.ToString());
            StartCoroutine(WaitForIt());
        }
        else if (hand == "L" && fingerNum == 4 && check)
        {
            check = false;


            Debug.Log("ring finger touch");
            gloveBluetoothManager.sendData(fingerNum.ToString());
            StartCoroutine(WaitForIt());
        }
        else if (hand == "L" && fingerNum == 5 && check)
        {
            check = false;


            Debug.Log("pinky finger touch");
            gloveBluetoothManager.sendData(fingerNum.ToString());
            StartCoroutine(WaitForIt());
        }
```

| 핵심소스코드(2)

Class: TouchDetector
사용자 제스처에 따른 색상 선택 팔레트 터치 시 피드백 및 블록생성 클래스

```
Unity 스크립트(자산 참조 1개)|참조 0개
public class TouchDetector : MonoBehaviour
{

    //private Vector3 installPos;
    public GameObject block;
    public GameObject player;
    private bool check = true;

    Unity 메시지|참조 0개
    private void OnCollisionEnter(Collision c)
    {

        if (c.gameObject.CompareTag("palette")&&
            check&& ExtendedFingerInfo.clickGuesture)
            //검지만 펼쳐진 클릭제스쳐이고 팔레트에 터치되었고
            //1.0f초 기다림이 끝난 상태일 때
        {

            check = false;
            Debug.Log("팔레트 collision enter");
            gloveBluetoothManager.sendData("2");
            StartCoroutine(WaitForIt());
            makeBlockInSky(c.gameObject.GetComponent<Renderer>
        }
    }
```

```
참조 1개
IEnumerator WaitForIt()
{
    yield return new WaitForSeconds(1.0f);
    check = true;
}
```

```
참조 1개
void makeBlockInSky(Material material)
{
        AudioSource sound = GetComponent<AudioSource>();
        sound.Play();

        Vector3 size = block.transform.lossyScale;

        Vector3 installPos = player.gameObject.transform.position;

        Random rand = new Random();

        installPos.y += 2;
        installPos.z += (float)rand.NextDouble() * (4 - 2) + 2;
        installPos.x -= (float)rand.NextDouble() * (2 - 0.2f) + 0.2f;

        Renderer rend = block.GetComponent<Renderer>();
        rend.enabled = true;
        rend.sharedMaterial = material;

        Instantiate(block, installPos, block.transform.rotation);
        string[] name = material.name.Split('(');
        Debug.Log("Make "+name[0]);
```

| 핵심소스코드(2)

Class: ExtendFingerInfo
펼쳐진 손가락 정보를 담은 클래스로
사용자의 손모션을 판단한다.

```
public class ExtendedFingerInfo : MonoBehaviour
{
    public static bool fistGesture = false;  //주먹쥔 모션인지 아닌지
    public static bool clickGuesture = false;   //클릭하는 검지를 펼친 모션인지 아닌지

    참조 0개
    public void fistMotionLeft()
    {
        fistGesture = true;
        if (CollisionDetector.collsionStay)
            Debug.Log("left hand grabbing the block");//Debug.Log("왼손 블록 잡음");
        //Debug.Log("주먹쥠");
    }

    참조 0개
    public void notFistMotionLeft()
    {
        fistGesture = false;
        if (CollisionDetector.collsionStay)  //방금까지 블록과 충돌되었는데 주먹을 펼친다 -> 블록을 놓음
            Debug.Log("left hand put down the block"); //Debug.Log("왼손 블록 놓음");
       // Debug.Log("주먹안쥠");
    }

    참조 0개
    public void fistMotionRight()
    {
        fistGesture = true;
        if (CollisionDetector.collsionStay)
            Debug.Log("right hand grabbing the block"); //Debug.Log("오른손 블록 잡음");
        //Debug.Log("주먹쥠");
    }
    참조 0개
    public void notFistMotionRight()
    {
        fistGesture = false;
        if (CollisionDetector.collsionStay)  //방금까지 블록과 충돌되었는데 주먹을 펼친다 -> 블록을 놓음
            Debug.Log("right hand put down the block");//Debug.Log("오른손 블록 놓음");
        // Debug.Log("주먹안쥠");
    }
```

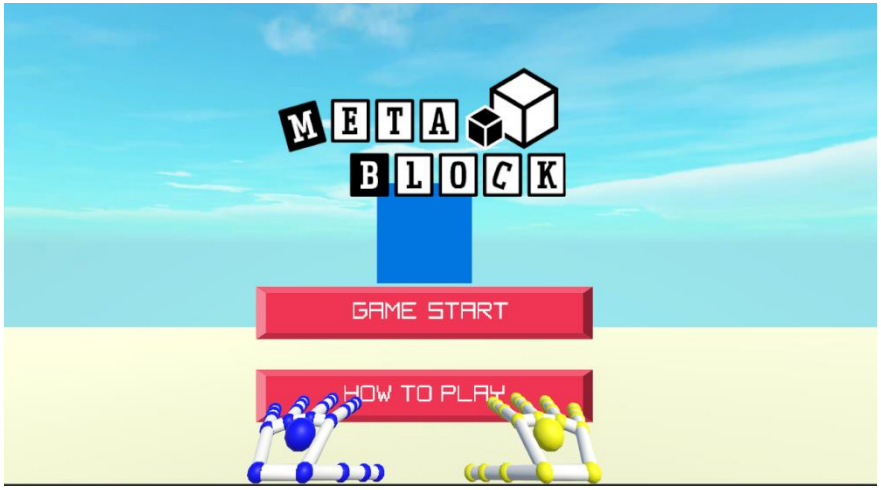
```
참조 0개
public void clickMotion()
{
    clickGuesture = true;
}

참조 0개
public void notClickMotion()
{
    clickGuesture = false;
}
```

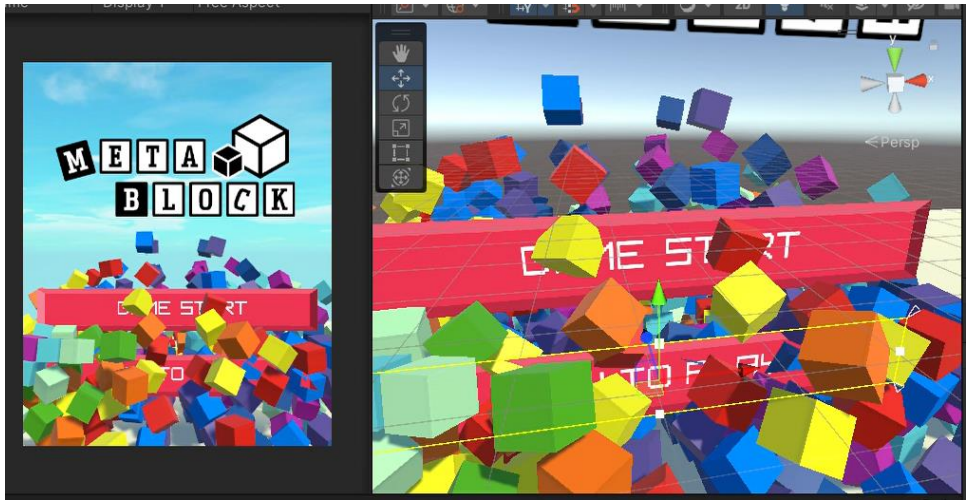
| 참조- 개발 환경 및 설명

구분		항목	적용내역
S/W 개발환경	메타버스 가상환경 개발	Unity3D	메타블록 가상공간 개발 tool로 유니티 에셋스토어를 통한 Leap Motion과의 연동도 적용하였다.
		C#	Unity3D 개발 언어로 Unity3D를 개발하는데 있어 가장 편리함을 준다.
		Visual Studio	C# 개발을 할 수 있는 tool 이다.
H/W 구성장비	햅틱 디바이스	진동 모듈	손목, 손바닥, 손가락에 진동을 주는 모듈이다.
		플렉서 센서	손가락이 기울어진 정도를 수치로 계산해주는 모듈이다.
		서보 모터	물체 모양에 따른 손 모양을 고정하기 위한 모듈이다.
		스위치 버튼	화면을 위아래로 컨트롤하기 위한 버튼이다.
		조이스틱	화면을 앞뒤좌우로 컨트롤하기 위한 모듈이다.
		Leap Motion	사용자의 손 모양을 인식하고 손 위치를 인식하기 위한 것이다.

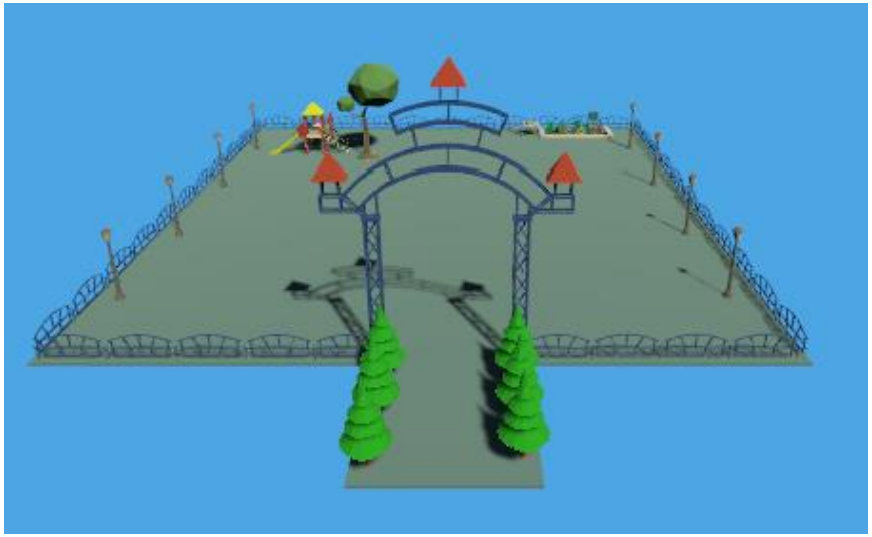
| 참조-S/W 기능 실사 사진

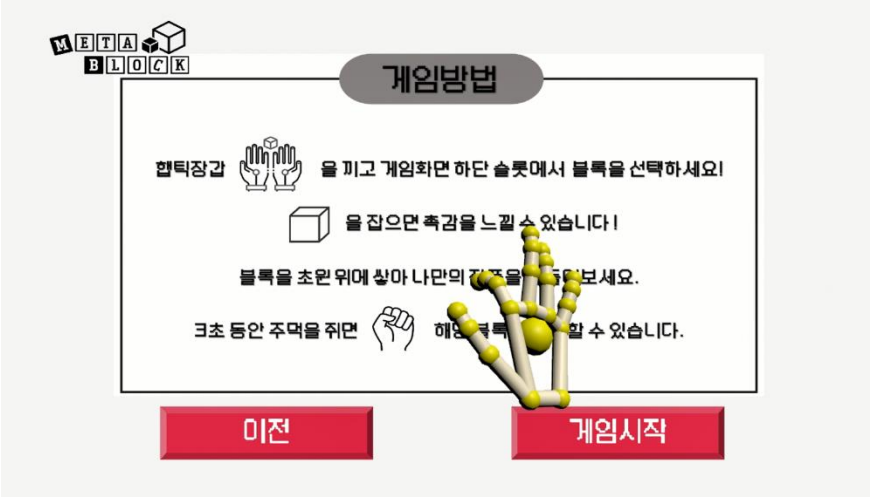
시작-인트로 화면

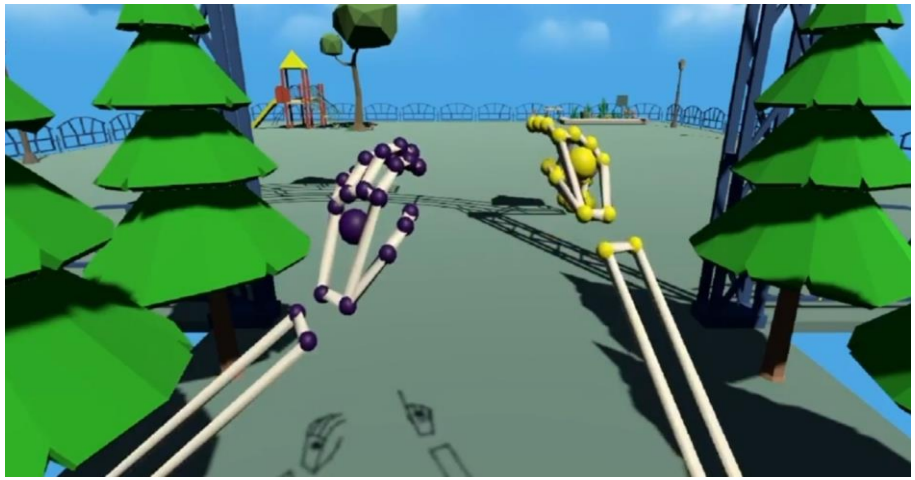




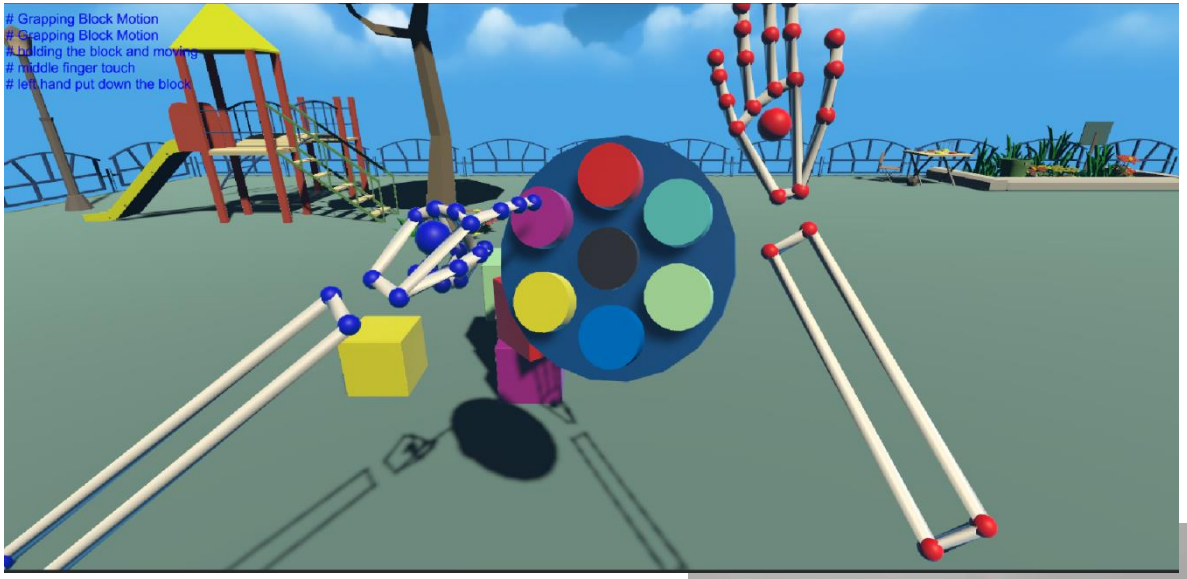
Playroom 입장, 게임방법, 위치이동



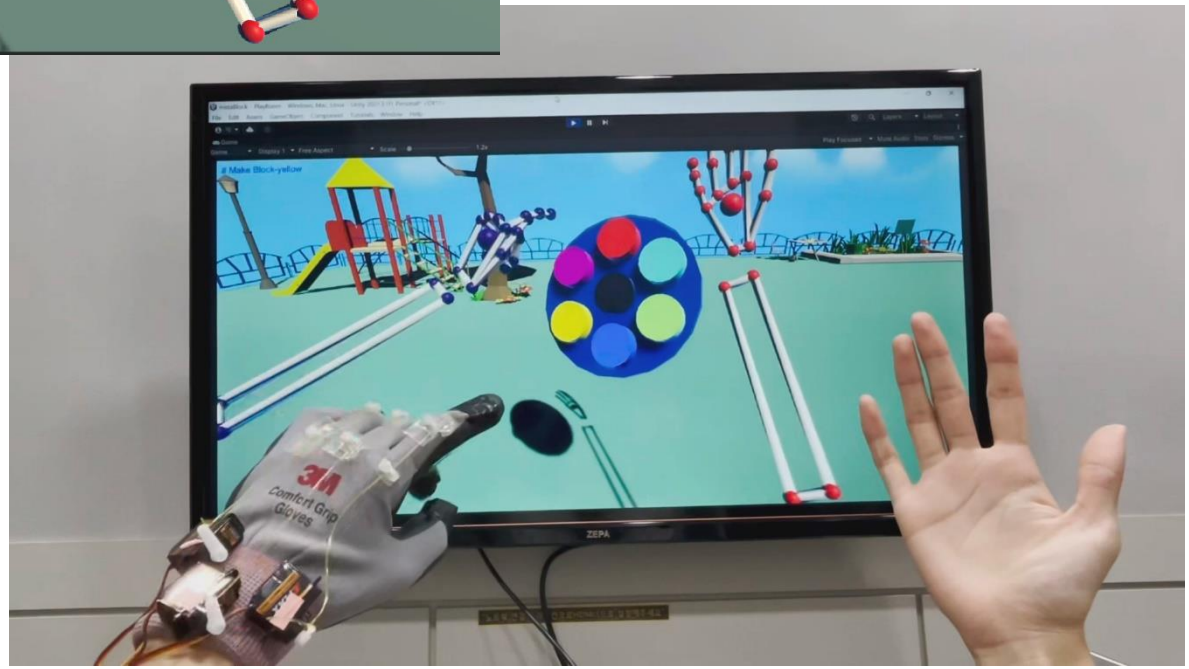




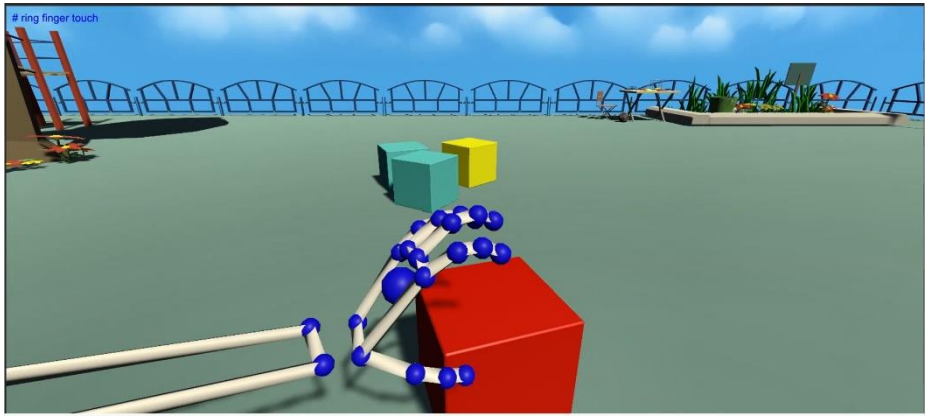
| 참조-S/W 기능 실사 사진

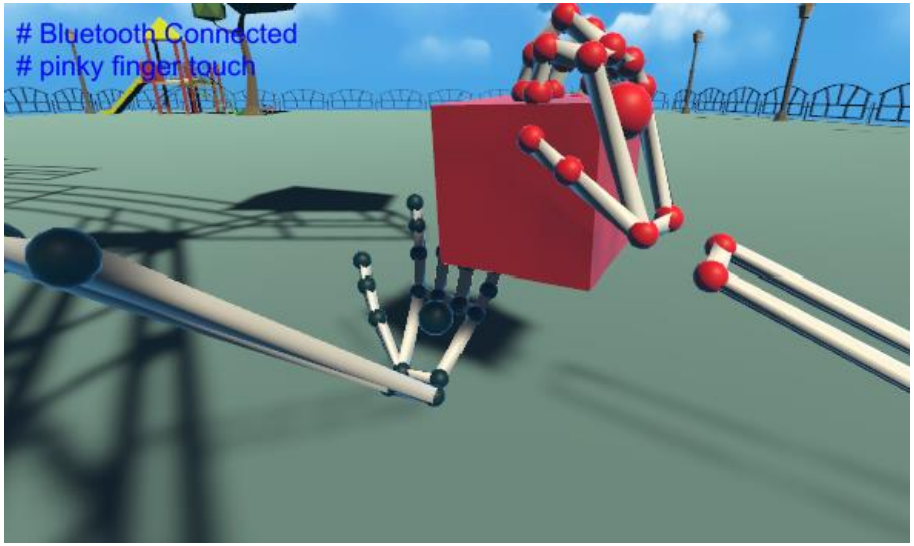


팔레트 펼치기, 블록생성

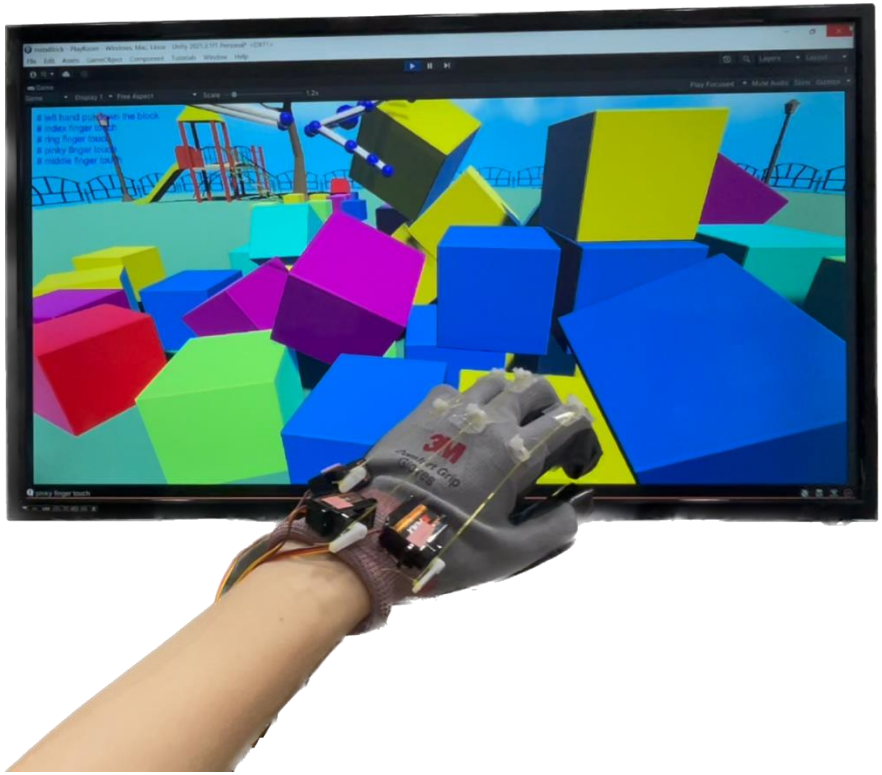


| 참조-S/W 기능 실사 사진

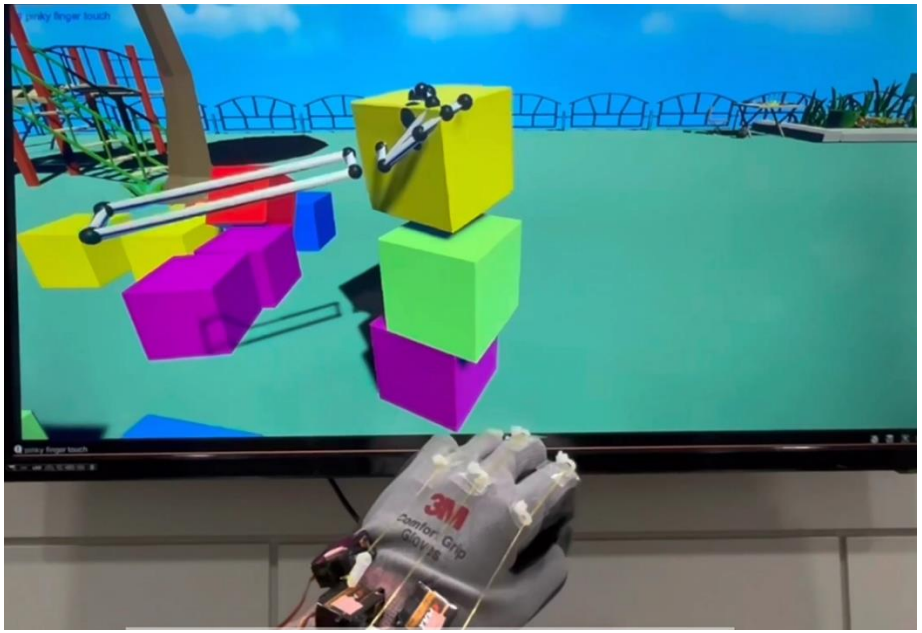


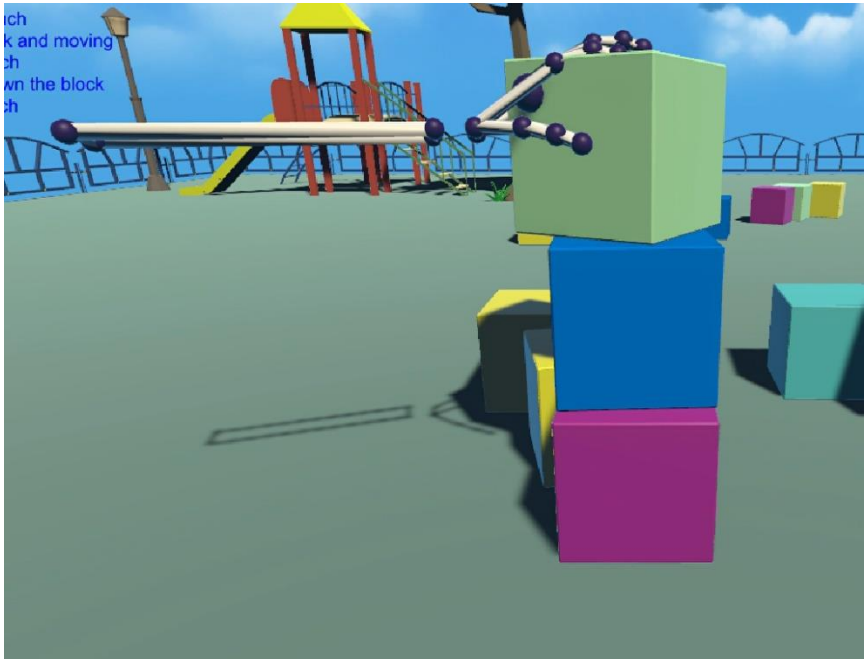


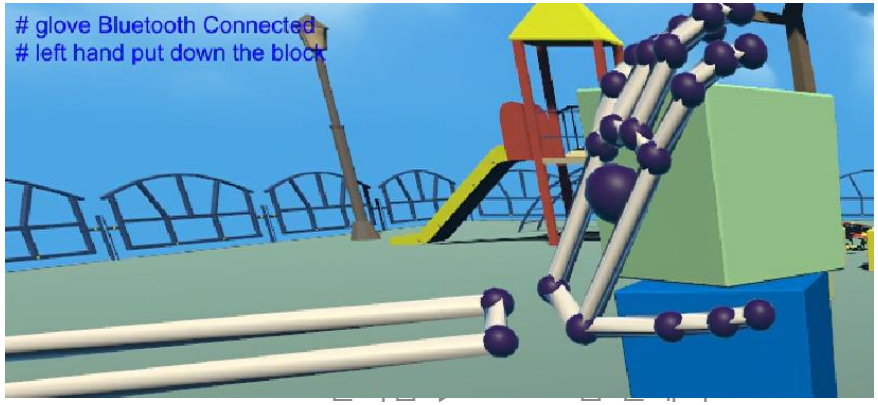
블록 만지기(터치)
블록 잡기, 들어올리기



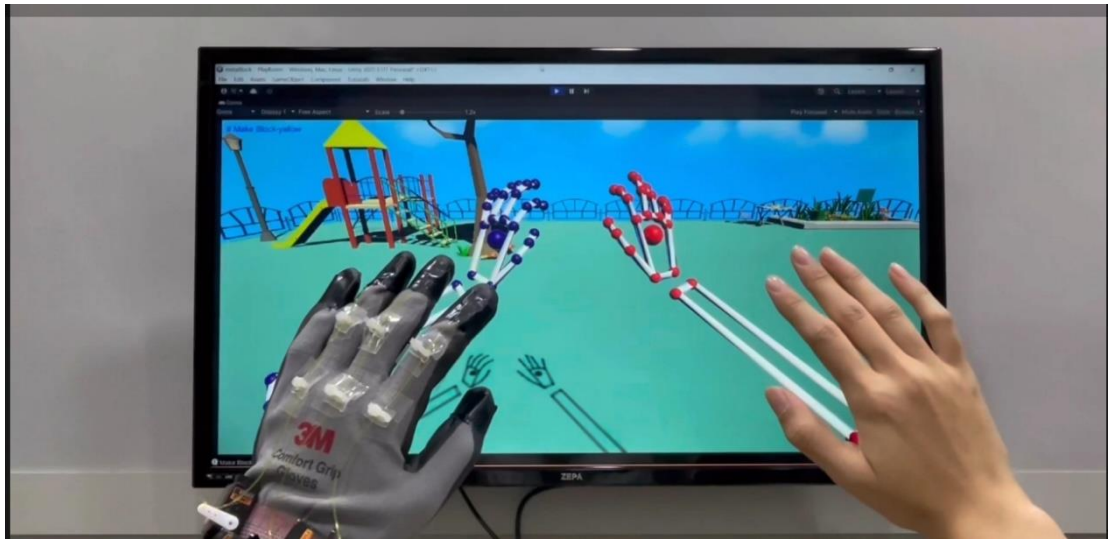
블록 쌓기, 블록 놓기




and moving
the block

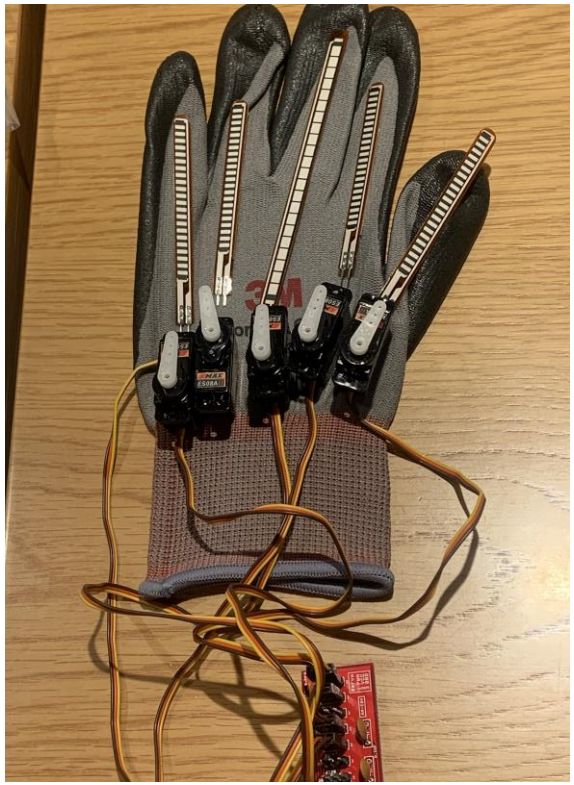

glove Bluetooth Connected
left hand put down the block

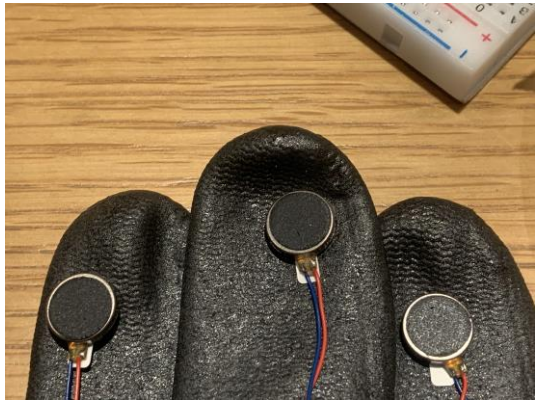
실제 손 시각화 모습

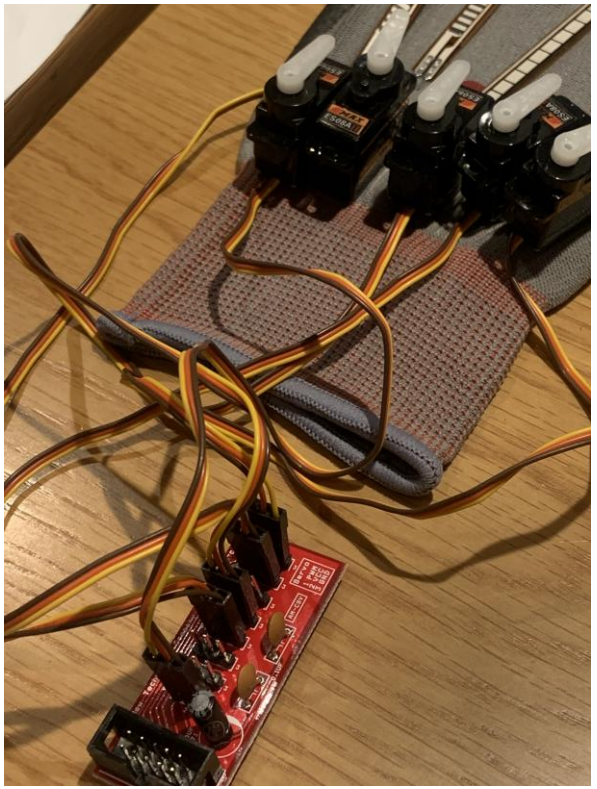


| 참조- H/W 기능 실사사진

<하드웨어 프로토타입>

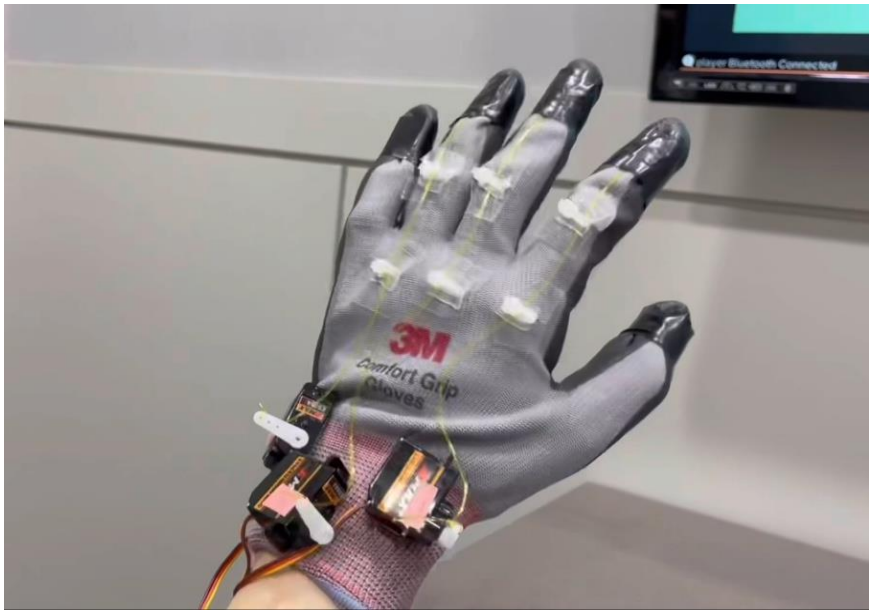


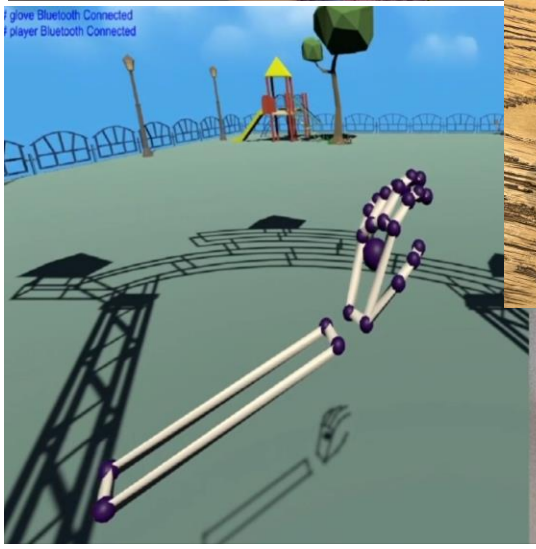


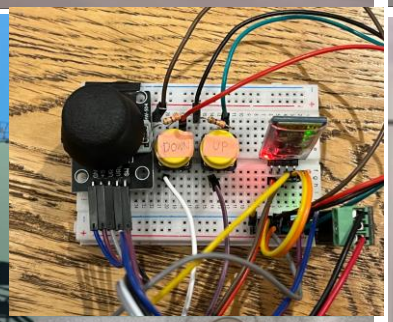


| 참조- H/W 기능 실사사진

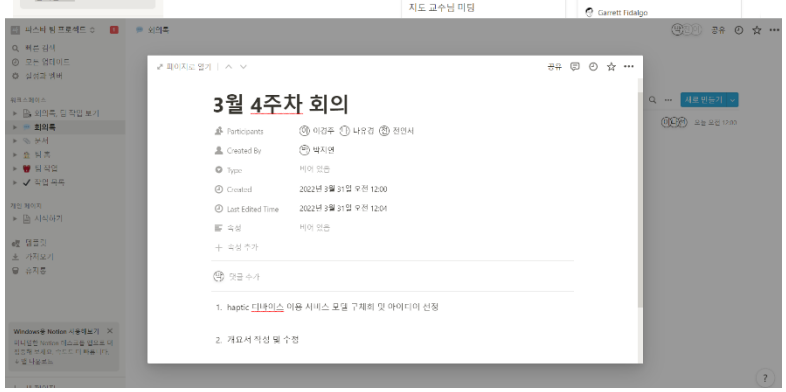
햅틱 장갑, 플레이어 컨트롤러

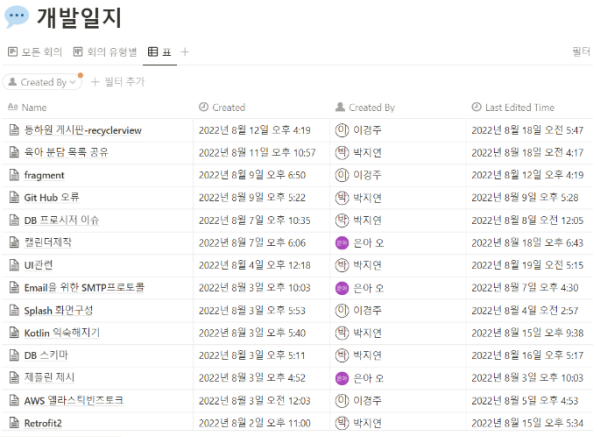






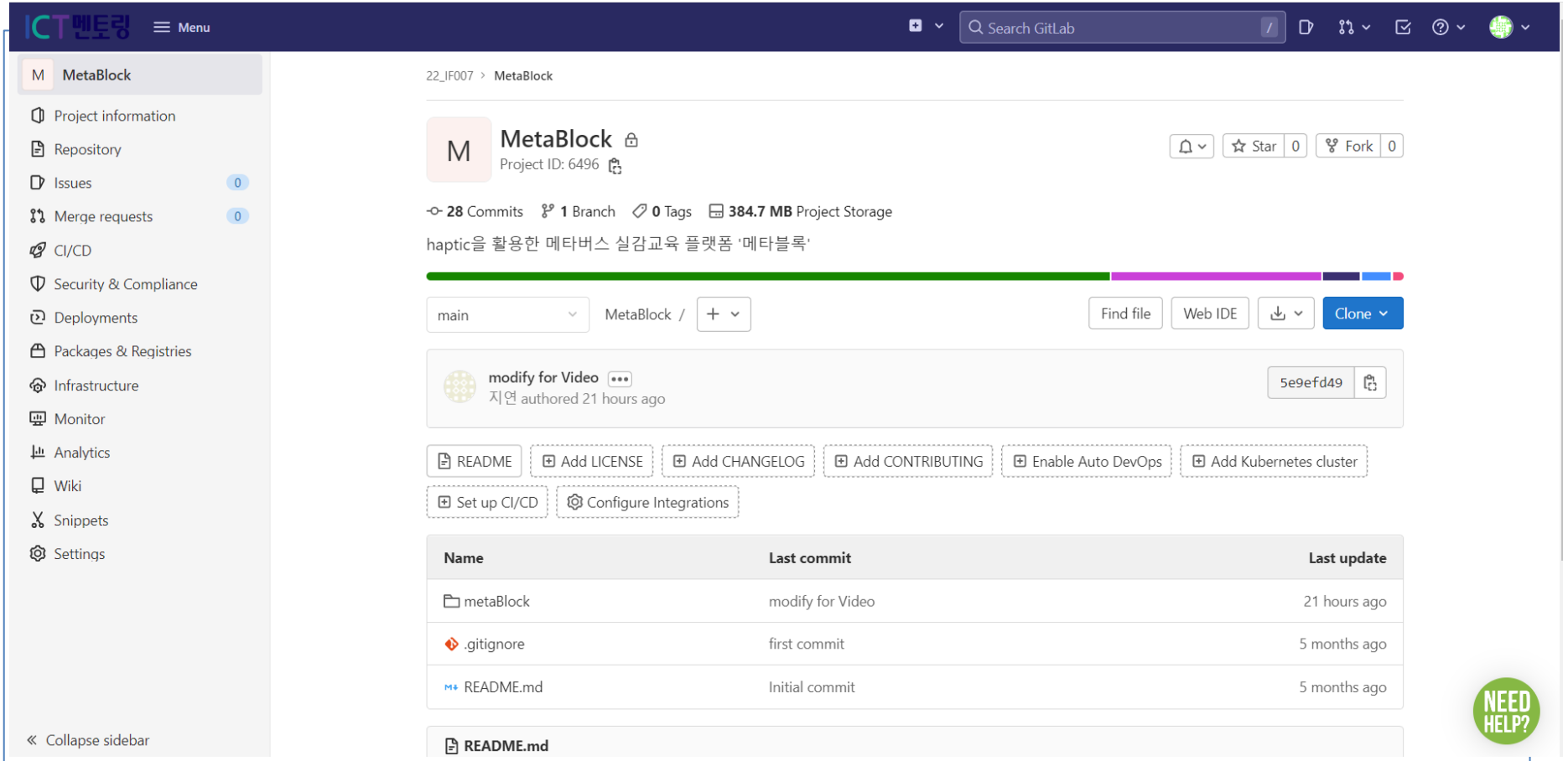
| 참조-프로젝트 관리





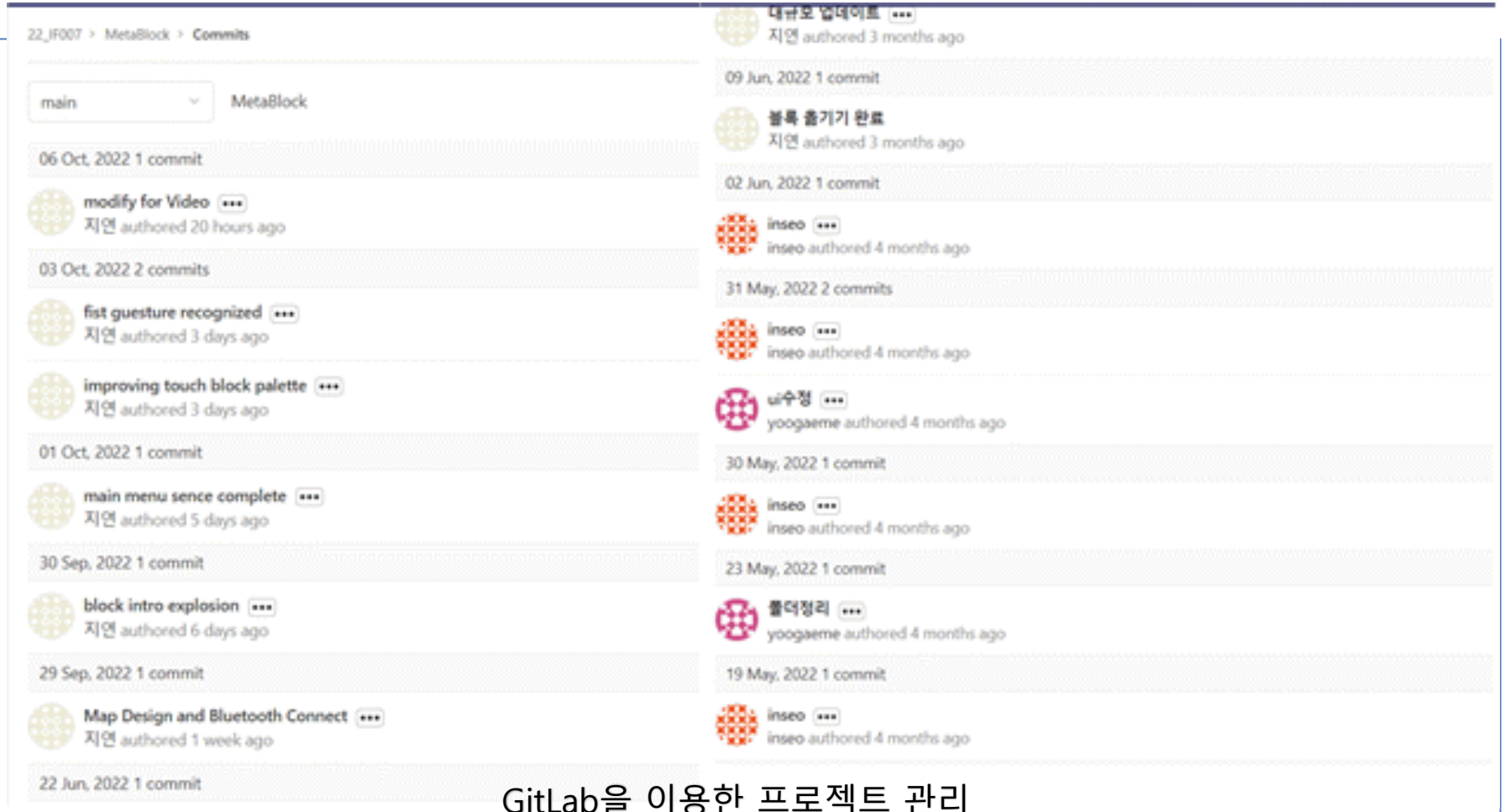
Notion을 이용한 프로젝트 일정 관리

| 참조-프로젝트 관리



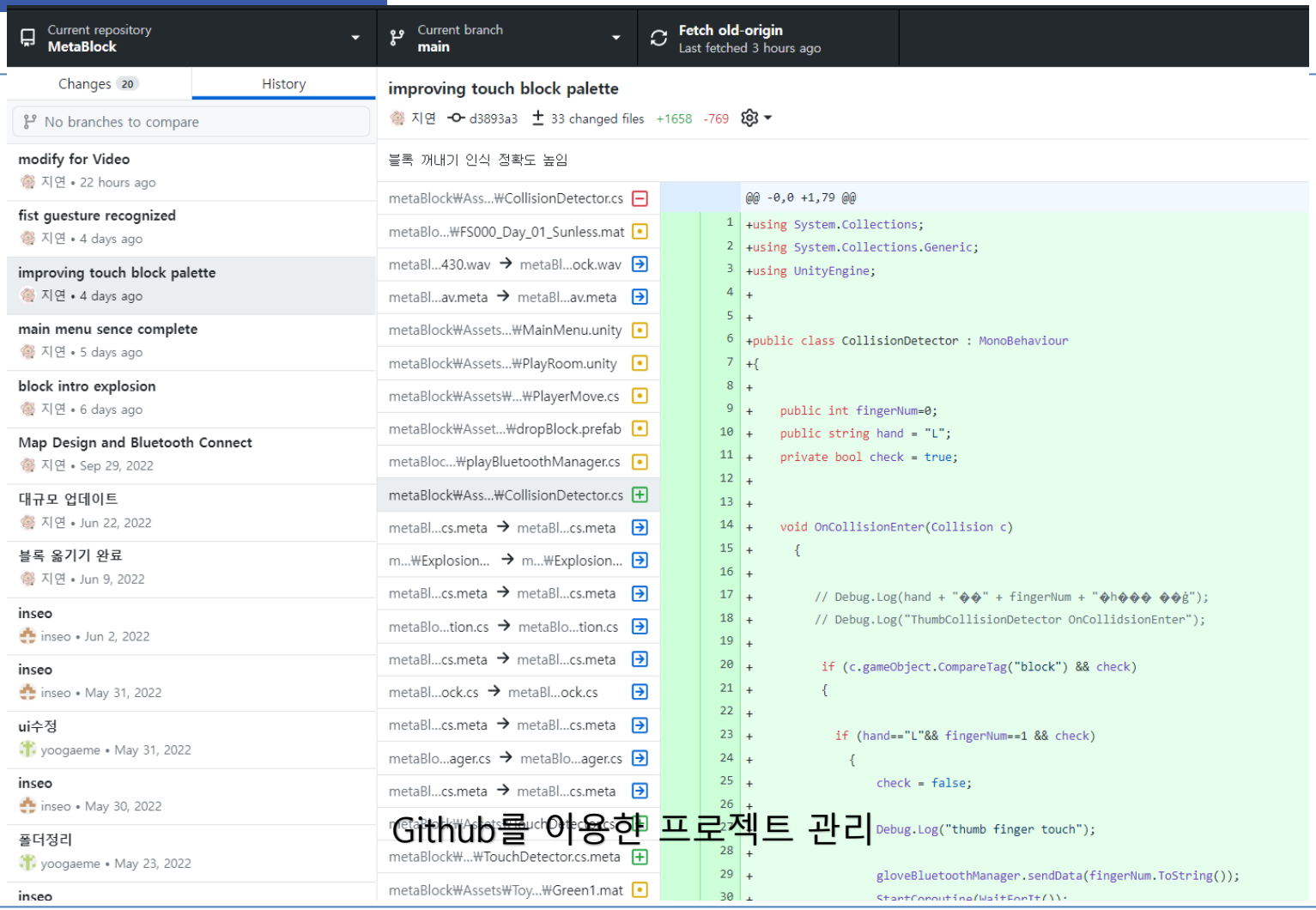
GitLab을 이용한 프로젝트 관리

| 참조-프로젝트 관리



GitLab을 이용한 프로젝트 관리

| 참조-프로젝트 관리



Github를 이용한 프로젝트 관리

Thank you