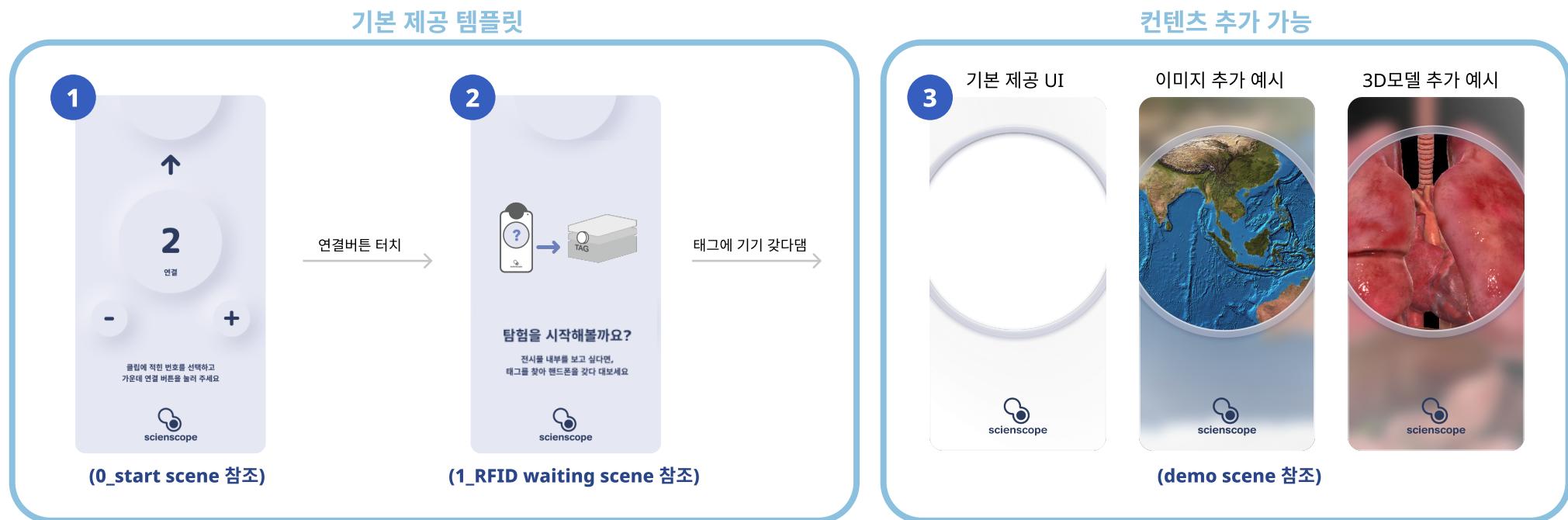


Scienscope Manual

1. 어플리케이션 구성



1 기기 연결 화면

설명

- 과학청진기 모듈과 스마트폰 연결 단계
- 모듈과 대응되는 번호를 선택한 후 가운데 연결버튼을 터치하면 연결됨

유의 사항

- 기기별로 대응되는 기기를 정해놓고 사용할 경우 스킵 가능

2 RFID 대기 화면

- RFID tag 인식을 대기 중인 단계
- 올바른 기기와 연결성공시 나타나는 화면
- 컨텐츠 관람 중, 기기가 태그로부터 멀어지면 다시 이 화면으로 돌아옴

- TAG_loading.cs script에서 태그와 이 다음에 넘어갈 scene index를 대응 시켜주어야 함

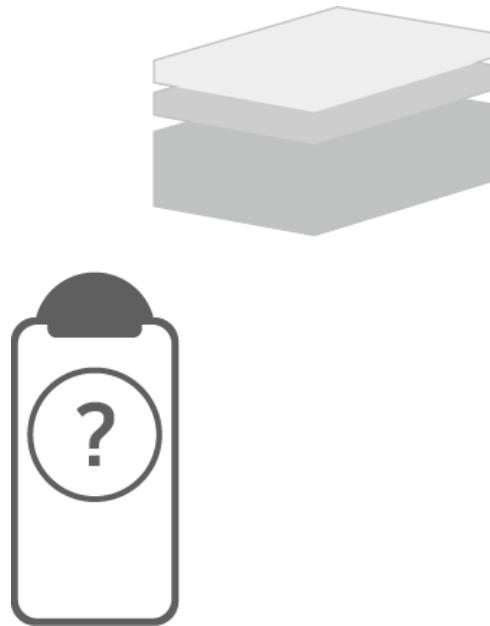
3 컨텐츠 출력 화면

- tag 인식된 후 대응되는 컨텐츠가 출력되는 화면
- 실질적으로 컨텐츠를 디자인하셔야 하는 구간

- 2D 이미지와 3D 모델링은 각각 사용되는 UI 가 다르므로, 사용법을 확인하고 각 컨텐츠에 해당하는 UI를 활성화하고 컨텐츠를 끌어다 놓아야함
(→ 자세한 사용법은 3-Template Scene 참고)

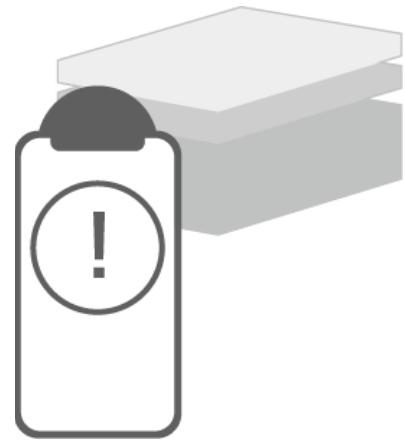
2. 시스템 구성

2-1. 과학청진기 모듈 데이터 출력



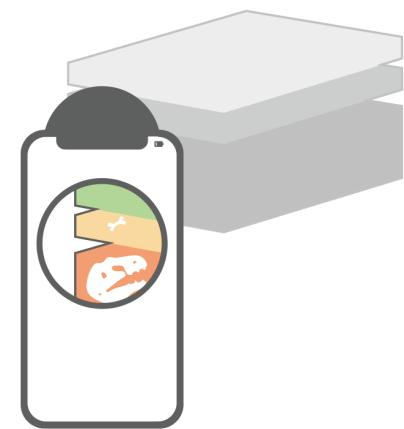
1. 사물 표면에서 거리가 있을 때

- 모듈로부터 **+40 ~ +255** 까지의 거리 데이터만 들어옴



2. 사물 표면에서 거리가 가까울 때

- 거리 센서 데이터가 +40 이하일 때
- START가 찍히고 RFID reading
- RFID가 읽히면
rft : (time) rfid : (----id-----) 와 같은 형태로 전송



3. RFID 인식 후에

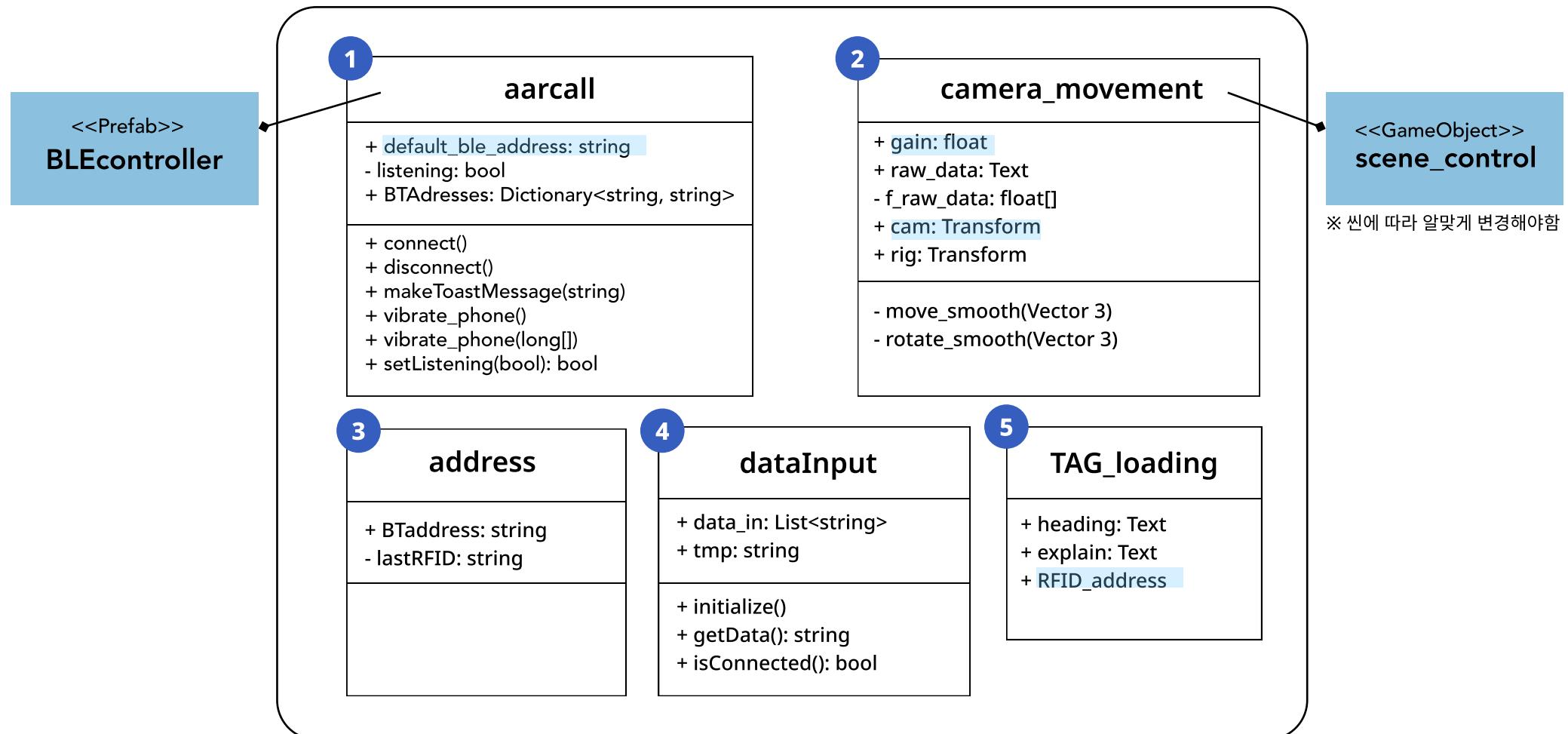
- 대응되는 컨텐츠 출력
- **q1 q2 q3 mx my** 형태로 값 전송
- RFID가 인식되지 않으면 출력 X

2. 시스템 구성

2-2. 주요 Script 구조 및 설명

: 사용자 셋업이 필요한 부분

common scripts : 과학청진기 모듈과 통신기능 및 센서값 관련 스크립트



1 : 과학청진기 모듈과의 블루투스 통신 관리
prefab이 scene 안에 있어야 연결, 데이터 송수신 가능

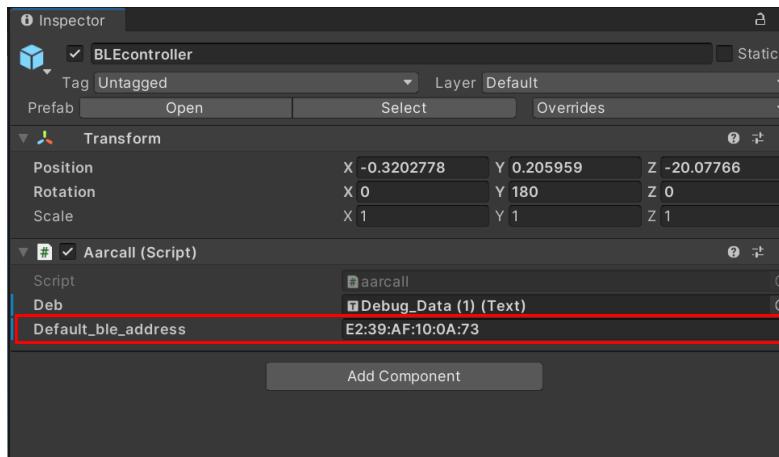
2 : 모듈로부터 넘어온 IMU 센서값과 mouse 센서 값을
cam_rig에 반영하여 회전, 이동시킴

3 : static class로 scene이 넘어가도 처음에 설정한
bluetooth module address를 저장하는 스크립트

4 : 과학청진기 모듈로부터 센서 값을 받아 List 형태로
저장하고서 getData()를 통해 하나씩 전달

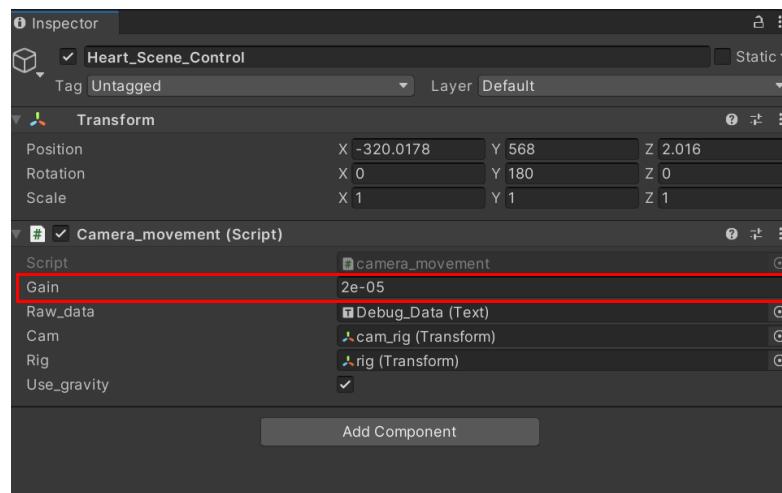
5 : RFID address를 관리, 모듈에서 넘어온 데이터가
RFID일때 대응되는 씬 로드하는 역할

2-3. 설정해야 하는 변수들 (모듈과의 연결)



- **default_ble_address**

- 앱이 실행될 때, 제일 기본으로 연결되는 디바이스의 ble address
- 스마트폰과 모듈을 대응시켜 놓고 (1번 화면 안거치고) 사용할 경우에 꼭 여기에 대응되는 기기의 bluetooth address를 입력해두어야 함



- **gain**

- 마우스 센서 값이 cam_rig를 움직일 때 곱해지는 계수 값
- 마우스의 감도와 같은 역할이라고 보시면 됨
- 0.00004 정도를 기본으로 생각하고 수치를 위아래로 바꿔 가며 원하는 속도가 나오도록 하면 됨

2-3. 설정해야 하는 변수들 (RFID 태그와 컨텐츠 대응)

```

public class TAG_loading : MonoBehaviour
{
    //태그 인식되면, 컨텐츠 로딩하는 스크립트

    /// <summary>
    /// Inspector 창에서 RFID_address에 원하는 RFID 태그의 주소값을 입력해주세요.
    /// 해당 RFID 태그 주소의 index +1의 scene id로 자동 대응되어 넘어갑니다.
    /// 혹은 아래 update 구문의 아래에, 직접 tag id와 scene build index를 대응시켜 주세요.
    /// </summary>
    public Text heading;
    public Text explain;

    public string[] RFID_address;

    void Start()
    {
        address.SetLastRFID("");
    }

    // Update is called once per frame
    void Update()
    {

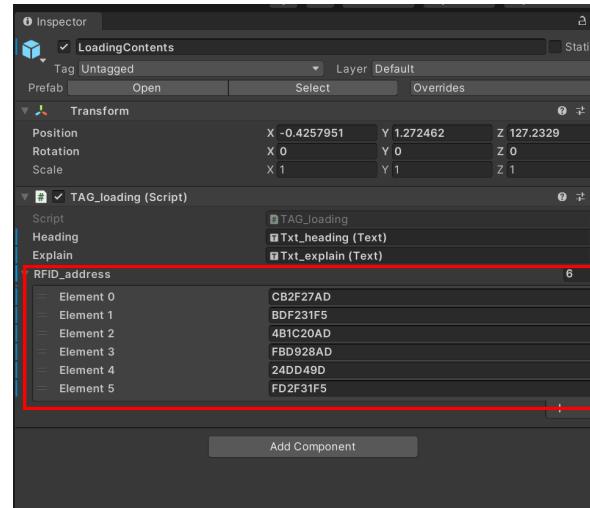
        string tmp = dataInput.getData();
        //Debug.Log(address.BTAddress);

        if (tmp!=null&&Application.platform == RuntimePlatform.Android)
        {
            heading.text = tmp;
            //Debug.Log(tmp);
            if (tmp.Contains("rfID"))
            {
                explain.text = "잠시만 기다려주세요...";

                /// move on contents scenes ///
                for(int i = 0; i < RFID_address.Length; i++)
                {
                    if(tmp.Contains[RFID_address[i]]) SceneManager.LoadScene ["DemoScene"] LoadSceneMode.Single;
                }
                인식해야할 tag id tag 인식되면 넘어갈 Scene
            }
        }

        /*int target_scene_num = address.GetCurrentSceneNumber(); //현재씬번호 받아오기
        if (target_scene_num != -1) //에러값이 아니면
        {
            heading.text = "전시를 안으로 가는중";
            explain.text = "잠시만 기다려주세요...";
            //씬전환
        }
    }
}

```



- 1_RFID_waiting scene에서 태그의 id와 디자인한 scene의 build index를 대응시켜주세요.

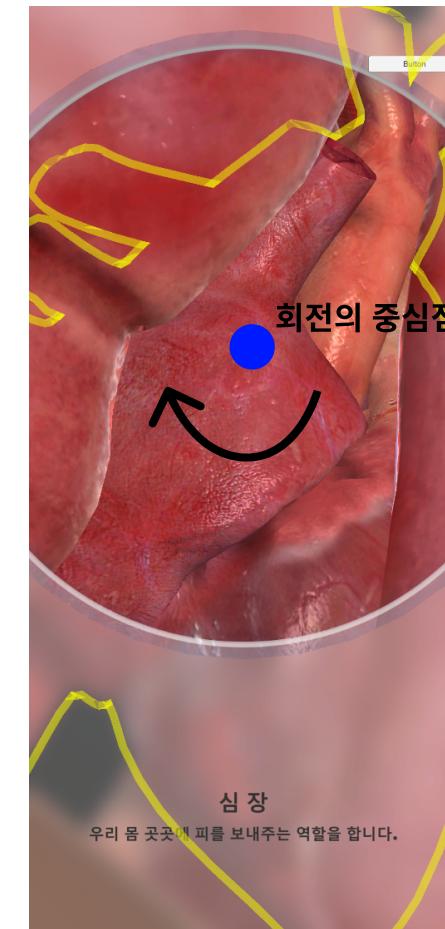
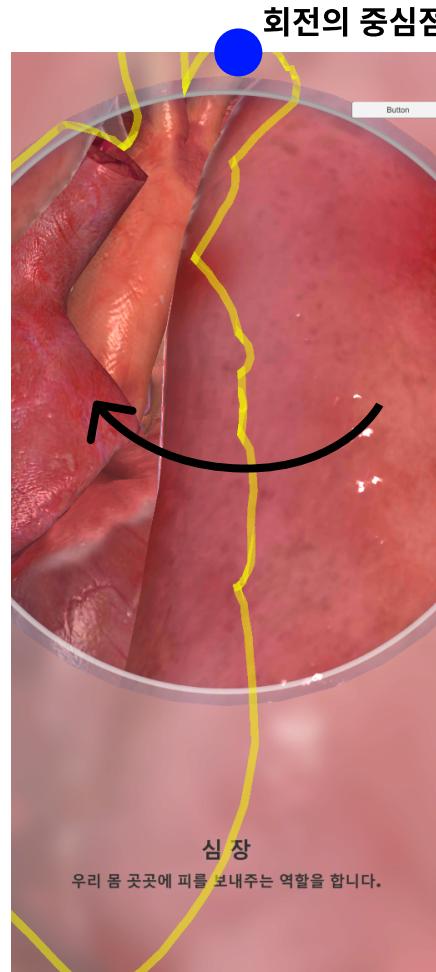
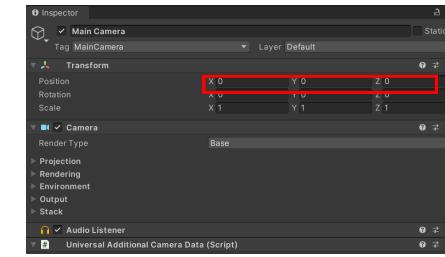
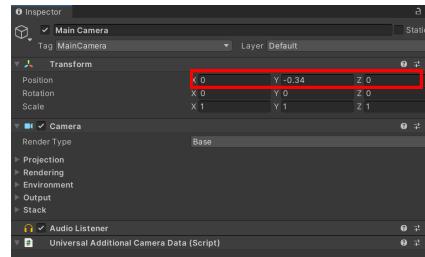
- 위의 inspector 창에서 rfid 값을 입력하시고,
- 왼쪽과 같이 TAG_loading 스크립트에서 대응되는 scene의 이름이나 id를 입력해주시면 됩니다.

2-3. 설정해야 하는 변수들 (화면 회전 중심점)

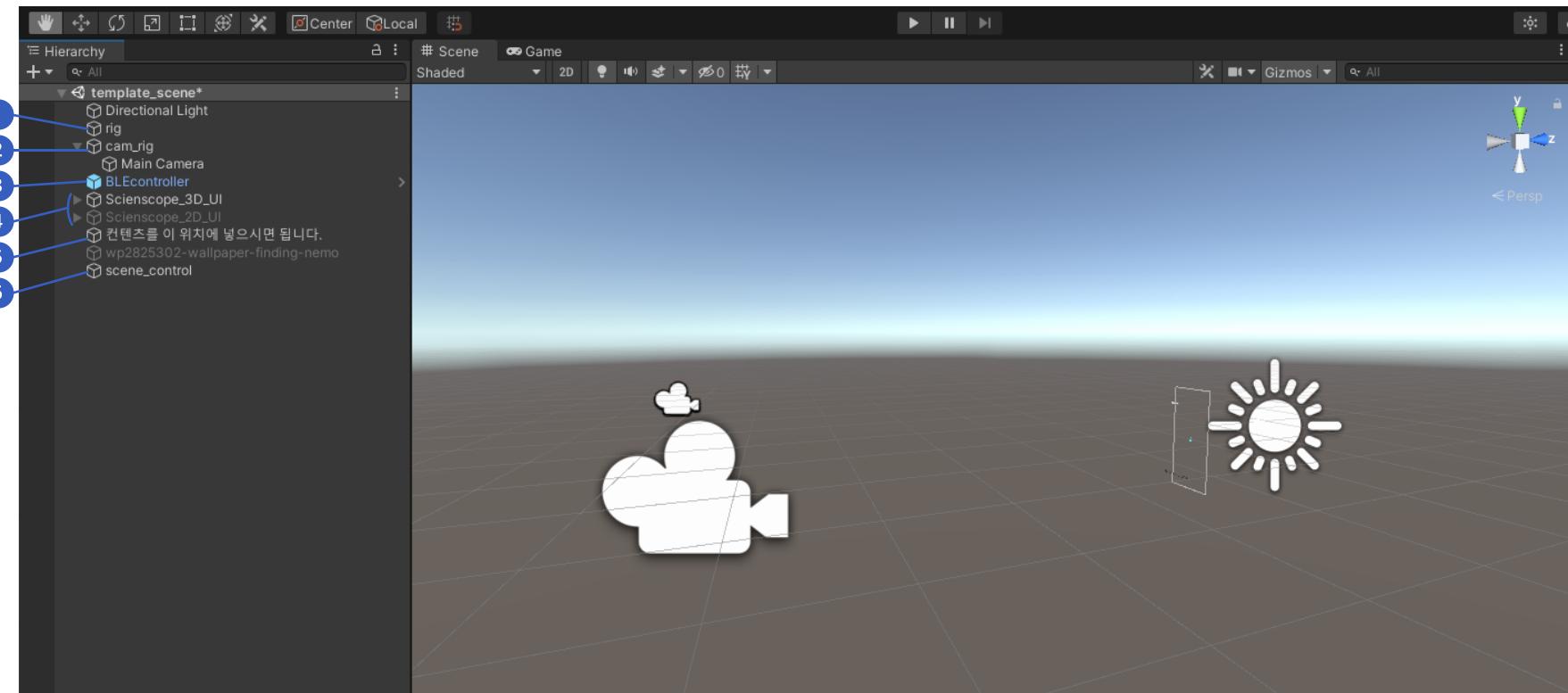
- cam_rig와 Main camera의 상대위치에 따라 회전의 중심점이 달라집니다.

ex)

Main camera의 y position을 -0.32 정도로 내리면 모듈이 체결되는 부위를 기준으로 화면이 회전하며, 0으로 설정하면 화면의 컨텐츠 뷰어 중심을 기준으로 회전합니다.



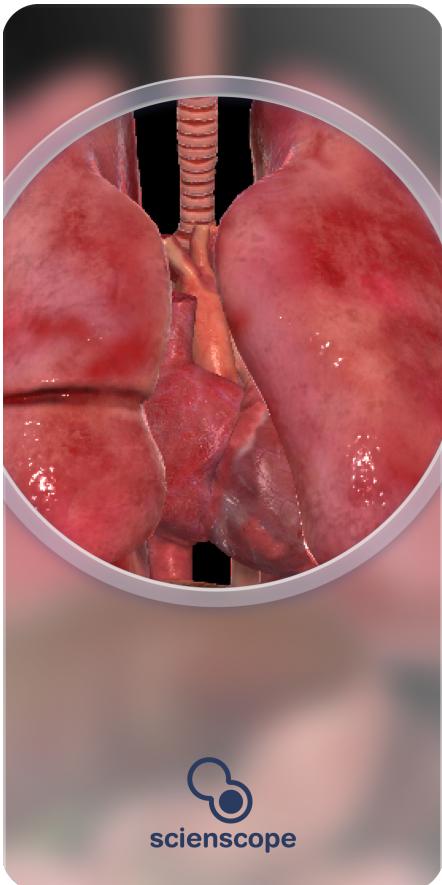
3. Template-Scene 설명



- ① **rig** : 블루투스를 통해서 넘어온 quaternion값을 반영하는 빈 객체, **cam_rig**에서 이 객체의 z축 회전만 전달 받아서 적용
- ② **cam_rig** : 메인 카메라와 회전의 중심점 조절을 위한 rig. Main camera와 rig의 y축 값 차이만큼 화면 회전의 중심점이 디바이스의 윗 방향으로 올라감.
- ③ **BLEcontroller** : 블루투스 통신 관리
- ④ **Scienoscope_3D_UI, 2D_UI** : 가장 UI 형태, 자유롭게 편집해서 사용하시면 됩니다.
- ⑤ **test_indicator** : 해당 씬을 export 했을 때, 움직임을 확인할 수 있도록 넣어둔 기본 큐브. (**컨텐츠를 이 위치에 넣으시면 됩니다.**)
- ⑥ **scene_control** : 각 씬마다 필요한 스크립트를 작성하기 위한 empty gameobject. 주로 **camera_movement**와 부차적인 스크립트들을 붙이면 됩니다.
(자세한 것은 **DemoScene**의 **Heart_Scene_Control** 참조)

※ Template Scene을 원하시는대로 편집 한 후, 다른 이름으로 저장해서 Scene을 활용하셔도 되고,
BLEcontroller, rig, cam_rig, scene_control 오브젝트들을 다른 씬으로 옮겨서 사용하셔도 됩니다.

4. DemoScene 설명 (3D 기준)



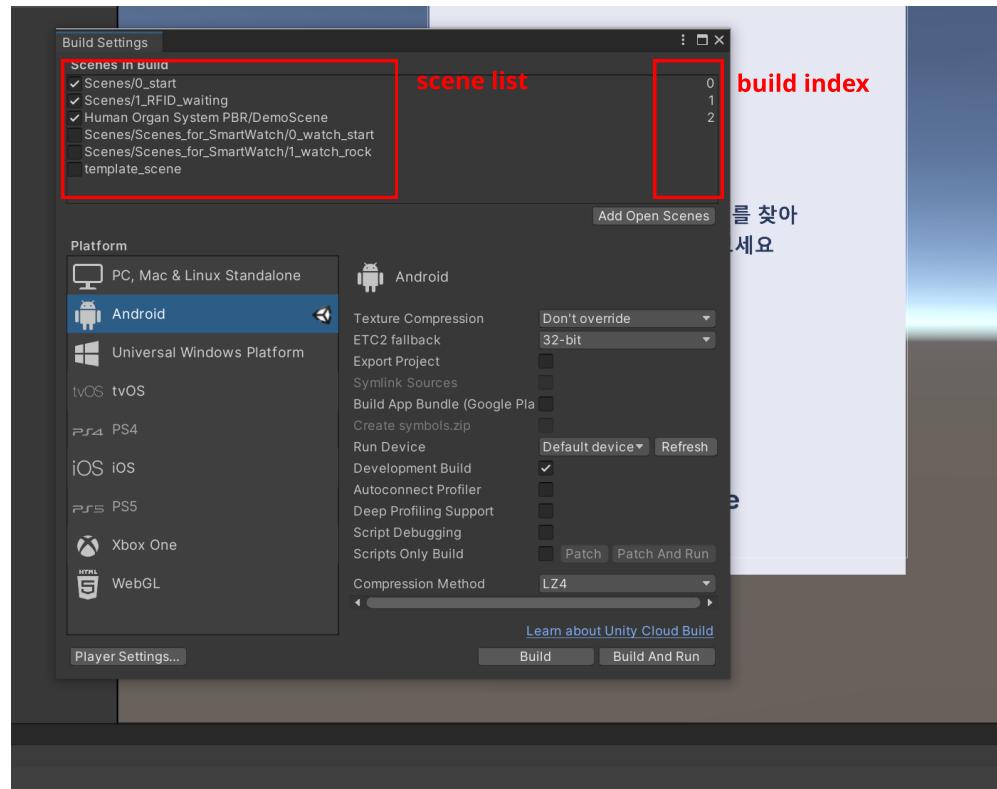
The image shows the Unity Editor interface with the Hierarchy and Scene windows open. The Hierarchy window displays the scene structure:

- 1 rig
- 2 cam_rig
- 3 Directional Light
- 3 Reflection Probe
- 3 human organ system
- 3 Scienscope_3D_UI
- 3 EventSystem
- 4 BLEController
- 5 Heart_Scene_Control
- 5 rotation_indicator

The Scene window shows a 3D view of the human organ system model (lungs, heart, liver, intestines) positioned above a grid floor. A camera is attached to the 'cam_rig' object, which is used to capture the 3D scene. The Unity interface includes various toolbars and panels at the top and sides.

1 rig는 센서를 통해 들어온 quaternion을 그대로 반영해서 rotation
2 cam_rig는 rig의 z축 회전만을 받아옵니다.
3 human organ system 모델이 과학청진기를 움직이면서 볼 컨텐츠
4 모듈과의 통신은 BLEController를 통해서 들어옴 (Prefab 이름 변경하면 X)
5 Heart_Scene_Control에 붙어있는 camera_movement.cs 가 cam_rig를 움직여서 진행

5. Build 관련 주의사항



- Android Studio는 기본적으로 설치되어 있어야 합니다. (SDK, JDK 포함)
- 0_start scene은 연결할 기기를 선택하는 단계로 skip 가능
(체크 해제, 스마트폰과 모듈을 1대1로 대응시켜서 전시할 경우)
- 1_RFID_waiting scene은 필수 (태그 인식되지 않았을 때, 대기화면)
- TAG_loading에서 각 scene들의 build index나 이름을 정확하게 입력해야 전환됩니다.

Appendix. Sequence Diagram

Common Script 간 호출 및 변수들의 흐름을 파악할 수 있다

