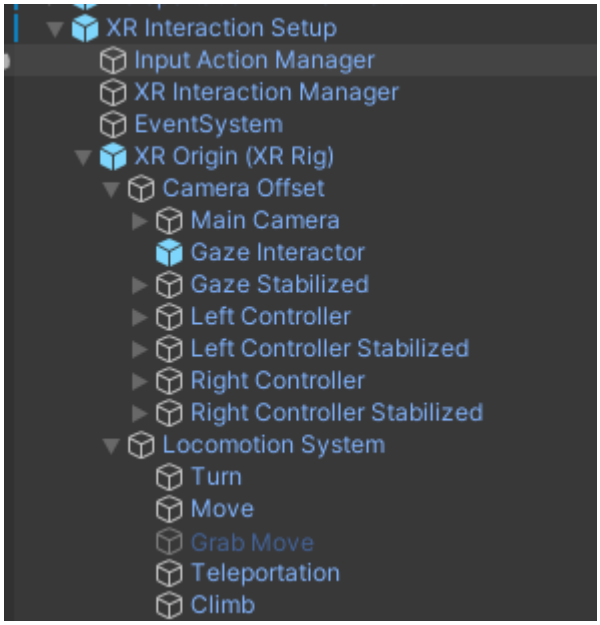
VR 기본 설명

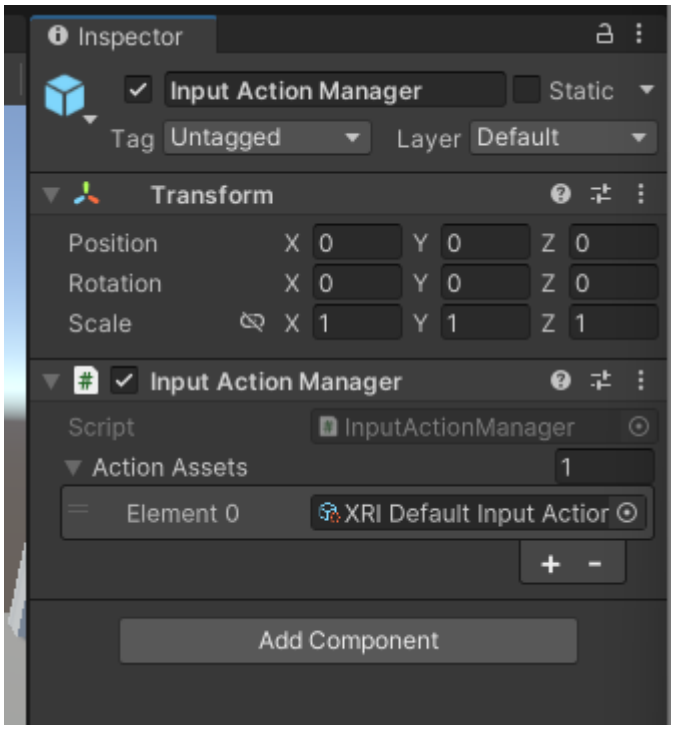
XR Interaction ToolKit

(1) Player Component



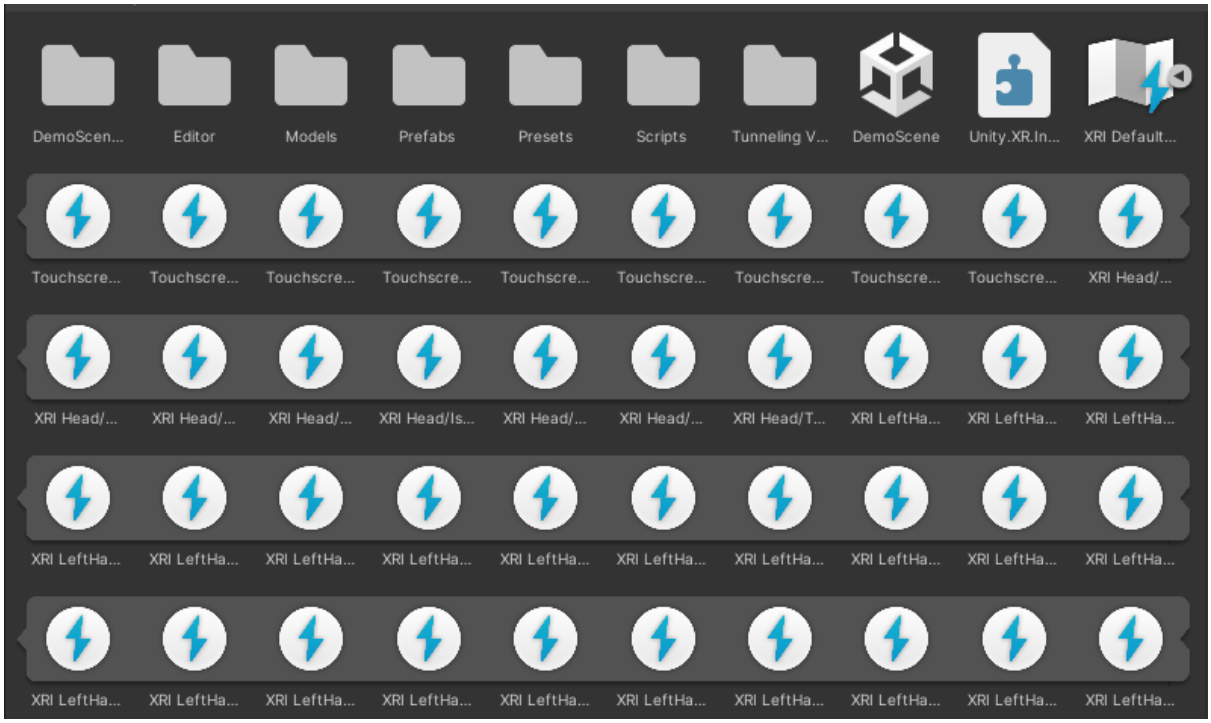
Player 기본 구조. (변경 X 여기서 필요한 기능만 추가로 컴포넌트로 붙이도록 한다.)

- Input Action Manager



플레이어가 설정한 컨트롤러 인풋을 액션을 사용하기 위한 매니져 스크립트.

Action Assets에 존재하는 XRI Default Input Action은 기본적으로 제공되는 디폴트 액션이며, 원한다면 직접 정의해서 추가해도 되지만 살펴본 결과 그대로 사용하고 해당 액션 내에서 필요한 액션만 따로 정의해도 될 듯 하다.



디폴트 액션을 펼쳐보면 이렇게 다양한 액션들이 정의 되어있다.

- XR Interaction Manager

Interactable과 Interaction하기 위해 필요한 매니져 스크립트

- EventSystem

플레이어가 UI와 상호작용하기 위한 이벤트 시스템.

XR UI Input Module은 VR 액션을 통해서 UI와 상호작용이 가능하도록 미리 액션이 정의되어있는 컴포넌트

- XR Origin (XR Rig)

플레이어 본체. 실제로 사용자가 움직이는 모델로 Character Controller 컴포넌트 기반으로 움직인다. (콜라이더와 리지드 바디가 안쪽에서 정의된 컴포넌트)

XR Origin 컴포넌트에서 플레이어의 머리에 해당하는 카메라를 제어할 수 있다. 머리의 위치를 바꾸고 싶다면 Camera Y Offset을 설정해준다.

Character Controller Driver 컴포넌트는 Character Controller와 밑에서 기술할 움직임을 담당해주는 Locomotion System의 연결을 담당해준다.

XR Input Modality Manager 컴포넌트는 사용자의 컨트롤러 트래킹과 핸드 트래킹을 전환해주는 일을 담당해준다. 현재 우리 게임에서는 필요 없는 기능이라 사실 지워도 되지만 혹시 모르니 없애지는 않았다.

XR Gaze Assistance 컴포넌트는 사용자의 컨트롤러의 Ray Interactor를 사용하지 못하는 상황(광선이 화면 밖을 나감) 상태일 때 사용자의 시선을 Ray Interactor로 대신 사용할 수 있게 해주는 기능

(그런데 시선 기능은 사용해본 결과 너무 눈이 아파서 사용할 만한 것이 못된다.)

- Camera Offset

카메라 위치 오프셋

- Main Camera

메인 카메라(플레이어의 머리)

- TunnelingVignette

Tunneling Vignette Controller는 사용자가 움직임을 취할 때 특수 효과를 내어주는 기능인데, 사용해보니 현실감이 떨어져 없애는 것을 추천한다.

- Gaze Interactor

시선 응시 + 눈 깜빡임으로 상호작용(쓸만한 것이 못됨)

- Gaze Stabilized

XR Transform Stabilizer 컴포넌트를 통해 Gaze Interactor의 위치 안정화 작업을 해주는 보조 컴포넌트

- Left Controller

사용자의 왼손 컨트롤러.

Action Based Controller Manager 컴포넌트로 왼손 컨트롤러 액션을 등록 + 보조. 미리 기본 기능에 대한 액션의 정의되어 있고, Locomotion Settings에서 Boolean 체크를 통해 이동 방식 제어가 가능하다. 또한 UI Settings를 통해서 왼손 조이스틱이나 그랩으로 스크롤링 제어가 가능하다. 해당 컴포넌트는 없애도 되지만 기본 기능을 사용하기 쉽게 도와주기에 있는게 좋아보인다.

XR Controller 컴포넌트는 무조건 있어야 하는 컴포넌트이고, 왼손 컨트롤러의 액션의 초기화와 사용할 것인지에 대해서 설정을 해주는 컴포넌트이다.

XR Interaction Group 컴포넌트는 사용자의 상호작용들 간의 우선 순위와 동시성을 제어해준다.

- Poke Interactor

XR Poke Interactor는 사용자가 컨트롤러를 통해 직접 "버튼을 누르는" 행위로 상호작용을 할 수 있게 해주는 컴포넌트이다.

- Direct Interactor

XR Direct Interactor는 컨트롤러에 Sphere Collider가 있어서 해당 콜라이더 안에 물체가 들어오면 상호작용을 할 수 있게 해주는 컴포넌트이다.

- Ray Interactor

XR Ray Interactor는 컴트롤러에서 나가는 레이에 물체가 닿으면 상호작용을 할 수 있게 해주는 컴포넌트이다.

- Teleport Interactor

XR Ray Interactor를 사용하지만, Ray의 Layer Mask를 텔레포트 지형으로 제한해서 사용할 수 있게 도와준다.

여기에 XR Controller 가 함께 있는데, 해당 컴포넌트에 액션이 텔레포트 기능으로 제한이 되어 있다.

- Left Controller Stabilized

왼손 컨트롤러의 위치 안정화

- Right Controller

오른손인 것 제외하고 왼손 컨트롤러와 동일

- Right Controller의 Interactor 기능 왼손과 동일

- Right Controller Stabilized

왼손 컨트롤러와 동일

- Locomotion System

VR 이동에 대한 기능을 담당해주는 컴포넌트. Locomotion System은 위에서 언급한

Character Controller Driver 을 통해서 Character Controller를 제어해 이동 기능을 사용한다.

- Turn

Snap Turn Provider는 컨트롤러의 조이스틱을 튕길 때 마다 사용자를 45도씩 회전시키는 기능을 가졌다.

Continuous Turn Provider는 컨트롤러의 조이스틱을 지속적으로 움직여 사용자를 회전시키는 기능을 가졌다.

- Move

Dynamic Move Provider는 사용자의 Continuous Move를 담당한다.

- Grab Move

그랩 + 무브를 가능하게 해준다 (사용 X)

- Teleportation

텔레포트 기능을 사용하게 해준다. Delay Time은 텔레포트를 사용하고 나서 또 텔레포트를 사용할 수 있게 되기 까지의 시간이다.

- Climb

사다리 타기와 같은 기능을 사용할 수 있게 해준다.

위 움직임 기능들은 현재로서는

move -> 왼손 조이스틱

turn-> 오른손 왼 오 뒤 조이스틱

teleportation -> 오른손 앞 조이스틱

으로 설정이 되어있다.

(2) Action Based

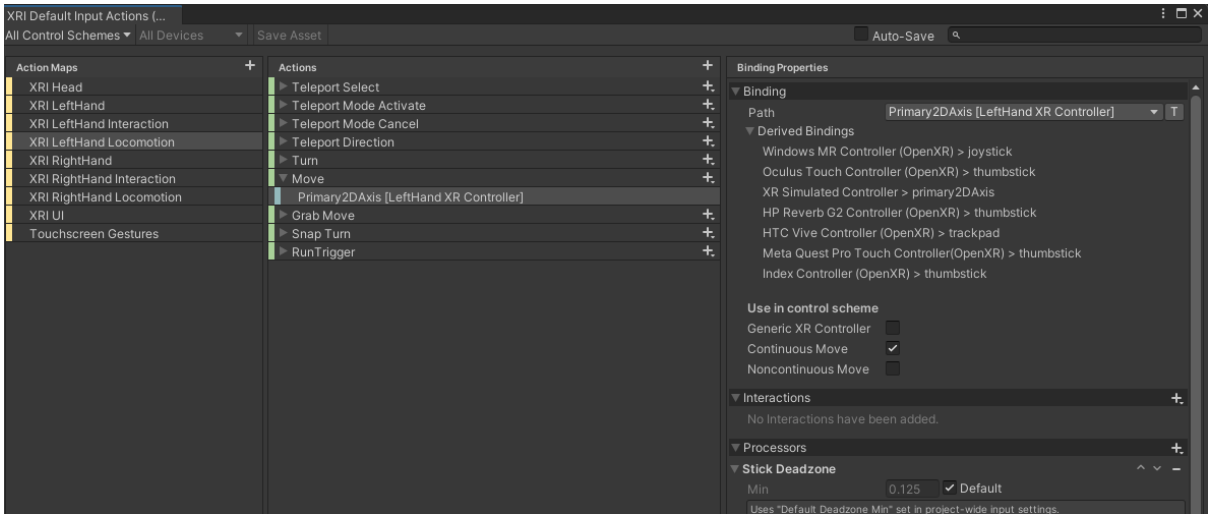
XR Interaction ToolKit의 컨트롤러 사용은 기본적으로 두가지로 이루어진다.

- Action Based

- Device Based

대부분의 경우 Action Based 방식을 사용해서 개발을 진행하게 되는데 그 이유는 플랫폼 별 호환이 잘 되어있기 때문이다 (다양한 VR 기기)

Action Based를 사용하는 방법은 미리 특정 Action을 정의



해서 해당 액션 SO 객체를 다른 스크립트에서 참조해 이벤트를 등록하고 해제하면서 사용한다.

```
// current action -> left controllor joy stick click
Unity 스크립트(자산 참조 1개)|참조 0개
public class ContinuousRunTrigger : MonoBehaviour {

    [Header("Action Variable")]
    [SerializeField, Tooltip("Register the action you want.")] private InputActionReference actionReference;
    [SerializeField, Tooltip("This function is used only on continuous movement.")] private  ContinuousMoveProviderBase cmBase;
    [SerializeField, Tooltip("This value is multiplied by the basic moving speed.")] private float mulSpeed;

    // event register
    Unity 메시지|참조 0개
    private void OnEnable() {
        actionReference.action.performed += SetRun;
        actionReference.action.canceled += SetOrigin;
    }

    // event unregister
    Unity 메시지|참조 0개
    private void OnDisable() {
        actionReference.action.performed -= SetRun;
        actionReference.action.canceled -= SetOrigin;
    }
```

예를 들어 이런식으로 InputActionReference를 참조해서 인스펙터 창에서 초기화를 해두고 이벤트를 등록한다.

Device Based는 복잡한 기능을 만드는데 유용하다. 예를 들어서 특정 컨트롤러의 어느 부분과 어느 부분을 동시에 누르면 어떤 기능이 나간다~~ 이런 것 처럼 말이다.

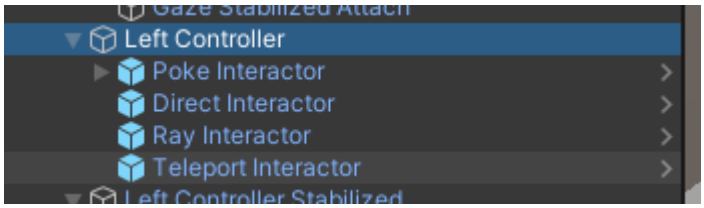
평소에 사용하는 " if (Input.GetKeyDown(KeyCode.Alpha1)) " 과 같은 방식이 이와 가장 유사하다. 만약 복잡한 컨트롤 조건으로 기능을 제어하고 싶다면 Device Based 방식을 사용하면 된다.

(3) Interaction

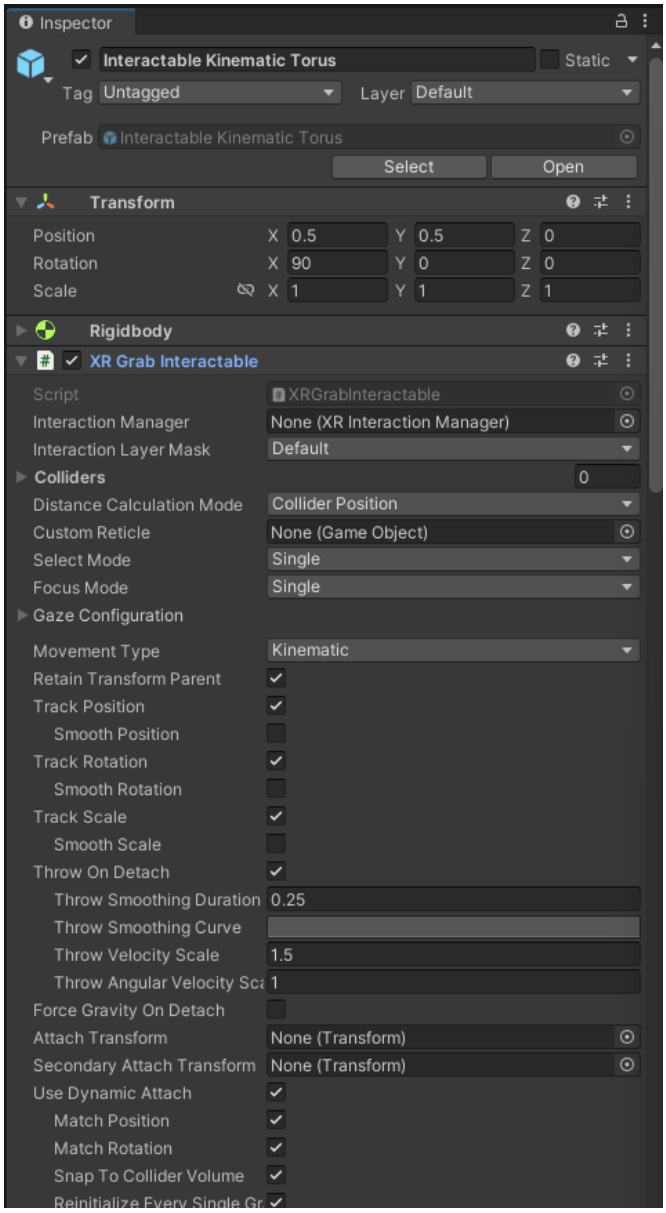
상호작용은 크게 두가지 컴포넌트로 나눌 수 있다.

- Interactor -> 사용자의 컨트롤러에 부착. 상호작용을 일으키는 객체

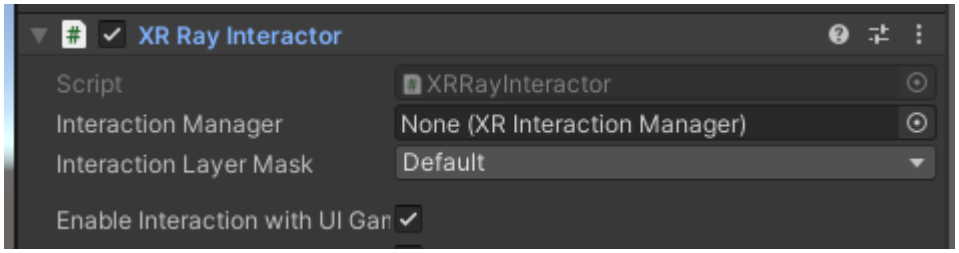
- Interactable -> 상호작용 하려는 물체에 부착. 실제 상호작용이 일어나는 객체

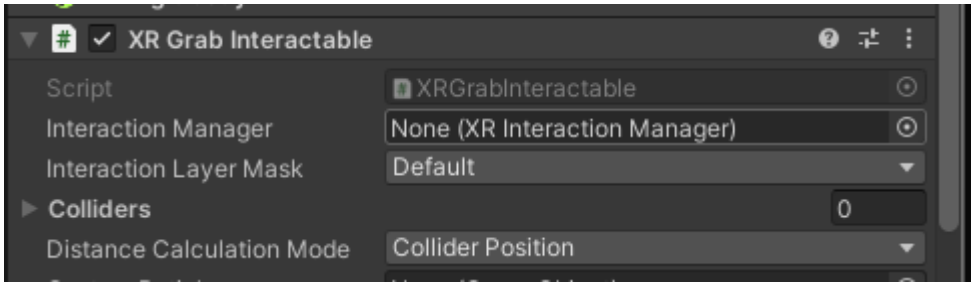


사용자의 Interactor는 이렇게 컨트롤러 쪽에 이미 부착이 되어있다.



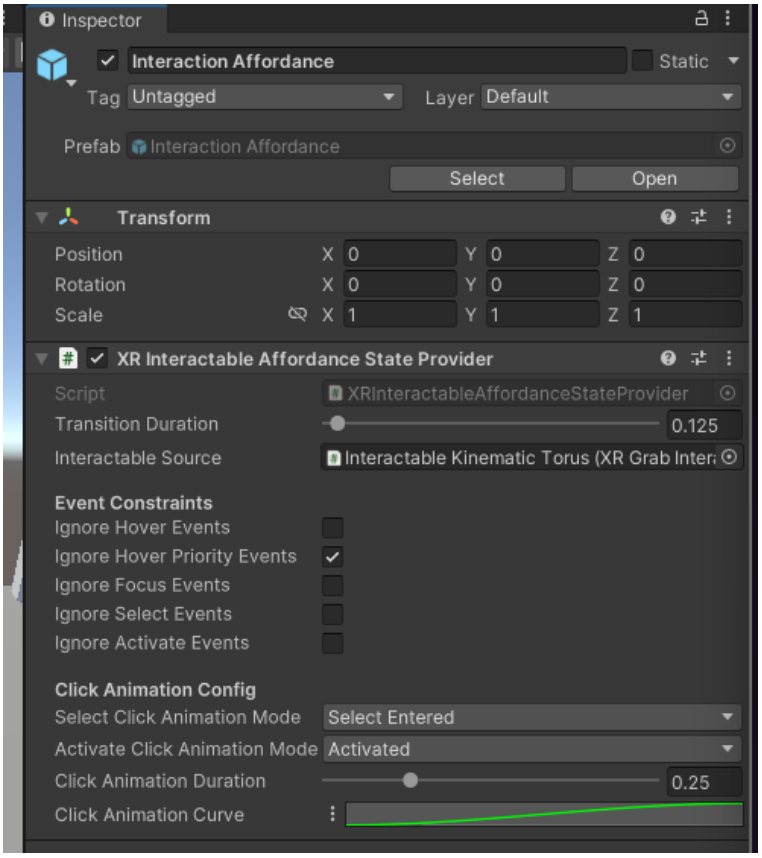
위 사진은 Grab Interactable의 예이다. 이 컴포넌트를 통해서 사용자는 잡기 상호작용이 가능하다. 상호작용이 되려면





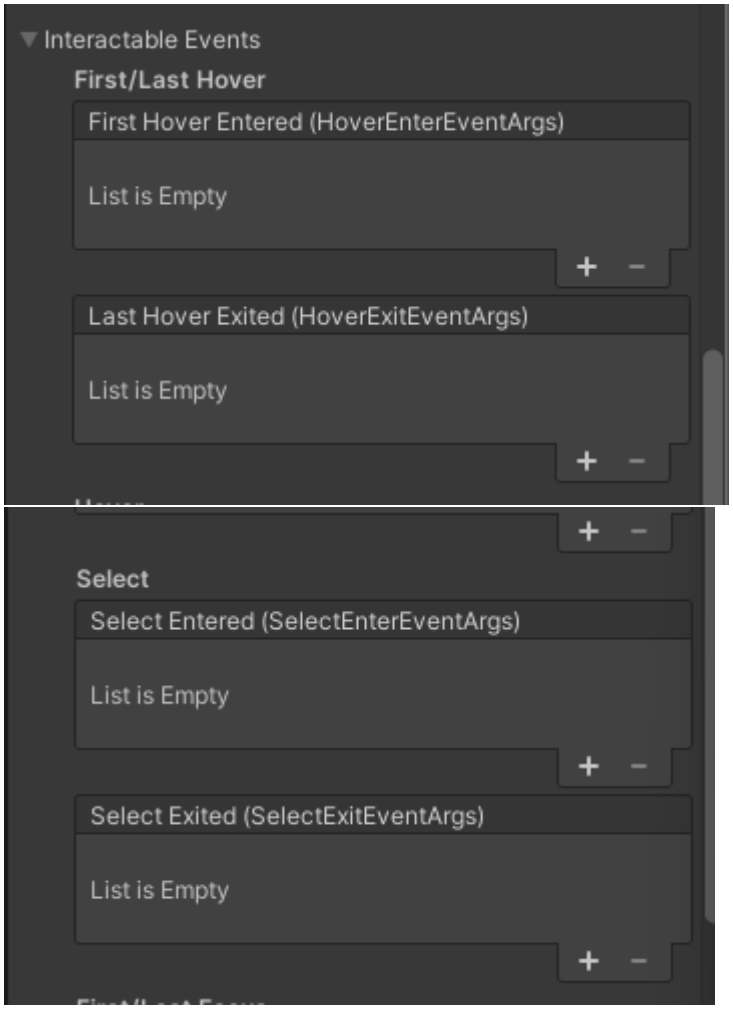
이렇게 Interactor와 Interactable의 Layer Mask가 동일해야 한다.

- XR Interaction Affordance State Provider



해당 컴포넌트는 하나의 Interactable 객체의 상태머신을 관리해주는 역할을 해준다.

예를 들면 모든 Interactable은



이런식으로 상태가 존재한다.

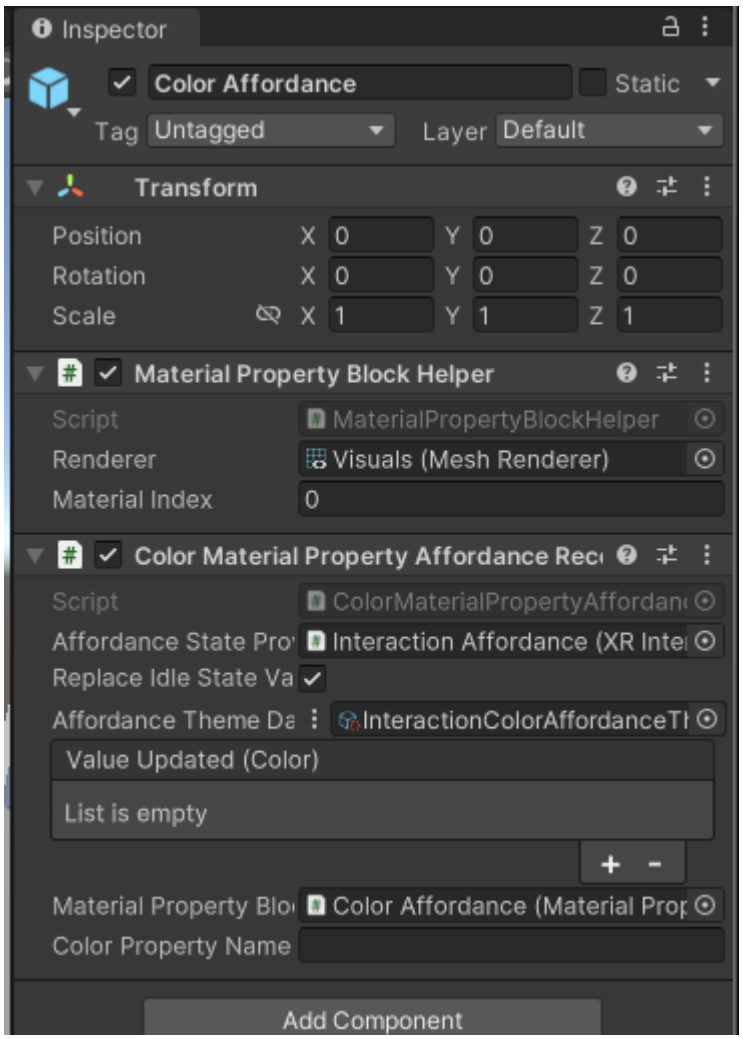
Hover -> Interactor가 Interactable의 상호작용 가능 범위 내에 들어온다.

Select -> Interactor가 Interactable과 상호작용 범위 내에서 그립 버튼을 눌렀다..

등등 존재한다. 해당 상태를 막아주거나, 상태 전이 속도를 조절해주는 등의 역할을 해준다.

- Color Affordance

Interactor의 상태에 따라 색상을 변경해준다.

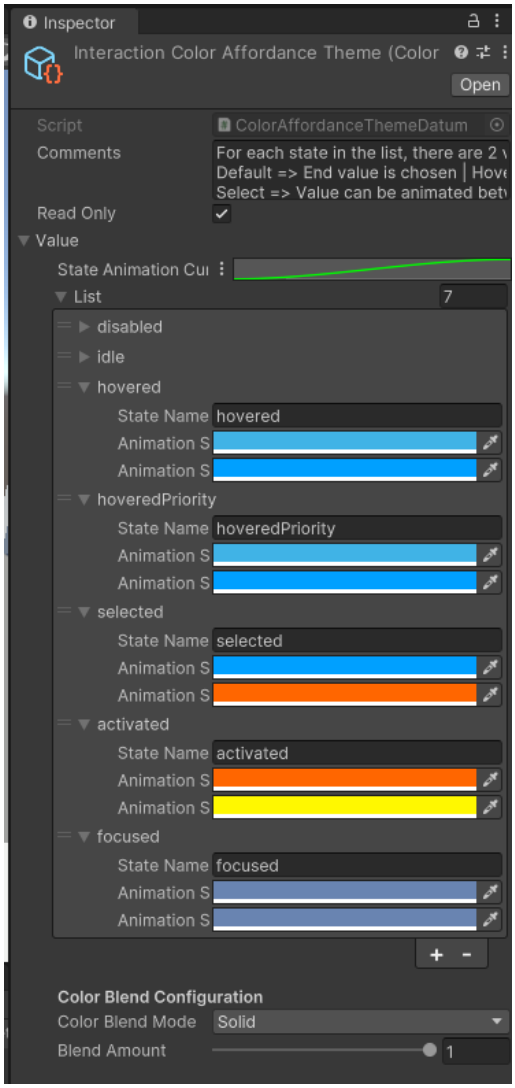


- Material Property Block Helper

색상을 변경할 매쉬 렌더러를 설정해준다.

- Color Material Property Affordance Receiver

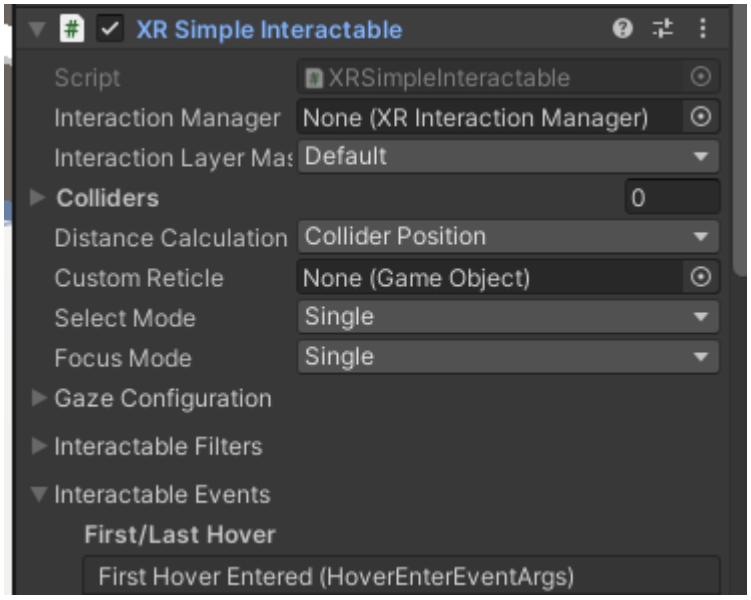
XR Interaction Affordance State Provider를 기반으로 Material Property Block Helper에 색상을 정해준다. 여기 보이는 Scriptable Object는 색상 정보가 담겨져 있다.



색상을 조절해주는 것은 사용자의 마음이지만 상태 머신을 제어해주는 컴포넌트는 필요해 보인다.

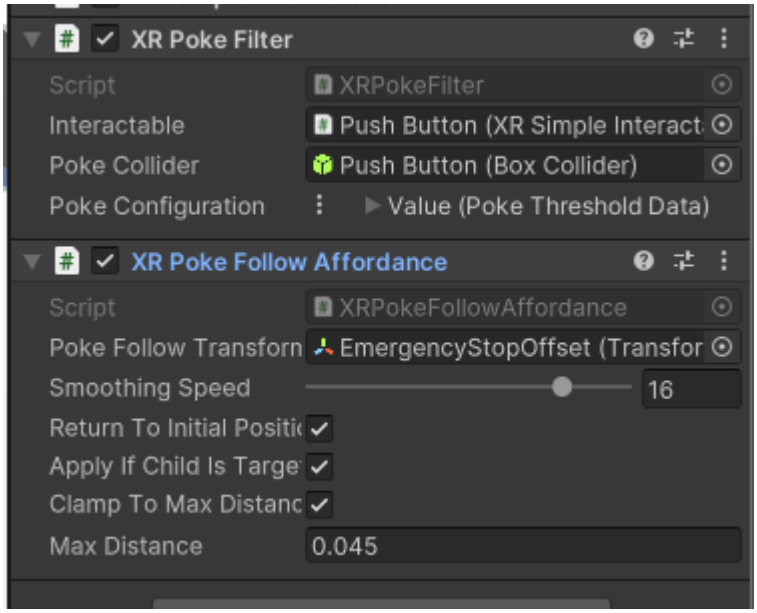
나머지 Interactable도 기본적으로 동일하다.

잡기, 등반과 같이 특수한 기능을 담은 Ineractable이 아니라 단순히 상호작용을 하고싶은 거라면



이렇게 Simple Interactable을 이용하면 된다.

이전 위에서 언급한 것과 같이 "실제 버튼을 누르는 행위" 와 같은 상호작용 방식을 Poke Interact 라고 하는데 이를 사용할 때에는 Interactor가 들어가는 객체에 필수적으로



이 두가지 컴포넌트가 들어가야 한다.

- XR Poke Filer

버튼을 누르는 등의 기능을 가능하게 해준다.

- XR Poke Follow Affordance

버튼을 누르면 버튼이 눌리면서 들어가게 되는데 이러한 애니메이션이 가능하도록 해주는 컴포넌트다.(없어도 된다는 소리)

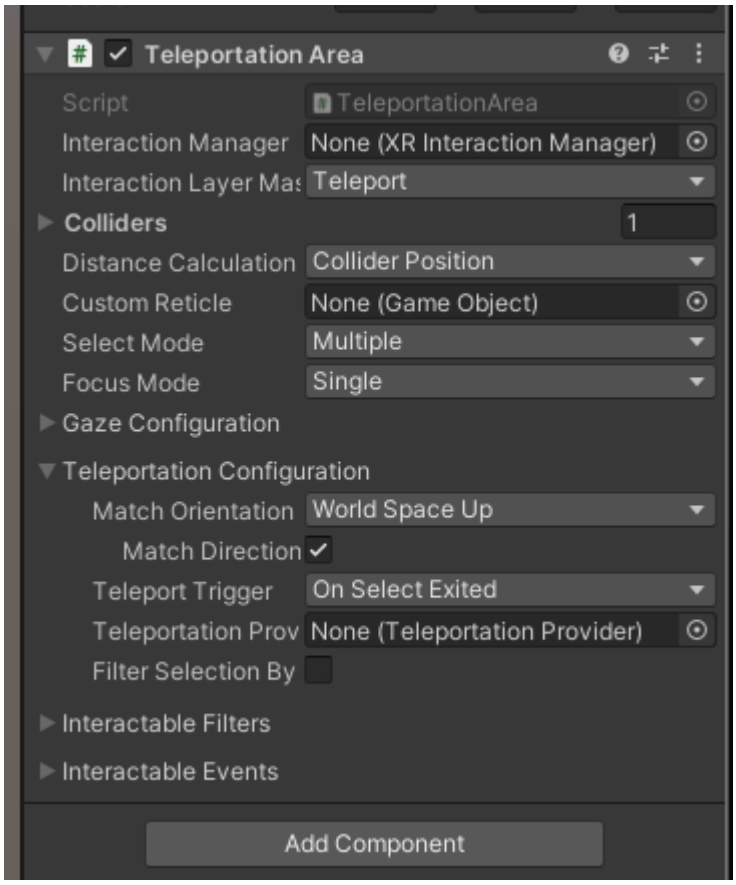
(4) Teleport

텔레포트가 가능한 장소는 정해져있다. 크게 두가지 경우가 있는데.

- Teleportation Area -> 부착 지면 전체 어디든 텔레포트 가능

- Teleportation Anchor -> 부착 지면의 특정 부분에 고정 텔레포트

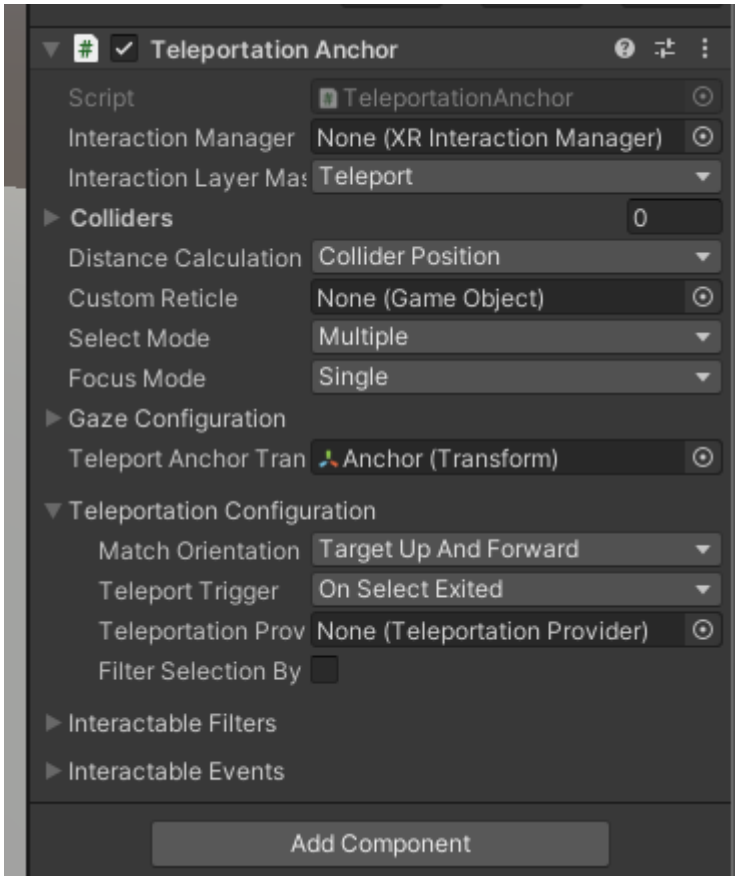
Teleportation Area는 지면을 만들고 그곳에



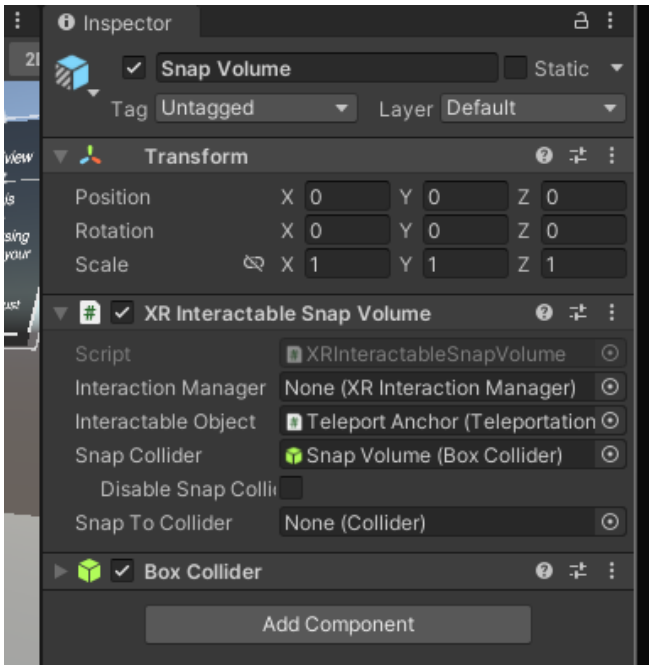
이렇게 Teleportation Area 컴포넌트를 붙이면 된다.

(항상 레이어 마스크를 확인하자)

Teleport Anchor는



이걸 붙이면 된다.



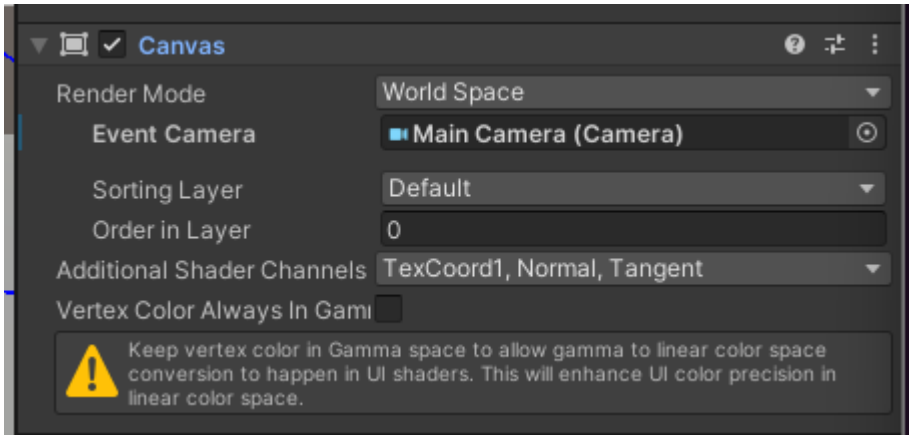
해당 부분은 시선을 사용해서 텔레포트를 하고싶을 때 시선 처리를 보조해주는 기능을 해준다

(없어도 된다는 소리)

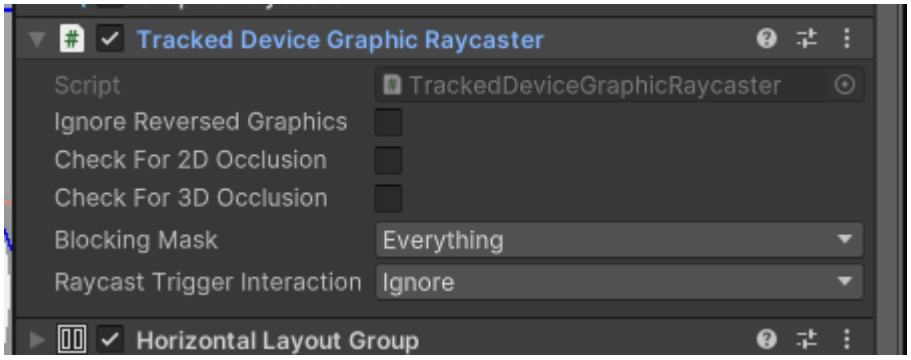
(5) VR UI

기본적으로 Canvas -> World Space 지정

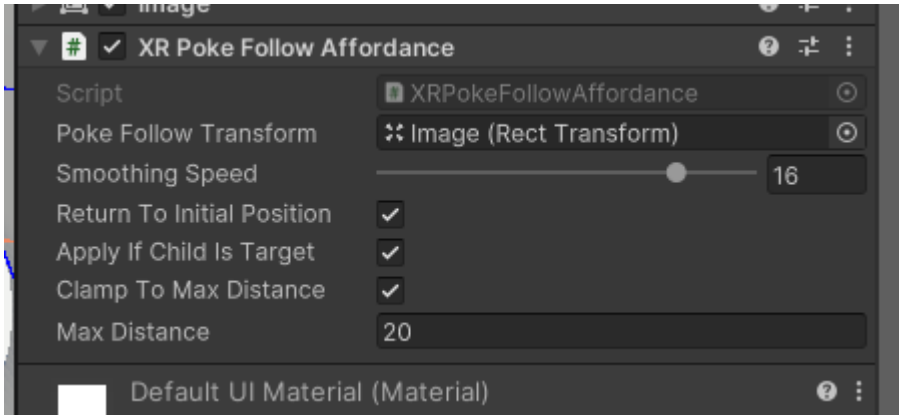
Event Camera -> Main Camera 로 지정해야 월드 좌표계에서 직접 눈으로 UI를 확인하는 것이 가능하다.



이후 해당 컴포넌트를 붙여야 컨트롤러의 상호작용으로 UI를 다루는 것이 가능해진다.



나머지는 똑같으나, 만약 Poke 기능으로 버튼이 눌리는 듯한 것을 연출하고 싶다면 버튼과 같은 UI 오브젝트에



해당 컴포넌트를 붙여준다.

(6) 참고 자료

XR Interaction Toolkit | XR Interaction Toolkit | 3.0.1 (unity3d.com)

(96) 눈코딩 - YouTube