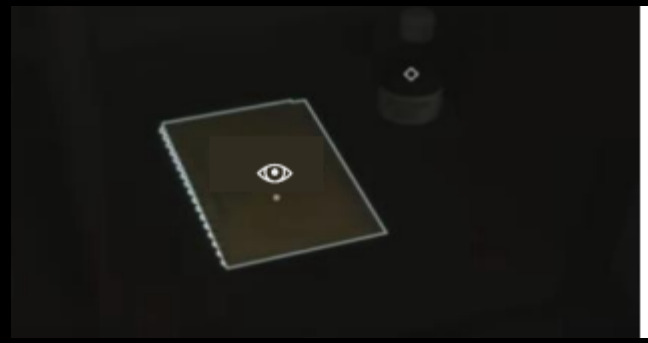
오브젝트 상호작용 UI

2023.08.28

작성자 : 신명지

GOAL

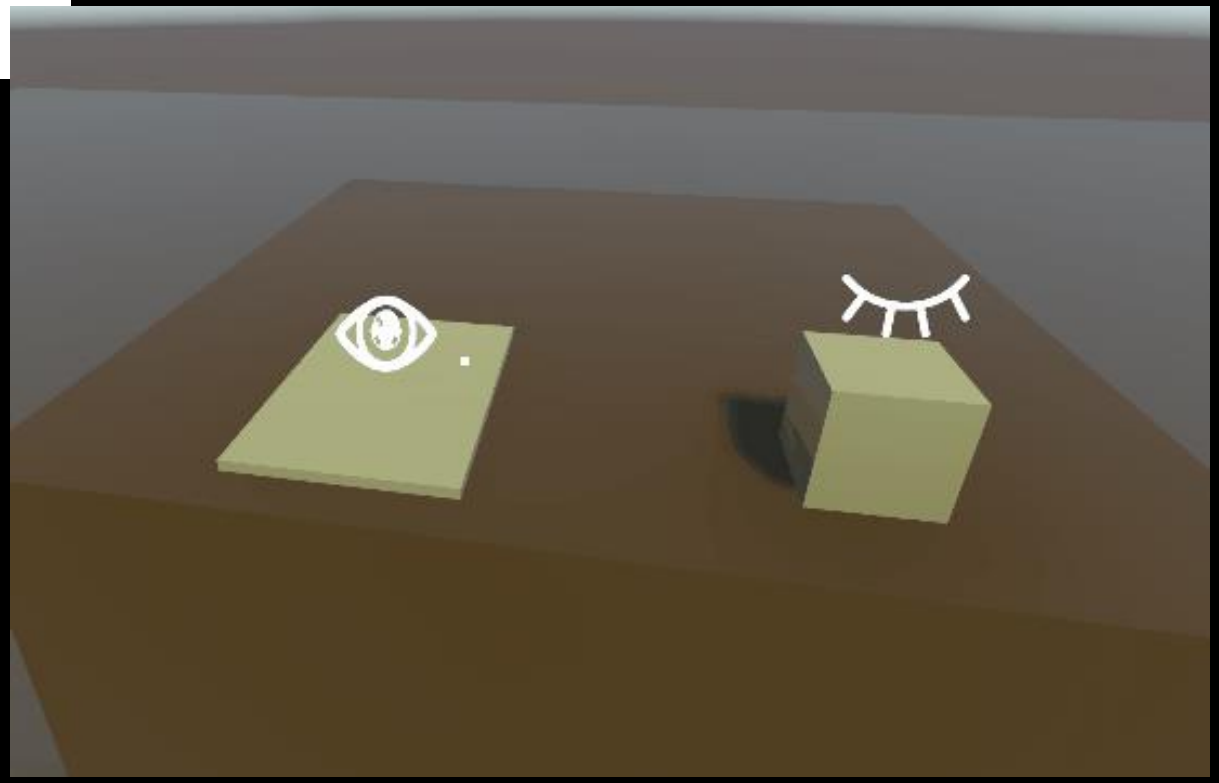


1. 상호작용 가능한 오브젝에 화면 중심점이 닿으면 왼쪽 사진과 같이 외각선이 밝게 보이며, 눈 아이콘이 표시된다.

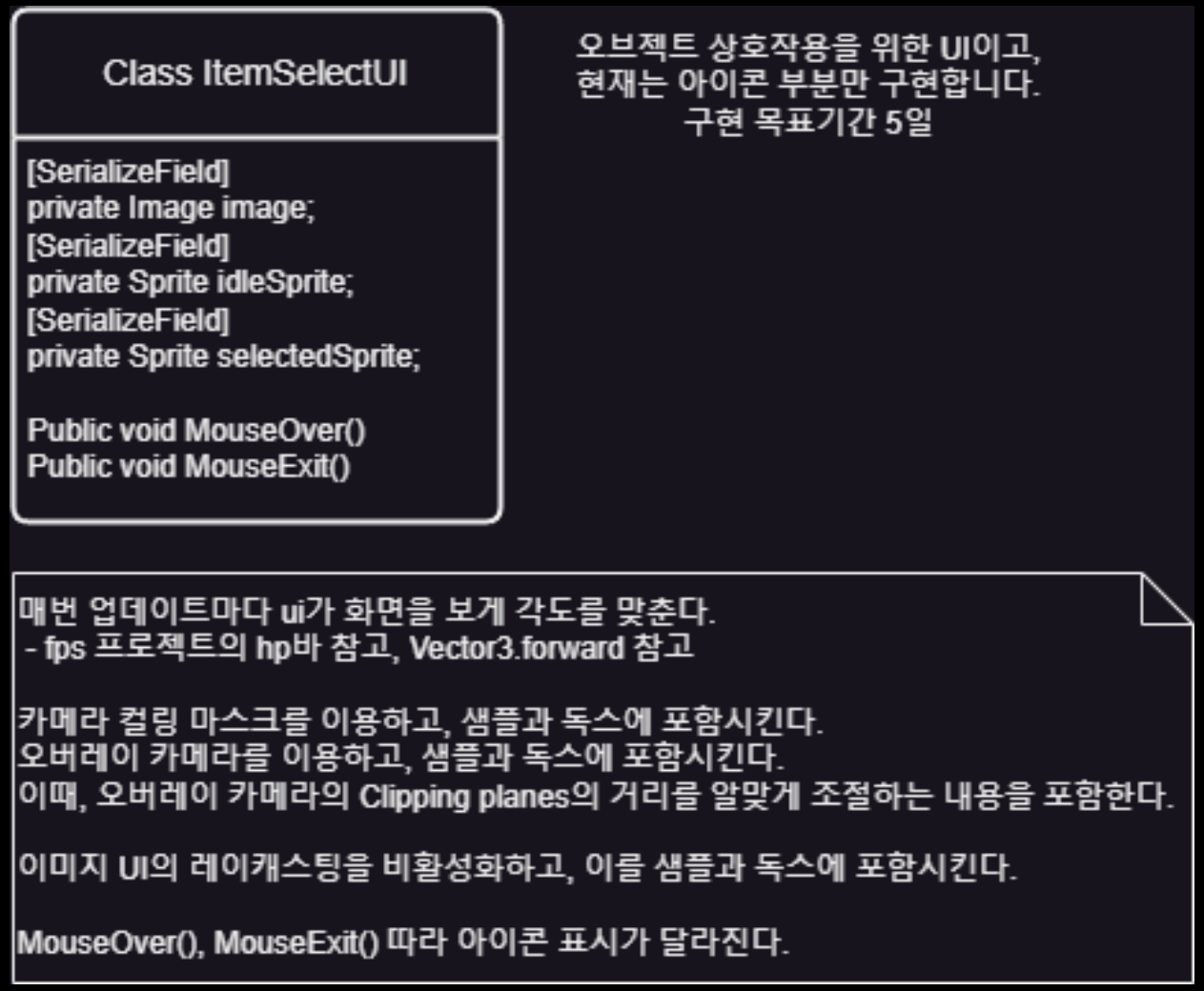
[왼쪽 마우스] 클릭 시 아래와 같이 오브젝트 상호작용.

플레이어가 상호작용이 가능한 오브젝트에 일정 거리 이내로 다가갈 경우 이와 같이 UI가 보여야 된다.

단, 초점이 오브젝트에 닿지 않았다면 눈 감은 이미지가 나타나고 닿았다면 눈 뜬 이미지가 보이게 한다.

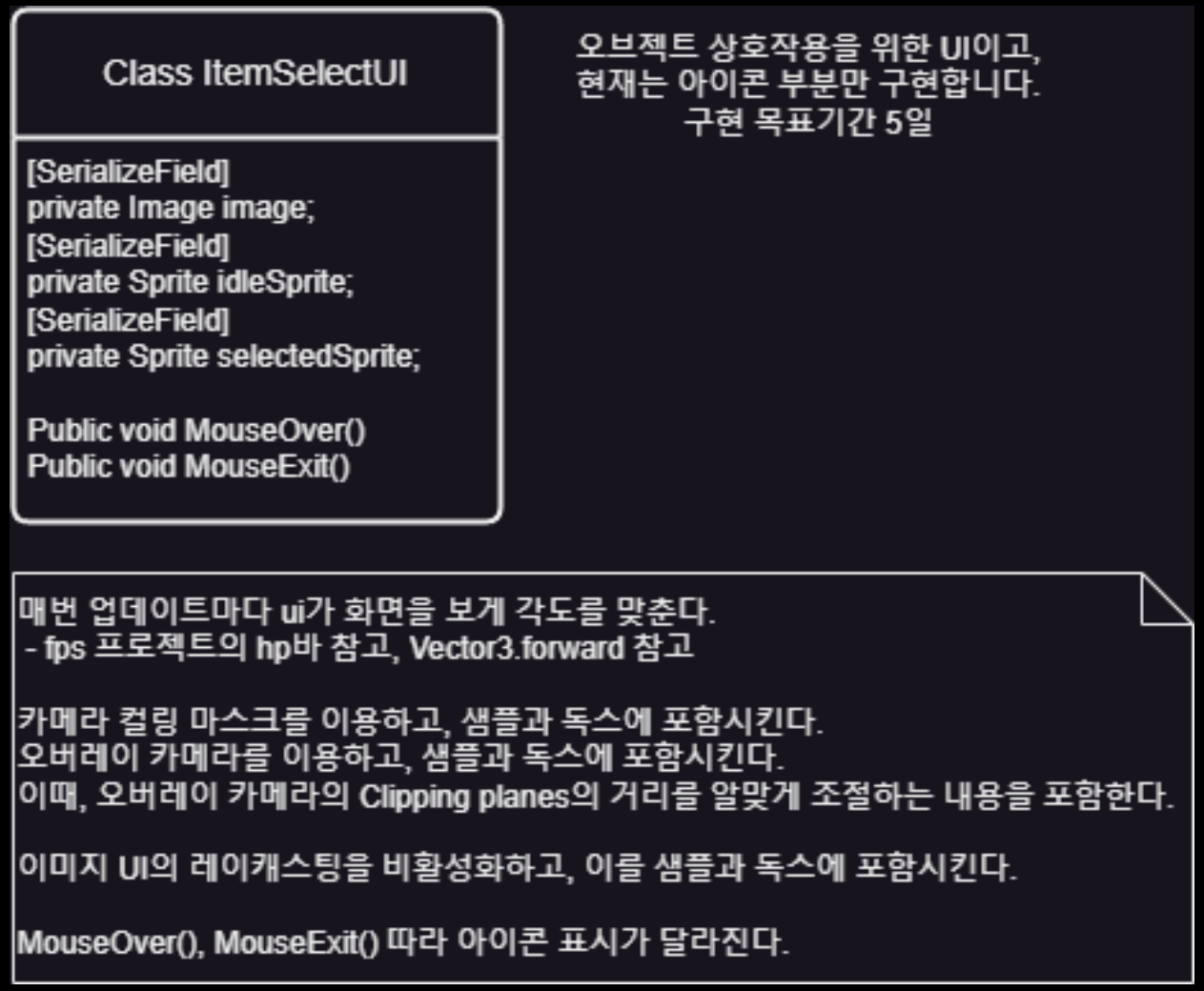


Mile Stone(Summary)



1. 눈 아이콘이 담길 UI는 오버레이로 들어가고, 플레이어 화면 각도에 따라 돌아야 한다.
2. UI 카메라를 추가하고, 카메라 컬링 마스크를 이용하여 메인 카메라에선 UI Layer를 제외한다.
   CullingMask : 원하는 Layer를 가진 Object들만 카메라로 볼 수 있게 하는 기능
3. ‘일정 거리 이내’로 들어온 것이 아니라면, 눈 아이콘이 보이지 않는다. 따라서 오버레이 카메라의 **Clipping Planes** 거리를 ‘일정 거리’로 설정한다.

Mile Stone(Summary)



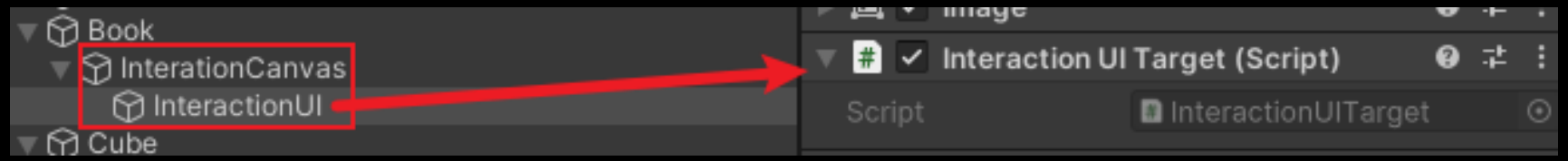
4. Raycast는 시선이 닿은 오브젝트를 찾을 때 쓰이는데, UI가 가리고 있음에도 Raycast가 동작한다면 기능이 어색할 수 있다. 따라서 UI를 감지하지 않도록 제한(비활성화) 한다.

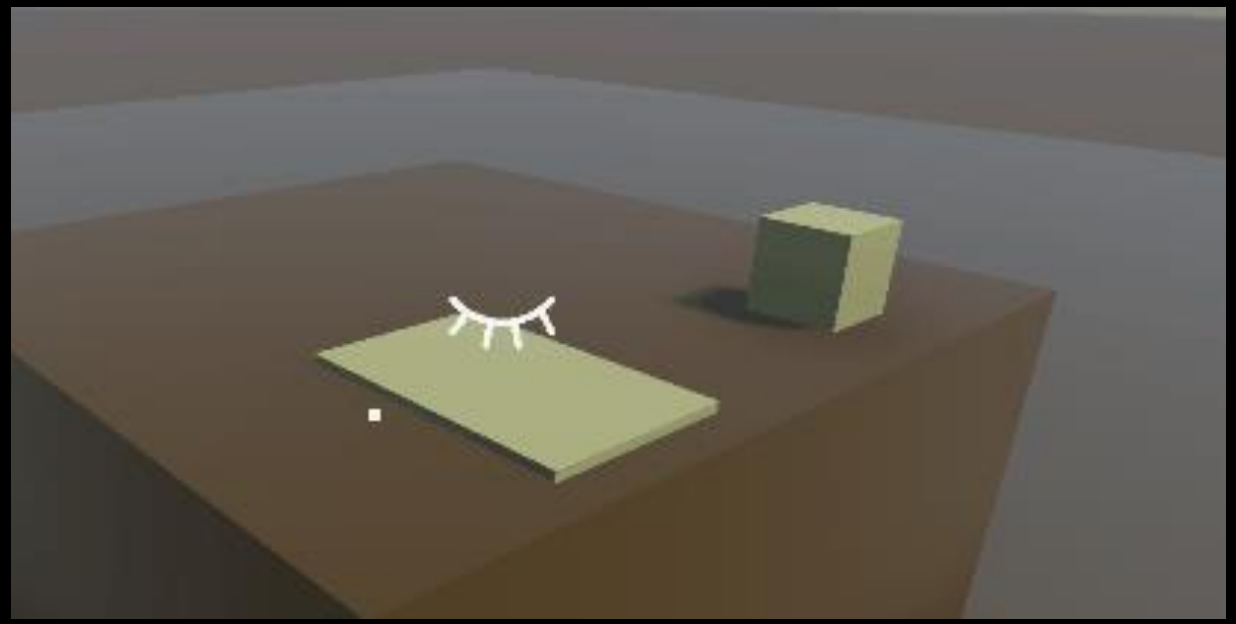
5. MouseOver() 시 눈 뜬 아이콘, MouseExit() 시 눈 감은 아이콘이 보이게 한다. (아이콘 두개)

Description

1. 화면 각도에 따라 UI 회전

**InteractionUITarget .cs (눈 이미지가 나타나는 'InteractionUI'에 넣는다)*





transform.forward = Camera.main.transform.forward;

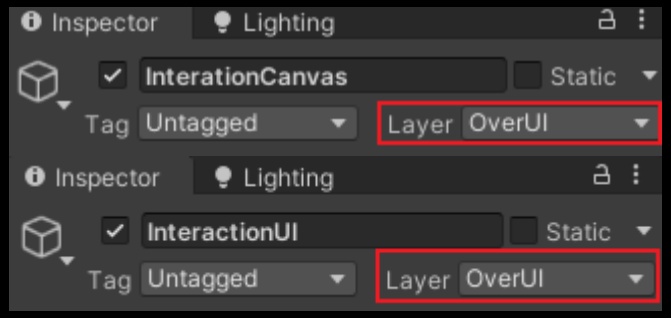
이와 같이 플레이어 화면에 따라
정면으로 보일 수 있게UI가 회전된다.

Description

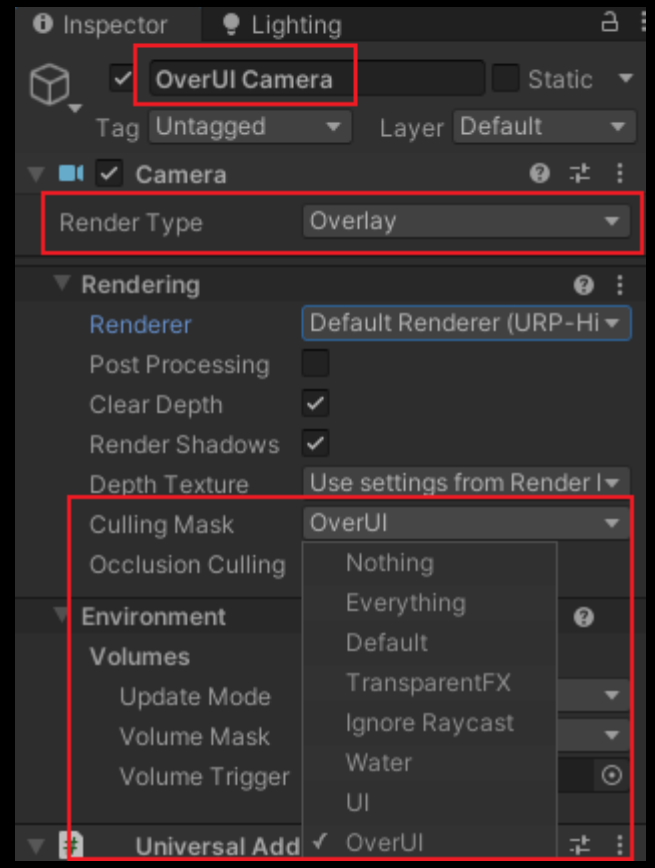
2-1. InteractionUI 전용 카메라 추가 및 설정

1. 카메라를 하나 더 만들고, Audio Listener를 제거
   (메인 카메라와 함께 움직이기 위해 메인 카메라의 자식으로 둔다)

2. Over UI 카메라의 Camera 컴포넌트 설정
   1) Layer에 OverUI 추가
      Interaction 캔버스, UI Image의 레이어를 OverUI로 설정



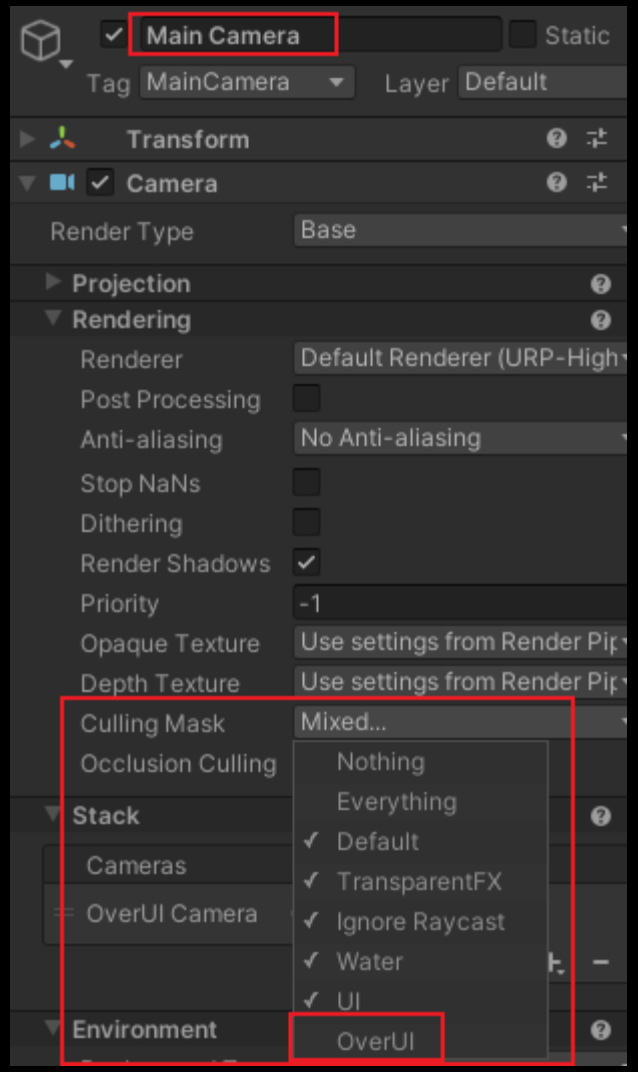
   2) OverUI 카메라 인스팩터 창에서 오른쪽과 같이 설정한다.
      - Render Type : `Overlay`
      - Culling Mask : `Over UI`



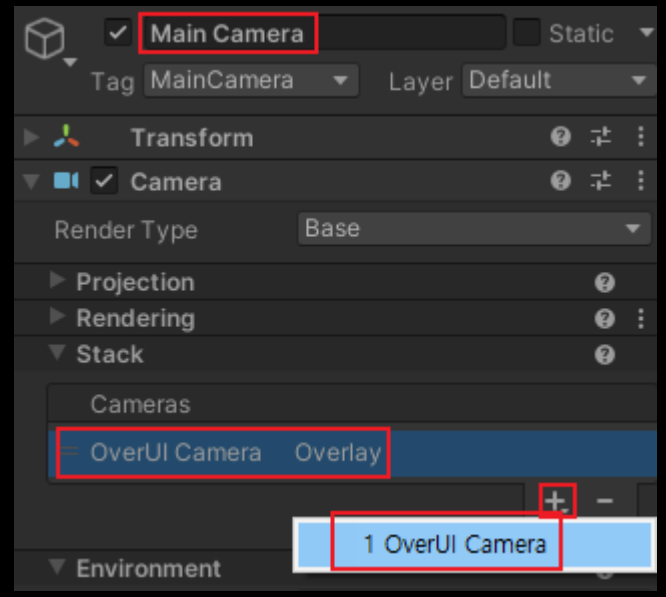
Description

2-2. 메인 카메라 설정

1. Culling Mask - Over UI만 제외



2. Stack - Over UI 카메라 추가



Description

2. 카메라 컴포넌트의 CullingMask

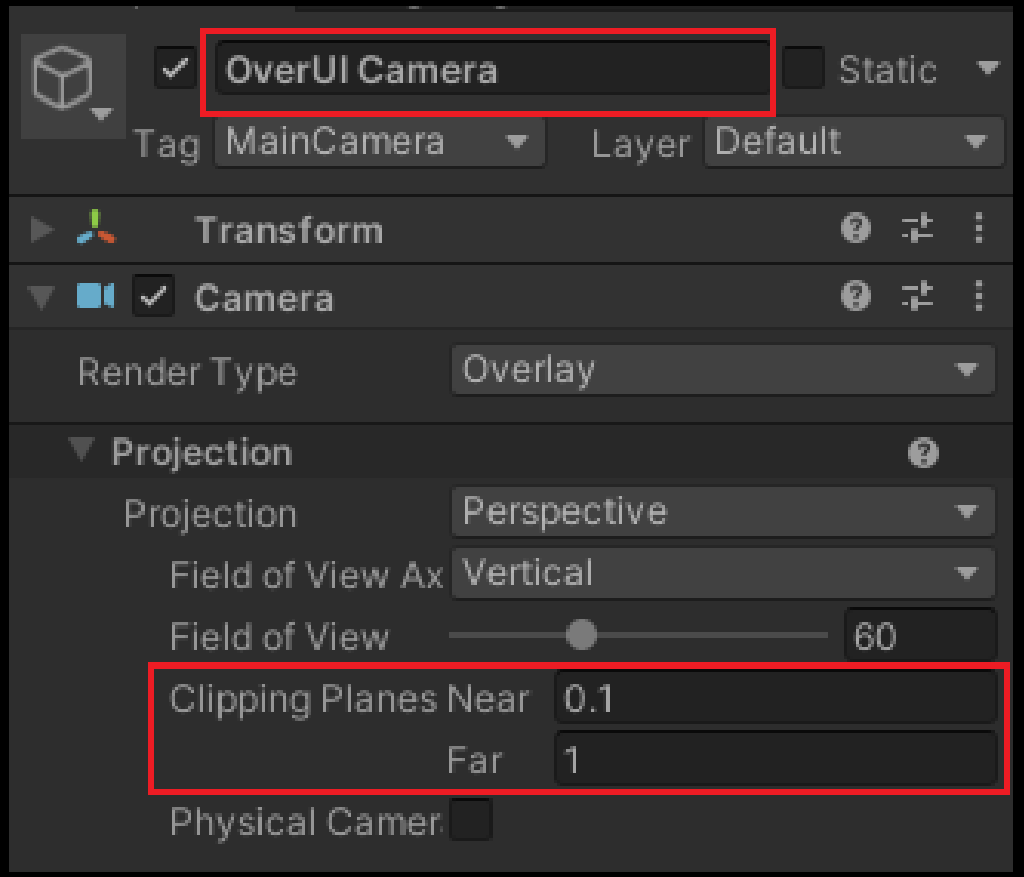
Culling Mask 결과

Main Camera

OverUI Camera

Description 3. 일정 거리 이내에 들어올 때만 보이게 하기

오버레이 카메라의 Clipping Planes 거리를 '일정 거리'로 설정한다.



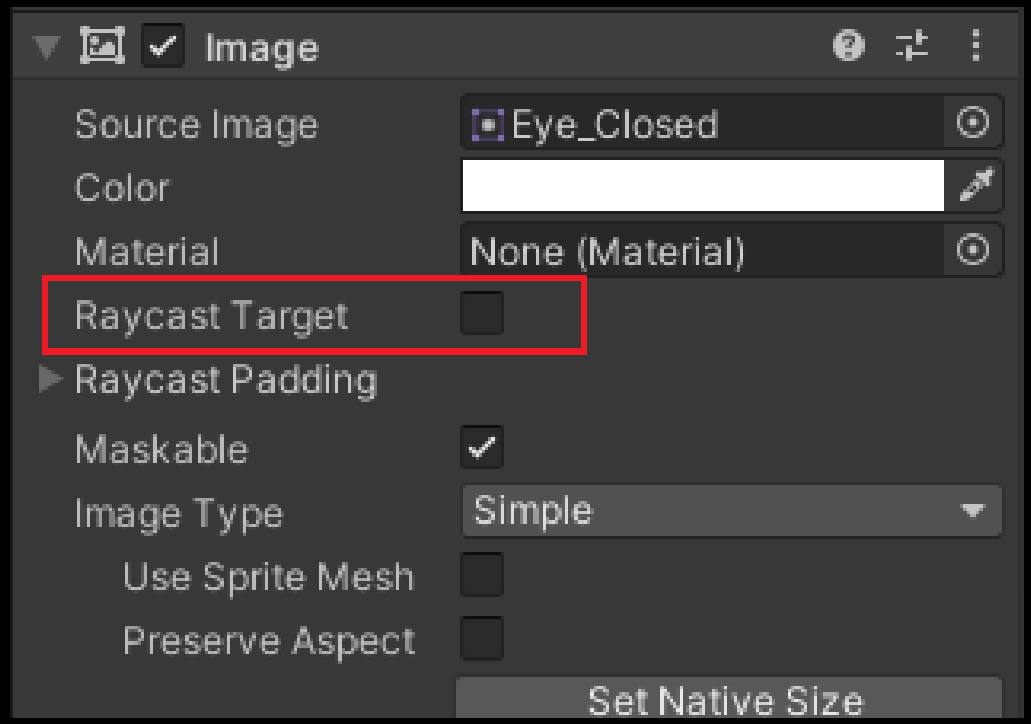
실제 게임에서 이 값을 조절하여 눈 이미지가 보이게 할 거리를 설정한다.

Description

4-1. UI는 Raycast에 감지되지 않게 하기

게임내 플레이어 초점에 상호작용 가능한 오브젝트가 닿았는 지 확인하기 위해 Ray를 사용

UI 이미지가 레이캐스트에 감지되지 않도록 체크 해제할 수 있다.



Description

4-2. 상호작용 가능한 오브젝트에 초점 닿은 여부 파악

**ItemSelectUI .cs (상호작용이 가능한 모든 오브젝트에 넣어야 한다.)*

Physics.Raycast() 메서드는 다양한 인자를 넣어 사용할 수 있다.

Physics.Raycast(ray, maxDistance, layerMask) 와 같이
Ray가 Casting될 최대 거리와 특정 Layer만 Casting되게 할 수 있다.
또한 Layermask 앞에 아래와 같이 ~를 붙여 해당 Layer만 제외할 수 있다.

```
// 부딪힌 오브젝트가 있다면
if (Physics.Raycast(ray, out hitInfo, rayMaxDistance, InteractionObj))
{
    ...
}
```

레이캐스팅된 오브젝트가 있다면 hitInfo에 저장된다

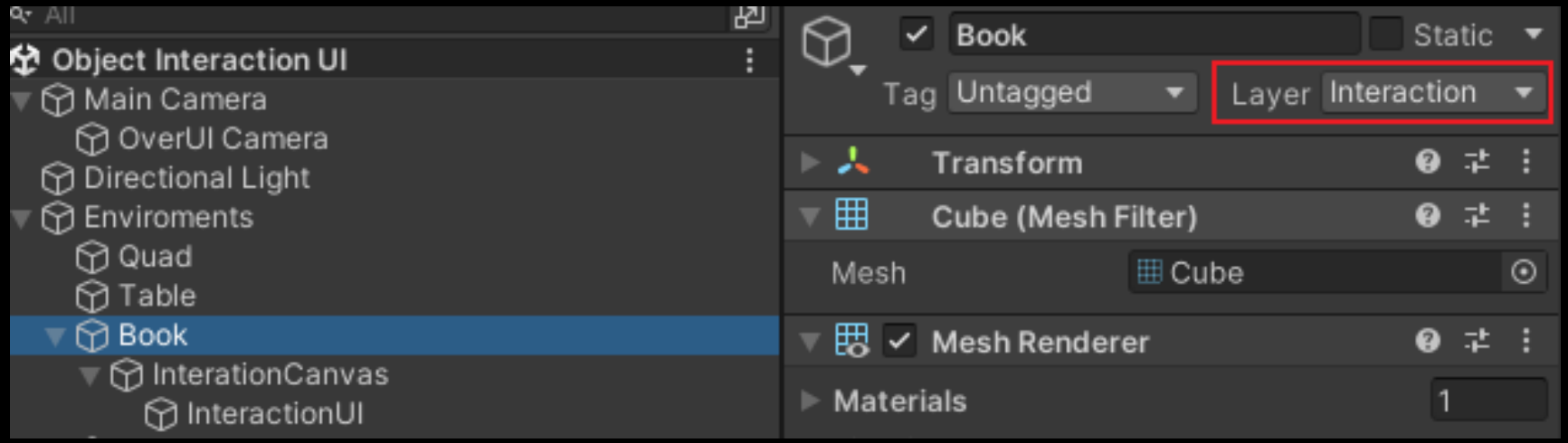
레이캐스팅 최대거리

이 레이어만 레이캐스팅 됨

Description

4-2. 상호작용 가능한 오브젝트에 초점 닿은 여부 파악

따라서 이와 같이 Interation Layer가 설정되었고,
이 레이어의 오브젝트만 레이캐스팅 되게 된다.



Description

4-2. 상호작용 가능한 오브젝트에 초점 닿은 여부 파악

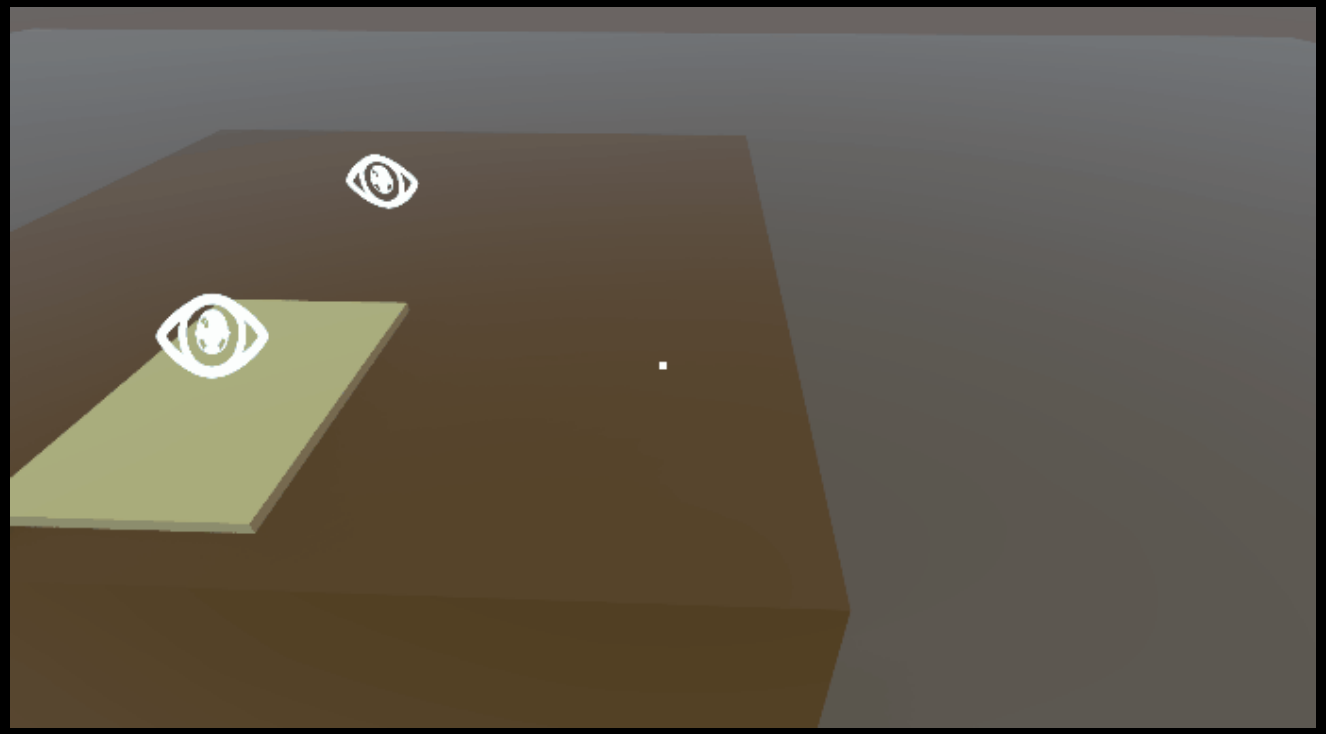
이 후, 상호작용 가능한 오브젝트에 레이캐스팅이 되었다면,
레이캐스팅 된(닿은) 오브젝트가 현재 스크립트의 게임 오브젝트가 맞는지 확인해야 한다.
(Interaction Layer인(상호작용 가능한) 다른 오브젝트 일 수 있으므로)

```
if (Physics.Raycast(ray, out hitInfo, rayMaxDistance, InteractionObj))
{
    // 부딪힌 물체가 현재 게임오브젝트(상호작용 가능한 오브젝트)라면
    if(hitInfo.collider.gameObject == gameObject)
    {
        // 현재 오브젝트는 눈 뜬 이미지로 표시
        OnMouseOver();
    }
    // 현재 오브젝트가 아니라면
    else
    {
        // 현재 오브젝트는 눈 감은 이미지로 표시
        OnMouseExit();
    }
}
```

Description

4-2. 상호작용 가능한 오브젝트에 초점 닿은 여부 파악

만약 이 과정을 생략하고 OnMouseOver() 함수만 넣는다면,
어떤 Interaction Layer를 가진 오브젝트이던 닿기만 하면
모든 상호작용 오브젝트의 이 스크립트에서 OnMouseOver() 를 실행하게 되므로,
아래와 같이 모두 눈 표시로 보이게 된다.



Description

4-2. 상호작용 가능한 오브젝트에 초점 닿은 여부 파악

마지막으로, 초점이 상호작용 가능한 오브젝트에 있다가 떠났을 경우에도
눈 감은 이미지로 바뀔 수 있도록 아래 코드가 추가되어야 한다.

```
if (Physics.Raycast(ray, out hitInfo, rayMaxDistance, InteractionObj))
{
    ...
}
// 애초에 상호작용 오브젝트에 닿지 않았다면
else
{
    // 현재 오브젝트는 눈 감은 이미지로 표시
    OnMouseExit();
}
```

Description

5. 마우스가 닿았을 때, 안 닿았을 때 처리

Interaction UI가 보이는 거리에 들어왔으나, 초점이 해당 오브젝트에 닿지 않은 경우에는
눈 감은 이미지가 보이고,
오브젝트에 초점이 닿은 경우는 눈 뜬 이미지가 보이게 한다.

```
public void OnMouseOver()
{
    // 현재 Image UI의 스프라이트 값을 눈 뜬 이미지로 바꾼다.
    image.sprite = selectedSprite;
}

public void OnMouseExit()
{
    // 현재 Image UI의 스프라이트 값을 눈 감은 이미지로 바꾼다.
    image.sprite = idleSprite;
}
```

오브젝트 상호작용 UI

-END-