가상현실 및 실습

사용자 인터페이스 및 레이캐스트 활용

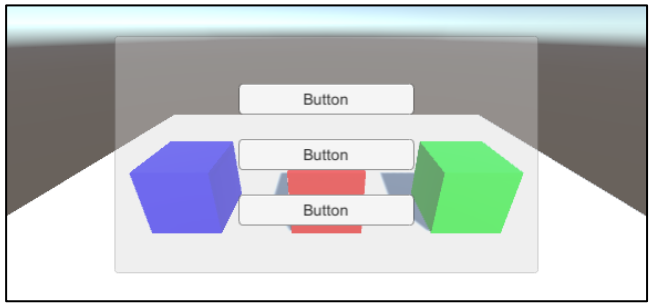
사용자 인터페이스 및 레이캐스트 활용

1) 사용자 인터페이스 (User Interface) 개념

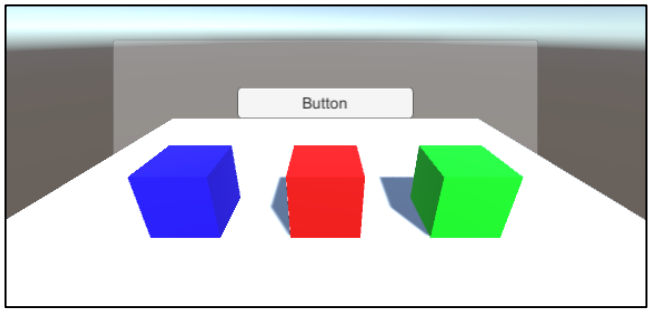
- 사람과 물체 또는 시스템 등이 서로 간에 통신하기 위해 사용되는 물리적/가상적 매체
- 즉, 사용자 인터페이스는 사람들이 컴퓨터와 상호작용하는 시스템
- 유니티에서는 **Canvas** 컴포넌트를 통해 사용자 인터페이스를 구현함

2) Canvas의 모드

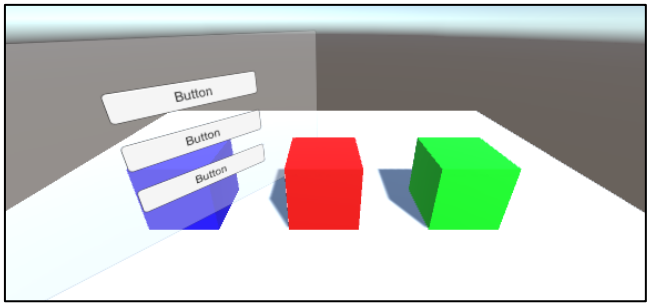
- Canvas 컴포넌트는 3가지 모드에 따라서 다른 특징을 가짐
  - ① Screen Space - Overlay
    - 3차원 오브젝트 위에 인터페이스를 렌더링하는 모드
  - ② Screen Space - Camera
    - 3차원 오브젝트와 인터페이스 간에 렌더링 순서를 정할 수 있는 모드
  - ③ World Space
    - 사용자 인터페이스가 2차원으로 취급되지 않고, 3차원 객체로써 적용되는 모드



Screen Space - Overlay



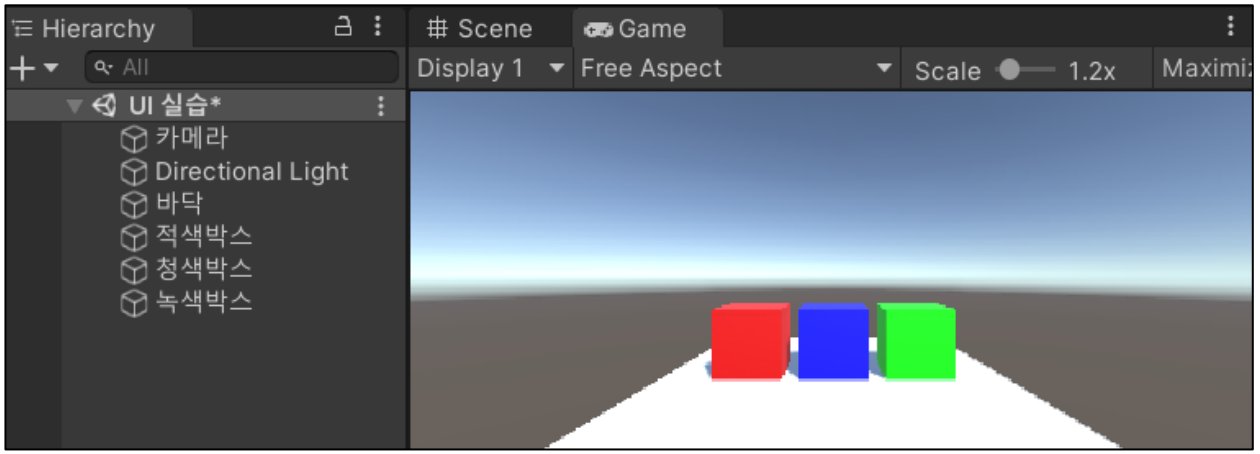
Screen Space - Camera



World Space

3) 오브젝트 배치

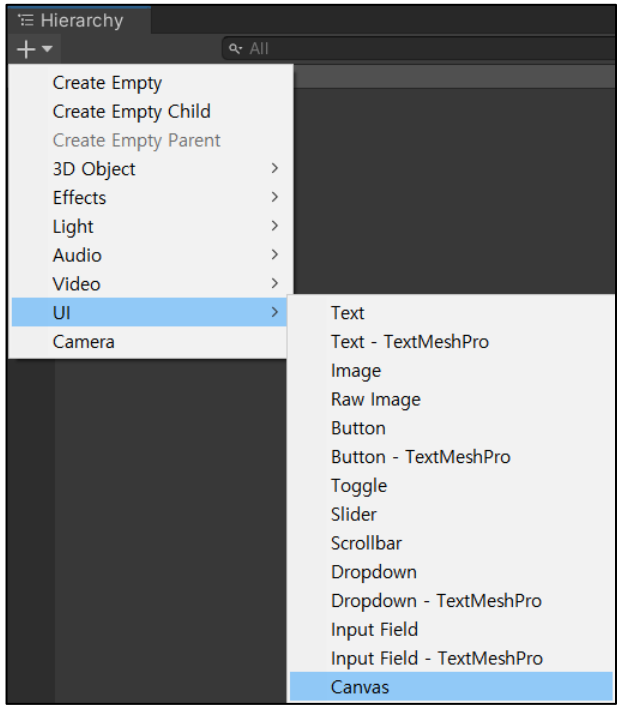
- 다음 표를 참고하여 게임 오브젝트를 생성/설정

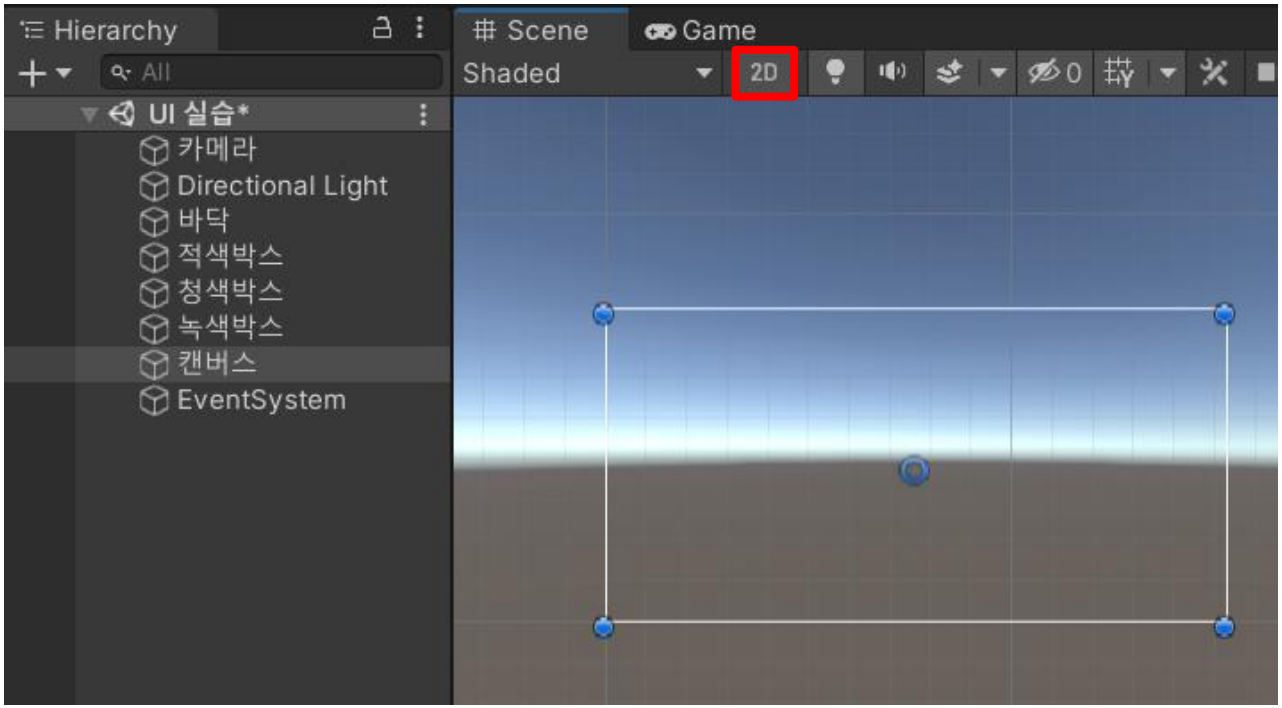


게임 오브젝트 이름	부모	Transform 속성			비고
		위치	회전	크기	
바닥	-	0, -2, 0	0, 0, 0	1, 1, 1	프리미티브 게임 오브젝트 – Plane 이용
적색 박스	-	-2.5, -1, 0	0, 0, 0	2, 2, 2	프리미티브 게임 오브젝트 – Cube 이용 재질을 이용하여 적색 색상 적용
청색 박스	-	0, -1, 0	0, 0, 0	2, 2, 2	프리미티브 게임 오브젝트 – Cube 이용 재질을 이용하여 청색 색상 적용
녹색 박스	-	2.5, -1, 0	0, 0, 0	2, 2, 2	프리미티브 게임 오브젝트 – Cube 이용 재질을 이용하여 녹색 색상 적용
카메라	-	0, 1. -10	0, 0, 0	1, 1, 1	Main Camera 게임 오브젝트

4) 사용자 인터페이스 설정

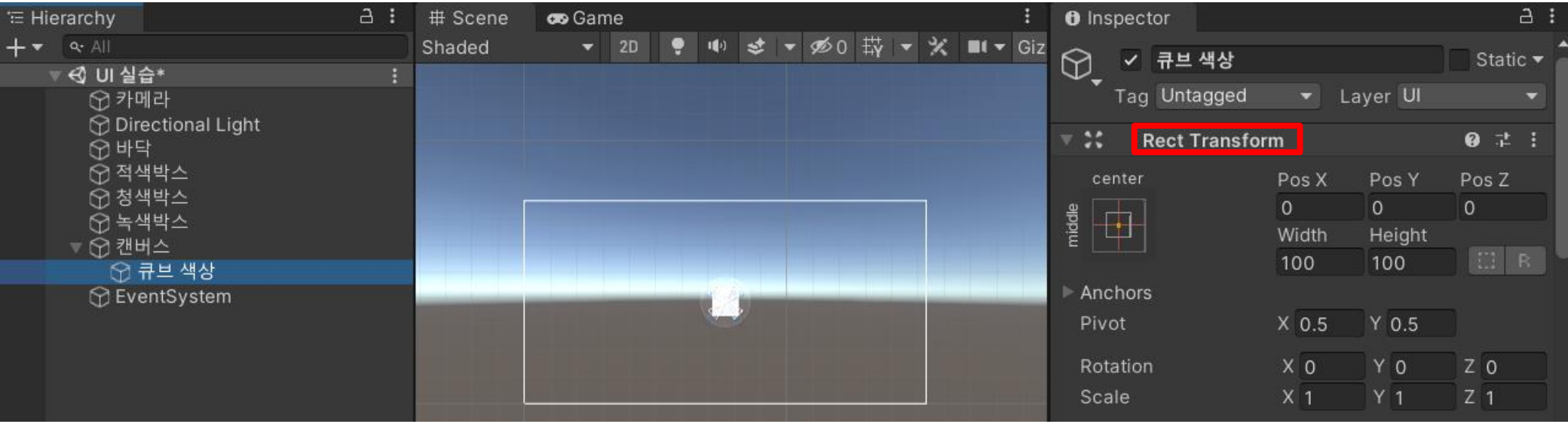
- 계층 창 > 마우스 우 클릭 > UI > Canvas를 선택하여 Canvas 게임 오브젝트를 생성
  생성된 Canvas 게임 오브젝트의 이름을 **캔버스**로 변경
- 인터페이스 작업을 쉽게 하기 위해서는 숫자 2 버튼을 선택하거나, 2D 버튼을 선택





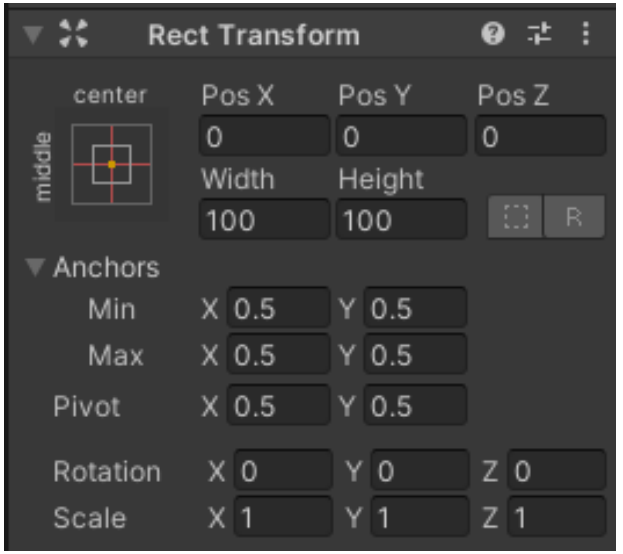
4) 사용자 인터페이스 설정

- **Canvas** 컴포넌트가 적용된 게임 오브젝트의 자식 게임 오브젝트에는 Transform 컴포넌트 대신 RectTransform 컴포넌트가 적용됨
- 계층 창 > **캔버스** 게임 오브젝트 선택 > 마우스 우 클릭 > UI > Image를 선택하여 **Image** 컴포넌트를 가지는 게임 오브젝트를 생성하고 이름을 **큐브 색상**으로 변경



5) RectTransform 개념

- 유니티의 모든 게임 오브젝트는 Transform 컴포넌트를 가지고 있음
- 유니티의 사용자 인터페이스를 구성하는 게임 오브젝트는 Transform을 상속 받은 Rect Transform 컴포넌트를 가지고 있음
- Rect Transform은 Transform 보다 2차원에서 조작이 보다 용이한 옵션을 가짐
  - ➢ Anchor Preset
  - ➢ Width, Height
  - ➢ Anchors
  - ➢ Pivot



사용자 인터페이스 및 레이캐스트 활용

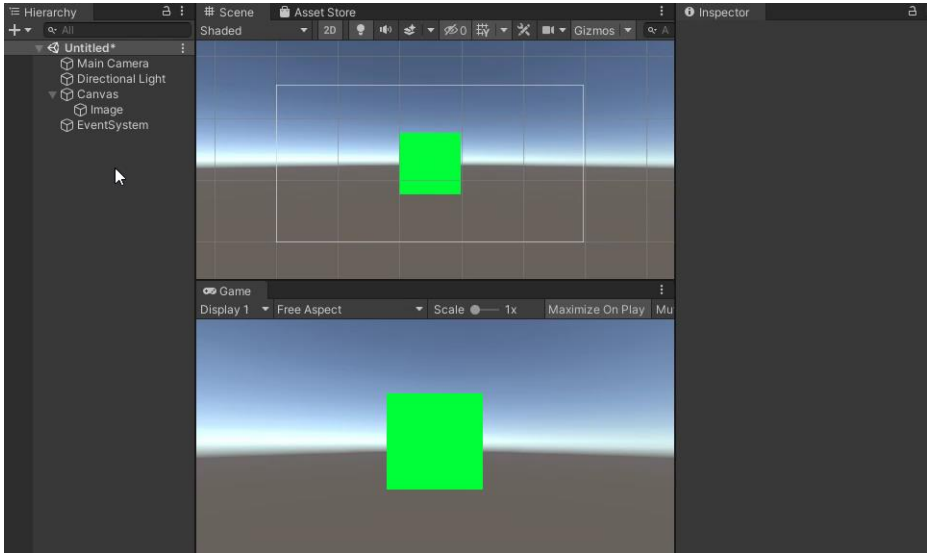
5) RectTransform 개념

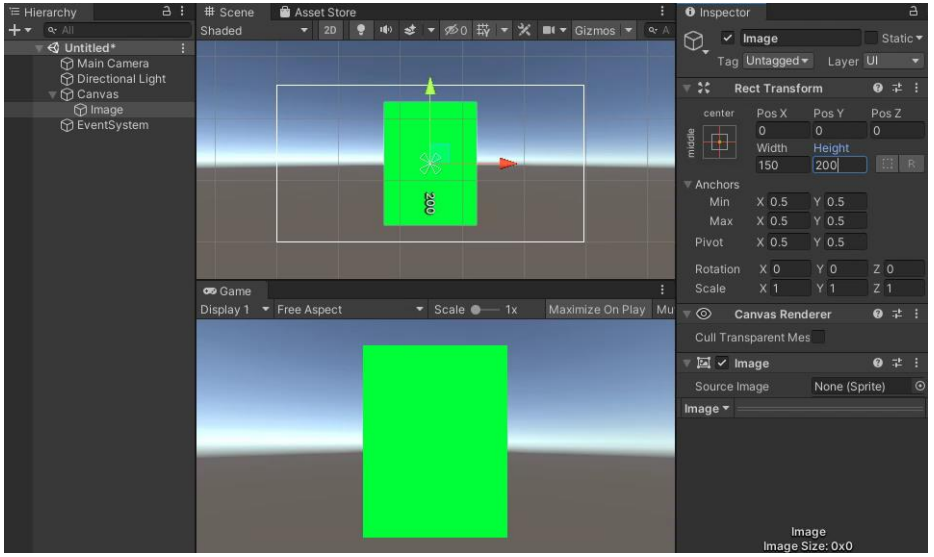
① Anchor Preset

- Anchor Preset은 사용자 인터페이스 구성 게임 오브젝트의 기준점을 정의하는 옵션
- 기본적으로는 상하 관계에서 중앙, 좌우 관계에서 중앙값을 가지고 있음

② Width, Height

- 사용자 인터페이스 구성 게임 오브젝트의 너비와 높이를 담당함

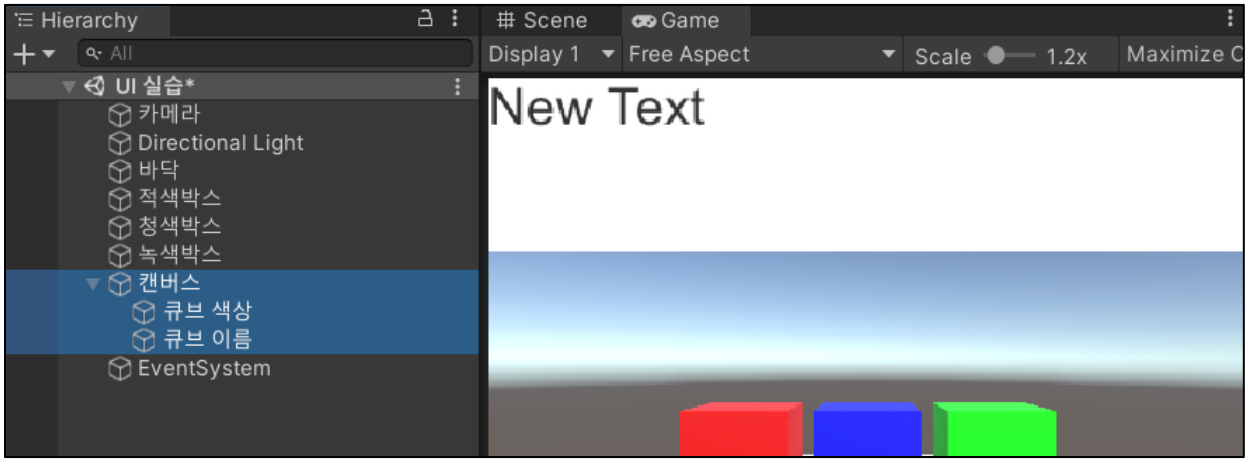


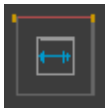


사용자 인터페이스 및 레이캐스트 활용

6) 사용자 인터페이스 배치

- 다음 표를 참고하여 게임 오브젝트를 생성/설정

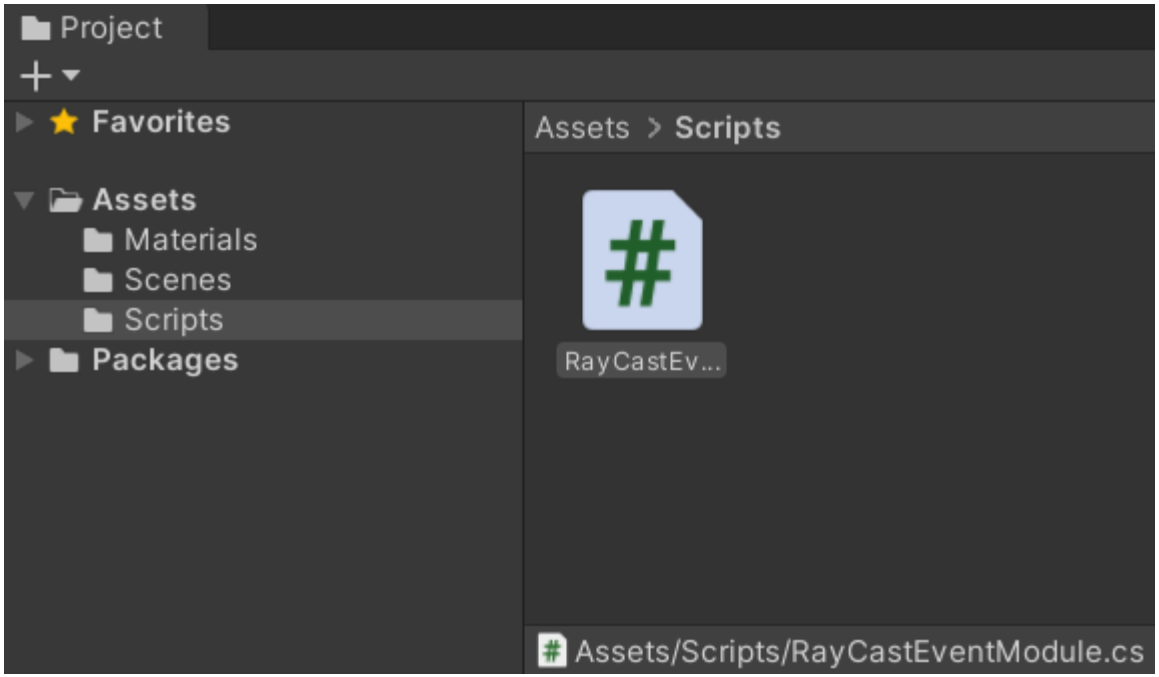


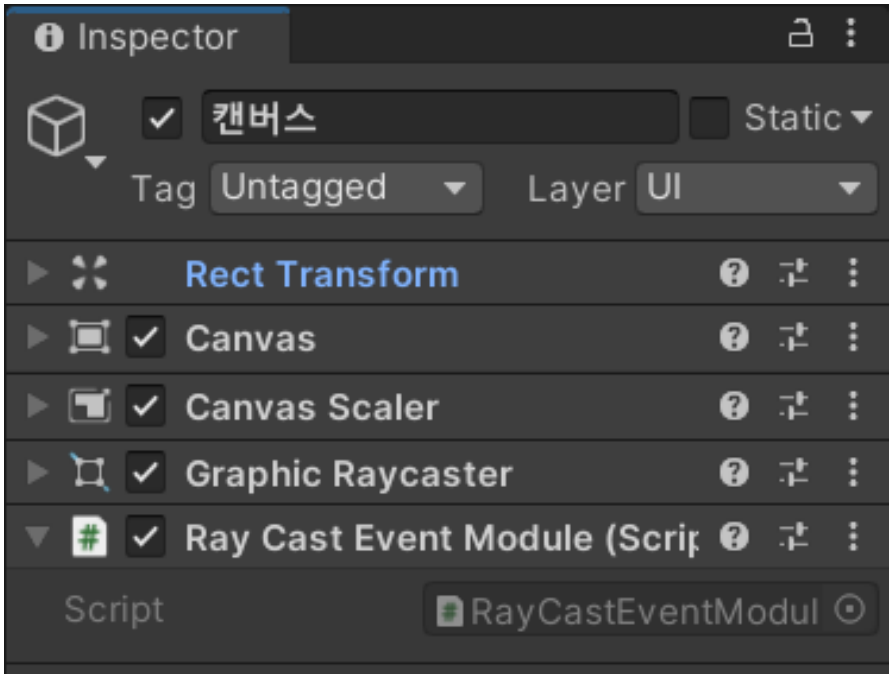
게임 오브젝트 이름	부모	RectTransform 속성	비고
큐브 색상	캔버스	Anchor Preset 모드:  Left: 0 Right: 0 Pos Y: -50 Height: 100	사용자 인터페이스 게임 오브젝트 - Image 이용
큐브 이름	캔버스	Anchor Preset 모드:  Pos X: 100 Pos Y: -40 Width: 200 Height: 80	사용자 인터페이스 게임 오브젝트 - Text 이용 Text 컴포넌트의 Font Size를 30으로 변경

사용자 인터페이스 및 레이캐스트 활용

7) 스크립트를 활용한 레이캐스트 이벤트 처리

- 새로운 스크립트 **RayCastEventModule** 생성 및 **캔버스** 게임 오브젝트에 적용
- **RayCastEventModule** 를 열어서 코드 편집기를 활성화





7) 스크립트를 활용한 레이캐스트 이벤트 처리

- 마우스로 색상 큐브를 선택하면 해당 큐브의 색상을 표현할 Image 멤버 변수(CubeColor)와, 선택한 큐브의 이름을 표현할 Text 멤버 변수(CubeName)을 생성함
- 사용자 인터페이스 관련 메서드/변수/자료형을 사용하기 위해서는 UnityEngine.UI 네임 스페이스를 추가해야 함

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

public class RayCastEventModule : MonoBehaviour
{
    public Text CubeName;
    public Image CubeColor;
}
```

7) 스크립트를 활용한 레이캐스트 이벤트 처리

- 레이캐스트 이벤트를 담당할 메서드를 생성(RayCastEvent)
- 다음과 같이 Update 메서드를 정의
  - bool Input.GetMouseButtonDown (int button) static 메서드:
    마우스 버튼이 눌러졌을 때 true, 그 외 false (0: 좌, 1: 우, 2: 휠)

```
public class RayCastEventModule : MonoBehaviour
{
    public Text CubeName;
    public Image CubeColor;
    private void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            RayCastEvent();

        }
    }
    private void RayCastEvent()
    {
    }
}
```

7) 스크립트를 활용한 레이캐스트 이벤트 처리

- 다음과 같이 RayCastEvent 메서드를 정의
  - Vector3 Input.mousePosition static 변수: 마우스의 현재 위치를 불러옴
  - Ray Camera.main.ScreenPointToRay (Vector3 pos) 메서드: pos 위치에서 화면에 직교한 방향으로 Ray를 생성
  - bool Physics.RayCast(Ray ray, RaycastHit hitInfo, float maxDistance) 메서드: maxDistance의 거리만큼 레이를 발사했을 때, 충돌체와 부딪히면 true, 그 외에는 false. 충돌 정보는 hitInfo에 저장
- 충돌 정보를 통해 사용자 인터페이스를 갱신할 메서드를 생성(UpdateUI)

```
private void RayCastEvent()
{
    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    if (Physics.Raycast(ray, out RaycastHit hitInfo, float.MaxValue))
    {
        UpdateUI(hitInfo);
    }
}
private void UpdateUI(RaycastHit hitInfo)
{

}
```

7) 스크립트를 활용한 레이캐스트 이벤트 처리

- 다음과 같이 UpdateUI 메서드를 정의
  - string Text.text 멤버 변수: 사용자 인터페이스 Text의 내용을 표현
  - Color Image.color 멤버 변수: 사용자 인터페이스 Image의 색상을 표현
  - Component.GetComponentInChildren<T>( ) 메서드: 자기 자신 및 자신의 자식 게임 오브젝트 중에서 T 컴포넌트를 불러옴. 없을 경우 null
  - MeshRenderer.material 멤버 변수: 그래픽 요소를 가지는 게임 오브젝트에 적용된 재질을 불러옴
  - Material.color 멤버 변수: 재질의 색상을 표현함

```
private void UpdateUI(RaycastHit hitInfo)
{
    Transform hittedObject = hitInfo.collider.transform;
    CubeName.text = hittedObject.name;
    CubeColor.color = hittedObject.GetComponentInChildren<MeshRenderer>().material.color;
}
```

사용자 인터페이스 및 레이캐스트 활용

7) 스크립트를 활용한 레이캐스트 이벤트 처리

- **캔버스** 게임 오브젝트의 **RayCastEventModule** 스크립트에서
  CubeName 변수에는 **큐브 이름** 게임 오브젝트를,
  CubeColor 변수에는 **큐브 색상** 게임 오브젝트를 설정