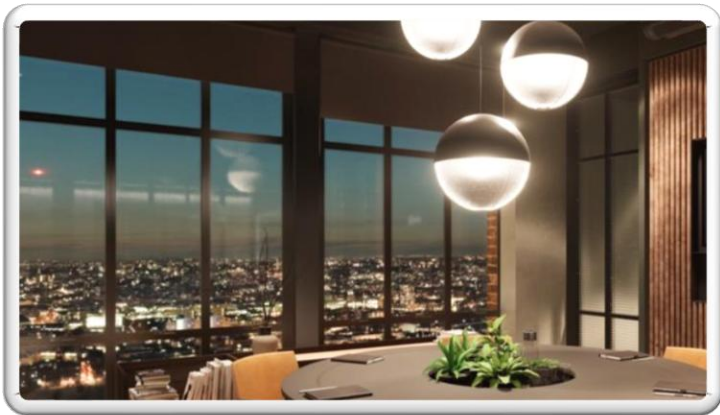




VR/AR 프로그래밍

- 3주차 -

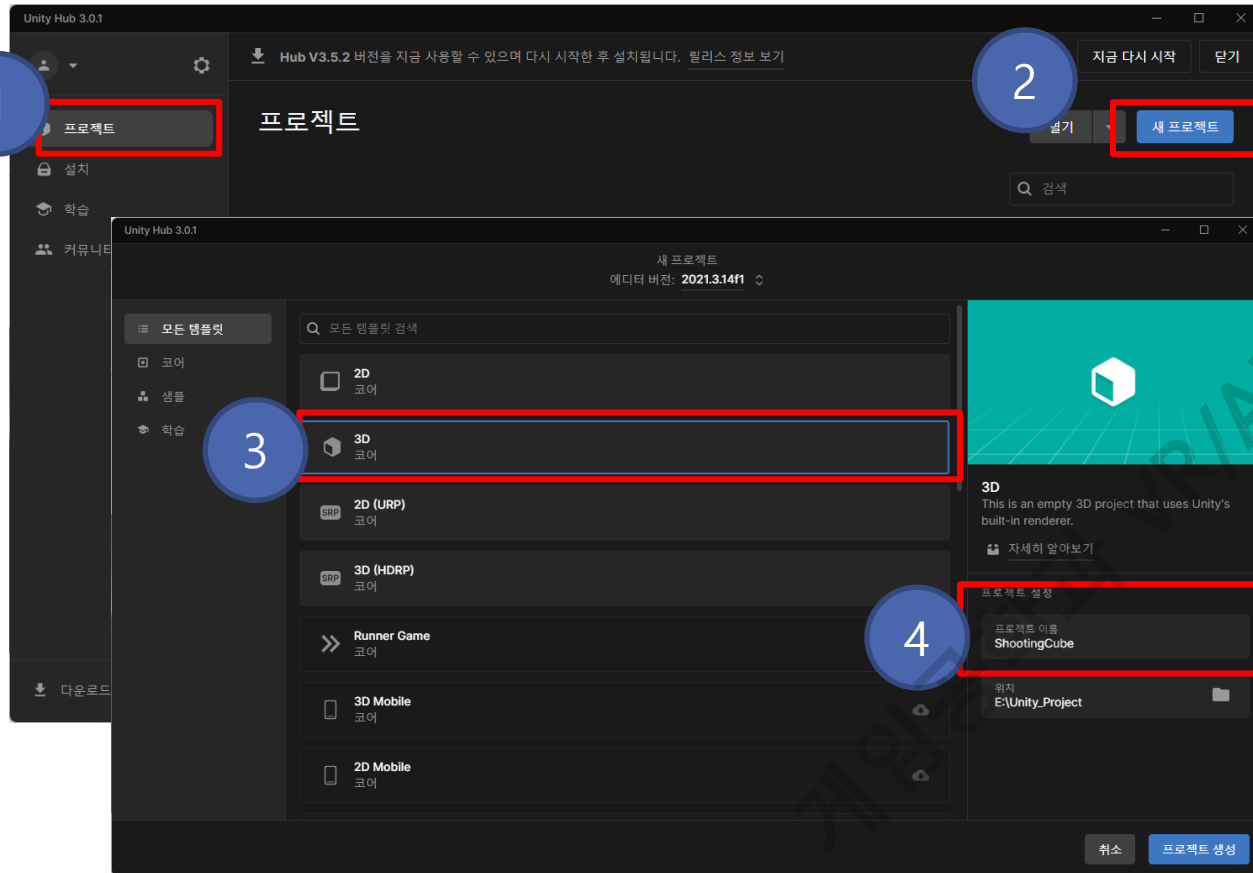
배재대학교 게임공학과



Shooting Cube (Meta Quest 2 버전)

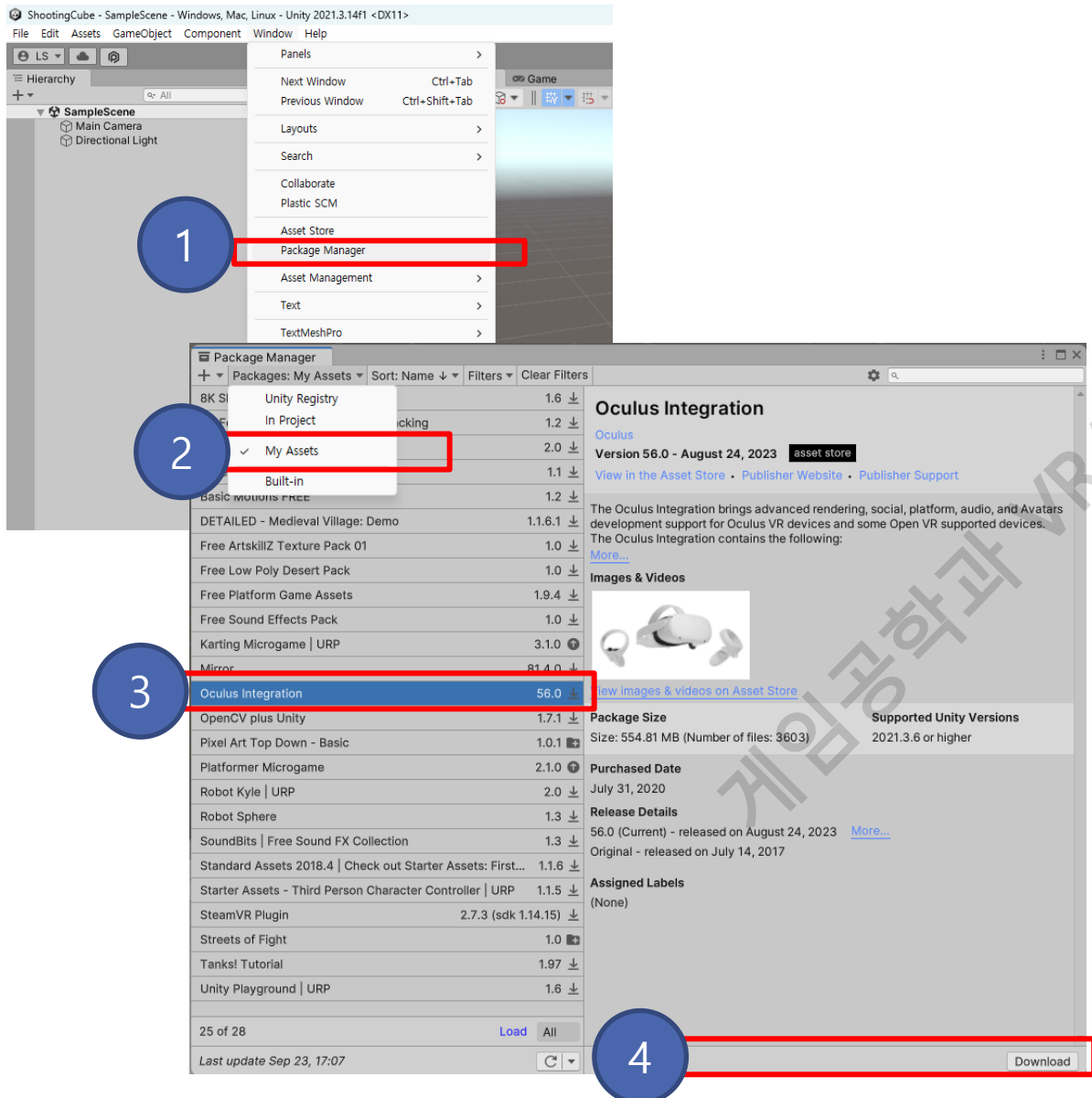
게임공학과 VR/AR 프로그래밍

프로젝트 생성



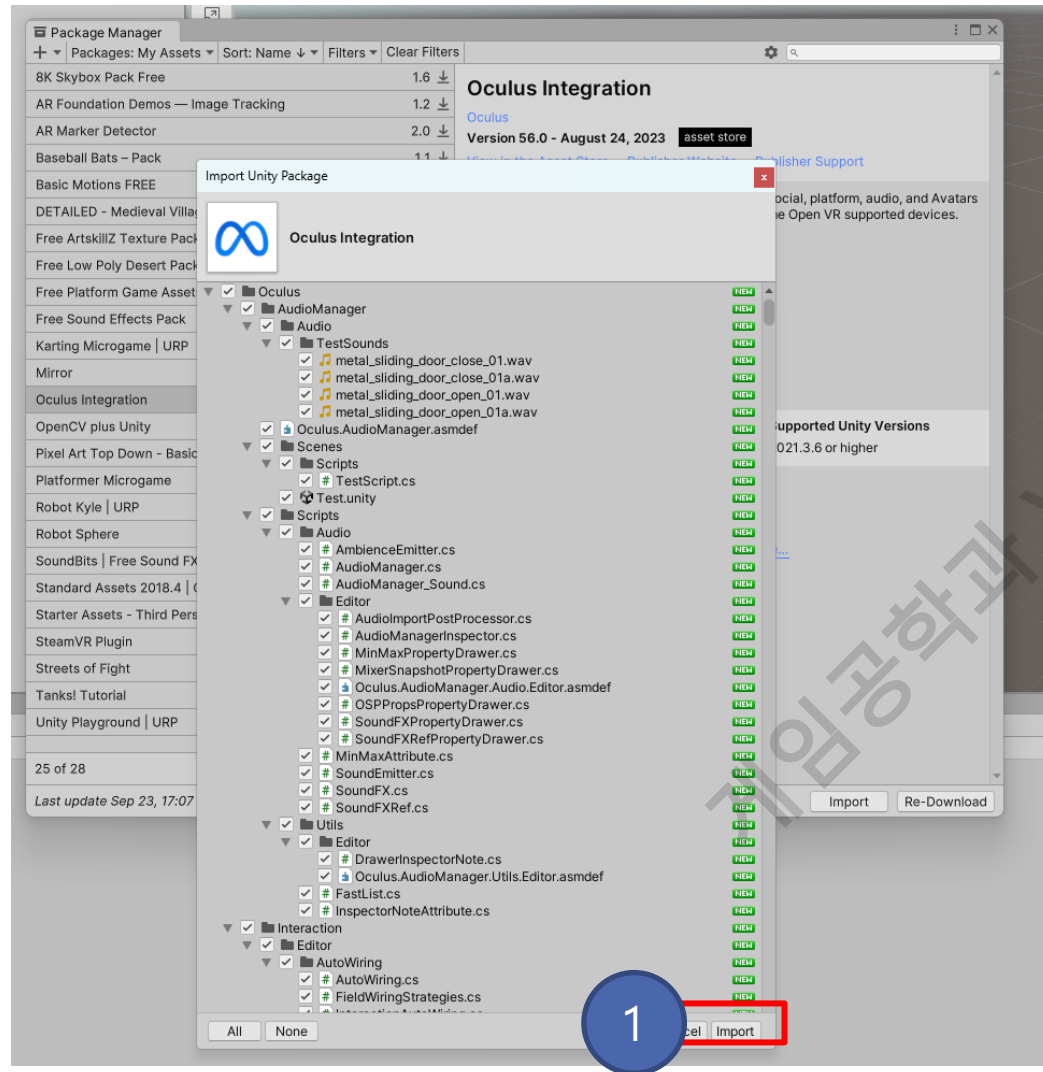
- ① 상단에 프로젝트 선택
- ② 오른쪽 상단에 새로 생성 선택
(2022.3.xxf 이상 버전)
- ③ 3D 체크 확인
- ④ 설정에 [프로젝트 이름] 지어주기
 - * 저장 위치에 **한글 폴더**는 없어야 함

프로젝트 설정

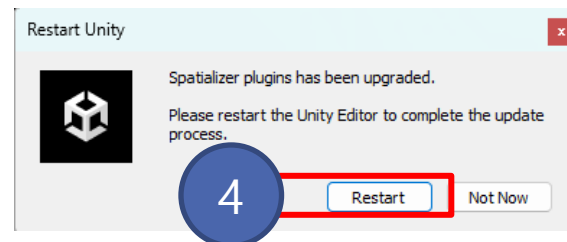
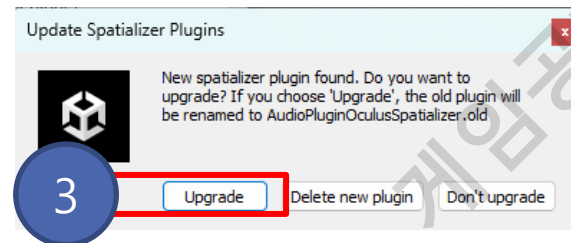
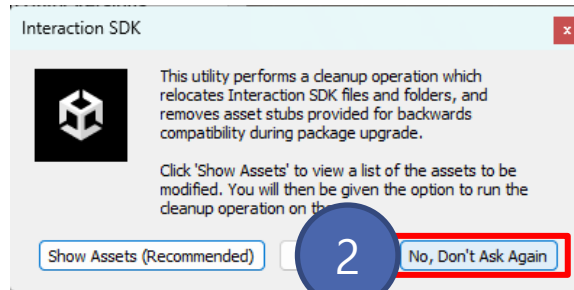
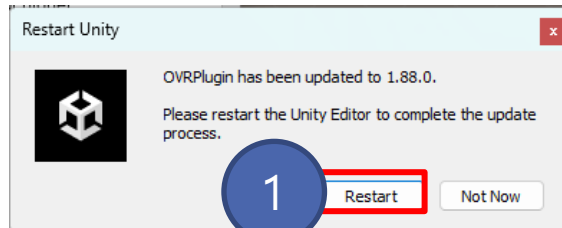


- ① Window -> Package Manager 클릭
- ② Packages 에서 My Assets 선택
- ③ Oculus Integration 선택 후
- ④ 다운로드 후 import를 눌러서 추가
- 또는 import를 눌러서 추가

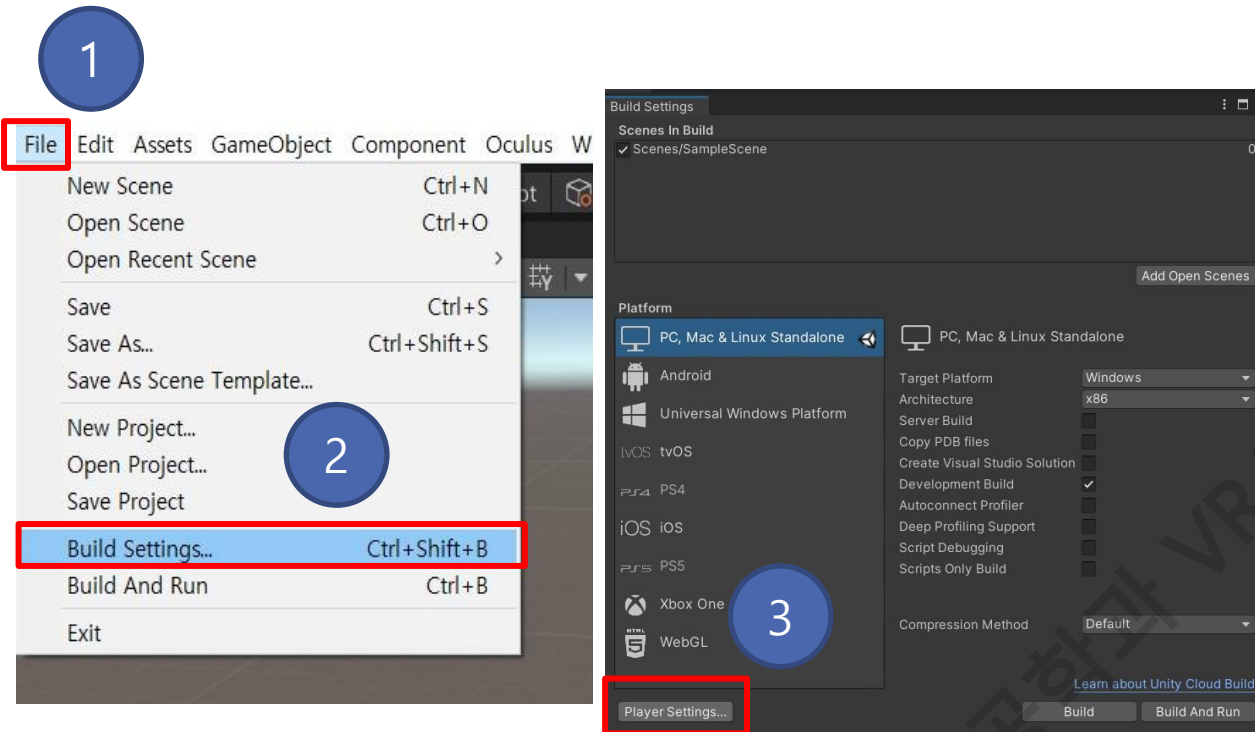
My Assets에 Oculus Integration이 없는 경우에는
2주차 수업을 진행하지 않았으므로
수업자료 참고해서 프로젝트 진행
(본인 계정으로 에셋 스토어에서 다운로드 받아야 함)



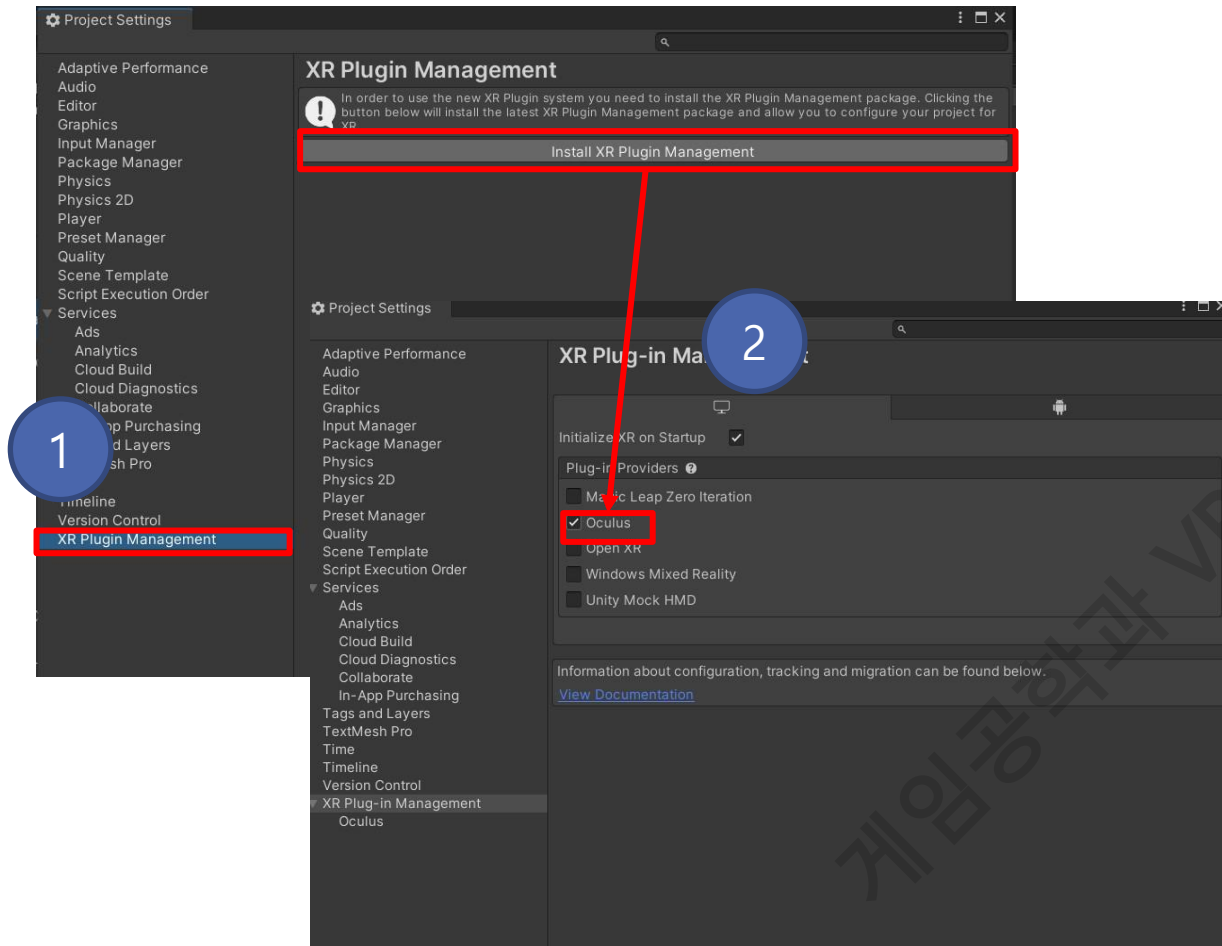
- ① Oculus Integration Import 진행



- ① Oculus Integration Import 진행

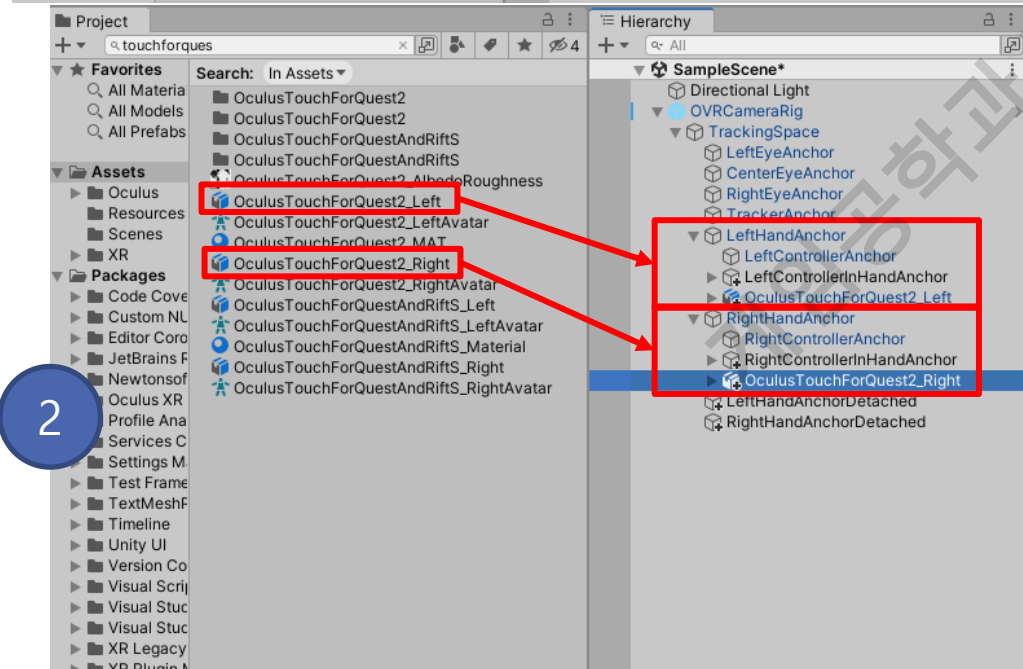
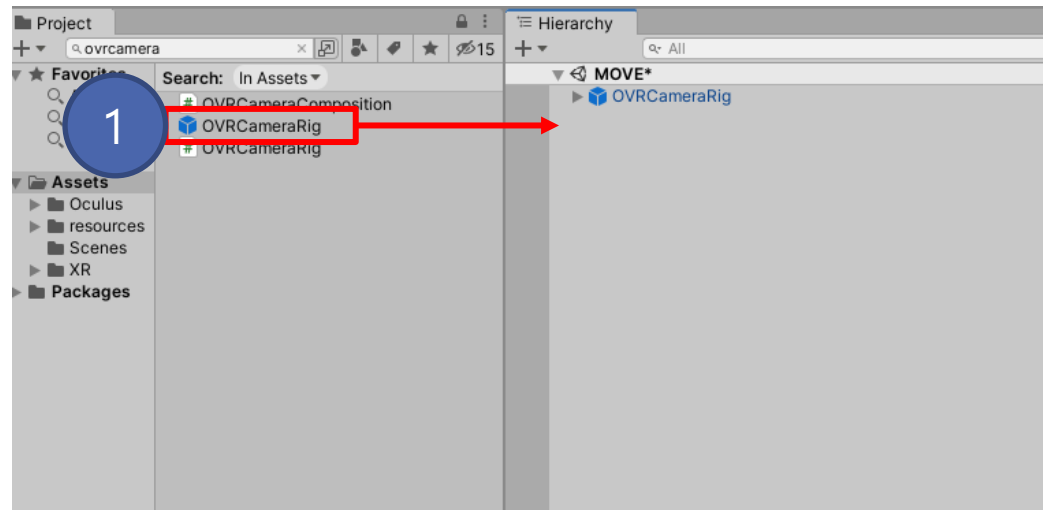


- ① 패키지를 Import
- ② File -> Build Settings... 클릭
- ③ 좌측 하단의 Player Setting 클릭



- ① XR Plugin Management 클릭 후, Install을 클릭하여 설치합니다.
- ② 설치가 끝난 후 나온 창의 Oculus 클릭

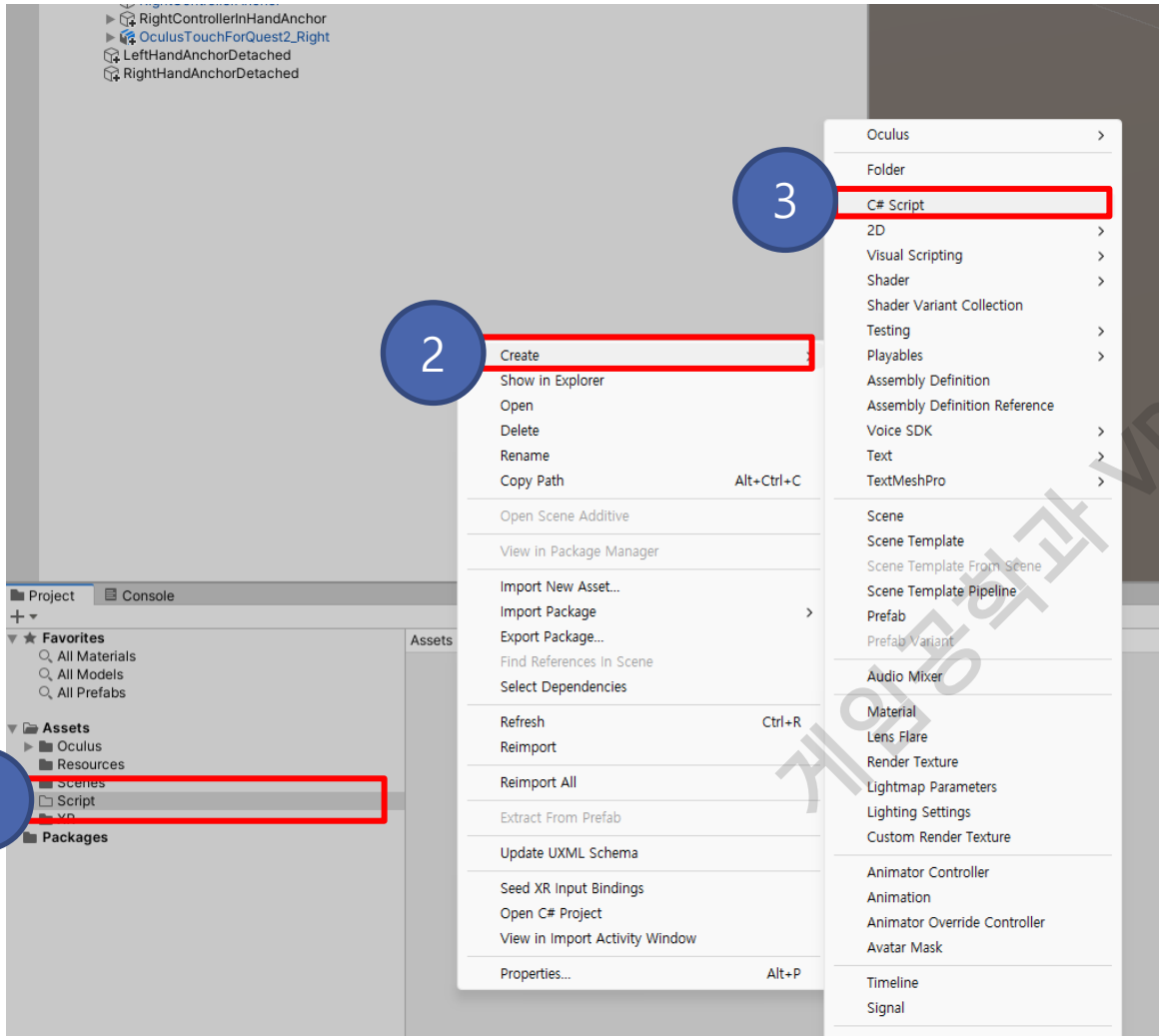
기본 오브젝트 생성



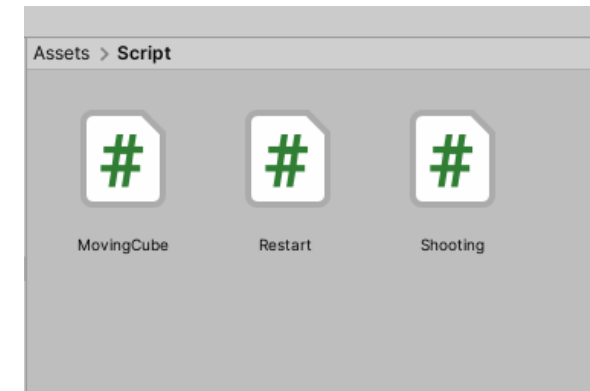
- ① Main Camera 삭제
- Project 창에서 [OVRcamerarig] 검색 후 Hierarchy 창에 추가
- ② Project창에 [Quest] 검색 후, OVRCameraRig를 펼친 후, Left(Right)HandAnchor 하위에 추가



- ① 3주차 수업자료에 있는
"Base_Resources.unitypackage" Import



- ① Assets 폴더에 Script 폴더를 추가한다
- ② Script 폴더에서 Create 메뉴 선택
- ③ C# Script를 선택하여 스크립트를 생성한다.
- 생성하는 스크립트명 :
 - Shooting, Restart, MovingCube



Shooting 스크립트 작성

```
MovingCube.cs Restart.cs Shooting.cs -x
Assembly-CSharp - Shooting

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 // Unity 스크립트 | 참조 4개
6 public class Shooting : MonoBehaviour
7 {
8     // 게임 시작 여부 확인
9     public bool _isGameOver = false;
10    // 오디오 클립 관리 변수
11    public AudioClip[] _acSoundPlay;
12    // 폭발효과
13    public GameObject _gbBombEffect;
14
15    // 게임 종료 확인
16    private bool _isFinish = false;
17    // UI 오브젝트
18    public GameObject _gbCanvasObject;
19    // 선택 게임 오브젝트
20    private GameObject _gbHit = null;
21    // 전방 물체 감지용 레이캐스트
22    private RaycastHit _rhHit;
23    // 새로 생성되는 오브젝트
24    public GameObject _gbSpawnObject;
25
26    // Start is called before the first frame update
27    // Unity 메시지 | 참조 0개
28    void Start()
29    {
30        SpawnObjectFunction(1, new Vector3(0, 0, 0));
```

```
32 // Unity 메시지 | 참조 0개
33 private void FixedUpdate()
34 {
35     // 플레이어가 바라보는 방향으로 50만큼 레이저를 쏘서 충돌하는 오브젝트 측정
36     if(Physics.Raycast(transform.position, transform.forward, out _rhHit, 50))
37     {
38         // 만약 빈 오브젝트라면
39         if(_gbHit == null)
40         {
41             // 선택되지 않음
42             SetValue(true);
43         }
44     }
45     else
46     {
47         if(_gbHit != null)
48         {
49             // 선택됨.
50             SetValue(false);
51         }
52     }
53
54     if (_isFinish)
55     {
56         //게임의 설정을 끝으로 지정
57         GameObject[] gbGameObj = GameObject.FindGameObjectsWithTag("Finish");
58         //게임 오브젝트의 갯수가 0개인 경우
59         if (gbGameObj.Length == 0)
60         {
61             // 게임 종료
62             _isGameOver = true;
63         }
64     }
65
66     // 게임 종료시
67     if(_isGameOver)
68     {
69         // UI 화면을 표시함
70         _gbCanvasObject.SetActive(true);
71     }
72     else
73     {
74         _gbCanvasObject.SetActive(false);
75     }
```

Shooting 스크립트 작성

```
참조 1개
77 public void SpawnFunction(float fScale, Vector3 transformObj)
78 {
79     // 선택된 오브젝트의 크기가 0.8보다 크다면
80     if(fScale > 0.8f)
81     {
82         for(int i = 0; i < 4; i++)
83         {
84             // 다음 생성되는 오브젝트의 크기를 0.75로 변경
85             SpawnObjectFunction(0.75f, transformObj);
86         }
87     }
88     else if(fScale > 0.6f)// 선택된 오브젝트의 크기가 0.6보다 크다면
89     {
90         for (int i = 0; i < 4; i++)
91         {
92             // 다음 생성되는 오브젝트의 크기를 0.75로 변경
93             SpawnObjectFunction(0.5f, transformObj);
94
95             // 마지막으로 생성되는 오브젝트 확인
96             _isFinish = true;
97         }
98     }
99 }
100
참조 3개
101 public void SpawnObjectFunction(float fScale, Vector3 transformObj)
102 {
103     // 오브젝트를 생성하고 세부사항을 입력해주기 위한 참조
104     GameObject gbChildObject = Instantiate(_gbSpawnObject);
105     // 생성될 오브젝트의 위치 초기화
106     gbChildObject.transform.position = transformObj;
107     gbChildObject.transform.localScale = new Vector3(fScale, fScale, fScale);
108     gbChildObject.AddComponent<MovingCube>();
109     gbChildObject.GetComponent<MovingCube>()._gbParentObject = this.gameObject;
110     gbChildObject.GetComponent<MovingCube>()._gbBombEffect = _gbBombEffect;
111     gbChildObject.AddComponent<MeshCollider>();
112     gbChildObject.GetComponent<MeshCollider>().convex = true;
113     gbChildObject.AddComponent<Rigidbody>();
114     gbChildObject.GetComponent<Rigidbody>().useGravity = false;
115     gbChildObject.GetComponent<Rigidbody>().drag = 1;
116
117     if(fScale < 0.6f)
118     {
119         gbChildObject.tag = "Finish";
120     }
121 }
122
```

```
참조 2개
123 private void SettingValue(bool isOnOff)
124 {
125     if(isOnOff)
126     {
127         //선택된 오브젝트 재선언
128         _gbHit = _rhHit.transform.gameObject;
129         //게임 오브젝트 선택
130         if(_gbHit.GetComponent<MovingCube>())
131         {
132             // 상태 변경
133             _gbHit.GetComponent<MovingCube>().isHit = true;
134         }
135
136         // UI 화면이 선택
137         if (_gbHit.GetComponent<Restart>())
138         {
139             // 상태 변경
140             _gbHit.GetComponent<Restart>().isSelect = true;
141         }
142         // 선택된 경우 색상을 파란색으로 변경
143         _gbHit.GetComponent<MeshRenderer>().material.color = Color.blue;
144     }
145     else
146     {
147
148         //게임 오브젝트 선택
149         if (_gbHit.GetComponent<MovingCube>())
150         {
151             // 상태 변경
152             _gbHit.GetComponent<MovingCube>().isHit = false;
153         }
154
155         // UI 화면이 선택
156         if (_gbHit.GetComponent<Restart>())
157         {
158             // 상태 변경
159             _gbHit.GetComponent<Restart>().isSelect = false;
160         }
161         // 선택이 해제된 경우 색상을 흰색으로 변경
162         _gbHit.GetComponent<MeshRenderer>().material.color = Color.white;
163
164         _gbHit = null;
165     }
166 }
167
168
169
170
```

Restart 스크립트 작성

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  Unity 스크립트 | 참조 4개
7  public class Restart : MonoBehaviour
8  {
9      public bool isSelect = false;
10     private float fDelTime = 0.5f; // 삭제 지연시간
11
12     // Update is called once per frame
13     Unity 메시지 | 참조 0개
14     void Update()
15     {
16         if(isSelect) // UI 화면을 선택중이라면
17         {
18             fDelTime -= Time.deltaTime; // 시간을 줄임
19         }
20         else
21         {
22             fDelTime = 0.5f; //초기화
23         }
24
25         if(fDelTime < 0.0f) // 시간이 0.5 초 지나면
26         {
27             RestartScene(); // 시나리오 재시작
28         }
29     }
30
31     참조 1개
32     private void RestartScene()
33     {
34         SceneManager.LoadScene(0);
35     }
36 }
```

MovingCube 스크립트 작성

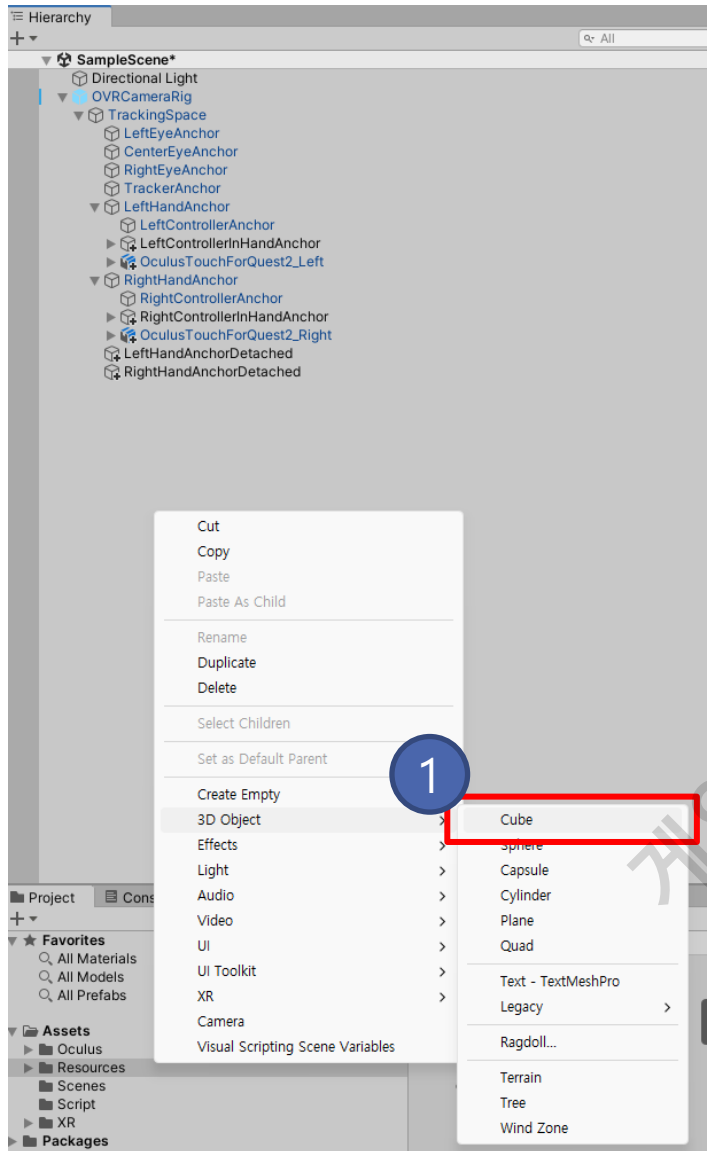
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  // Unity 스크립트 | 참조 7개
6  public class MovingCube : MonoBehaviour
7  {
8      public GameObject _gbParentObject; // 부모 오브젝트
9      public GameObject _gbBombEffect; // 폭발 효과
10     private float fDelTime = 0.5f; // 삭제 지연 시간
11     public float fSpeed = 0.0f; // 이동 속도
12     public bool _isHit = false; // 선택 여부
13     private float _fRandomDelayTime = 3.0f; // 랜덤 이동 유지 시간
14     private Vector3 _vecRandom; // 3차원 공간의 랜덤 위치
15
16     // Start is called before the first frame update
17     // Unity 메시지 | 참조 0개
18     void Start()
19     {
20         NewCalculateVectorPos(); // 랜덤 위치 지정 호출
21     }
22
23     // 참조 3개
24     private void NewCalculateVectorPos()
25     {
26         _vecRandom = new Vector3(Random.Range(-4.0f, 4.0f), Random.Range(-4.0f, 4.0f), Random.Range(3.0f, 6.0f));
27         _fRandomDelayTime = 3.0f;
28     }
29 }
```

```
28 // Update is called once per frame
29 // Unity 메시지 | 참조 0개
30 void Update()
31 {
32     // 변경되는 위치까지 선형보간을 통해 이동
33     transform.position = Vector3.Lerp(transform.position, _vecRandom, Time.deltaTime);
34     _fRandomDelayTime -= Time.deltaTime; // 시간 줄임
35
36     if (_fRandomDelayTime < 0.0f) // 만약 시간이 0보다 작다면
37     {
38         NewCalculateVectorPos();
39     }
40
41     if (_isHit) // 레이캐스트로 선택된 경우
42     {
43         _fDelTime -= Time.deltaTime;
44     }
45     else
46     {
47         _fDelTime = 0.5f;
48     }
49
50     if (_fDelTime < 0.0f) // 0.5초 이상 선택이 되었다면...
51     {
52         ChildExplosion(); // 하위 객체 삭제
53     }
54
55     if (_gbParentObject.GetComponent<Shooting>()._isGameOver) // 게임이 종료되었다면
56     {
57         ChildExplosion(); // 하위 객체 삭제
58     }
59 }
60
61 // Unity 메시지 | 참조 0개
62 private void OnCollisionEnter(Collision collision)
63 {
64     if (collision.gameObject.name.Contains("Target"))
65     {
66         NewCalculateVectorPos(); // 랜덤으로 위치 지정 호출
67     }
68 }
```

MovingCube 스크립트 작성

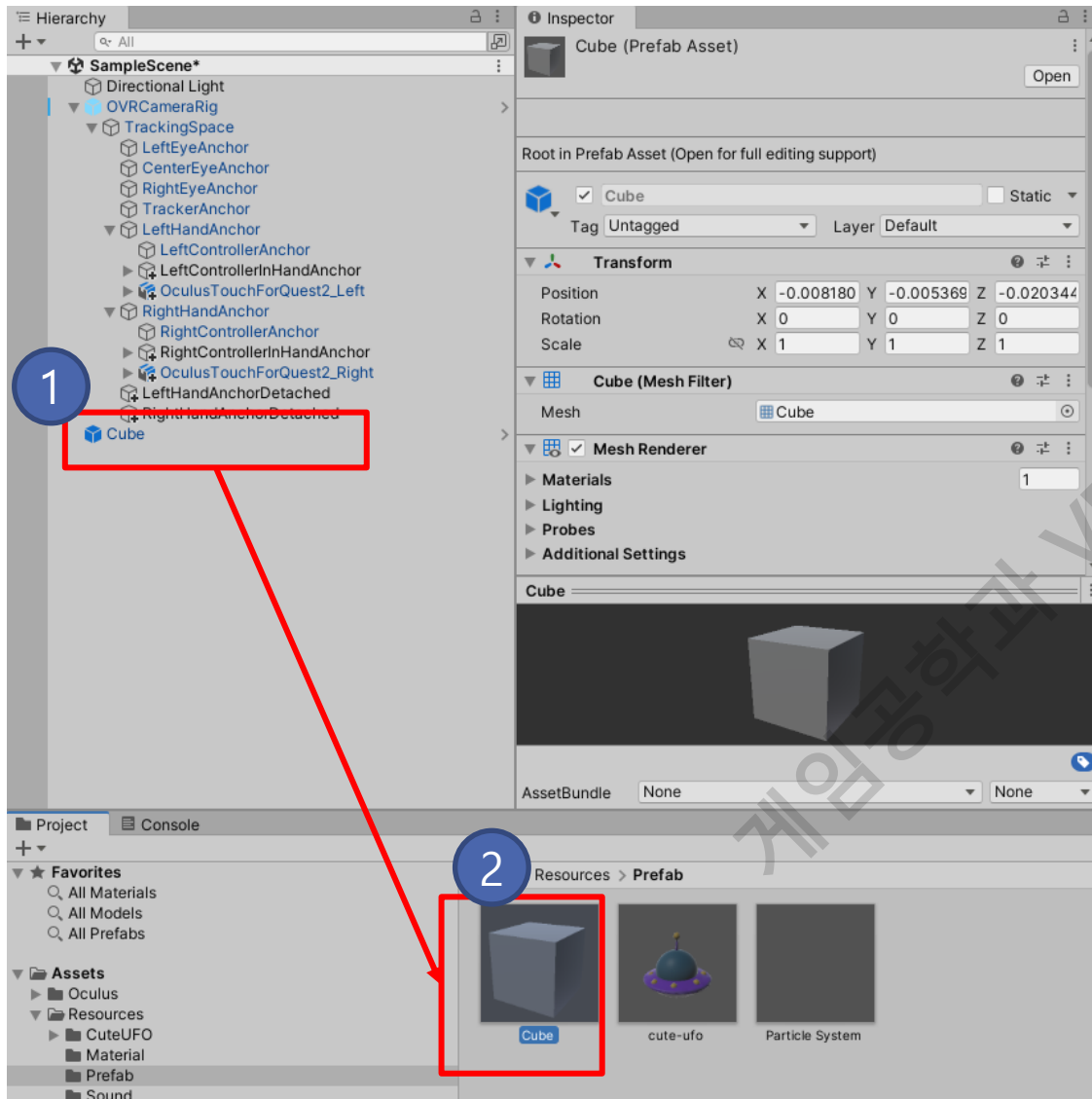
```
참조 2개
62 private void Child_Explotion()
63 {
64     if(this.gameObject.transform.localScale.y > 0.6) // 게임 오브젝트의 사이즈가 0.6보다 크다면
65     {
66         _gbParentObject.GetComponent<Shooting>().
67             SpawnFunction(this.gameObject.transform.localScale.y, this.gameObject.transform.position); //초기 설정
68     }
69
70     child_Fundction(); // 공통 동작 호출
71 }
72
참조 1개
73 private void child_Fundction()
74 {
75     GameObject gb = Instantiate(_gbBombEffect);
76     _gbParentObject.GetComponent<AudioSource>().clip
77         = _gbParentObject.GetComponent<Shooting>().
78             _acSoundPlay[Random.Range(0, _gbParentObject.GetComponent<Shooting>()._acSoundPlay.Length)]; // 사운드 설정
79     _gbParentObject.GetComponent<AudioSource>().Play(); // 사운드 플레이
80     gb.transform.position = this.gameObject.transform.position; //위치 변경
81     gb.SetActive(true); // 폭발 효과 오브젝트 가시화
82     Destroy(this.gameObject); // 선택 오브젝트 삭제
83 }
```


돌아다니는 큐브 만들기

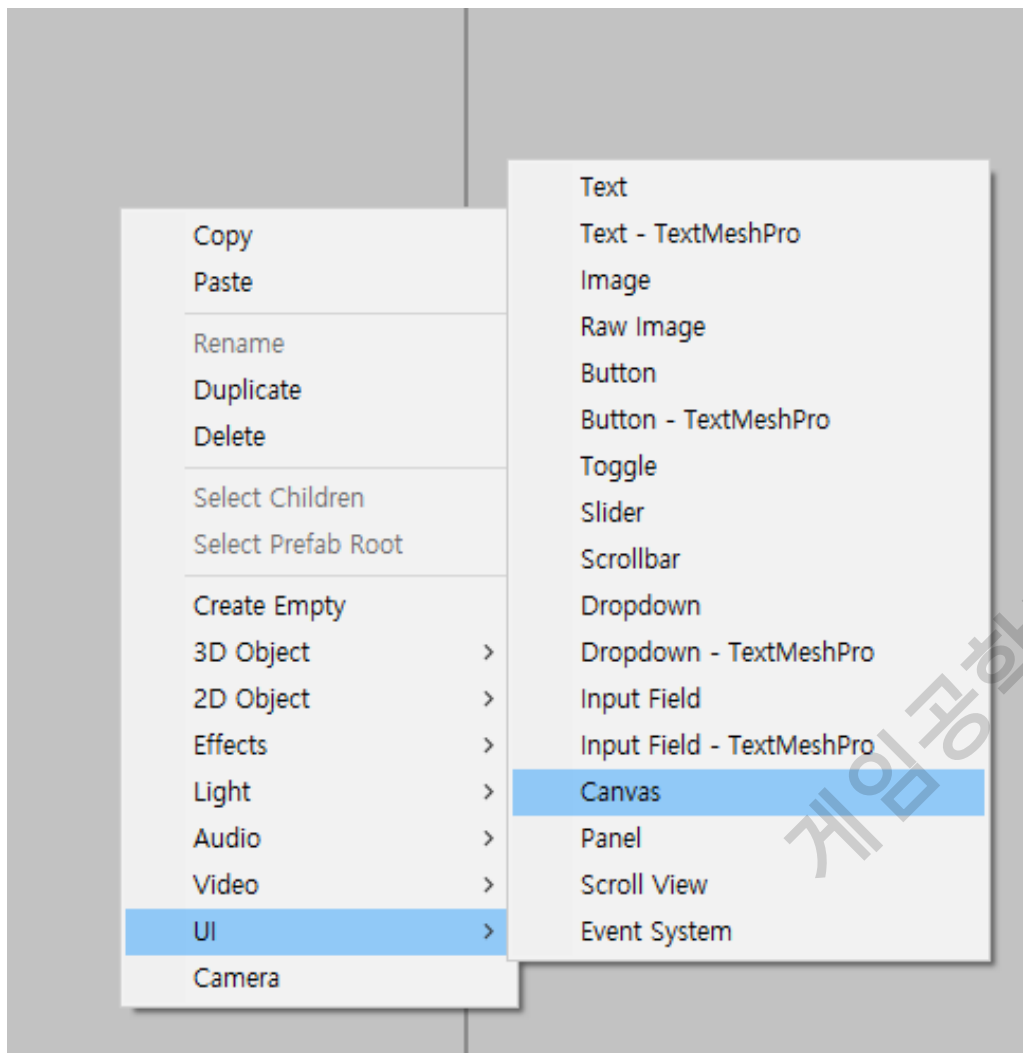


- ① Hierarchy 에서 Cube를 생성

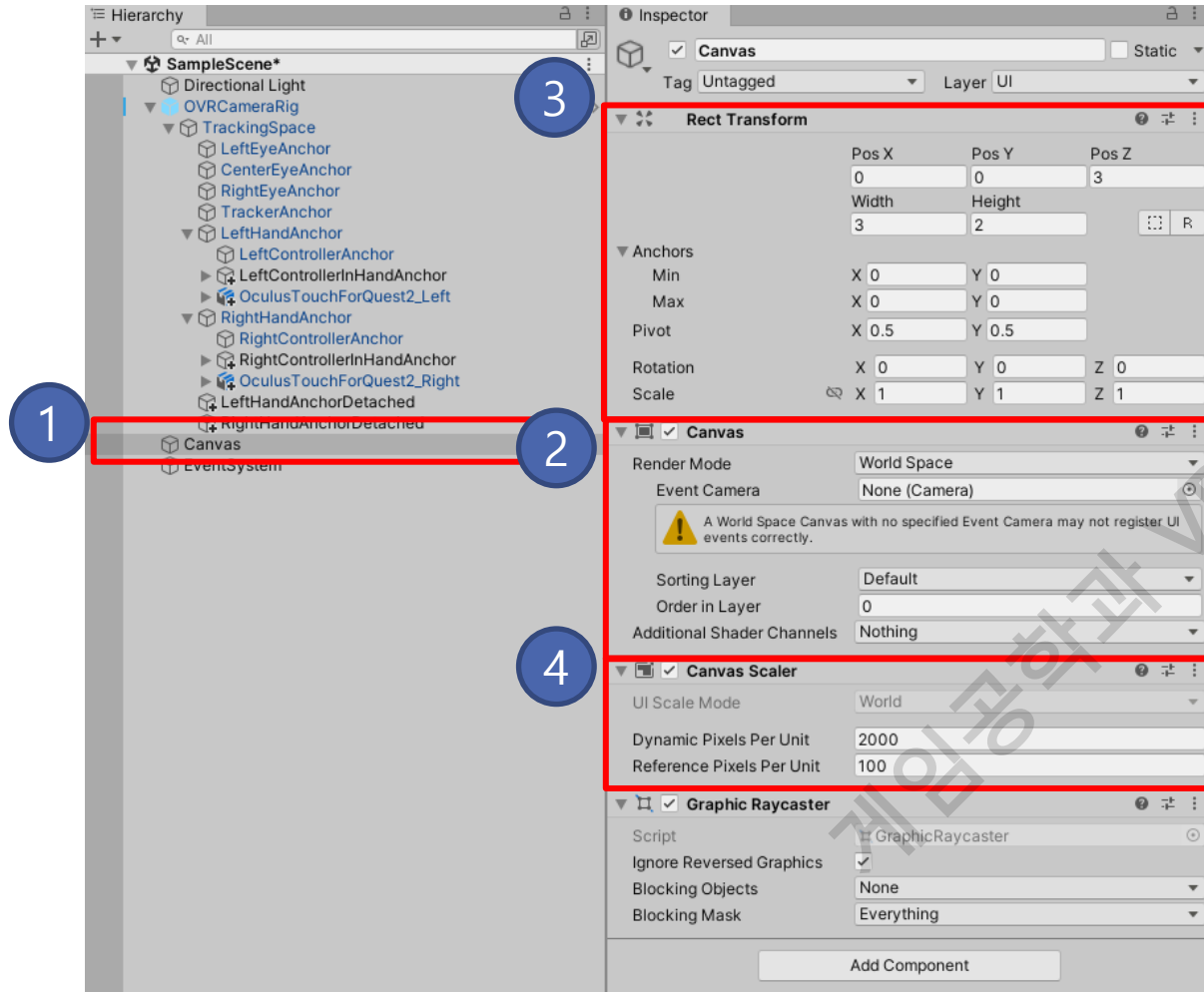
돌아다니는 큐브 만들기



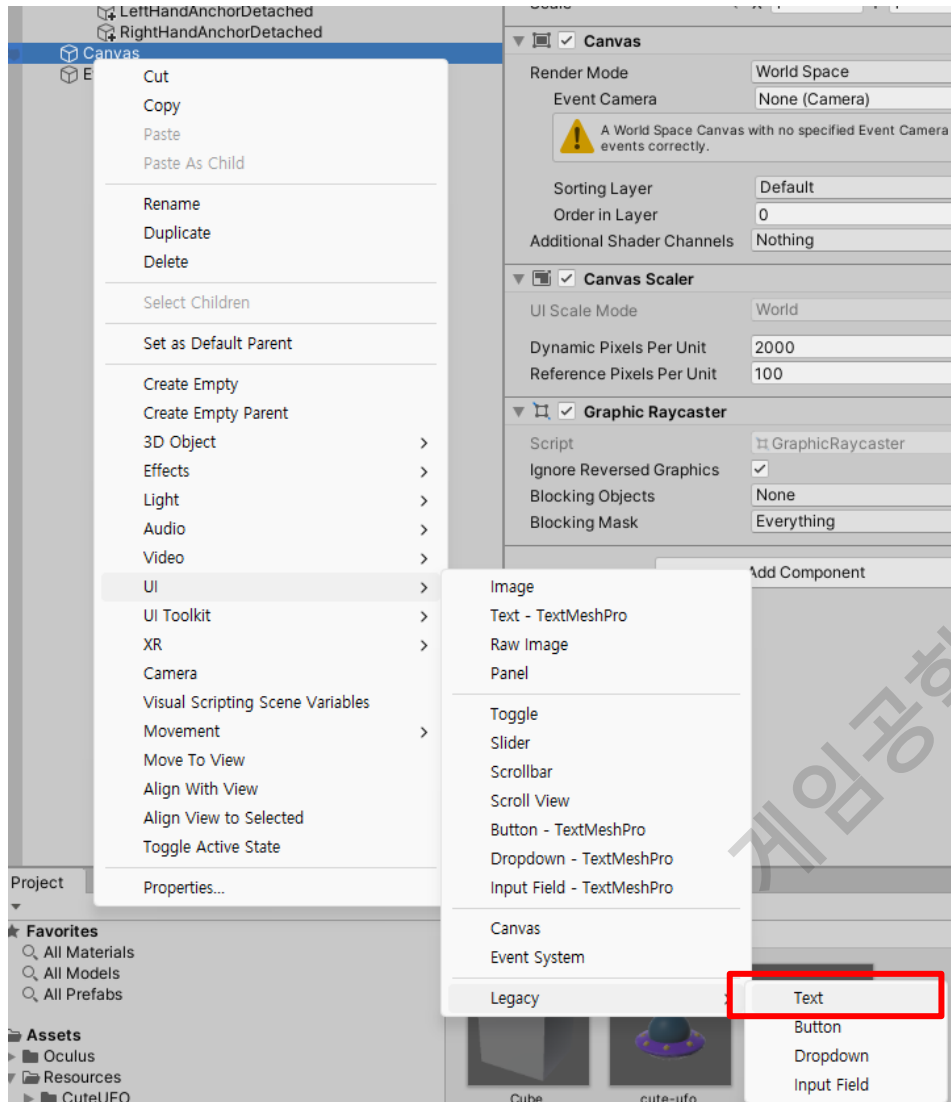
- ① Hierarchy 에서 Cube를 선택합니다.
- ② Cube를 Assets -> Resources -> Prefab 폴더로 이동시켜 프리팹으로 만듭니다.
- 이동 후 Hierarchy에서 Cube 삭제!!



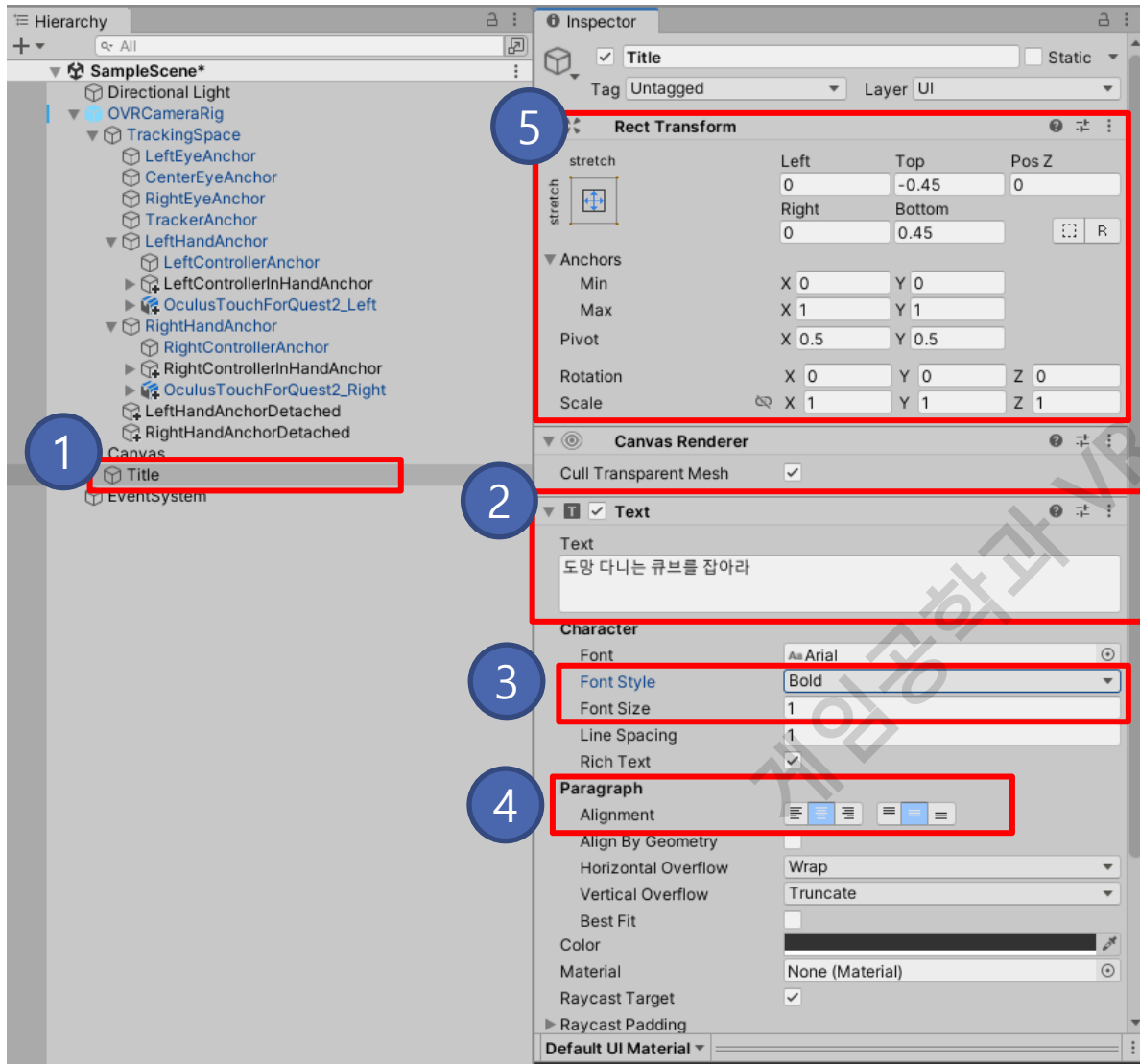
- ① Hierarchy 빈공간에서 UI 선택 후 Canvas 생성



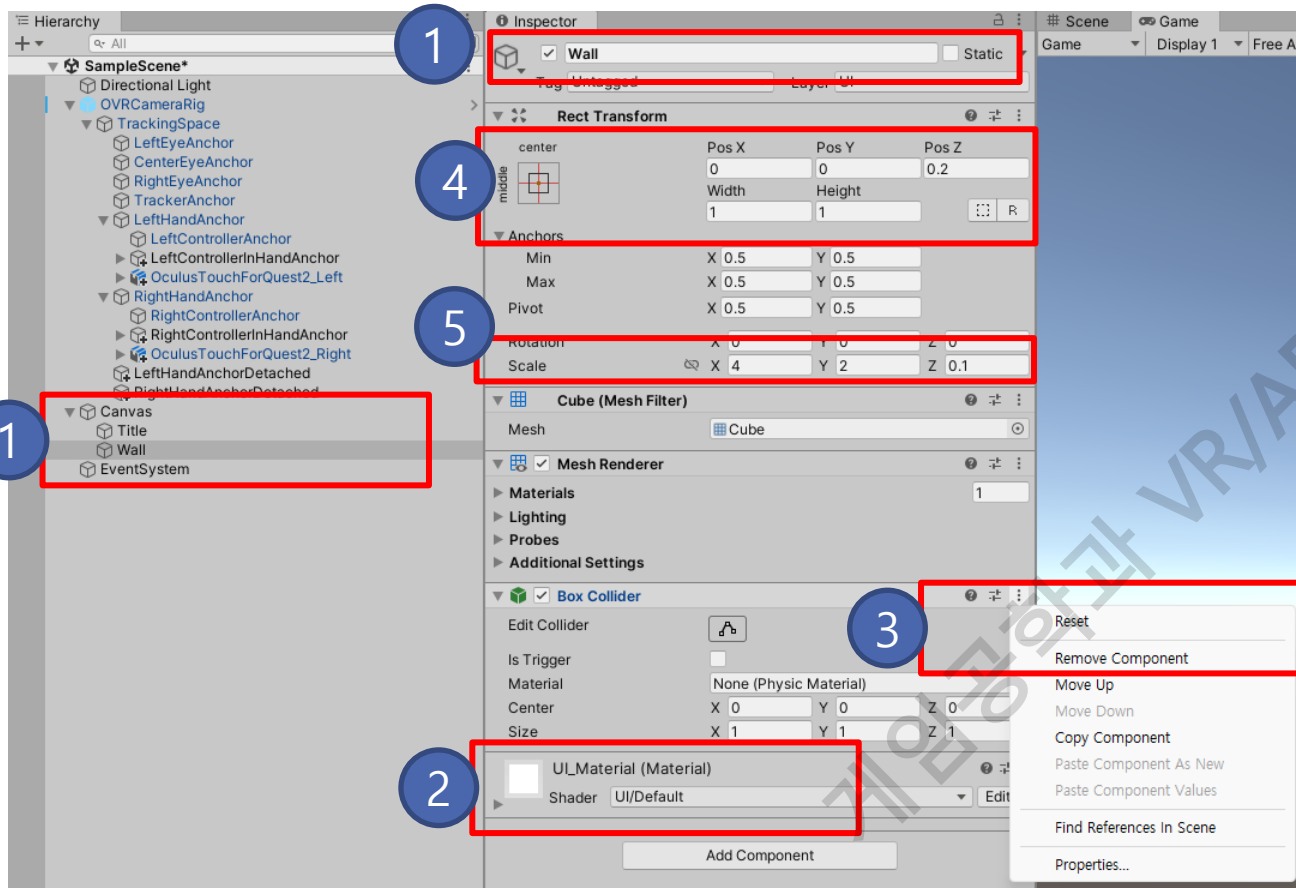
- ① 생성된 Canvas 선택
- ② Canvas 의 Render Mode를 World Space 로 변경
- ③ Rect Transform 의 값을 0 0 3 3 2 값으로 변경
- ④ Canvas Scaler의 Dynamic Pixels Per Unit 의 값을 2000으로 변경



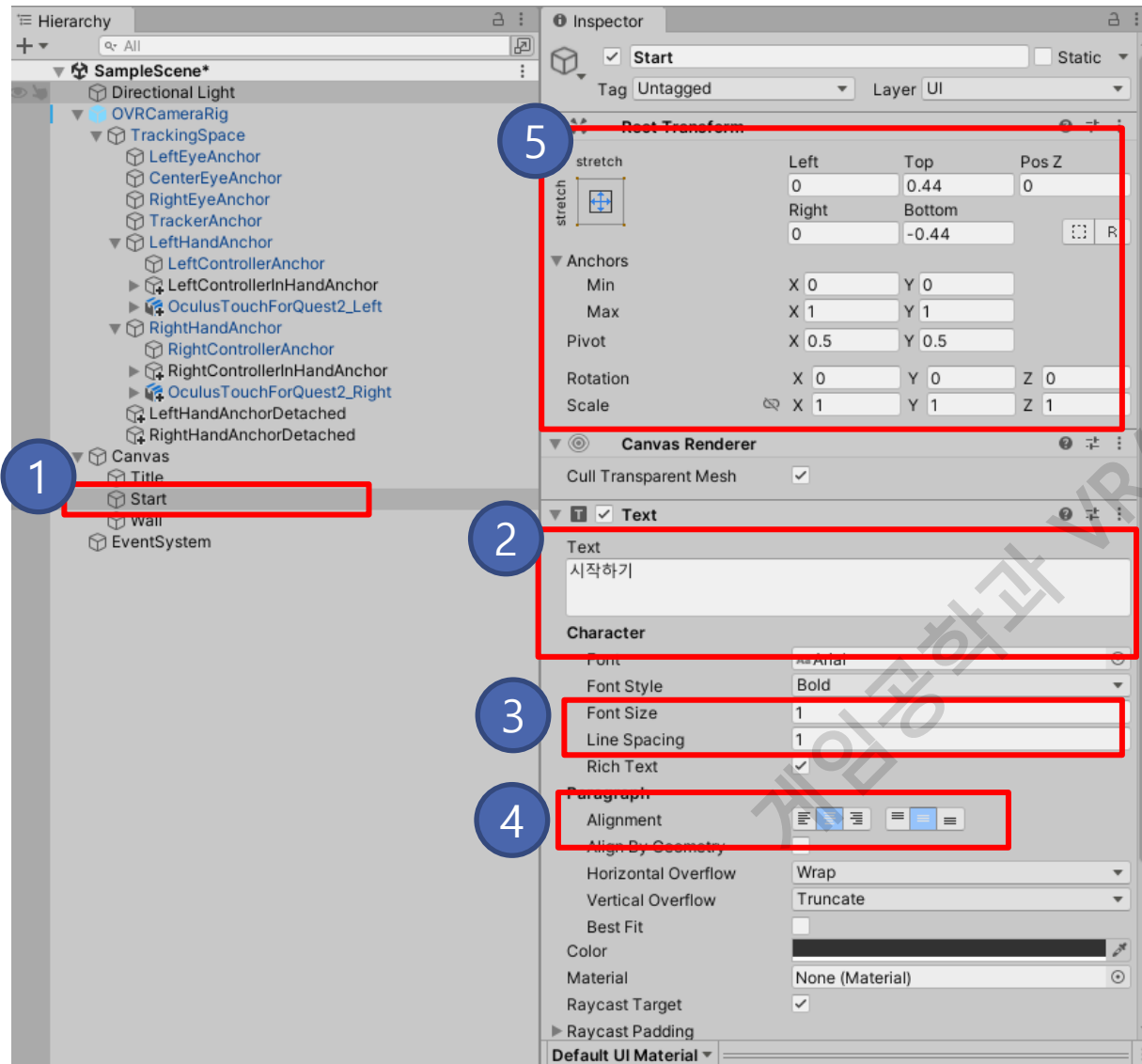
- ① Canvas 선택 후 오른쪽 마우스 클릭
- ② UI -> Legacy 선택
- ③ Text를 선택하여 추가



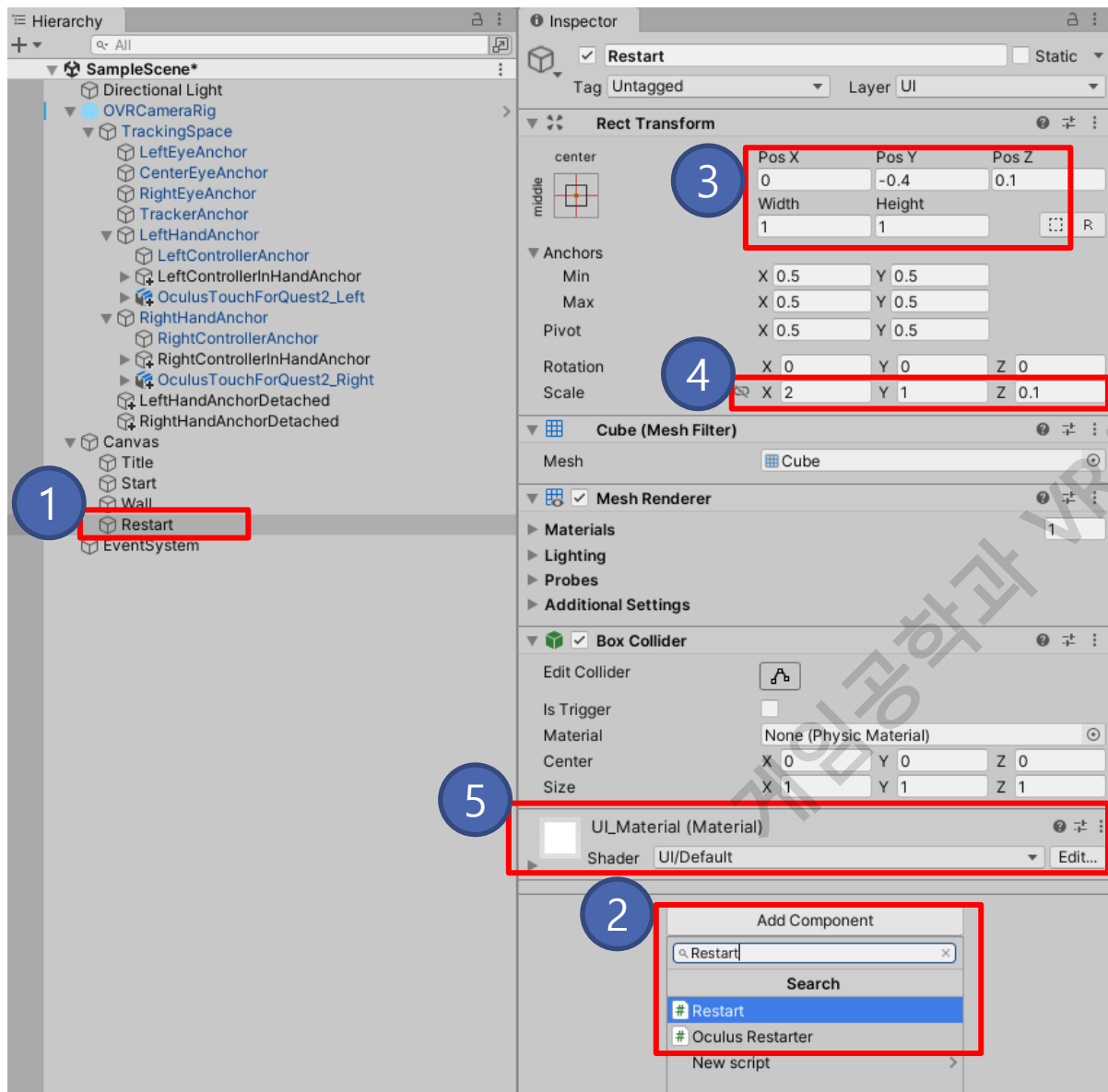
- ① 생성된 Text의 이름을 Title로 변경
- ② Inspector 에서 Text를 “도망 다니는 큐브를 잡아라” 입력
- ③Font Size를 1, Font Style을 Bold로 변경
- ④Paragraph 에서 글자 정렬을 모두 가운데로 맞춤
- ⑤Rect Trasnfrom 에서 Top를 0.45, Bottom을 -0.45로 변경
- Anchors Min 값을 모두 0으로, Max 값을 모두 1로 변경



- ① Canvas에 3D Object->Cube를 생성 후 Inspector에서 Wall로 이름 변경
- ② Project 탭에서 Assets – resources – Material 의 UI_Material 을 Wall 의 Inspector 빈공간에 끌어오기
- ③ Wall의 Inspector에서 Box Collider를 오른 쪽 마우스 클릭하여 Remove Component 하여 지우기
- ④ Rect Transform 의 Pos 값을 0, 0, 0.2, 1, 1 로 변경하고
- ⑤ Scale 값을 4, 2, 0.1 로 변경

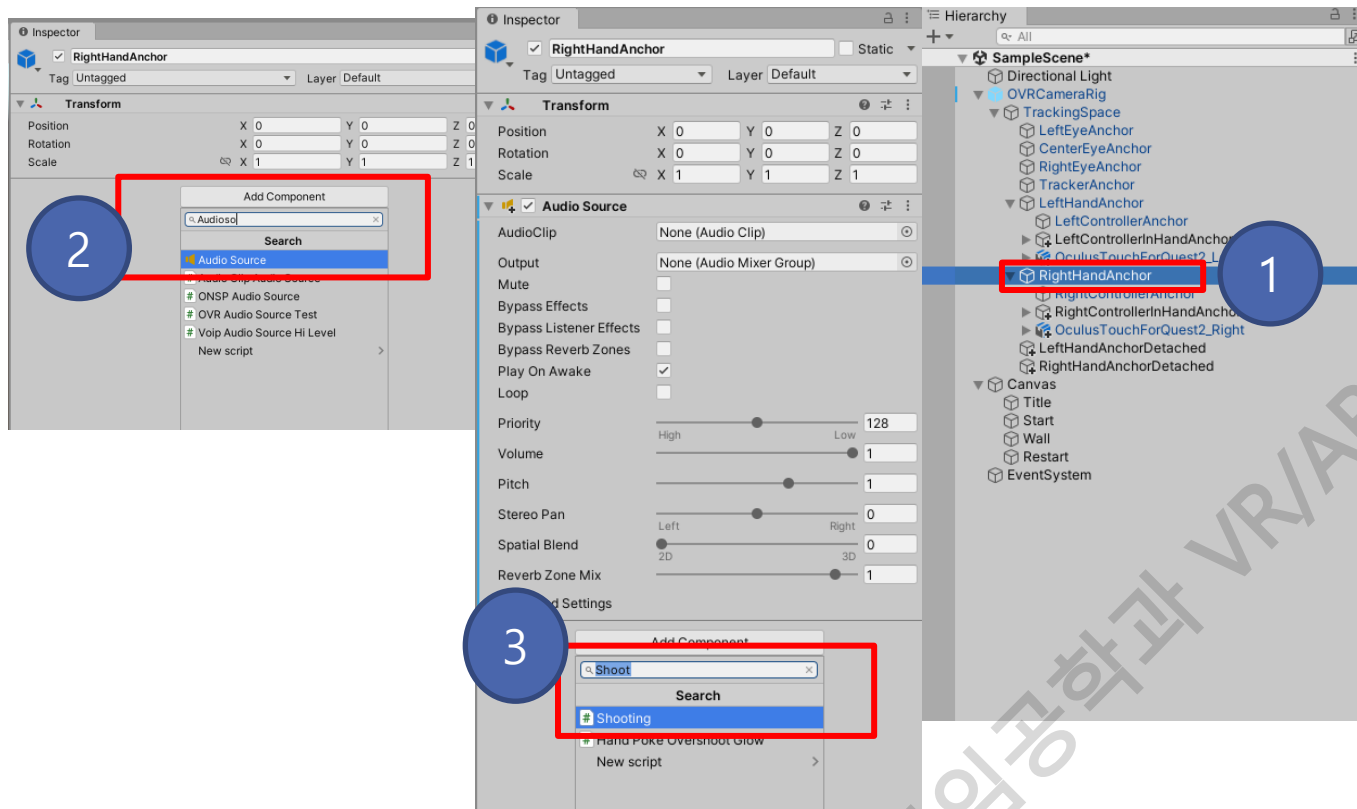


- ① 생성된 Text의 이름을 Start로 변경
- ② Inspector 에서 Text를 "시작하기" 입력
- ③Font Size를 1, Font Style을 Bold로 변경
- ④Paragraph 에서 글자 정렬을 모두 가운데로 맞춤
- ⑤Ract Trasnfrom 에서 Top를 0.44, Bottom을 -0.44로 변경
- Anchors Min 값을 모두 0으로, Max 값을 모두 1로 변경



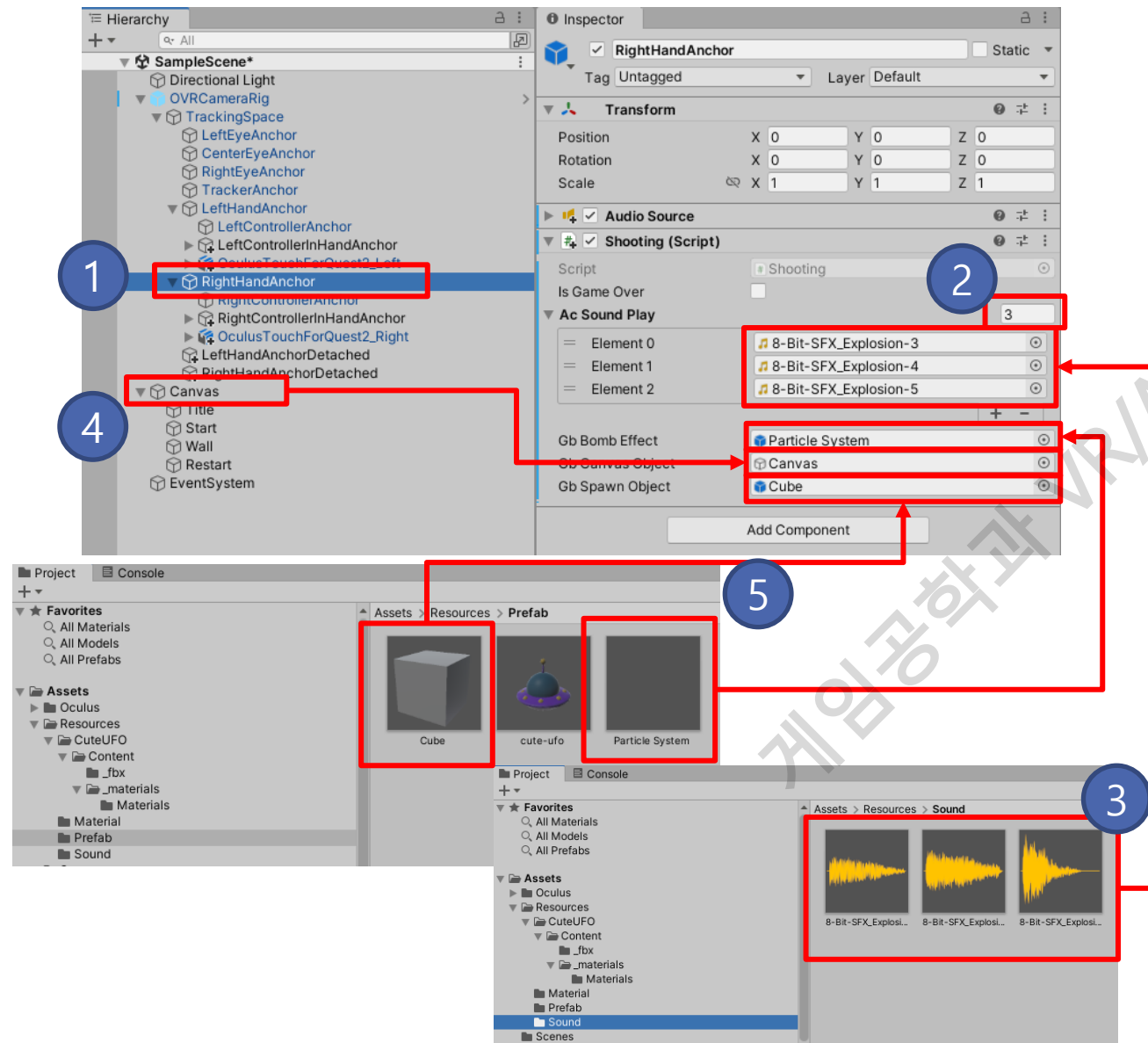
- ① Canvas에 3D Object->Cube를 생성 후 Inspector에서 Restart로 이름 변경
- ② Inspector 하단에 Add Component 를 선택 하여 Restart 스크립트를 추가하여 Inspector 안에 생성되는지 확인합니다
- ③ Rect Transform 의 Pos 값을 0, -0.4, 0.1, 1, 1 로 변경하고
- ④ Scale 값을 2, 1, 0.1 으로 수정
- ⑤ Project 탭에서 Assets – resources – Material 의 UI_Material 을 Wall 의 Inspector 빈공간에 끌어오기

슈팅 기능 제작



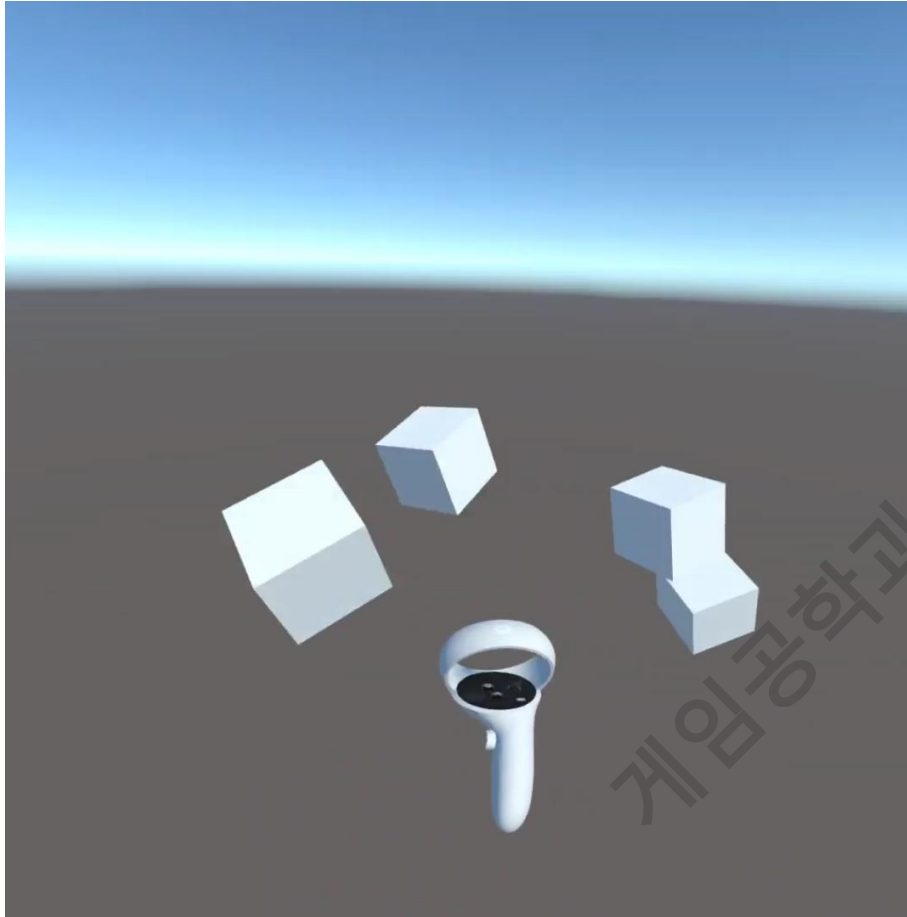
- ① OvrCameraRig 하위에 RightHandAnchor 을 선택합니다.
- ② Inspector 하단에 Add Component 를 선택 하여 Audio Source 컴포넌트 추가
- ③ Inspector 하단에 Add Component 를 선택 하여 Shooting 스크립트를 추가하여 Inspector 안에 생성되는지 확인합니다.

슈팅 기능 제작

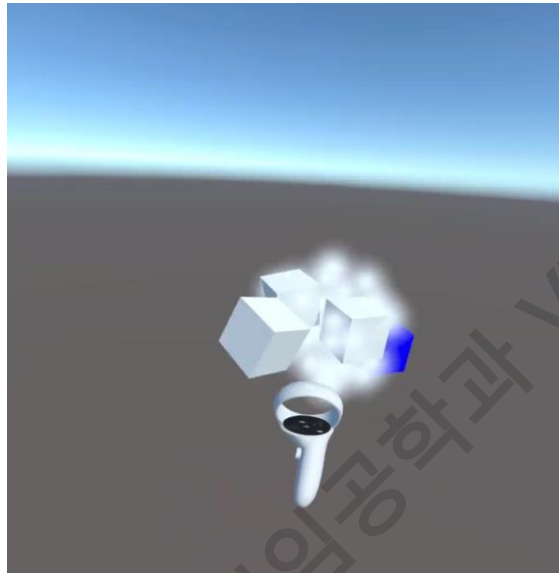
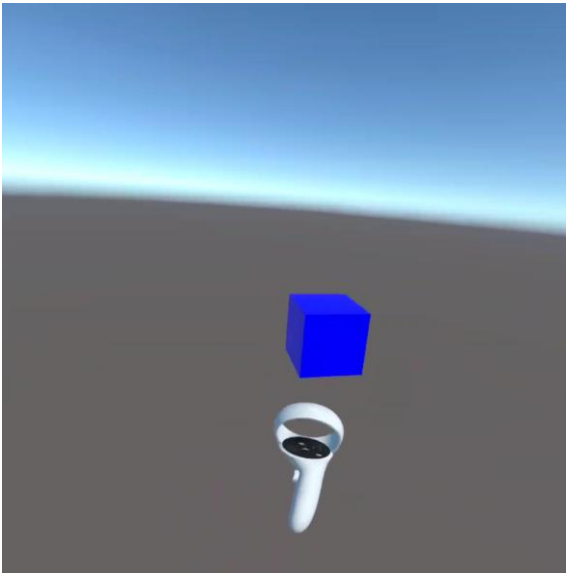


- ① OvrCameraRig 하위에 RightHandAnchor 을 선택합니다.
- ② Ac Sound Play 숫자를 3으로 설정합니다.
- ③ Project탭에서 Assets -> resources -> Sound 내의 사운드를 Element 0~2에 설정합니다.
- ④ Hierarchy에서 Canvas를 Shooting 의 Gb Canvas Object에 이동 시킵니다.
- ⑤ Project탭에서 Assets - resources - Prefab 안에 있는 Cube를 Shooting 의 Gb Spawn Object로 Particle System을 Gb Bomb Effect로 이동 시킵니다.

2주차 수업자료를 이용하여 케이블 링크로 실행합니다.



- ① 오른손 컨트롤러를 잡고 겨냥을 합니다.
- ② 컨트롤러의 전면에서부터 겨냥 레이저가 나갑니다.(보이지는 않음)
- ③ 컨트롤러의 전면을 큐브에 맞추며 잡기 위한 조준점을 맞춘다



- ① 조준점을 물체에다가 가져다 두면 물체의 색상이 파랗게 변하면서 선택된다
- ② 선택된 오브젝트는 0.3초간 바라보면 터지는 이펙트와 함께 4개로 나누어집니다.
- ③이렇게 나누어지는 오브젝트를 모두 잡으면 게임이 끝납니다.