3D Interactive Contents
(Chapter 5)

Jin-Mo Kim
jinmo.kim@hansung.ac.kr

3D 콘텐츠 제작

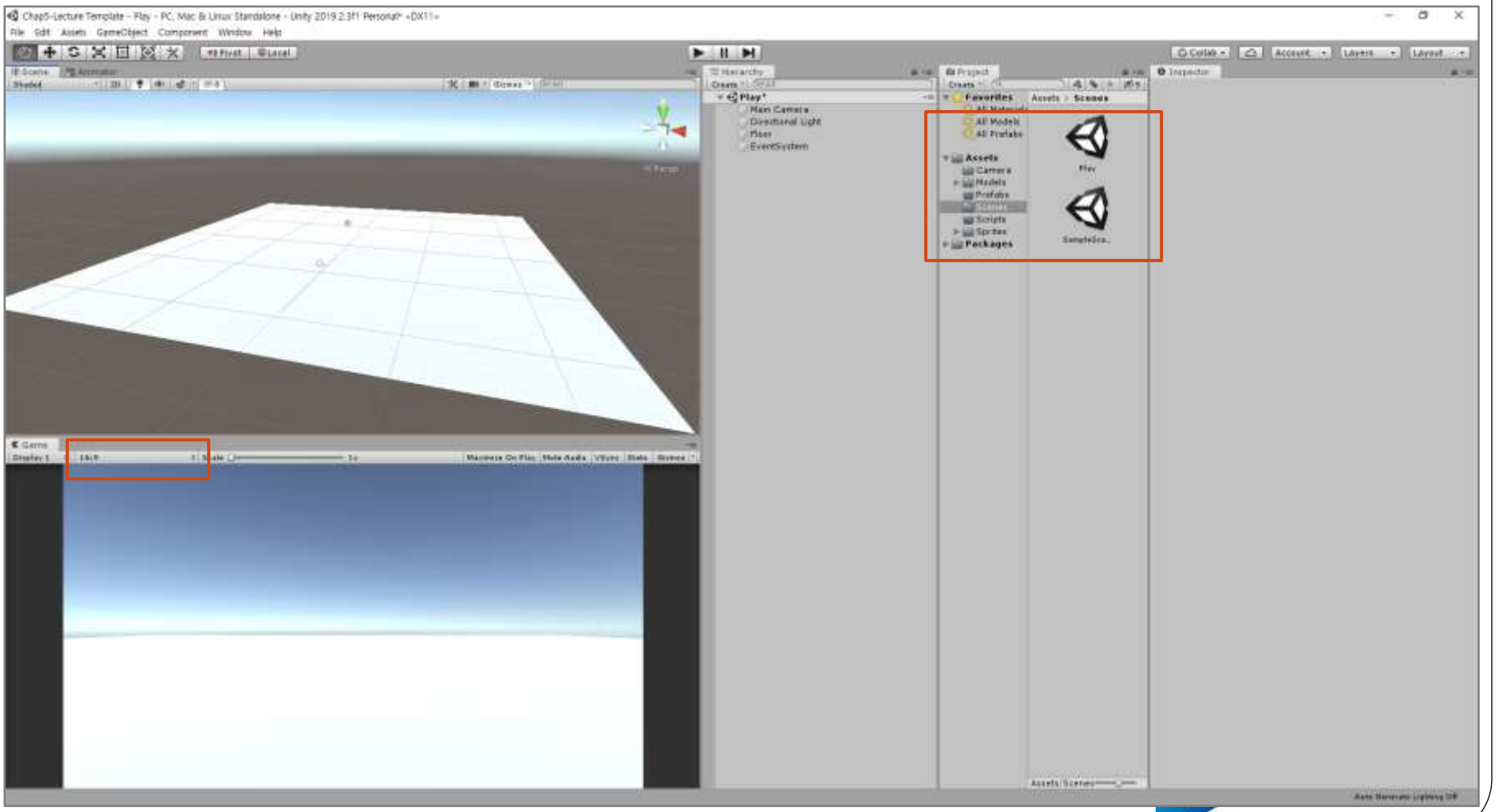
- 3D 콘텐츠 제작 과정
  - 간단한 3D 액션 게임 콘텐츠를 제작하는 과정을 통해 3D 콘텐츠에서 사용되는 용어나 기법을 학습
    - PC 뿐 아니라 모바일에서 제어 가능한 3D 콘텐츠 제작
    - 캐릭터 애니메이션 처리 및 가상 패드, 카메라 구현 방법
    - 간단한 파티클 애니메이션 처리 방법

3D 콘텐츠 제작

- 준비 과정
  - Camera, Models, Sprites 폴더를 Project에 복사
  - 배경
    - Hierarchy → Create → Plane
      - 이름 : Floor
      - Scale : 5, 1 ,5
  - 장면 저장
    - File → New Scene
      - Save As → Scenes 폴더에 이름을 Play로 하여 저장
  - 폴더 생성
    - Project → Create → Folder
      - Scripts, Prefabs 두 개의 폴더 생성
  - 게임(Game) 뷰 해상도 조절
    - Free Aspect → 16:9로 변경

3D 콘텐츠 제작

- 준비 과정



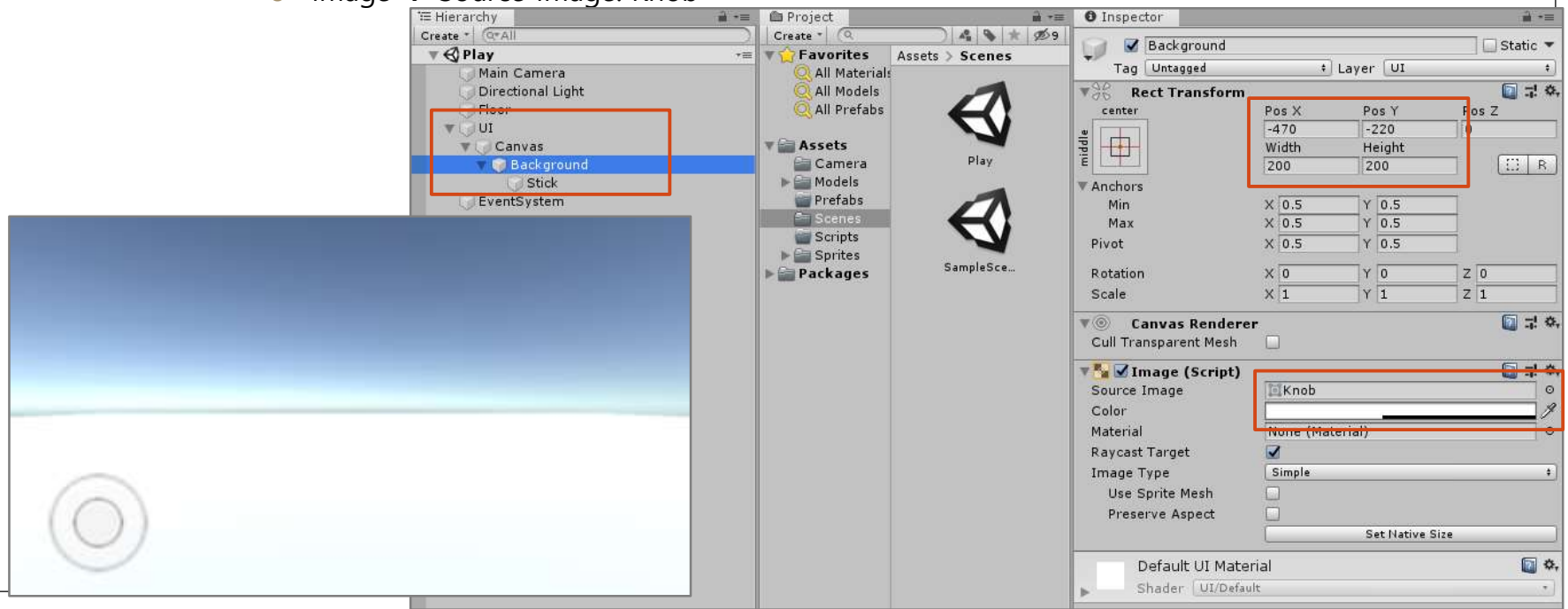
3D 콘텐츠 제작

- 가상 패드
  - Hierarchy → Create → Create Empty
    - 이름: UI
    - UI 자식 객체
      - Hierarchy → Create → UI → Canvas
        - Canvas Scaler → UI Scale Mode : Scale With Screen Size
        - Canvas Scaler → Reference Resolution : (X: 1280, Y: 720)
    - 가상 패드 이미지 추가
      - Hierarchy → Create → UI → Image
        - 이름 : Background
        - PosX, PosY : -470, -220
        - Width, Height : 200, 200
        - Image → Source Image: Knob
        - Color → Alpha : 110

3D 콘텐츠 제작

참고로 이미지의 좌표와 화면의 좌표는 다르다

- 가상 패드
  - 가상 패드 이미지 추가
    - Background 자식 객체로 이미지 추가
      - Hierarchy → Create → UI → Image
        - 이름 : Stick
        - PosX, PosY : 0, 0
        - Width, Height : 100, 100
        - Image → Source Image: Knob



3D 콘텐츠 제작

- 조이스틱 스크립트
  - Project → Create → C# Script
    - 이름 : cshJoystick
    - 스크립트를 Background 이미지에 등록

```
using UnityEngine.UI;
using UnityEngine.EventSystems;

public class cshJoystick : MonoBehaviour, IDragHandler, IPointerUpHandler, IPointerDownHandler
{
    private Image imgBG;
    private Image imgJoystick;
    private Vector3 vInputVector;

    // Start is called before the first frame update
    void Start()
    {
        imgBG = GetComponent<Image>();
        imgJoystick = transform.GetChild(0).GetComponent<Image>();
    }
}
```

3D 콘텐츠 제작

C#은 자바처럼, 다중 상속이 불가
즉, 골라진 4개는 클래스가 아니라
인터페이스다.

- 조이스틱 스크립트

MonoBehaviour의 start
함수 처럼 자동으로 알아서
동작하는 인터페이스가 있는
반면,

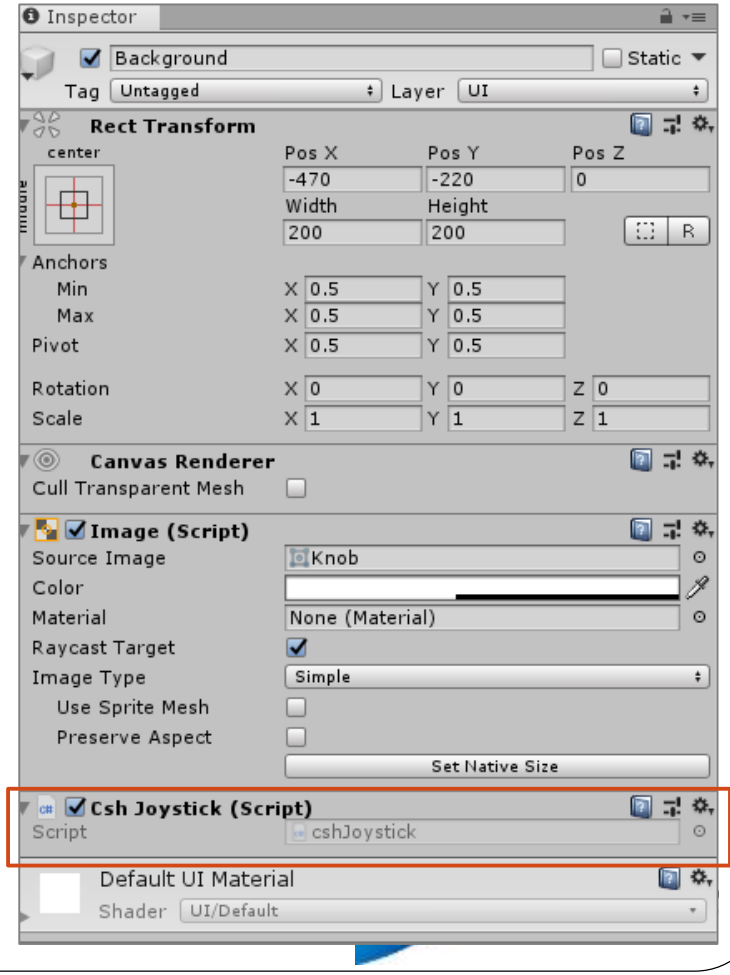
나머지 3개는 반드시
(버튼에따라)재정의를 해주어야 함

```
public class cshJoystick : MonoBehaviour, IDragHandler, IPointerUpHandler, IPointerDownHandler
{
    public void OnDrag(PointerEventData eventData)
    {
        Debug.Log("Joystick >>> OnDrag()");
        Vector2 pos;

        //배경 영역에 터치가 발생할 때
        if (RectTransformUtility.ScreenPointToLocalPointInRectangle(imgBG.rectTransform, eventData.position, eventData.pressEventCamera, out pos))
        {
            //Debug.Log(imgBG.rectTransform.sizeDelta);
            //터치된 로컬 좌표값을 pos에 저장
            //배경 이미지의 size로 나누어 pos.x: -1~1, pos.y: -1~1 으로 변환
            pos.x = (pos.x / imgBG.rectTransform.sizeDelta.x);
            pos.y = (pos.y / imgBG.rectTransform.sizeDelta.y);

            vInputVector = new Vector3(pos.x, pos.y, 0);
            vInputVector = (vInputVector.magnitude > 1.0f) ? vInputVector.normalized : vInputVector;

            //Joystick Image 움직임
            imgJoystick.rectTransform.anchoredPosition = new Vector3(vInputVector.x * (imgBG.rectTransform.sizeDelta.x / 2),
                                                                     vInputVector.y * (imgBG.rectTransform.sizeDelta.y / 2));
        }
    }

    public void OnPointerDown(PointerEventData eventData)
    {
        OnDrag(eventData);
    }
    public void OnPointerUp(PointerEventData eventData)
    {
        vInputVector = Vector3.zero;
        imgJoystick.rectTransform.anchoredPosition = Vector3.zero;
    }
}
```

3D 콘텐츠 제작

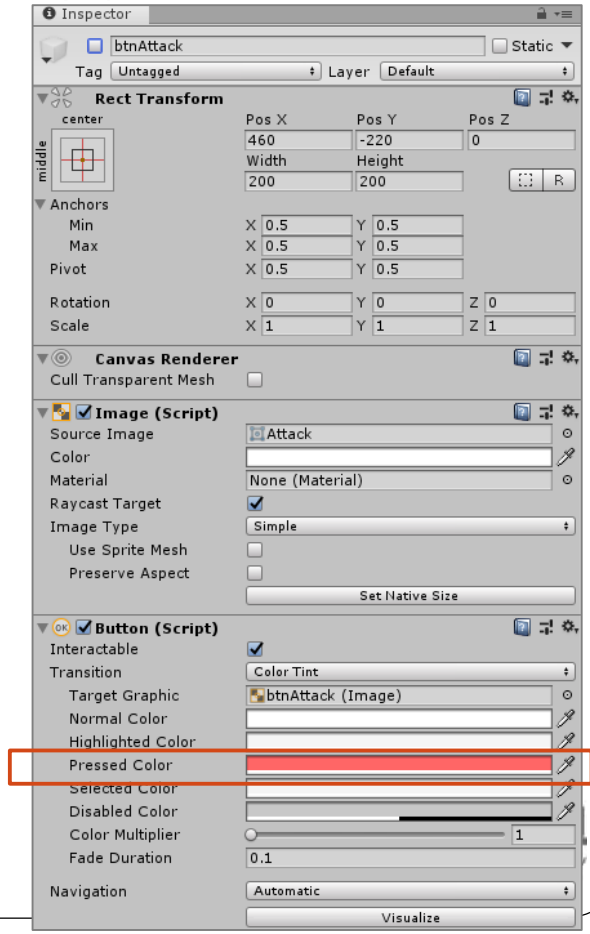
- 조이스틱 스크립트

```
public class cshJoystick : MonoBehaviour, IDragHandler, IPointerUpHandler, IPointerDownHandler
{
    //PlayerController 에서 입력 값을 넘겨주기 위한 함수
    public float GetHorizontalValue()
    {
        return vInputVector.x;
    }
    public float GetVerticalValue()
    {
        return vInputVector.y;
    }

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

    }
}
```



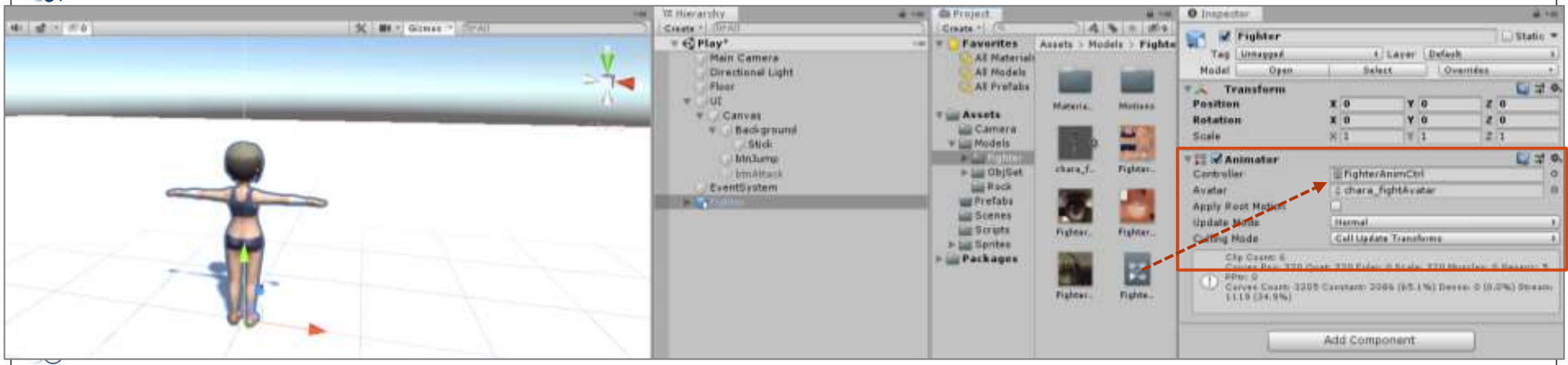
3D 콘텐츠 제작

- 가상 패드
  - 점프, 공격 버튼 추가
    - Hierarchy → Create → UI → Button 두 개 버튼 각각 생성
      - 이름: btnJump, btnAttack
      - PosX, PosY: 460, -220
      - Width, Height: 200, 200
      - Image → Source Image : Jump, Attack 각각 지정
      - 버튼 하위 객체인 text는 삭제
      - btnAttack 버튼
        - Pressed Color를 붉은 계열 색으로 변경
        - 비활성화
    - UI 객체의 Layer를 UI로 변경
      - Change Layer 다이얼로그
        - Yes, change children 선택



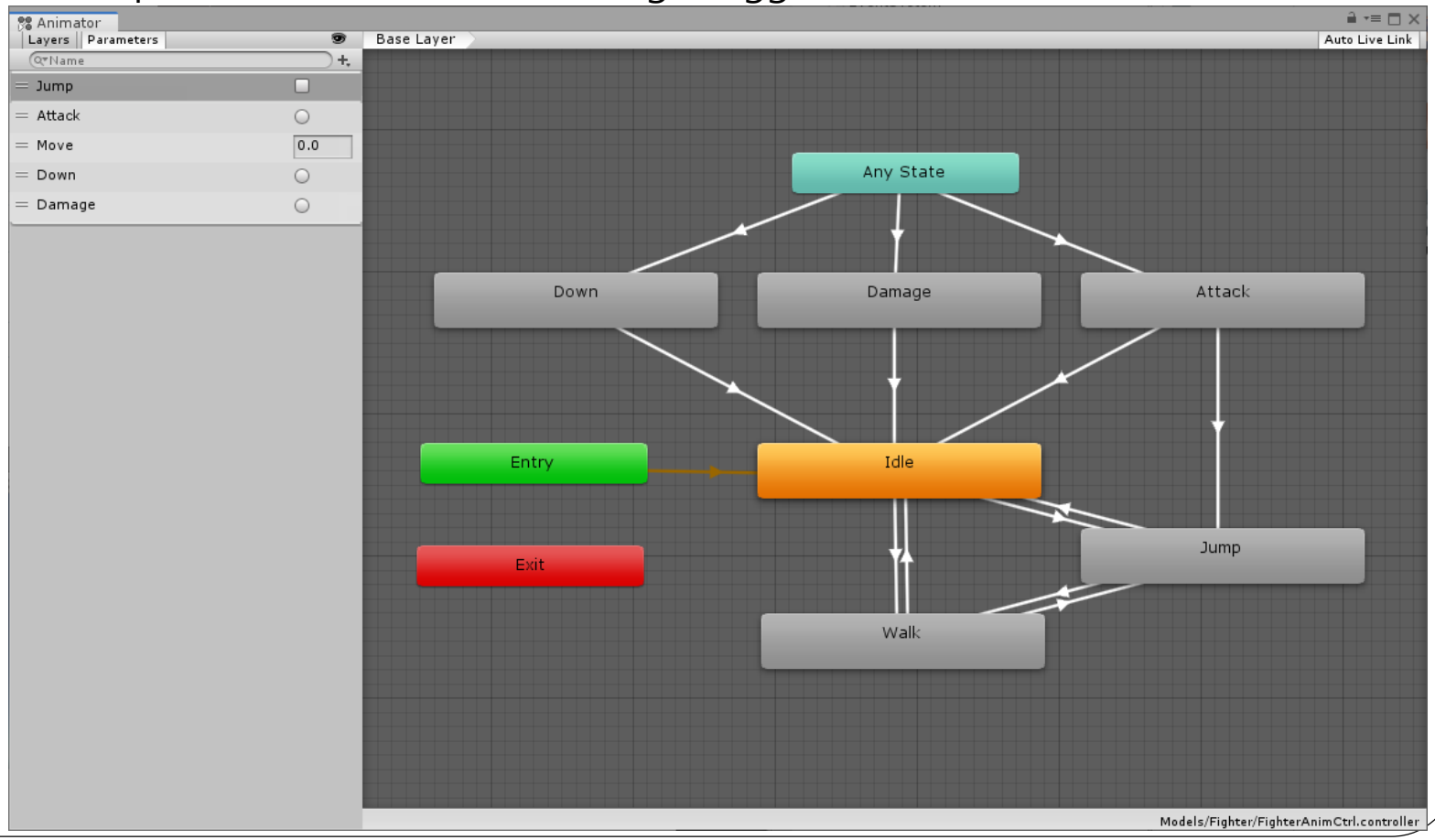
3D 콘텐츠 제작

- 캐릭터 설정
  - Models → Fighter → chara_fight.fbx
    - Rig: Generic
    - Scene으로 등록
      - 이름: Fighter로 변경
      - Fighter → FighterAnimCtrl : 미리 설정된 애니메이터 컨트롤러를 활용



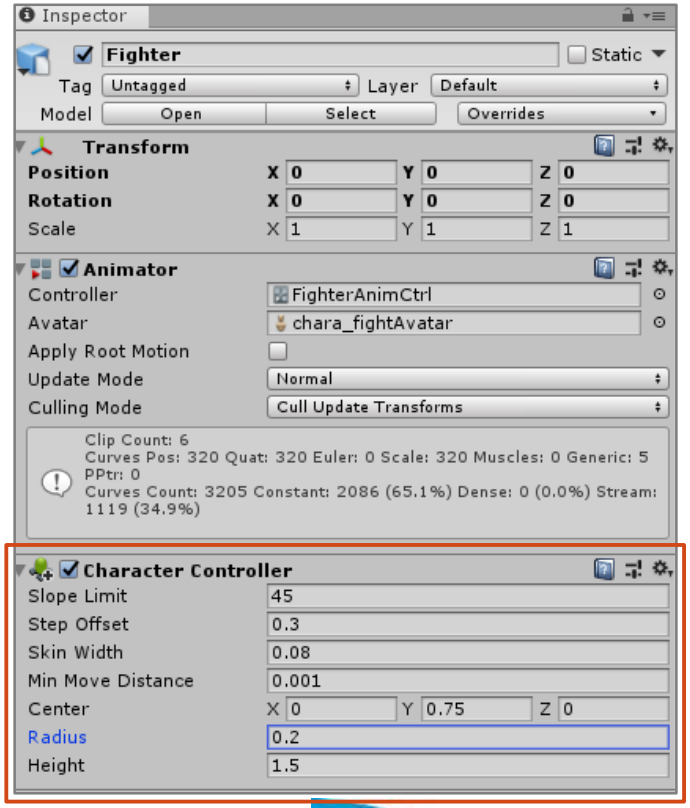
3D 콘텐츠 제작

- 캐릭터 설정
  - 파라미터 설정
    - Jump(Bool), Attack, Down, Damage(Trigger), Move(Float)



3D 콘텐츠 제작

- 캐릭터 설정
  - Fighter 캐릭터에 캐릭터 컨트롤러 속성 추가
    - Center: 0, 0.75, 0
    - Radius: 0.2
    - Height: 1.5
  - 플레이어 컨트롤러 스크립트 생성
    - Project → Create → C# Script
      - 이름: cshPlayerController



3D 콘텐츠 제작

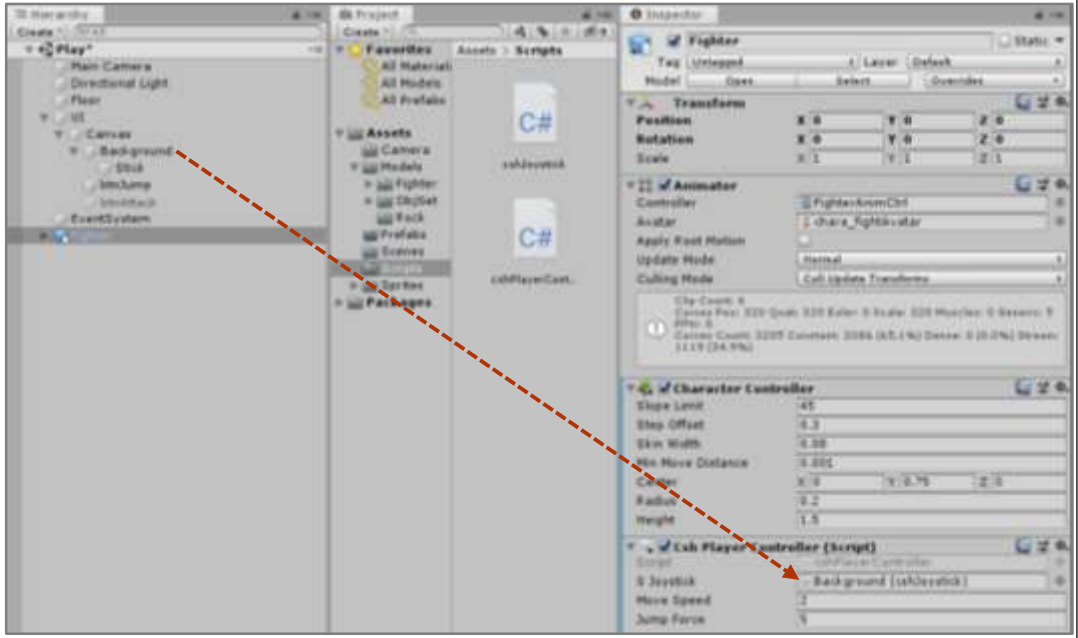
- 캐릭터 설정
  - 플레이어 컨트롤러 스크립트 생성

```
public class cshPlayerController : MonoBehaviour
{
    private Animator m_animator;
    private Vector3 m_velocity;
    private bool m_isGrounded = true;
    private bool m_jumpOn = false;

    public cshJoystick sJoystick;
    public float m_moveSpeed = 2.0f;
    public float m_jumpForce = 5.0f;

    void Start()
    {
        m_animator = GetComponent<Animator>();
    }

    void Update()
    {
        PlayerMove();
        m_animator.SetBool("Jump", !m_isGrounded);
    }

    public void OnVirtualPadJump()
    {

        if (this == null) { return; }
        const float rayDistance = 0.2f;
        var ray = new Ray(transform.localPosition + new Vector3(0.0f, 0.1f, 0.0f), Vector3.down);
        if (Physics.Raycast(ray, rayDistance))
        {
            m_jumpOn = true;
        }
    }
}
```

3D 콘텐츠 제작

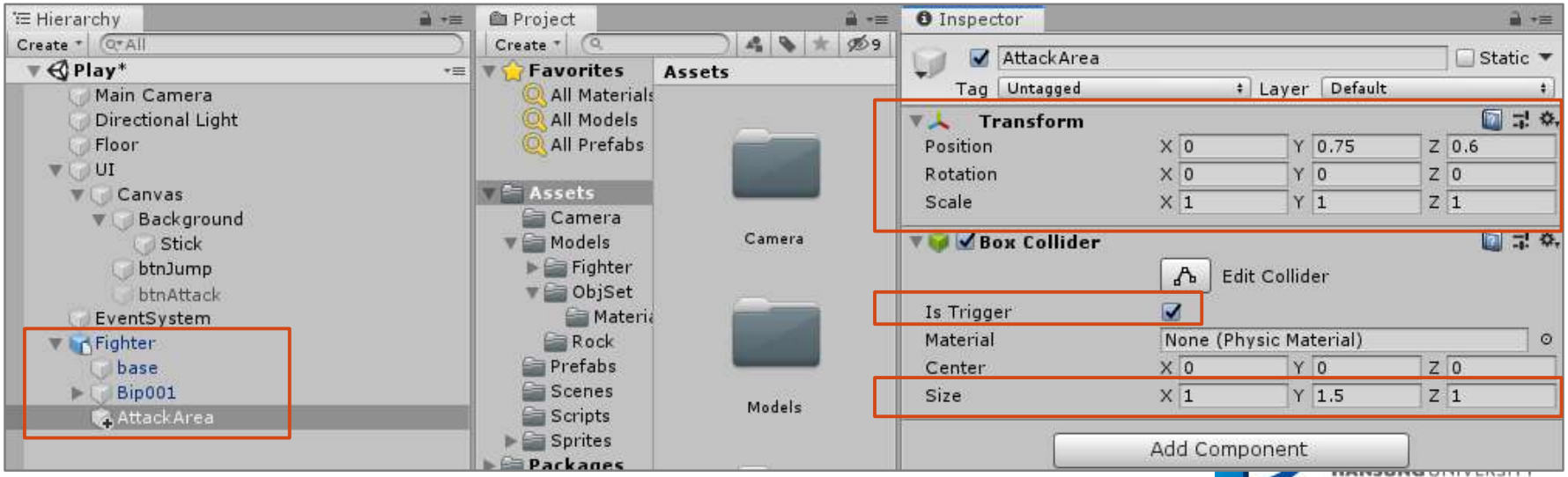
- 캐릭터 설정
  - 플레이어 컨트롤러 스크립트 생성

```
public class cshPlayerController : MonoBehaviour
{
    private void PlayerMove()
    {
        CharacterController controller = GetComponent<CharacterController>();
        float gravity = 20.0f;

        if (controller.isGrounded)
        {
            float h = sJoystick.GetHorizontalValue();
            float v = sJoystick.GetVerticalValue();
            m_velocity = new Vector3(h, 0, v);
            m_velocity = m_velocity.normalized;

            m_animator.SetFloat("Move", m_velocity.magnitude);

            if (m_jumpOn)
            {
                m_velocity.y = m_jumpForce;
                m_jumpOn = false;
            }
            else if (m_velocity.magnitude > 0.5)
            {
                transform.LookAt(transform.position + m_velocity);
            }
        }

        m_velocity.y -= gravity * Time.deltaTime;
        controller.Move(m_velocity * m_moveSpeed * Time.deltaTime);

        m_isGrounded = controller.isGrounded;
    }
}
```



3D 콘텐츠 제작

- 캐릭터 설정
  - 공격 설정
    - Hierarchy → Create → Create Empty
      - 이름: AttackArea
      - Position: 0, 0.75, 0.6
      - BoxCollider 속성 추가
        - Size: 1, 1.5, 1
        - Is Trigger: 체크
      - Fighter의 자식 객체로 설정



3D 콘텐츠 제작

- 캐릭터 설정
  - 공격 범위 스크립트 생성
    - Project → Create → C# Script
      - 이름: cshAttackArea → AttackArea 객체에 등록

```
public class cshAttackArea : MonoBehaviour
{
    public List<Collider> colliders
    {
        get
        {
            if (0 < colliderList.Count)
            {
                //  현재 colliders 리스트에 객체중 null인 것은 제거하여 colliderList에 저장 후 반환
                colliderList.RemoveAll(c => c == null);
            }
            return colliderList;
        }
    }
    private List<Collider> colliderList = new List<Collider>();
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("BreakableObject") || other.CompareTag("Enemy"))
        {
            colliders.Add(other);
        }
    }
    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("BreakableObject") || other.CompareTag("Enemy"))
        {
            colliders.Remove(other);
        }
    }
}
```

3D 콘텐츠 제작

- 버튼 이벤트 설정
  - 점프, 공격 버튼에 대한 이벤트 스크립트 생성
    - Project → Create → C# Script
      - 이름: cshButton → UI 객체에 등록

```
public class cshButton : MonoBehaviour
{
    public Button btnJump;
    public Button btnAttack;
    public cshPlayerController sPlayer;
    void Start()
    {
        btnJump.gameObject.SetActive(true);
        btnJump.onClick.RemoveAllListeners();
        btnJump.onClick.AddListener(OnClickJumpButton);
        btnAttack.gameObject.SetActive(false);
        btnAttack.onClick.RemoveAllListeners();
        btnAttack.onClick.AddListener(OnClickAttackButton);
    }
    void Update()
    {
        UpdateButton();
    }

    private void UpdateButton()
    {
        bool canAttack = sPlayer.CanAttack();
        btnAttack.gameObject.SetActive(canAttack);
        btnJump.gameObject.SetActive(!canAttack);
    }

    private void OnClickJumpButton()
    {
        sPlayer.OnVirtualPadJump();
    }

    private void OnClickAttackButton()
    {
        sPlayer.OnVirtualPadAttack();
    }
}
```

3D 콘텐츠 제작

- 플레이어 스크립트 수정
  - cshPlayerController 변수 및 함수 추가

```
public class cshPlayerController : MonoBehaviour
{
    private cshAttackArea m_attackArea = null;

    void Start()
    {
        m_animator = GetComponent<Animator>();
        m_attackArea = GetComponentInChildren<cshAttackArea>();
    }

    public bool CanAttack()
    {
        return 0 < m_attackArea.colliders.Count;
    }
```

```
    public void OnVirtualPadAttack()
    {
        if (this == null) { return; }

        m_animator.SetTrigger("Attack");

        Vector3 center = Vector3.zero;
        int cnt = m_attackArea.colliders.Count;
        int cntBreak = 0;

        for (int i = 0; i < m_attackArea.colliders.Count; ++i)
        {
            var collider = m_attackArea.colliders[i];
            center += collider.transform.localPosition;

            var obj = collider.GetComponent<cshBreakableObject>();
            if (obj != null)
            {
                obj.PlayEffect();
                cntBreak++;
            }
            var enemy = collider.GetComponent<cshEnemyController>();
            if (enemy != null)
            {
                enemy.Damage();
                if (enemy.GetHP() <= 0) m_attackArea.colliders.Clear();
            }
            else
            {
                Destroy(collider.gameObject);
            }
        }
        if (cntBreak > 0) m_attackArea.colliders.Clear();

        center /= cnt;
        center.y = transform.localPosition.y;
        transform.LookAt(center);
    }

}
```

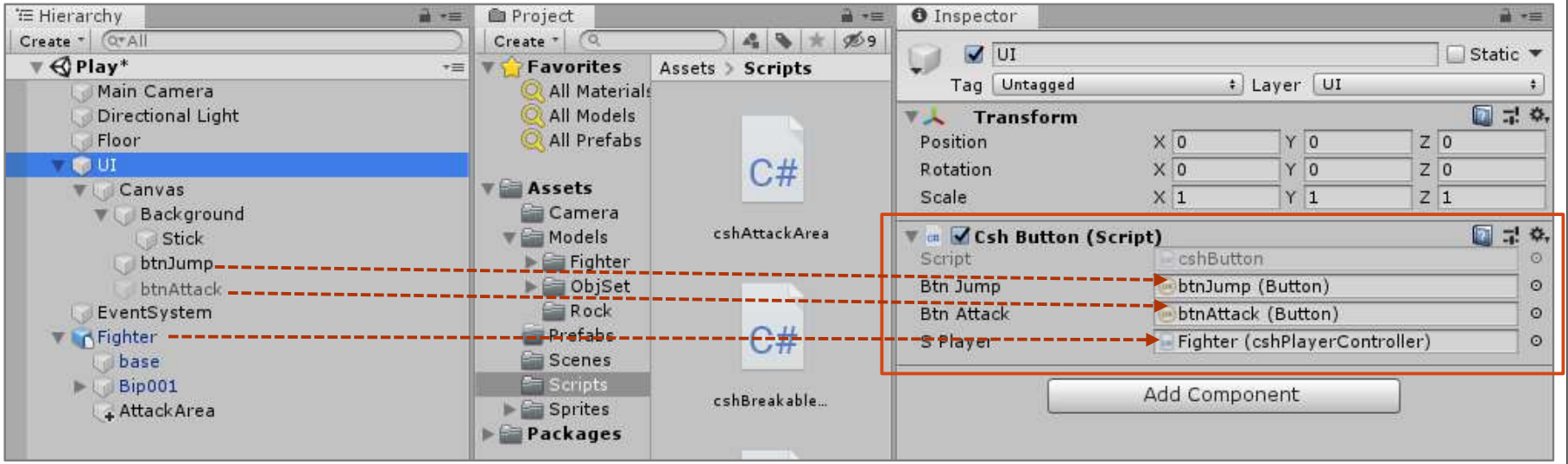
3D 콘텐츠 제작

- 적, 데미지 오브젝트 처리 스크립트 추가
  - Project → Create → C# Script
    - cshBreakableObject, cshEnemyController 각각 추가

```
public class cshBreakableObject : MonoBehaviour
{
    public GameObject destroyEffectPrefab;

    public void PlayEffect()
    {
        Instantiate(destroyEffectPrefab, transform.localPosition, Quaternion.identity);
    }
}

public class cshEnemyController : MonoBehaviour
{
    private int hp = 3;

    public void Damage()
    {
        hp--;
        if (hp <= 0)
        {
            Destroy(gameObject);
        }
    }

    public int GetHP()
    {
        return hp;
    }
}
```

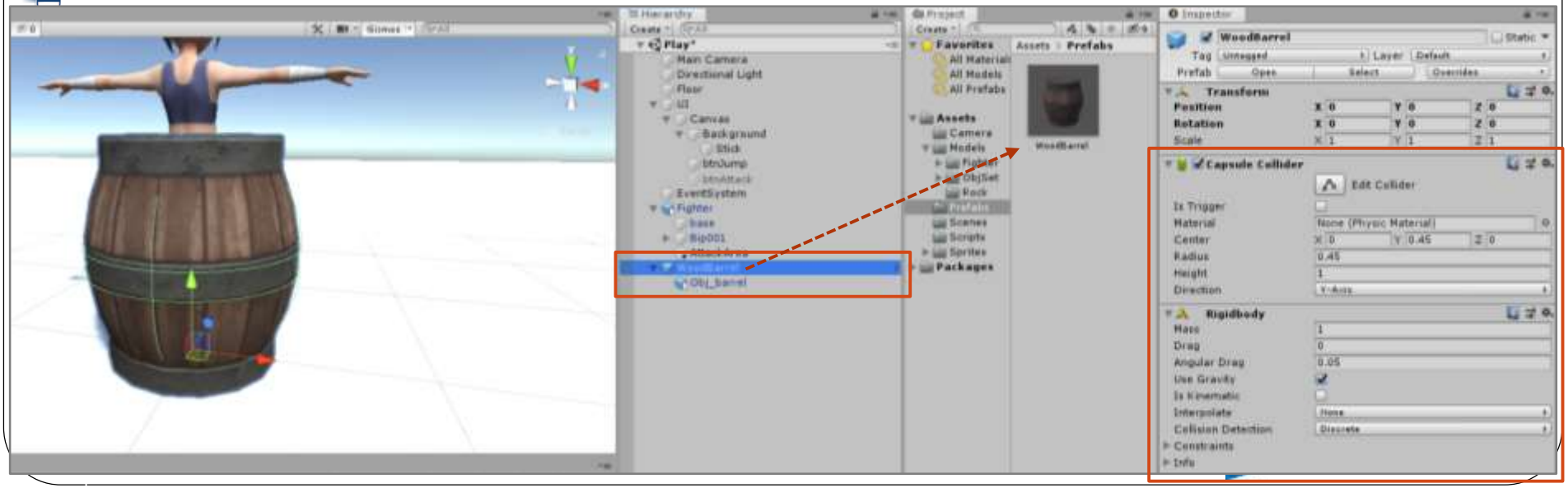
3D 콘텐츠 제작

- 버튼 스크립트 속성 설정
  - 버튼, 캐릭터 관련 속성 변수 등록



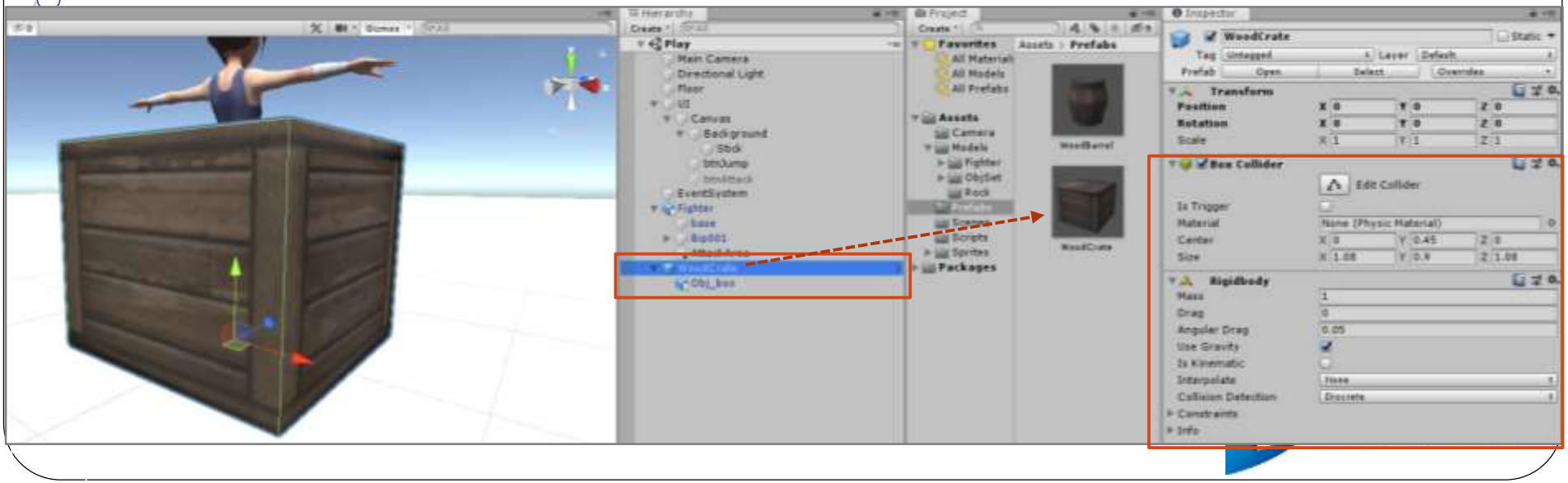
3D 콘텐츠 제작

- 나무 통 프리팹 생성
  - Hierarchy → Create → Create Empty
    - 이름: WoodBarrel
      - Project → Models → ObjSet → Obj_barrel 객체를 자식 객체로 등록
      - Capsule Collider 추가
        - Center: 0, 0.45, 0
        - Radius: 0.45
      - Rigidbody추가
    - 프리팹으로 등록



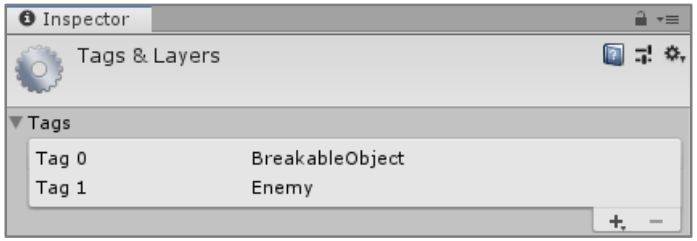
3D 콘텐츠 제작

- 나무 상자 프리팹 생성
  - Hierarchy → Create → Create Empty
    - 이름: WoodCrate
      - Project → Models → ObjSet → Obj_box 객체를 자식 객체로 등록
      - Box Collider 추가
        - Center: 0, 0.45, 0
        - Size: 1.08, 0.9, 1.08
      - Rigidbody추가
    - 프리팹으로 등록



3D 콘텐츠 제작

- 나무, 통 객체의 충돌 이벤트를 위한 태그 추가
  - Tag → Add Tag
    - BreakableObject, Enemy 태그 추가



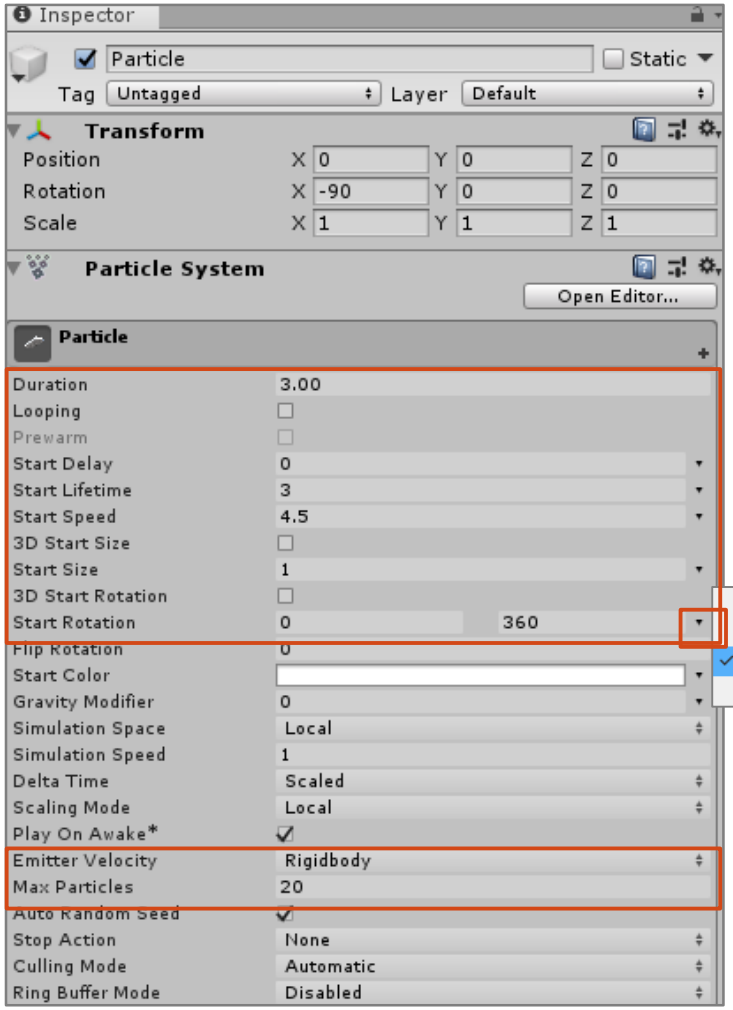
  - WoodBarrel, WoodCrate 프리팹 태그
    - BreakableObject로 변경

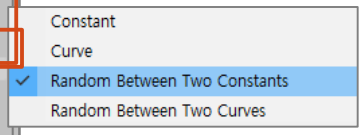
3D 콘텐츠 제작

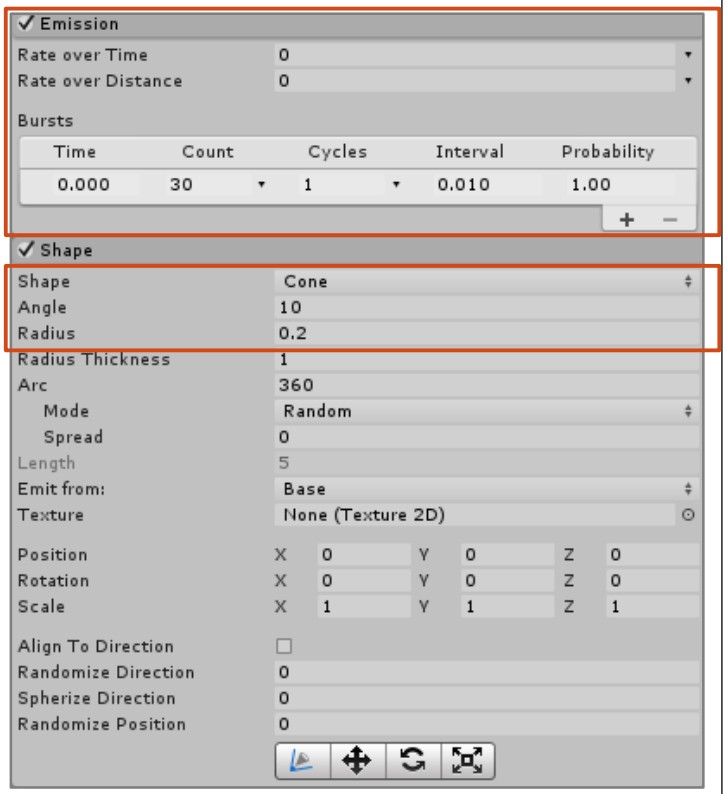
- 부숴지는 나무 파티클 생성
  - Hierarchy → Create → Create Empty
    - 이름: WoodBreak
      - Hierarchy → Create → Effects → Particle System
        - 이름: Particle
        - WoodBreak의 자식 객체로 설정
        - 파티클 시스템의 속성을 이미지를 참고하여 수정

3D 콘텐츠 제작

- 부숴지는 나무 파티클 생성





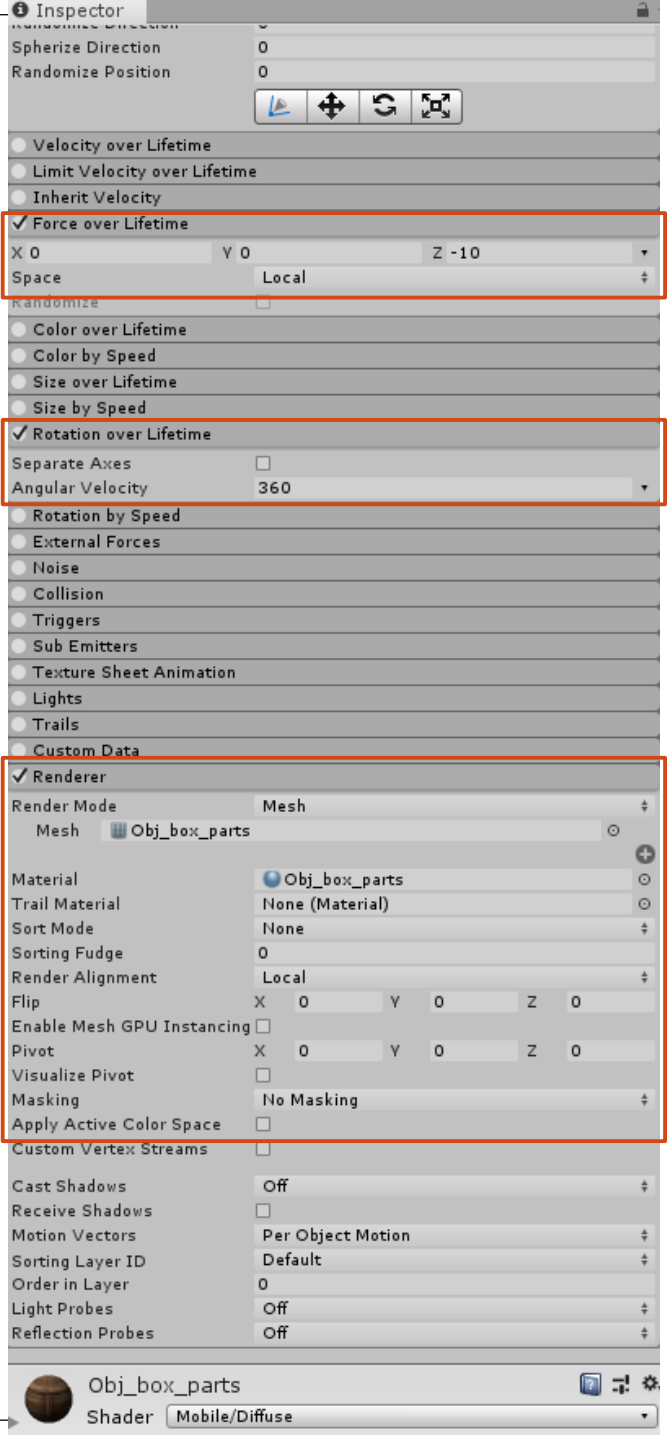


Digital Contents Design & Production

한성대학교 HANSUNG UNIVERSITY

3D 콘텐츠 제작

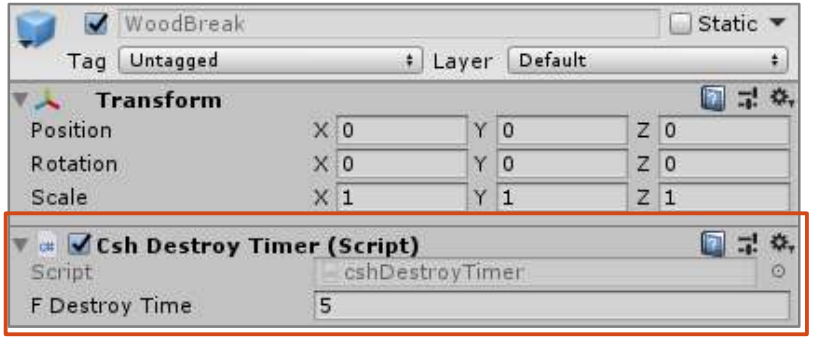
- 부숴지는 나무 파티클 생성

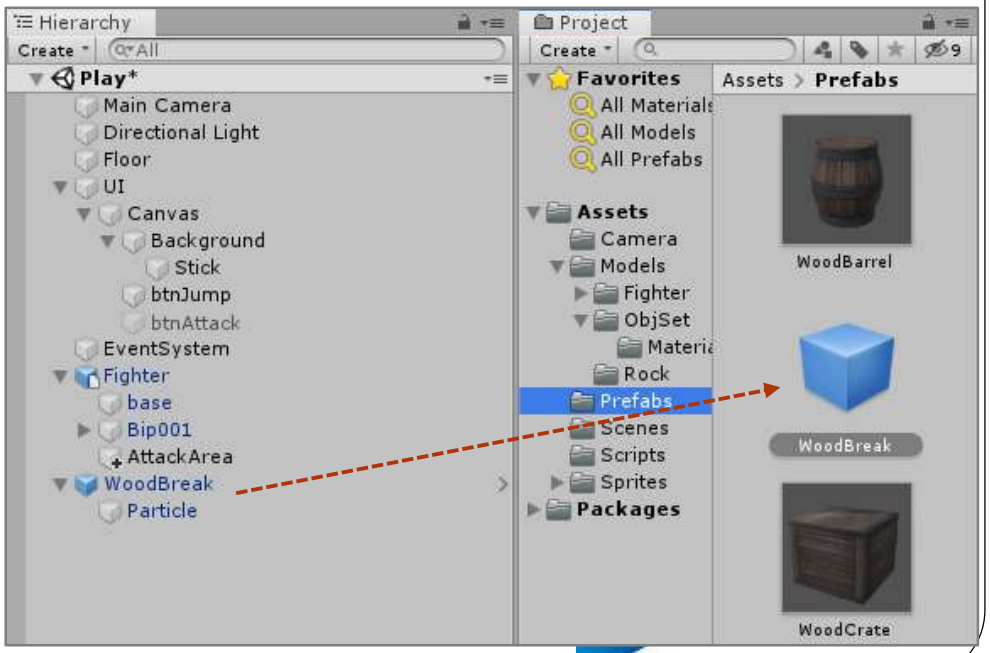


3D 콘텐츠 제작

- 부숴지는 나무 파티클 생성
  - 파티클 삭제 스크립트 추가
    - Project→ Create→C# Script
      - 이름: cshDestroyTimer
    - WoodBreak 객체에 등록
      - FDestroyTime : 5로 수정
  - WoodBreak → 프리팹 등록

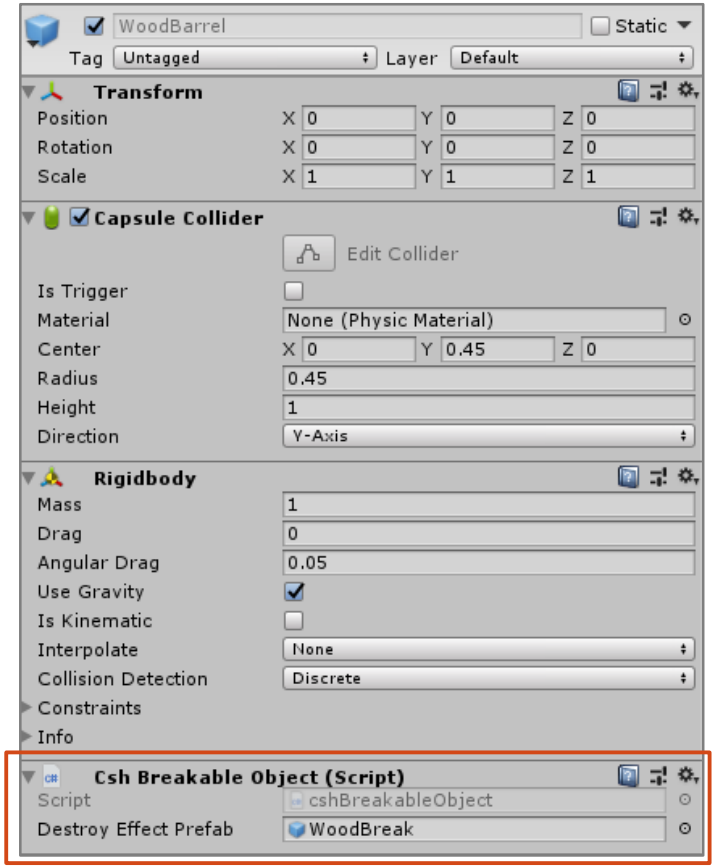
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class cshDestroyTimer : MonoBehaviour
{
    public float fDestroyTime;
    // Start is called before the first frame update
    void Start()
    {
        Destroy(gameObject, fDestroyTime);
    }

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

    }
}
```

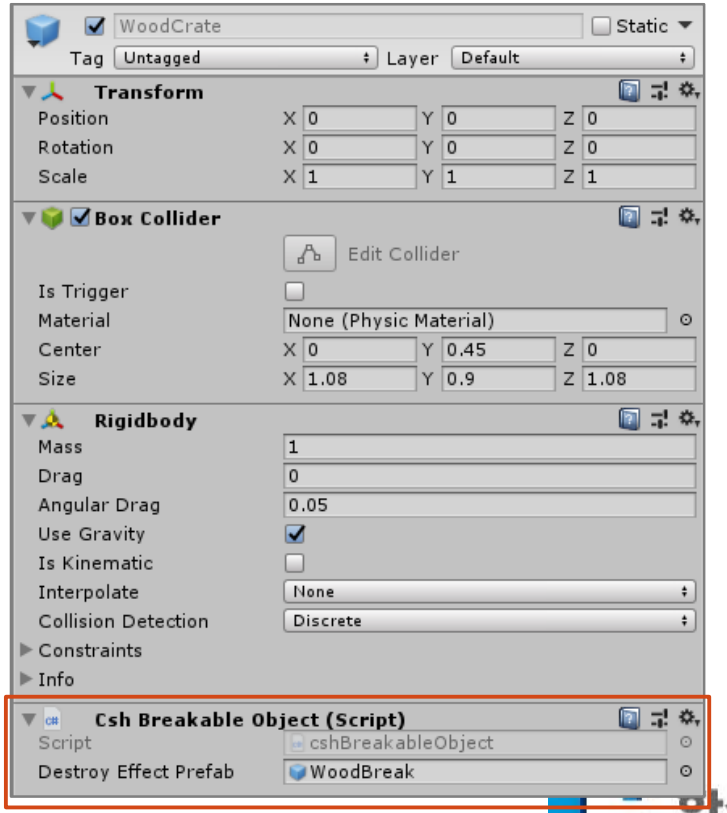




3D 콘텐츠 제작

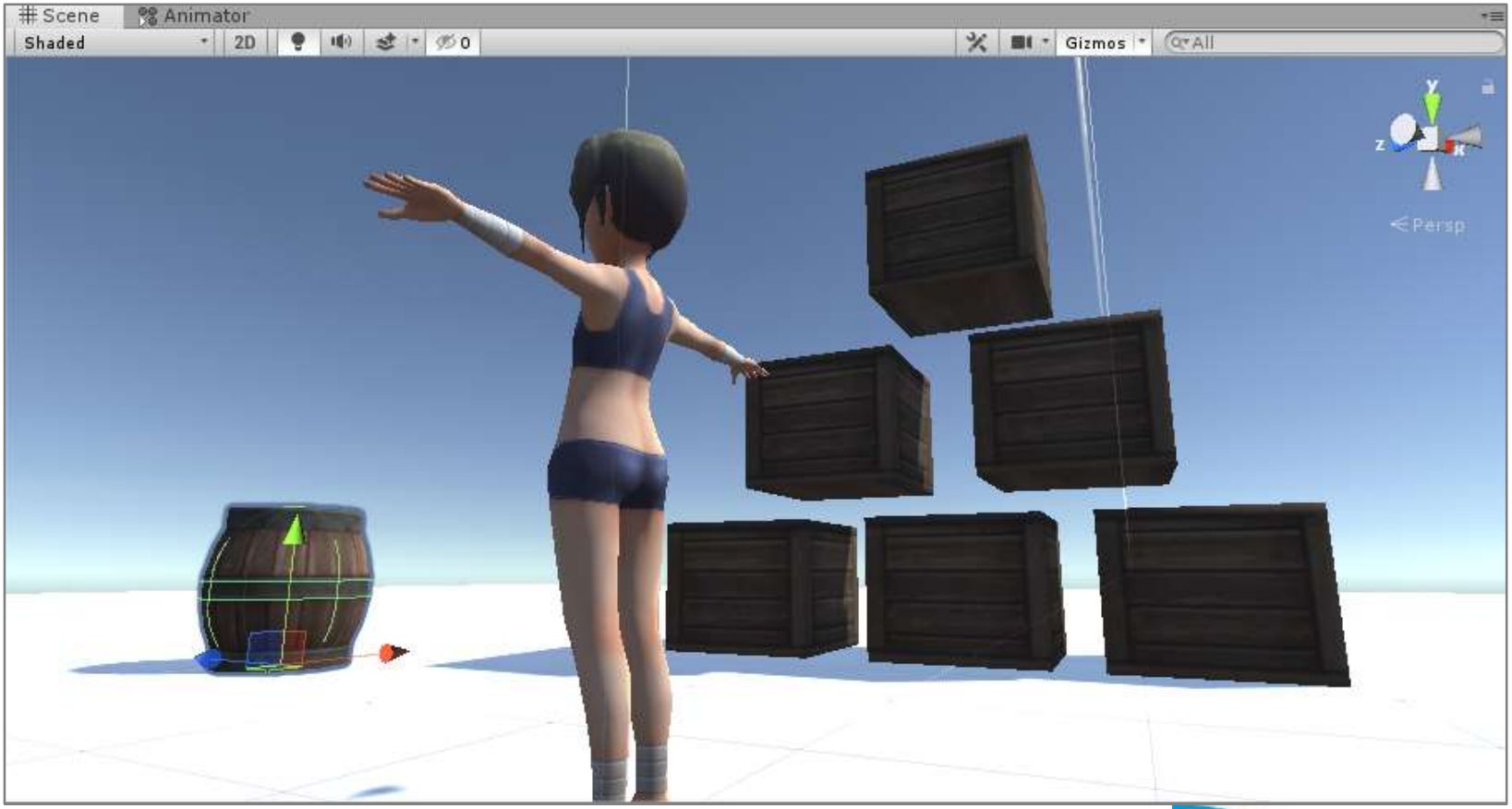
- 나무 상자, 통에 스크립트 추가
  - cshBreakableObject 스크립트를 상자, 통 프리팹에 추가
    - DestroyEffectPrefab에 WoodBreak파티클 등록





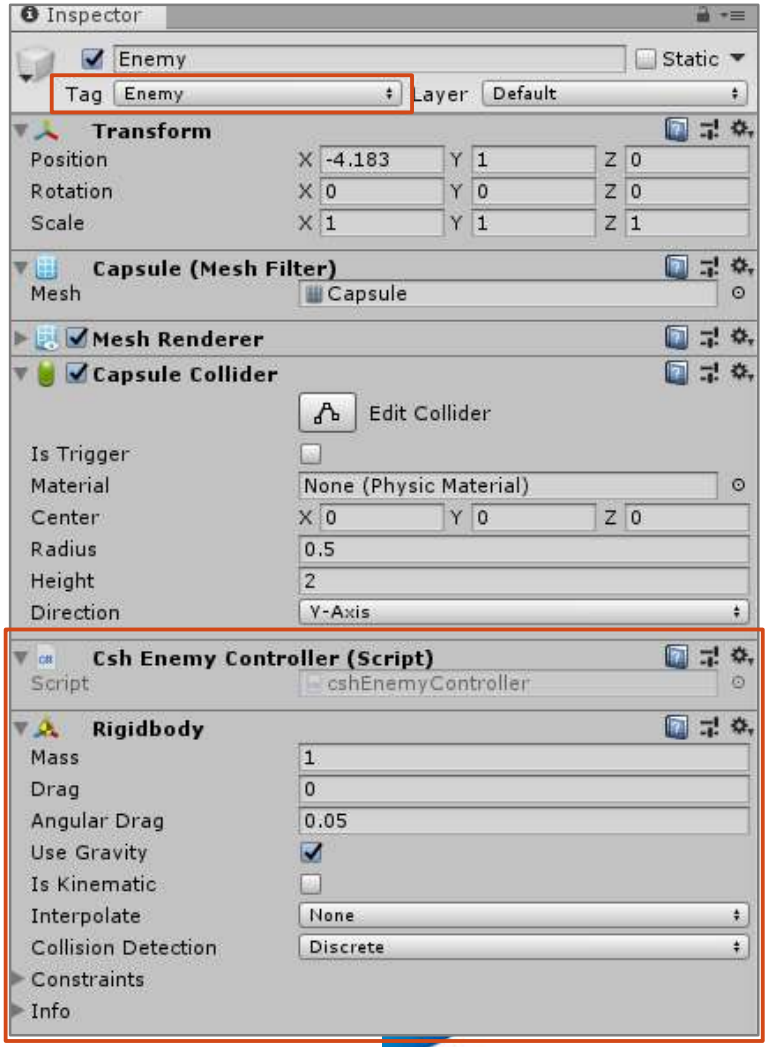
3D 콘텐츠 제작

- 통, 나무 객체를 적당한 위치에 배치



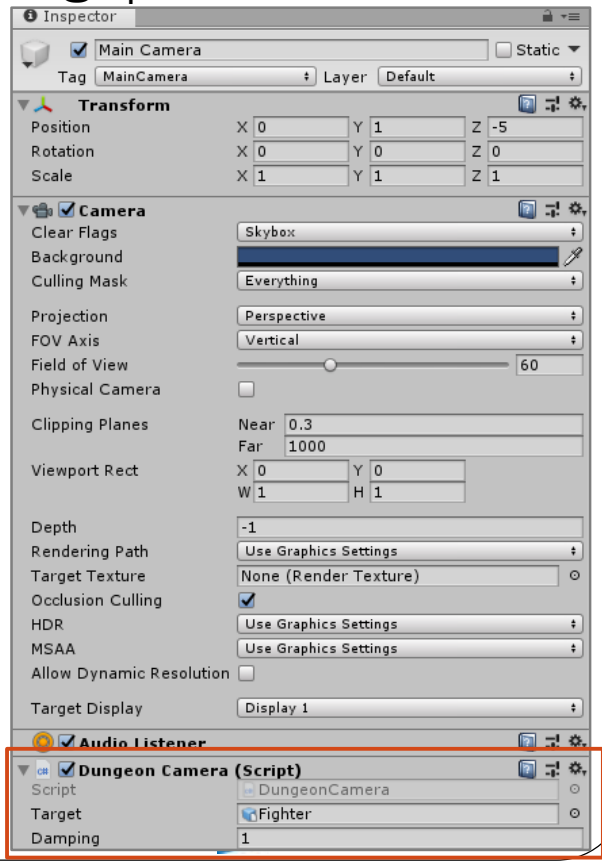
3D 콘텐츠 제작

- 적 객체 생성
  - Hierarchy → Create → Capsule
    - 이름: Enemy → 적당한 위치에 배치
    - Tag: Enemy로 수정
    - cshEnemyController 스크립트 추가
    - Rigidbody추가



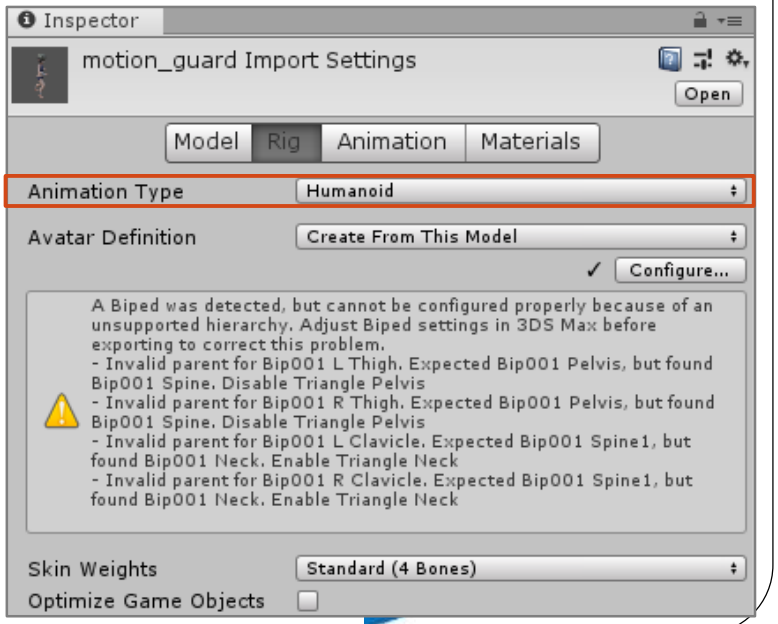
3D 콘텐츠 제작

- 카메라 설정
  - Main Camera
    - Position: 0, 1, -5
    - Project → Camera → DungeonCamera 스크립트 등록
      - Target : Fighter로 설정



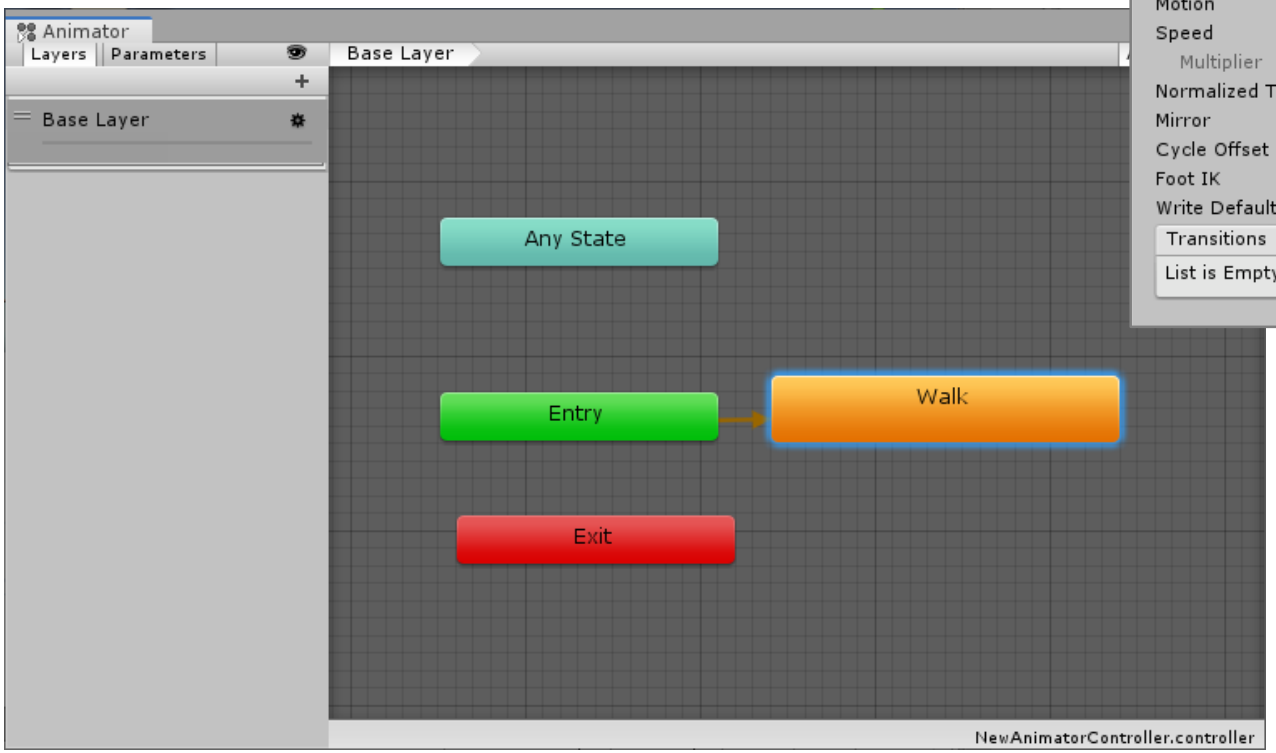
3D 콘텐츠 제작

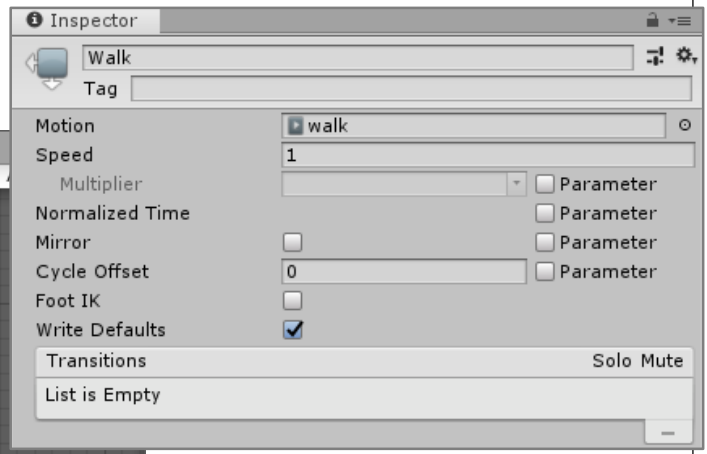
- 애니메이션 응용
  - 제작된 애니메이션 동작을 합성하여 새로운 동작을 생성
    - guard 동작과 walk 동작을 합성한 동작
    - Project → Models → Fighter
      - chara_fight 선택
        - Rig → Animation Type: Humanoid 변경
    - Project → Models → Fighter → Motion
      - motion_guard, motion_walk 동작 선택
        - Rig → Animation Type: Humanoid 변경



3D 콘텐츠 제작

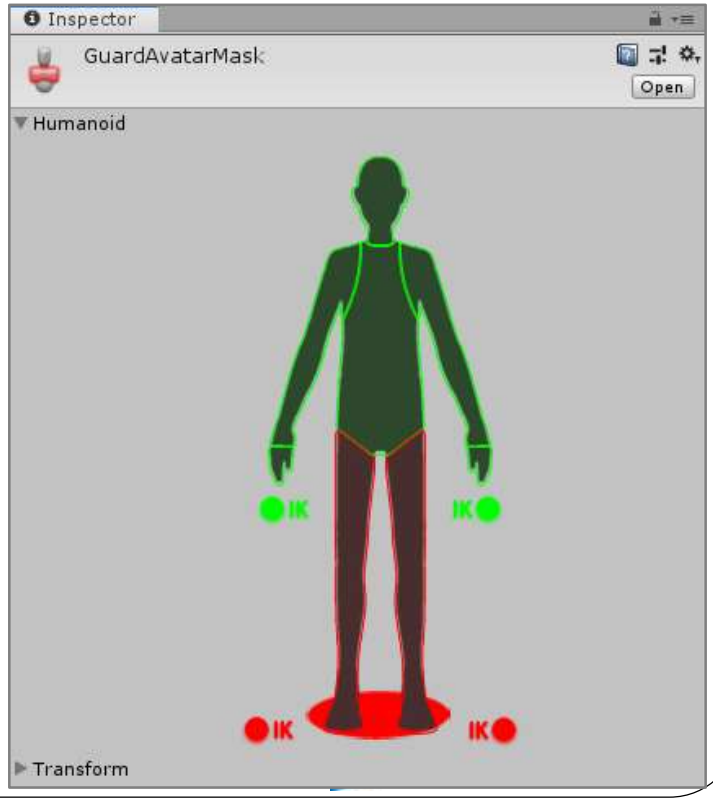
- 애니메이션 응용
  - Project → Create → Animator Controller
    - 이름: NewAnimatorController
      - Walk 동작 추가





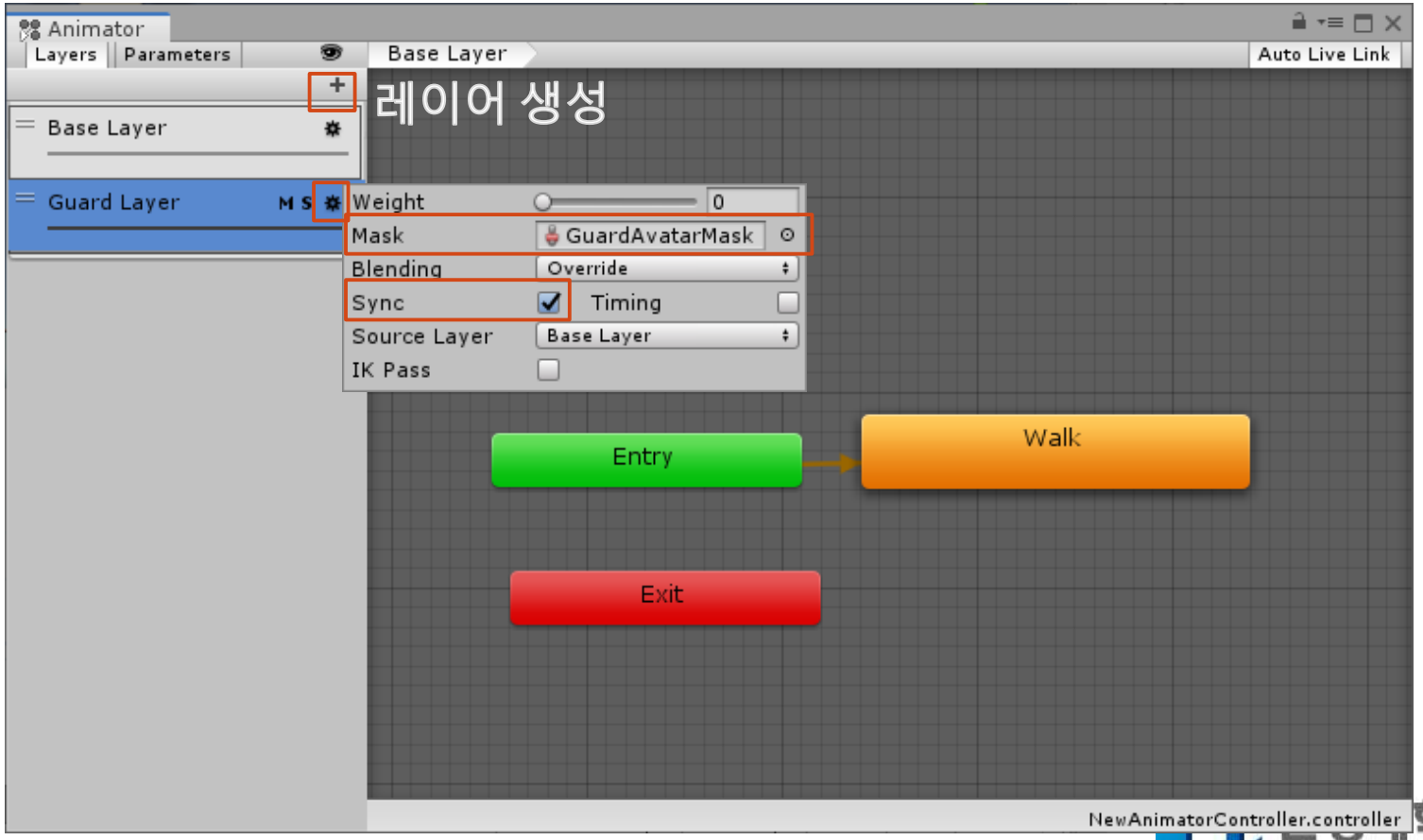
3D 콘텐츠 제작

- 애니메이션 응용
  - 합성을 위한 마스크 아바타 생성
    - Project → Create → Avatar Mask
      - 이름: GuardAvatarMask
      - Guard 동작 중 상반신만 사용하기 위해 하반신을 비활성화



3D 콘텐츠 제작

- 애니메이션 응용
  - 애니메이터 컨트롤러에 합성 레이어 생성 및 설정



3D 콘텐츠 제작

- 애니메이션 응용
  - Guard Layer의 Walk 동작
    - 합성을 위한 guard 동작을 추가
    - Guard Layer → Weight : 1로 변경

3D 콘텐츠 제작

- 애니메이션 응용
  - chara_fight 선택
    - Animator → Controller : NewAnimatorController 등록