05.가상현실 제작 실습 4/9

한 일

- ~~가상현실 제작 실습 3/9~~
    - ~~컴포넌트 할당하기~~
        - ~~컴포넌트의 자료형~~
        - ~~GetComponent<T>()~~
    - ~~Mesh~~
        - ~~Mesh 교체하기~~
    - Renderer
        - Material 교체하기

할 일

- 패키지 내보내기/불러오기
- Renderer
    - Material 교체하기
- Collider 와 Rigidbody 에 대한 이해
    - 충돌판단하기
    - 중력

과제

- 제목
  - Terrain 이 있는 씬 만들기
- 내용
  - Terrain 이 있는 씬을 만들고, Play 모드에서 키보드와 마우스로 카메라의 Position 과 Rotation 을 조작할 수 있게 함
- 요구사항
  - 과제 폴더를 만들고 관련 파일들(Scene, Terrain, Script 등)을 해당 폴더에 모을 것
    - 과제 폴더의 이름은 다음의 형식을 따를 것
    - 예) 학번이 [1234567890]이고 이름이 [홍길동]일 경우, 폴더 이름은 [HW01_1234567890_HGD]으로 정할 것
  - Terrain 게임 오브젝트를 활용하여 가상의 환경을 제작할 것
    - [Paint Terrain]의 [Set Height] 기능으로 평지를 만들 것
    - [Paint Terrain]의 [Raise or Lower Terrain] 기능으로 다채로운 고저를 표현할 것
    - [Paint Terrain]의 [Smooth Height] 기능으로 지형의 고저 중 일부를 완만하게 처리할 것
    - [Paint Terrain]의 [Paint Texture] 기능으로 3 개 이상의 [Terrain Layer]를 적용할 것
    - [Paint Tree]의 기능으로 3 종류 이상의 Tree 를 Terrain 에 심을 것
  - Scene 에 FPSController 를 추가하여 런타임(Play 모드) 중 키보드와 마우스로 Camera 의 위치와 방향을 조작하여 씬을 둘러 볼 수 있게 할 것
  - FPSController 와 충돌하면 사라지는(Destroy) 게임 오브젝트를 만들 것
  - Button 을 누르면 현재 Scene 을 다시 Loading 하게 할 것
  - 작업물을 Unity 의 패키지 내보내기 기능을 사용하여 파일(*.unitypackage)로 저장할 것
  - Play 모드에서 Scene 의 특장점을 잘 보여줄 수 있도록 조작하며 화면 캡처를(동영상) 할 것
- 제출물
  - 다음 2 가지를 제출할 것
  - 패키지 파일
    - 과제 폴더를 [Export Package]로 내보내고 Zip 포맷으로 압축하여 사이버 캠퍼스에 제출할 것

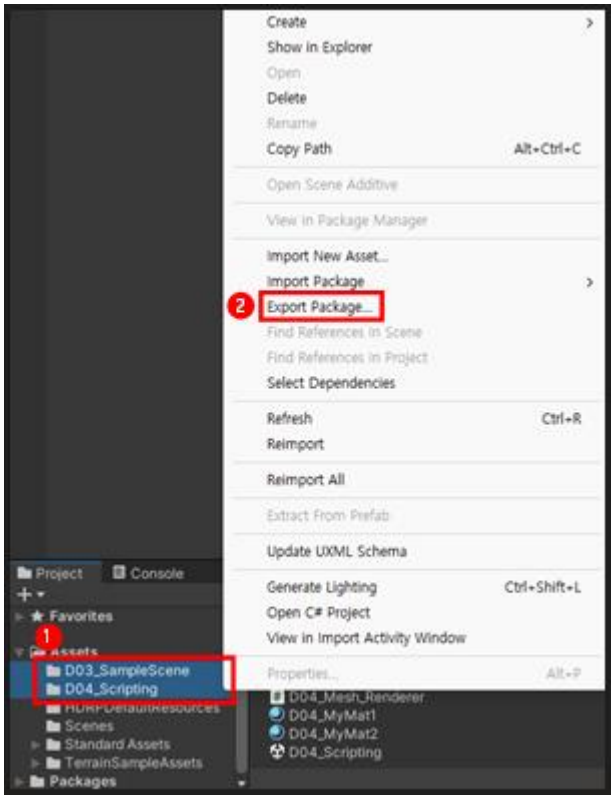
        - 용량이 제출 허용치를 초과할 경우, 다운로드 가능한 링크를 제출해도 무방함
    - 화면 캡처(동영상) 파일
        - Play 모드를 동영상으로 캡처하고 mp4 포맷으로 저장하여 제출
        - 캡션이나 안내, 설명 등은 넣을 필요 없음
        - 스마트폰으로 촬영해도 무방함
- 제출 기한
    - 일요일 23:59
- 제출 방법
    - 사이버 캠퍼스
- 주의
    - 제출한 패키지가 제대로 열리는지 반드시 확인할 것
        - 가령, 새로운 유니티 프로젝트를 만들고 해당 프로젝트에서 자신의 패키지가 제대로 열리는지 확인할 것
        - 만약 열리지 않을 경우 미제출로 처리함
    - 링크를 제출할 경우, 교수자가 다운로드 가능한 링크인지 확인할 것
        - 만약 다운로드가 불가하다면 미제출로 처리함

패키지 내보내기

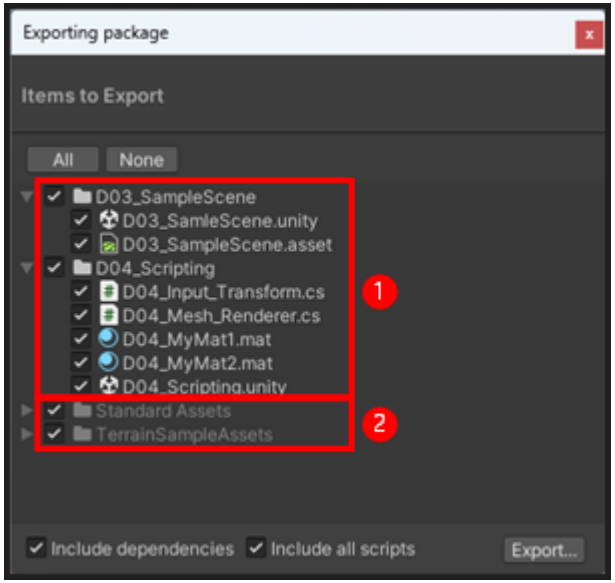
- 작은 규모의 작업물은 유니티의 패키지 내보내기/불러오기 방식으로 관리할 수 있음
- 패키지를 내보내기 할 때는 의존성 파일들의 선택에 주의를 해야 함

과정

1. 유니티 > [Project] 창에서 내보내기 할 대상 선택하고 [Export Package] 클릭



2. [Export package] 창에서 내보내기 할 목록 확인

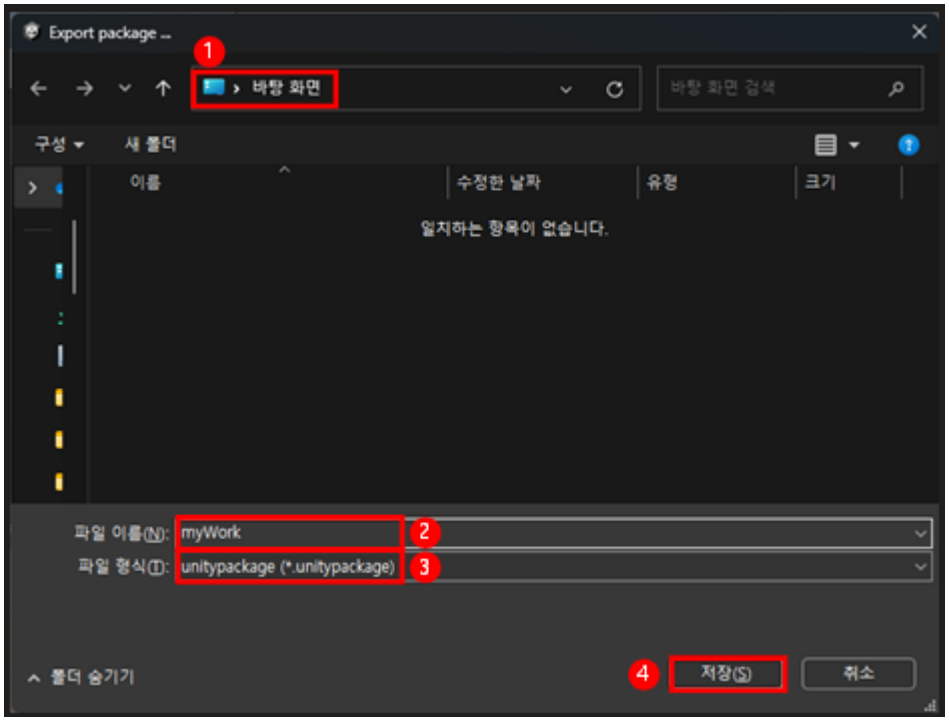


의존관계 X → 체크 X

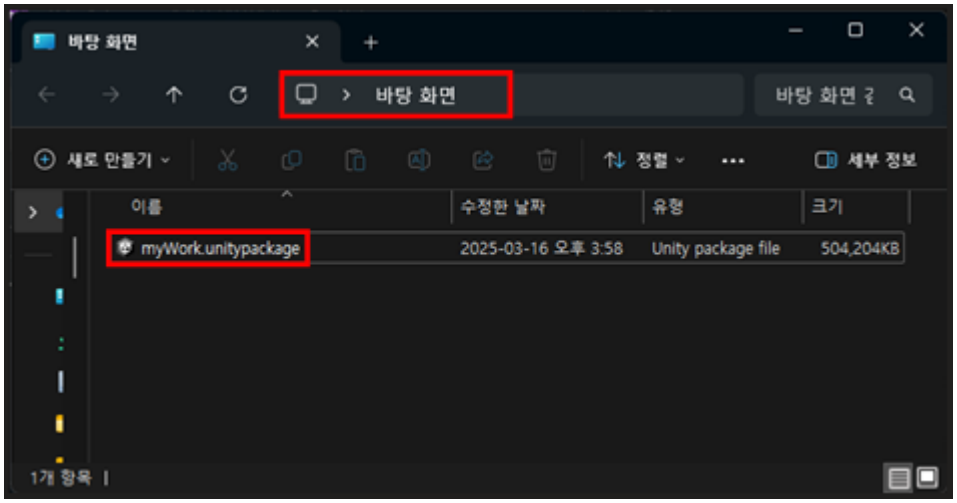
괜찮

- 내보내기 위해 선택한 대상 뿐만 아니라 대상이 의존하고 있는 다른 애셋들도 포함되어 있는 것을 볼 수 있음
- 만약, 이런 의존성 파일을 내보내기에 포함하지 않고 내보낼 경우, 불러오기를 했을 때 문제가 발생할 수 있으니 주의해야 함

3. [Export package] 창에서 목록을 확인했다면 [Export] 클릭하고 내보내기 진행

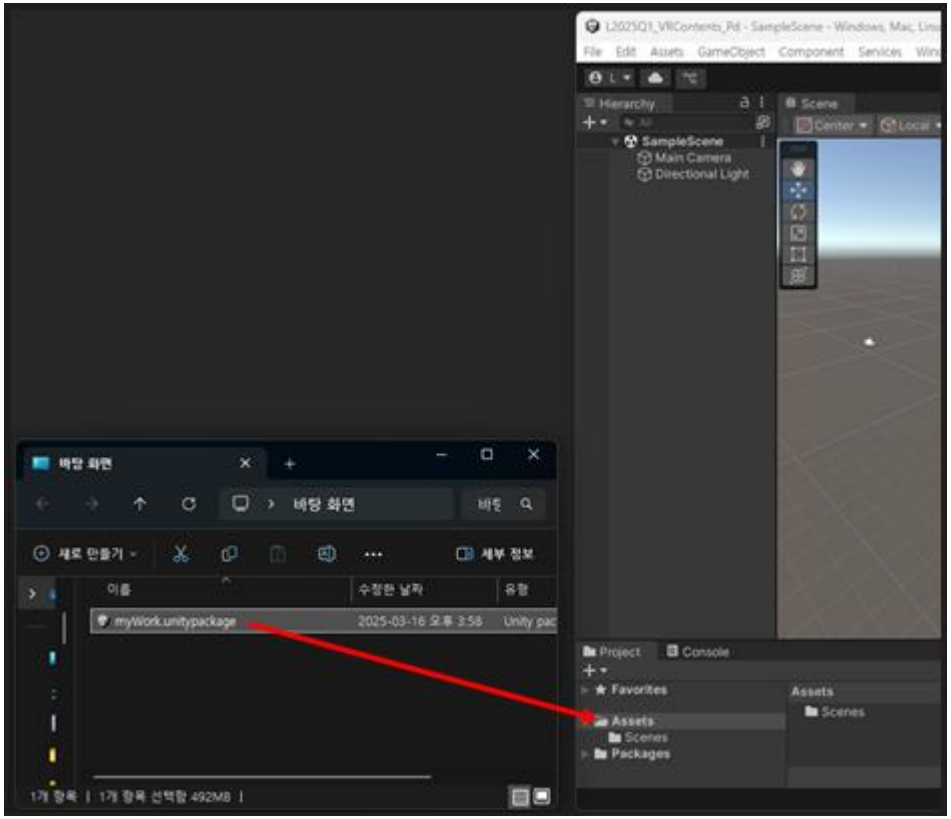


1. 저장한 위치
2. 패키지의 이름
3. 패키지의 형식
4. 저장
5. 지정한 경로에 자신의 패키지가 생성되었는지 확인

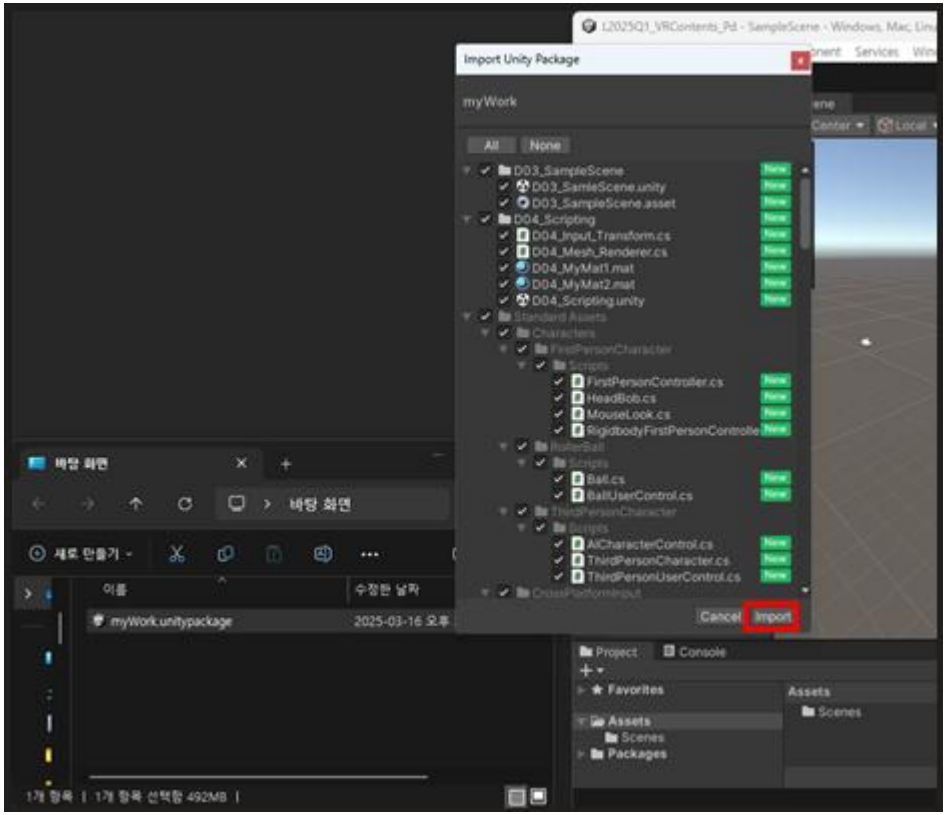

바탕 화면
바탕 화면
바탕 화면 검
새로 만들기
정렬
세부 정보
이름
수정한 날짜
유형
크기
myWork.unitypackage
2025-03-16 오후 3:58
Unity package file
504,204KB
1개 항목

패키지 불러오기

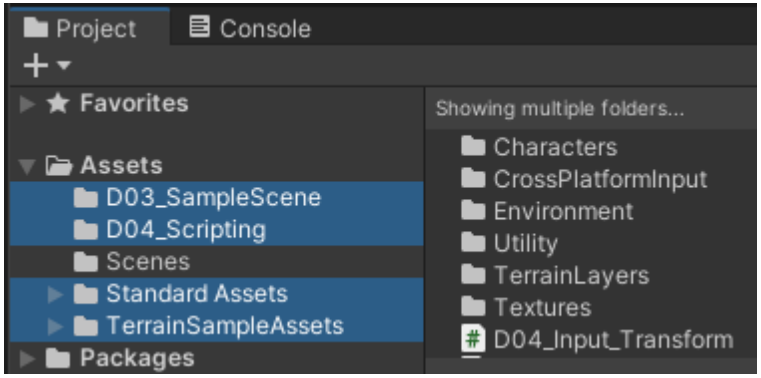
1. 불러올 패키지 파일(*.unitypackage)을 유니티 > [Project] 창 > [Assets]에 드래그드롭



2. [Import Unity Package] 창이 열리면 목록 확인 후 [Import] 클릭

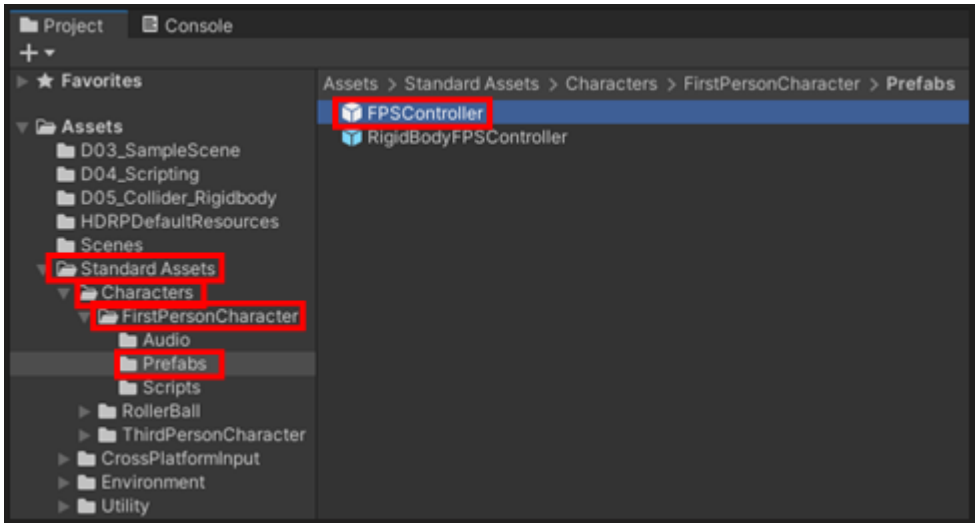


3. 불러오기를 완료하면 [Project] 창에 해당 내용물이 나타남



FPSController 활용하기

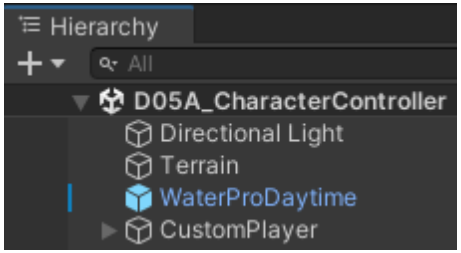
- Player 가 조종하는 Character 는 이동, 회전 뿐만 아니라 매우 다양한 기능이 필요함
    - 가령, 점프, 계단/비탈 오르기 제한 등
- [StandardAssets]에 포함되어 있는 [FPSController]는 이러한 기능이 이미 구현되어 있는 Prefab

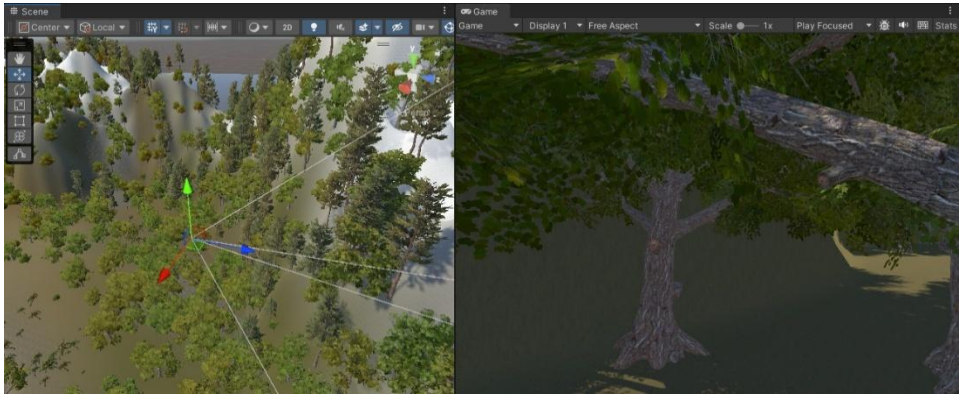


FPSController 실습

- Player 가 조종할 수 있는 Custom Player 를 구현하고, Unity 가 제공하는 [FPSController]와 비교함

- Scene 구성





- [CustomPlayer] 게임 오브젝트에 아래 스크립트 추가

- D05A_CustomPlayerController.cs

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

public class D05A_CustomPlayerController : MonoBehaviour
{
    public float translateSpeed, rotateSpeed, jumpSpeed;

    void Update()
    {
        if (Input.GetKey(KeyCode.UpArrow))
        {
            transform.Translate(0, 0, translateSpeed * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            transform.Translate(0, 0, -translateSpeed * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            transform.Rotate(0, -rotateSpeed * Time.deltaTime, 0);
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            transform.Rotate(0, rotateSpeed * Time.deltaTime, 0);
        }
        if (Input.GetKeyDown(KeyCode.Space))
        {
            Rigidbody rb = GetComponent<Rigidbody>();
            rb.AddForce(0, jumpSpeed, 0);
        }
    }
}
```

- Play 실행하고 방향키, Space 키를 눌러 움직임 확인

비교하기

- [CustomPlayer] 게임 오브젝트를 비활성화하고 [StandardAssets]에 포함되어 있는 [FPSController]를 씬에 추가
- 추가했다면 Play 모드에서 마우스와 키보드로 Player 의 시점 조작

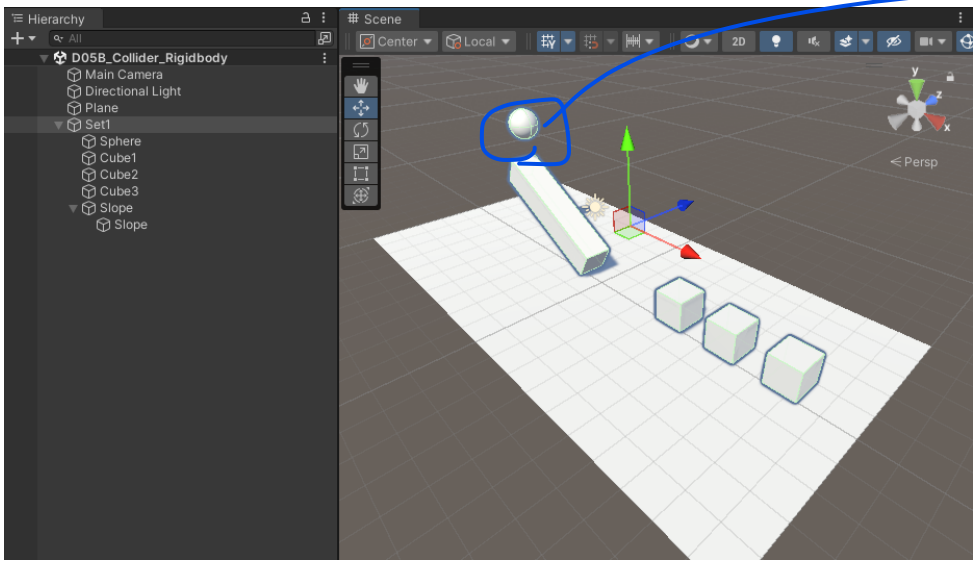
Collider 와 Rigidbody

- Collider
  - 게임 오브젝트가 다른 오브젝트와 충돌할 수 있게 해주는 컴포넌트
  - Trigger 를 활성화하면 물리적인 반응 없이 충돌을 감지할 수 있음

- Rigidbody
  - 게임 오브젝트에 물리적 속성(중력, 질량, 힘 등)을 부여하는 컴포넌트
  - 중력(Gravity)
    - Rigidbody 를 가진 오브젝트에 중력을 적용할 수 있음
  - 힘(Force)
    - 게임 오브젝트에 외부 힘을 가하여 해당 게임 오브젝트를 움직이거나 회전시킬 수 있음
  - 질량(Mass)
    - 질량은 다른 오브젝트와의 충돌 시 반응에 영향을 줌
  - 속도(Velocity)
    - 게임 오브젝트의 이동 속도와 방향을 설정할 수 있음
  - 회전(Rotation)
    - 게임 오브젝트가 회전할 수 있도록 함

실습

- Scene 구성



Mass 100

힘 부여

반 튕김

✓ 스크립트 작성하기

- Sphere, Cube1, Cube2, Cube3 에 아래 스크립트 추가
- D05B_CollisionDetection.cs

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

public class D05B_CollisionDetection : MonoBehaviour
{
    private void OnCollisionEnter(Collision collision)
    {
        print($"{gameObject.name} has detected CollisionEnter: {collision.gameObject.name}");
    }

    private void OnTriggerEnter(Collider other)
    {
        print($"{gameObject.name} has detected TriggerEnter: {other.name}");

        if (gameObject.name == "Cube2")
        {
            PushUp(500);
        }
    }

    private void Update()
    {
        if (transform.position.y < -2f)
        {
            PushUp(100);
        }
    }

    void PushUp(float speed)
    {
        Rigidbody rb = GetComponent<Rigidbody>();
        if (rb != null)
```

충돌체가 필요하다

트리거! collider → isTrigger

둘 중 하나만

```
        {
            rb.velocity = new Vector3(0, 0, 0);
            rb.angularVelocity = new Vector3(0, 0, 0);
            rb.AddForce(Vector3.up * speed);
        }
    }
}
```

- Play 모드로 전환하고 콘솔에 출력되는 메시지 확인