Unity를 이용한 2D 게임프로그래밍

Lecture 6
간단한 아케이드 게임을 위한 플레이어-적 비행체 제어

강영민

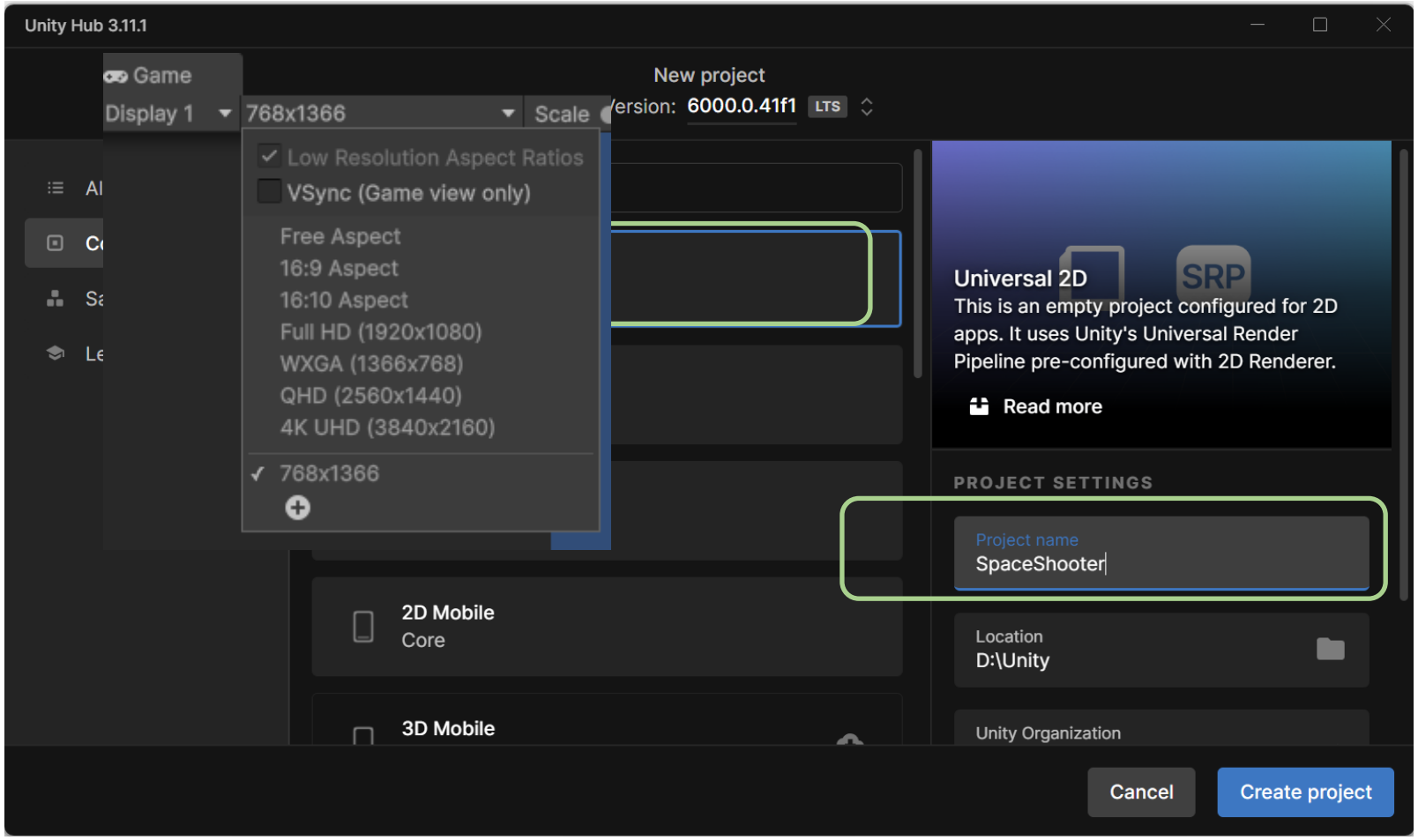
동명대학교 게임공학과

학습목표

- 전통적인 아케이드 게임 프로젝트를 만들어 본다
- 사용자 입력을 처리하기 위한 New Input System을 사용하자
- Prefab에 대해 이해하자
- Prefab 객체의 생성과 소멸
- 루프를 이용하여 적의 움직임 제어
    - 따라서 움직임 지점을 파악하여 움직이기
    - 여러 객체가 해당 움직임을 따르게 하기
    - 다수의 길을 준비하여 동작시키기
    - 적기 생성이 끝나버리지 않게 무한히 반복시키기

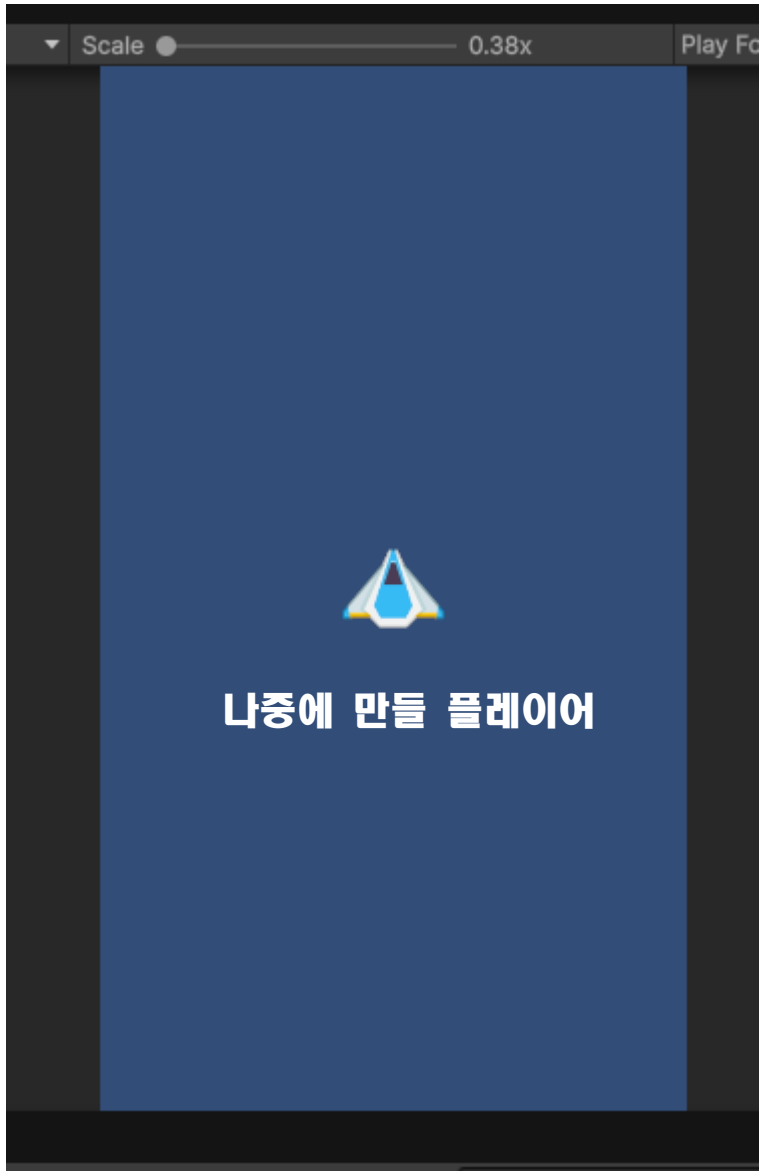
프로젝트를 생성하자

- 전통적 슈팅 게임
  - Space Shooter



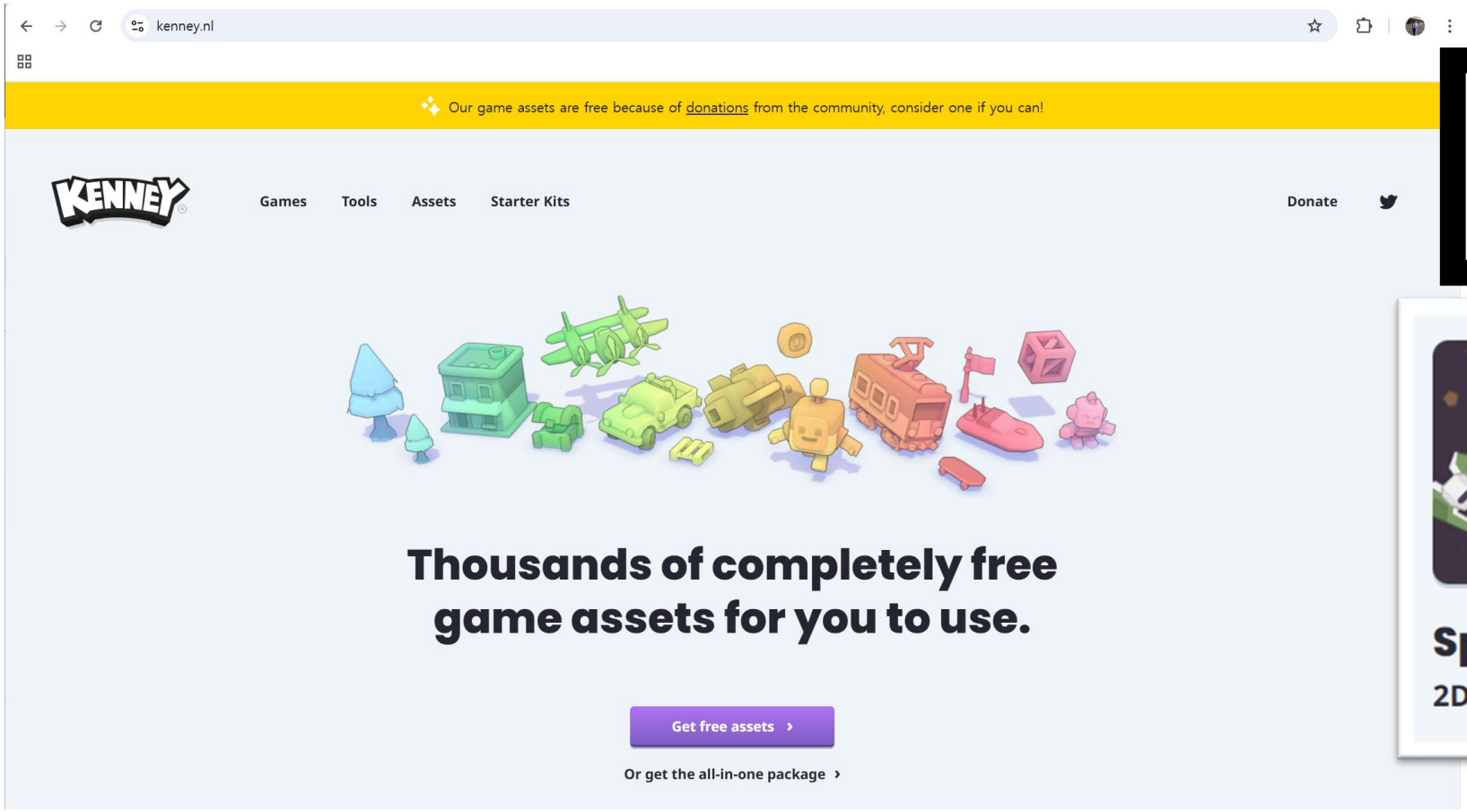
프로젝트 해상도 설정

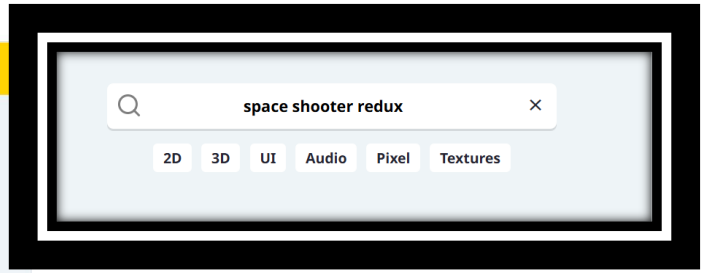
- 변경

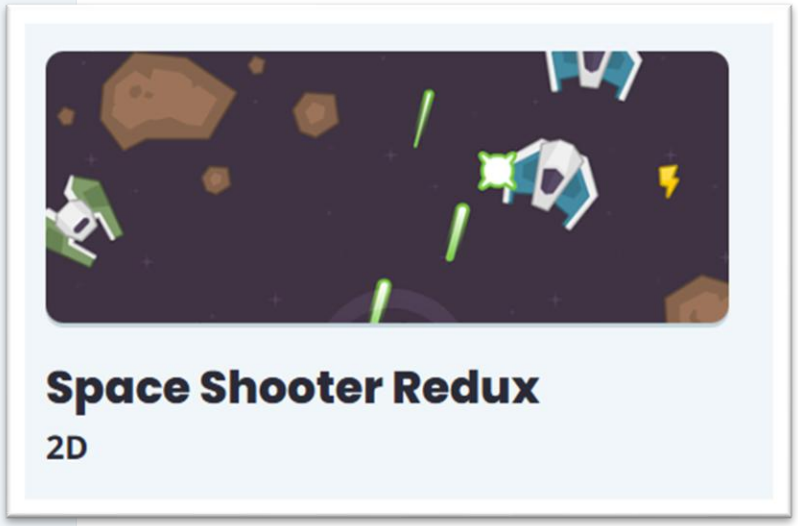


게임 리소스

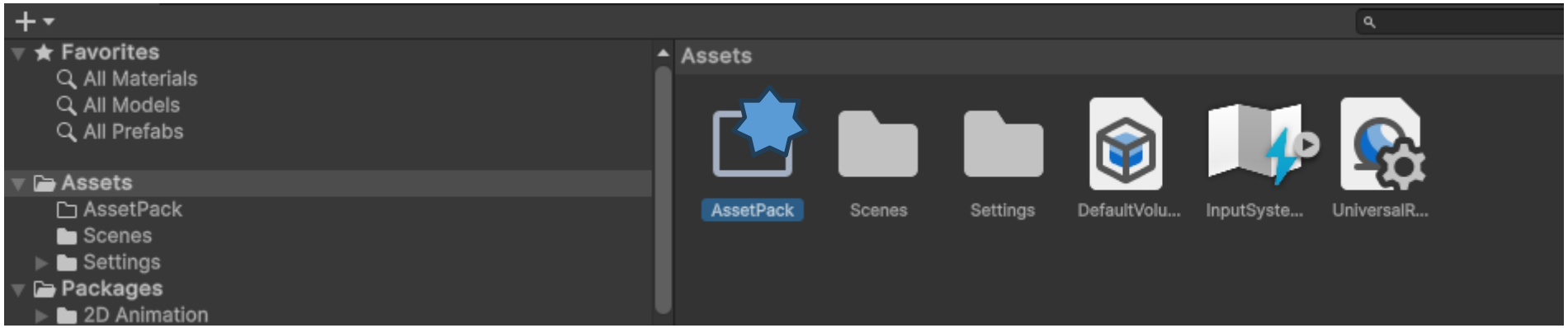
- Kenney is a good guy ☺ (kenney.nl)

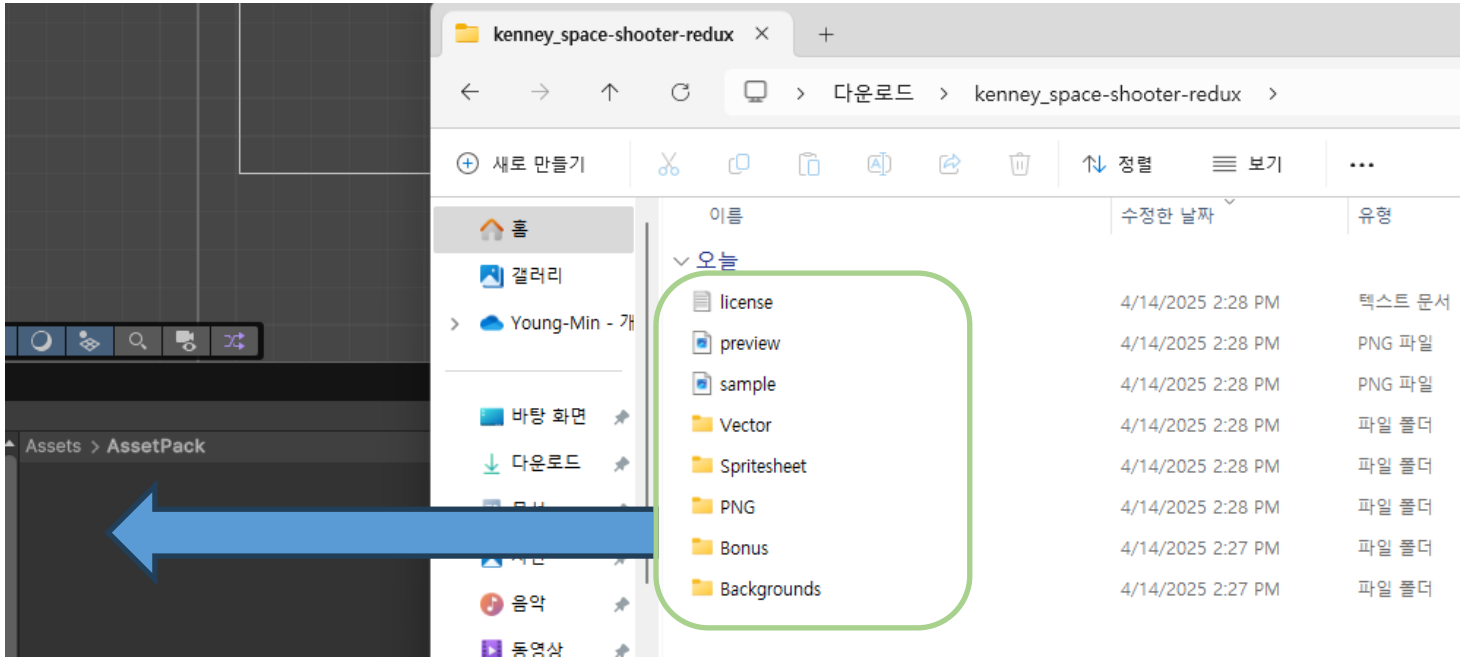






Import assets!

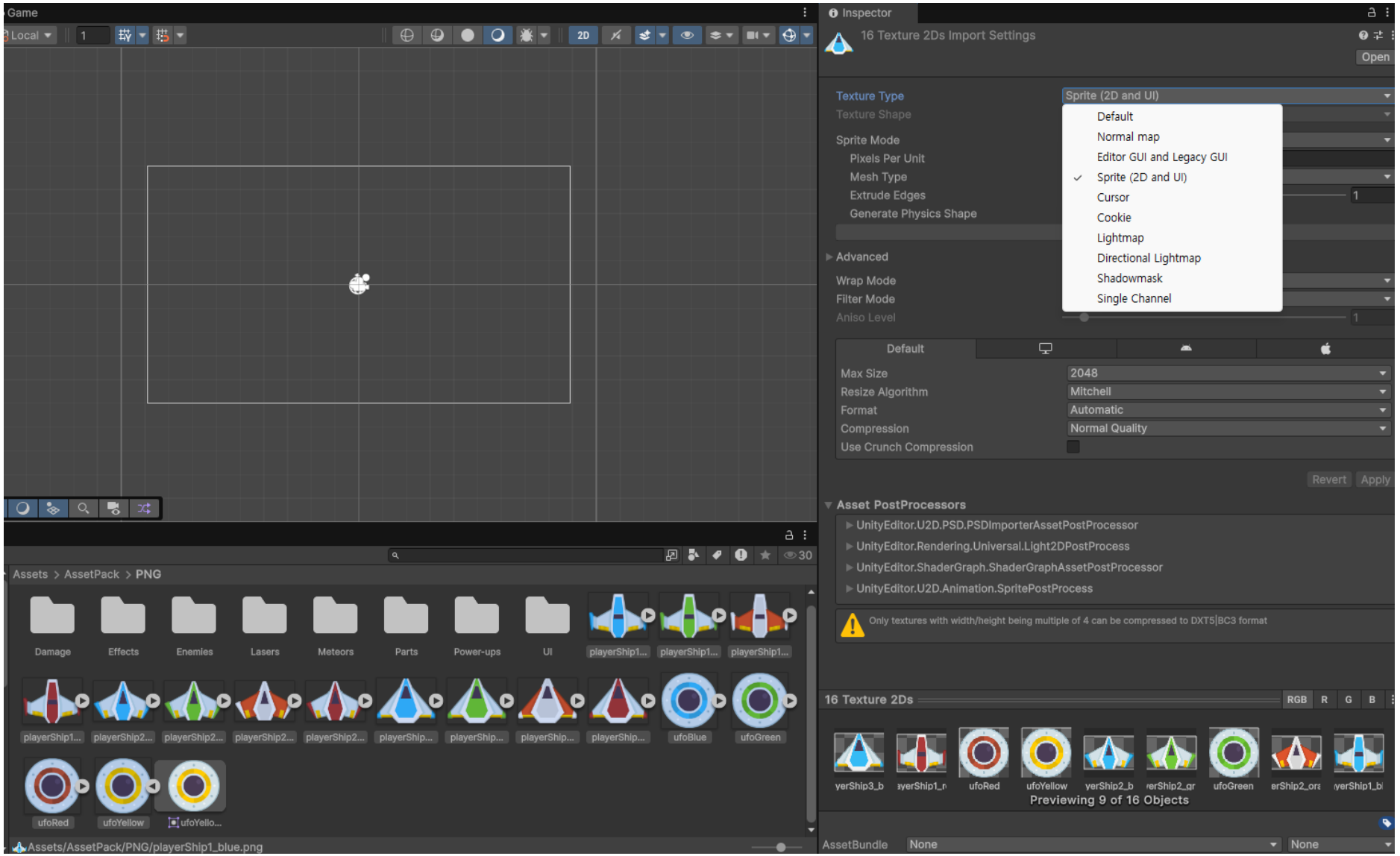




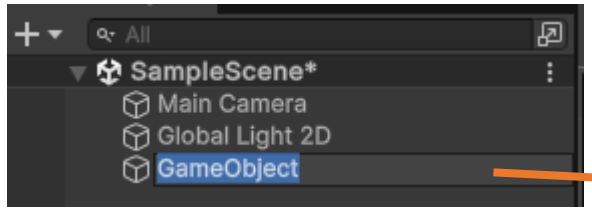
아주 Nice!


Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
AssetPack
Backgrounds
Bonus
PNG
Damage
Effects
Enemies
Lasers
Meteors
Parts
Power-ups
UI
Spritesheet
Vector
Assets > AssetPack > PNG
Damage
Effects
Enemies
Lasers
Meteors
Parts
Power-ups
UI
playerShip1...
playerShip1...
playerShip1...
playerShip1...
playerShip2...
playerShip2...
playerShip2...
playerShip2...
playerShip...
playerShip...
playerShip...
playerShip...
ufoBlue
ufoGreen
ufoRed
ufoYellow

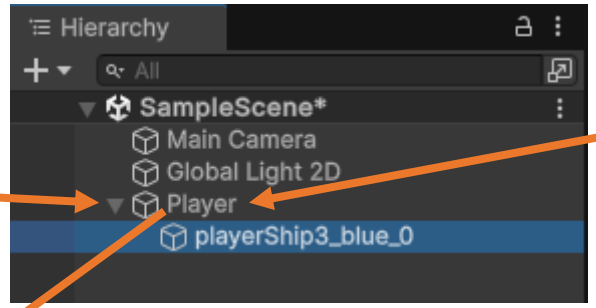
PNG 파일들이 Sprite Texture type임을 확인



Prefab 제작 1 – Player

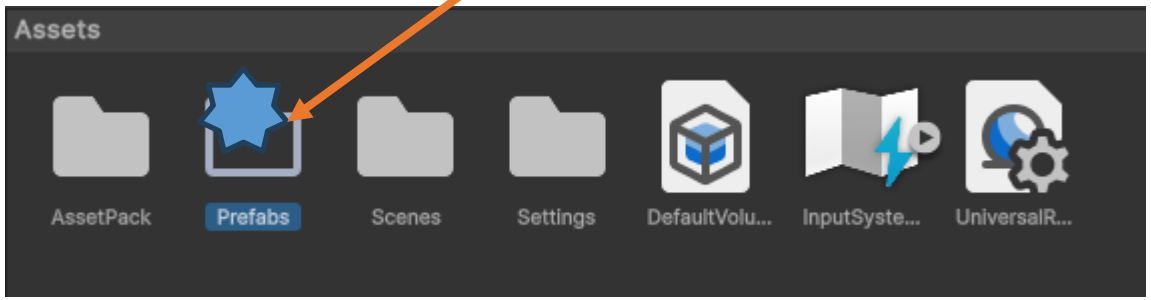


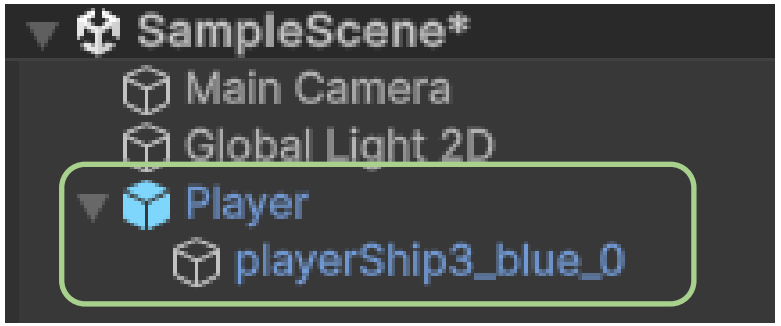
1. Rename



2. Drag & Drop

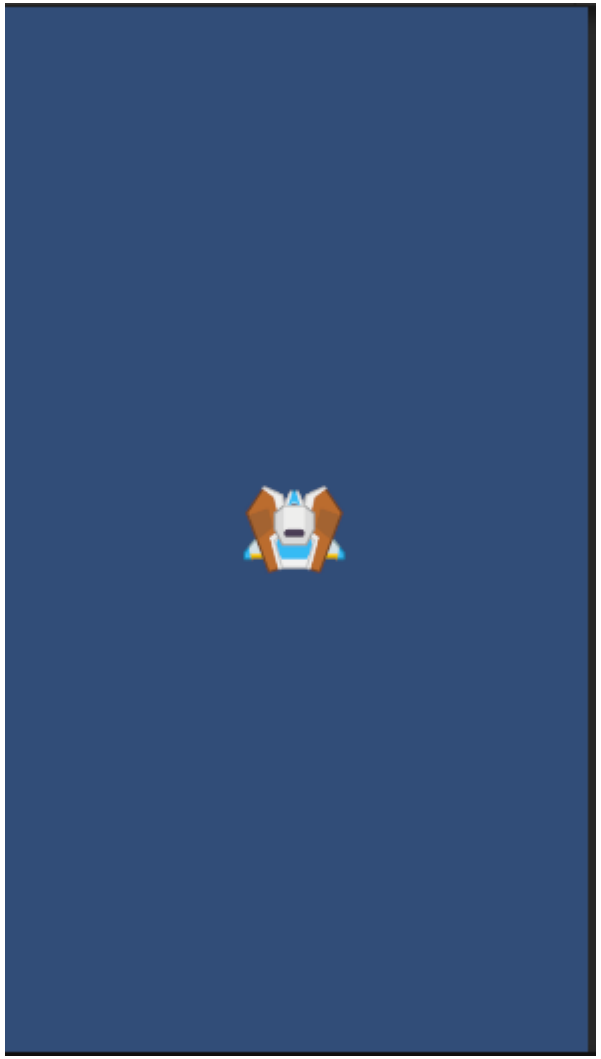
3. Create a folder for Prefabs and
Drag & Drop “player”

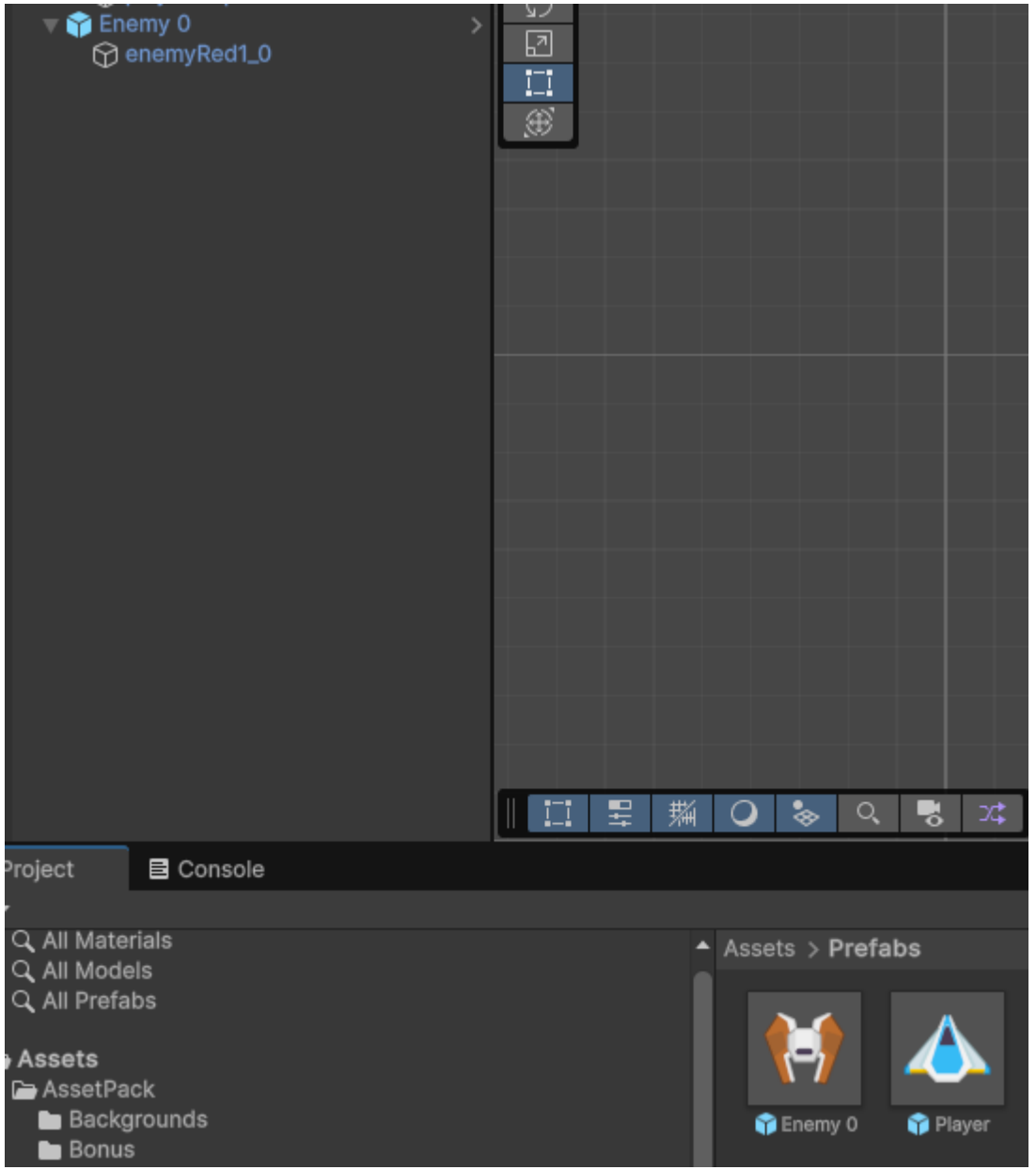




4. Prefab in Hierarchy

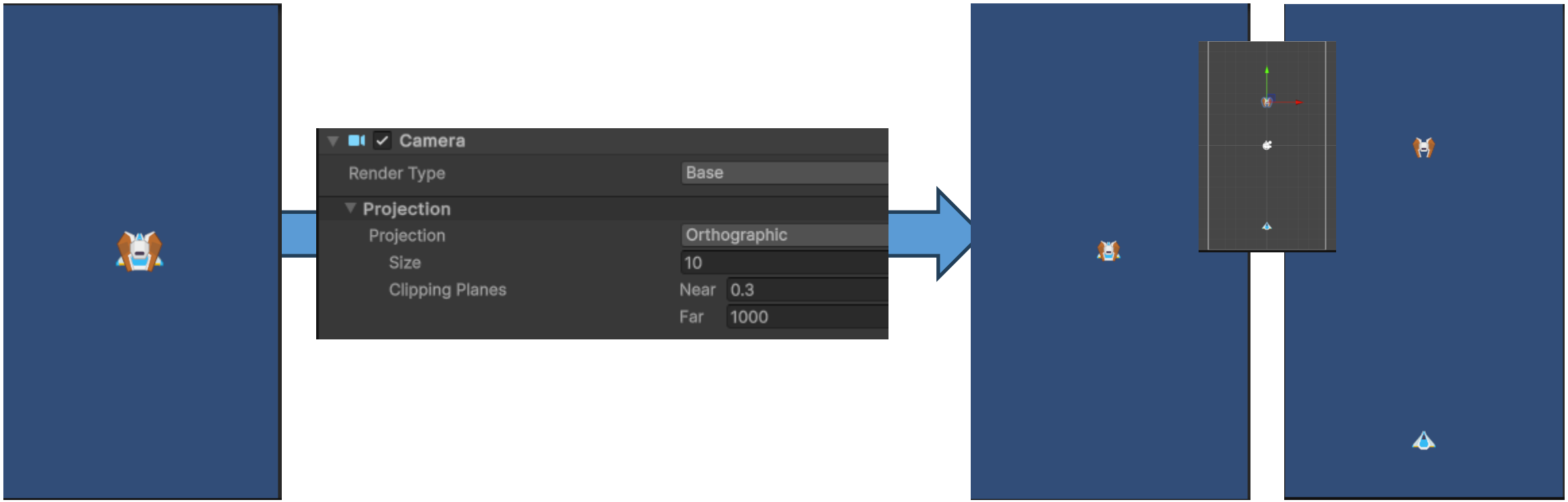
Prefab 제작 2 – Enemy 0





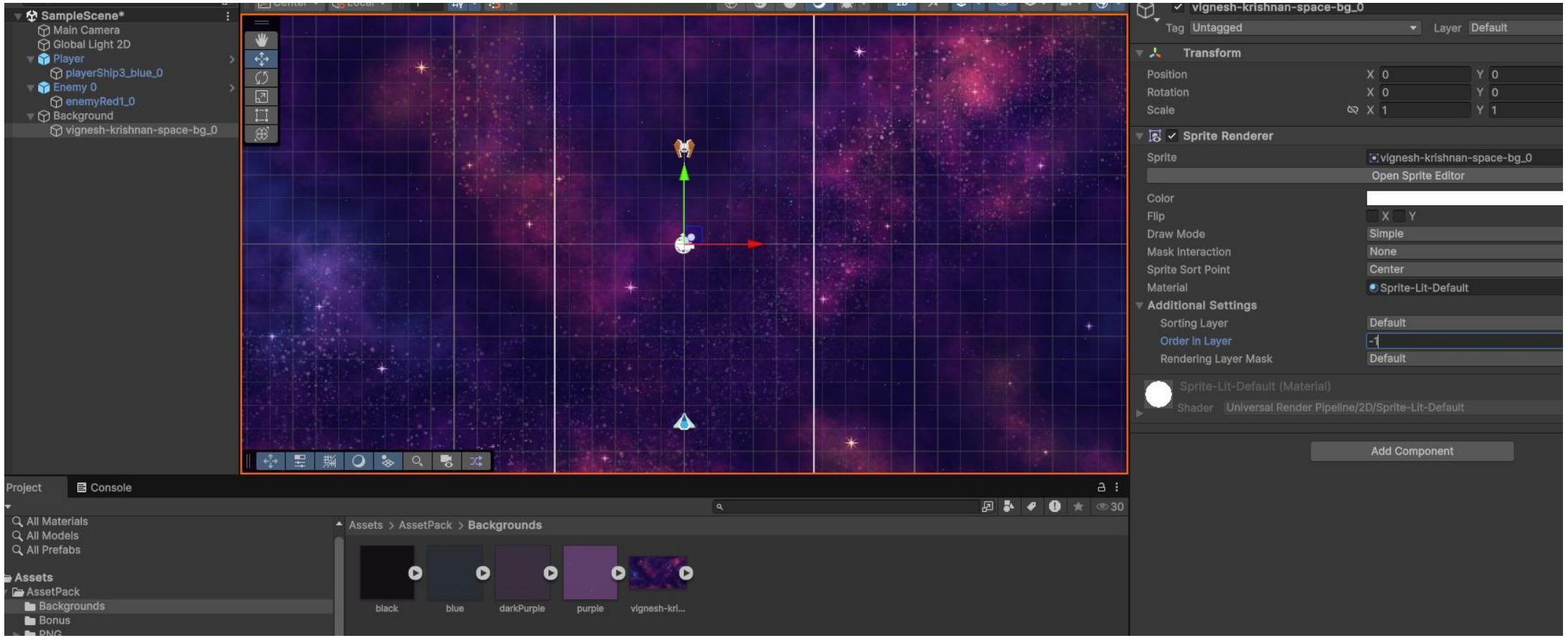
문제점

- 게임 공간에 비해 캐릭터가 너무 크다 → 카메라 설정 변경
- 두 캐릭터가 같은 위치를 점유 중 → 프리팹의 위치 변경



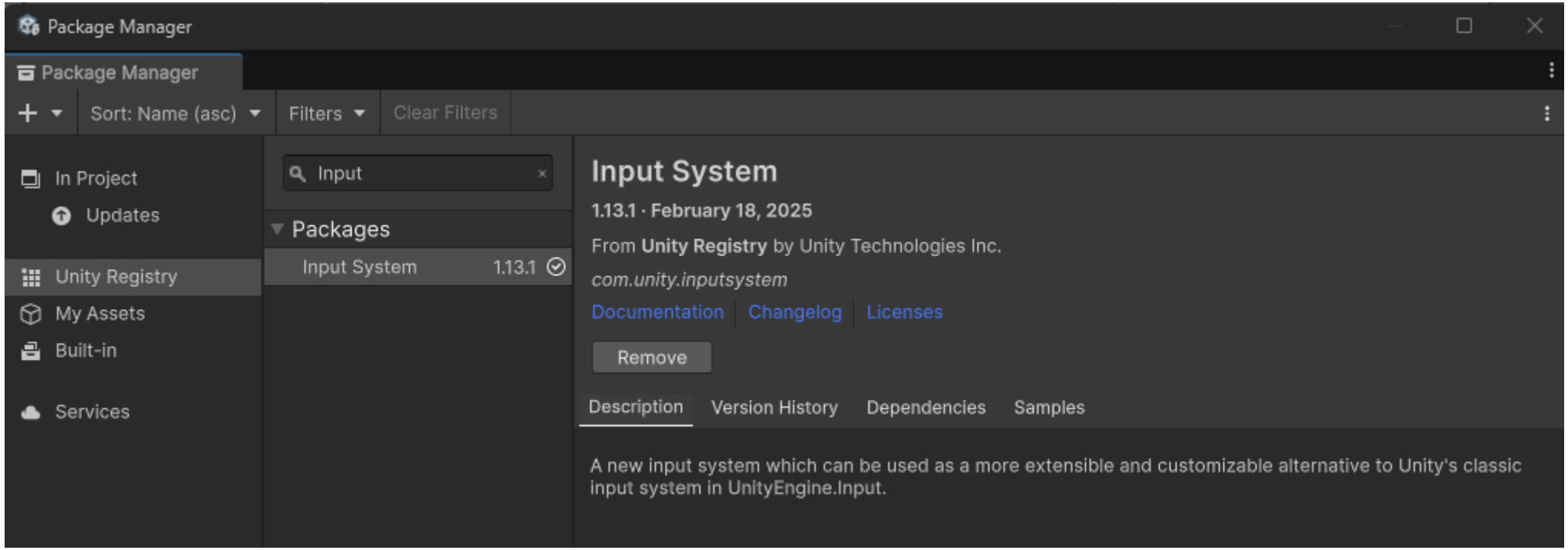
배경 넣기

- 원하는 이미지로 배경 설정해 보기
- 배경은 sorting order를 -1로 설정하여 다른 것들을 가리지 않게 함



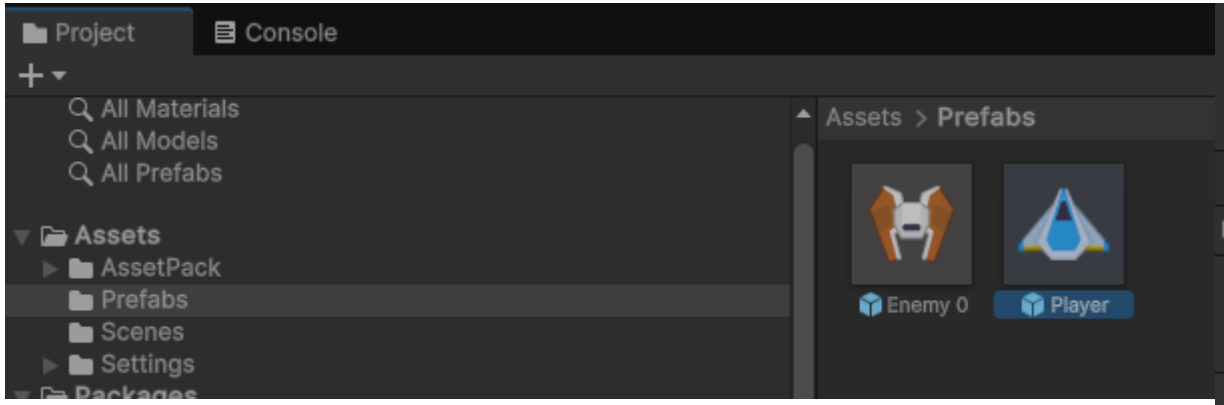
입력 시스템 – New Input System을 사용해 보자

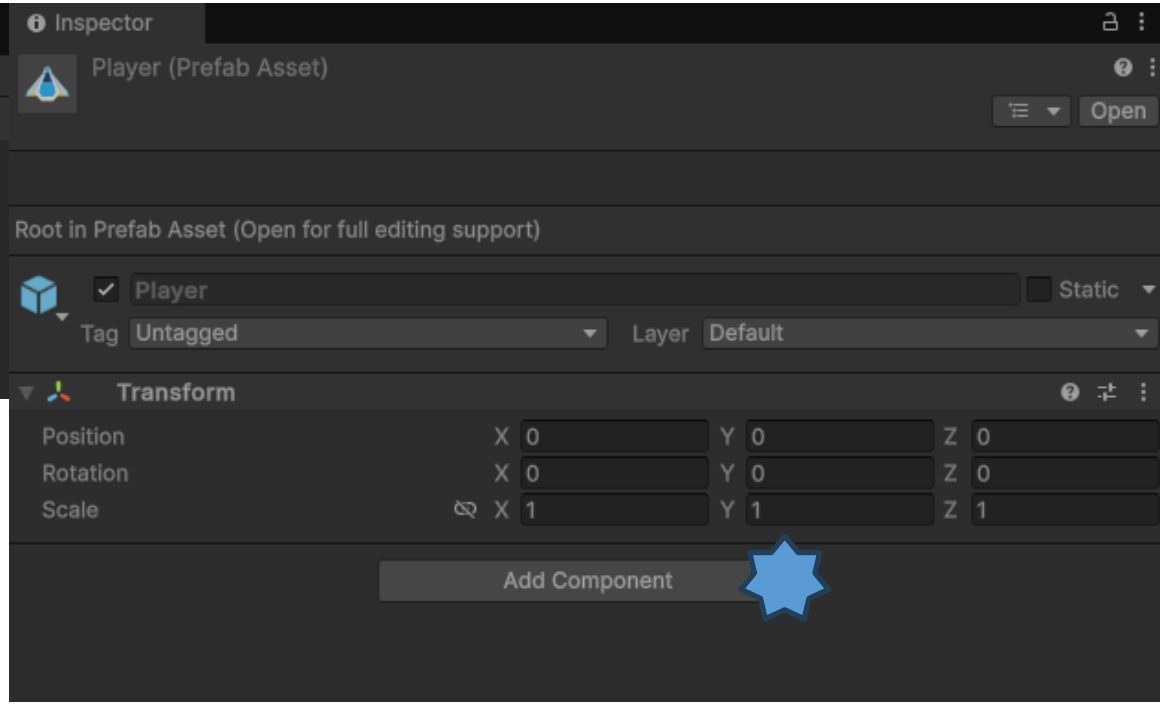
- Windows → Package Manager
  - Input System 패키지 설치를 확인 (없으면 설치)

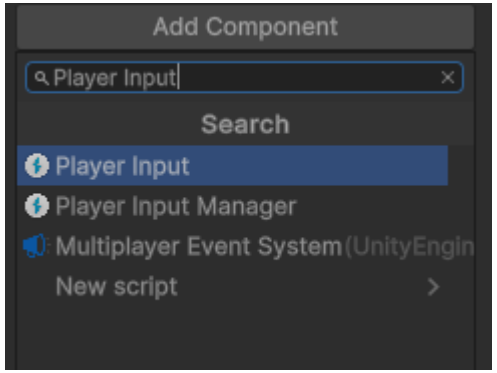


입력 시스템을 플레이어에 적용

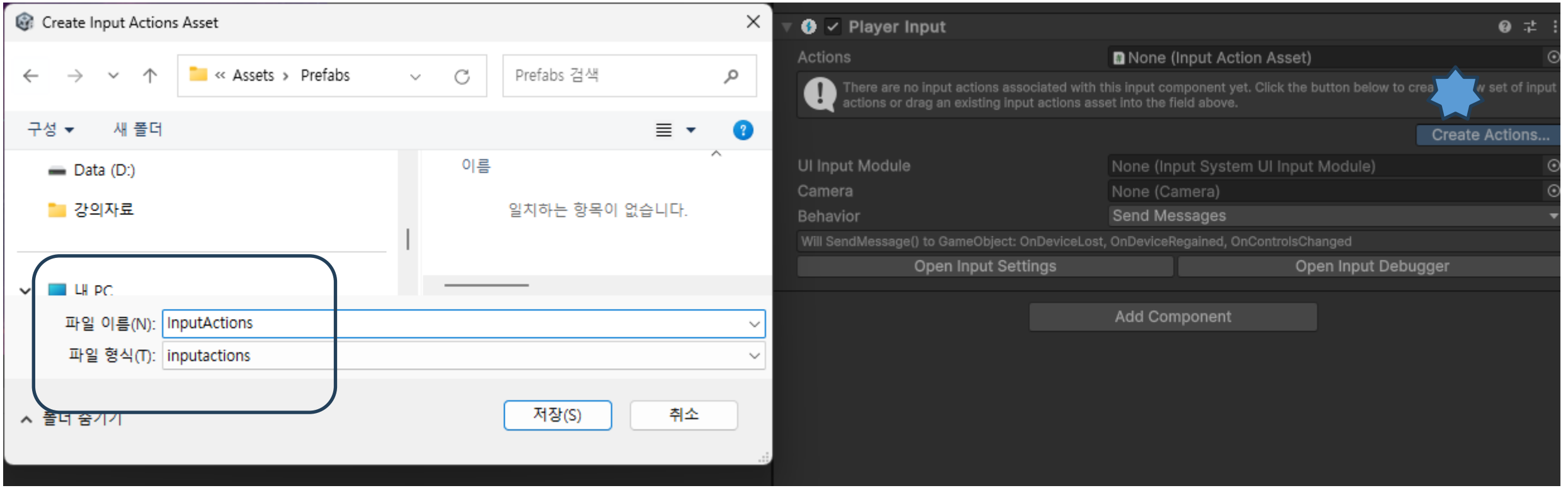
- Hierarchy에 있는 특정 객체가 아니라 Prefab을 변경



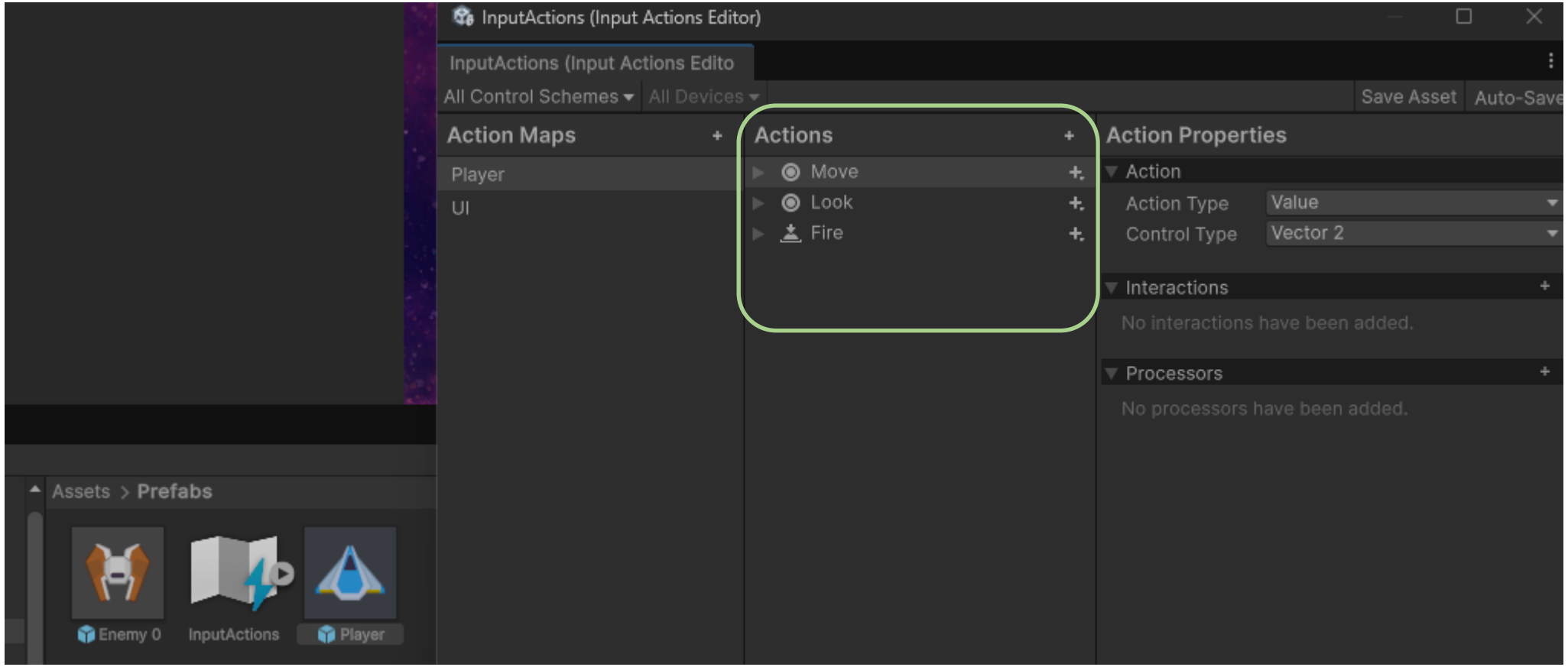




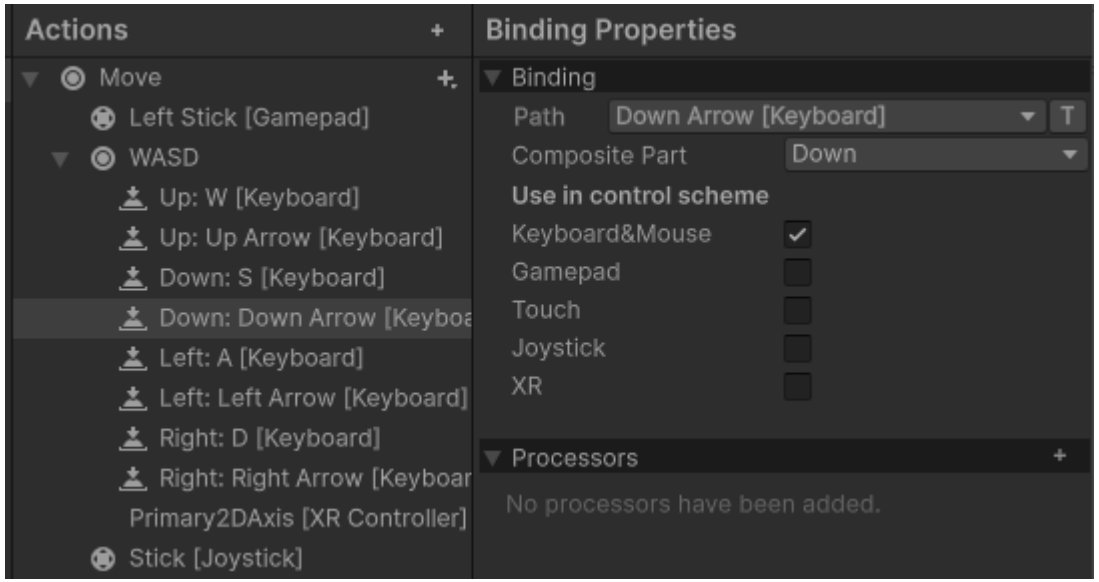
Player Input 설정



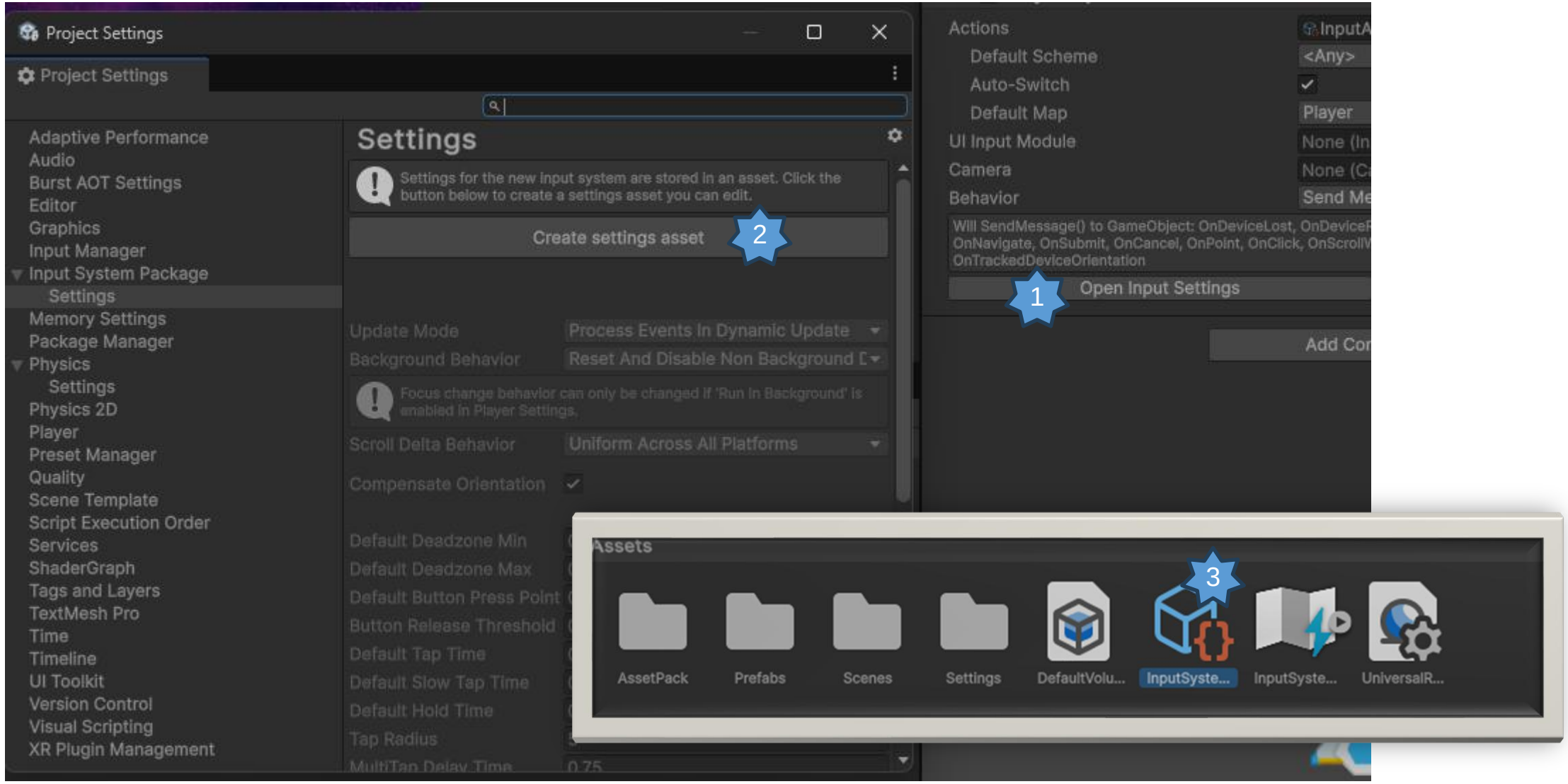
Input Actions



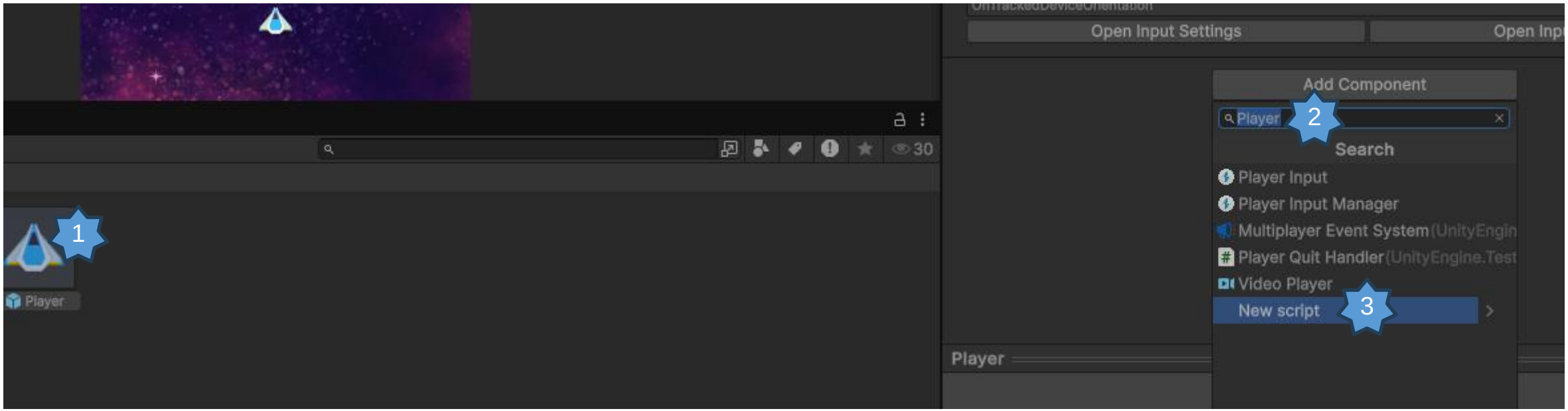
Key-bindings



Player Input이 동작하게 만들기



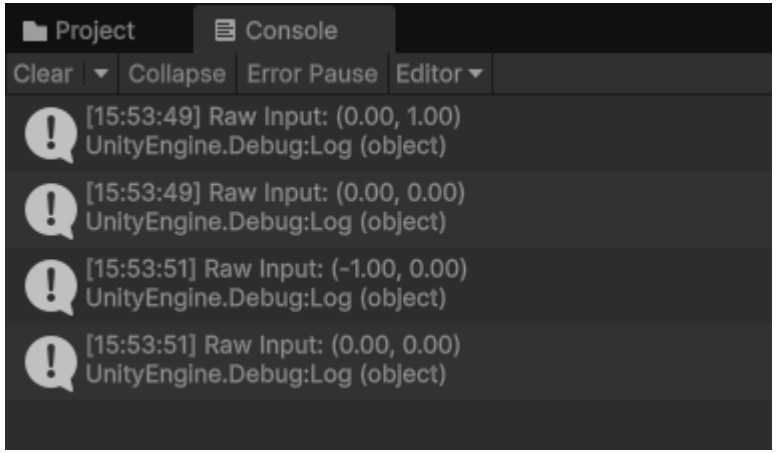
Message를 가로챌 코드 작성



Player.cs 코드가 Player Prefab과 연결되어 있는지 반드시 체크

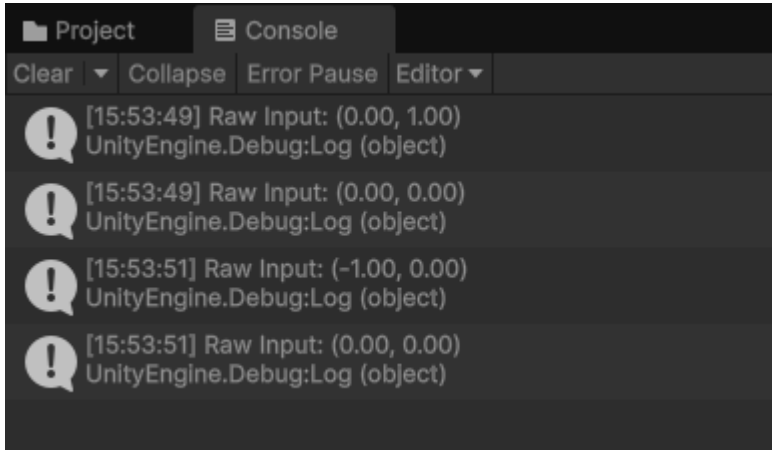
OnMove에 의해 실행될 내용 담기

```
using UnityEngine;

using UnityEngine.InputSystem; // Required for Input System

public class Player : MonoBehaviour
{
    private Vector2 rawInput; // Store the raw input value

    void Update()
    {

    }

    void OnMove(InputValue value)
    {
        rawInput = value.Get<Vector2>();
        Debug.Log("Raw Input: " + rawInput);
    }
}
```

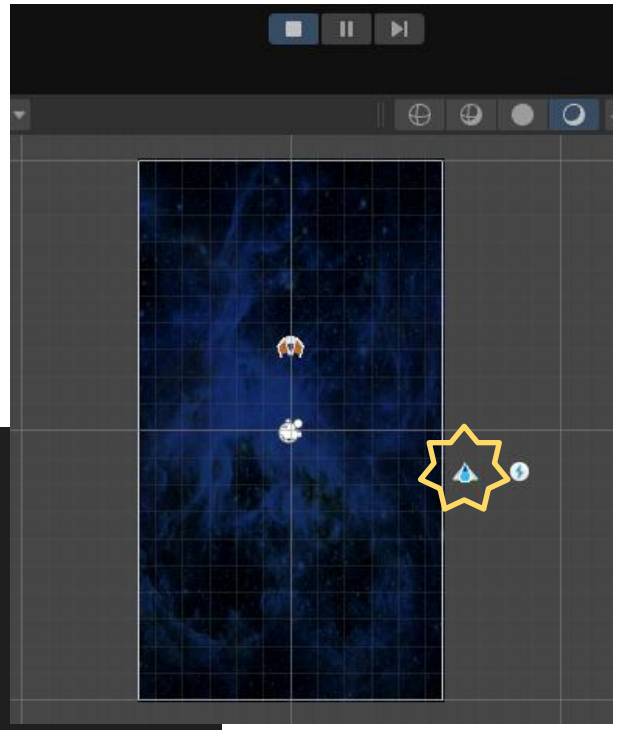


플레이어 동작에 연결

```
using UnityEngine;

using UnityEngine.InputSystem; // Required for Input System

public class Player : MonoBehaviour
{
    private Vector2 rawInput; // Store the raw input value

    void Update()
    {

    }

    void OnMove(InputValue value)
    {
        rawInput = value.Get<Vector2>();
        Debug.Log("Raw Input: " + rawInput);
    }
}
```



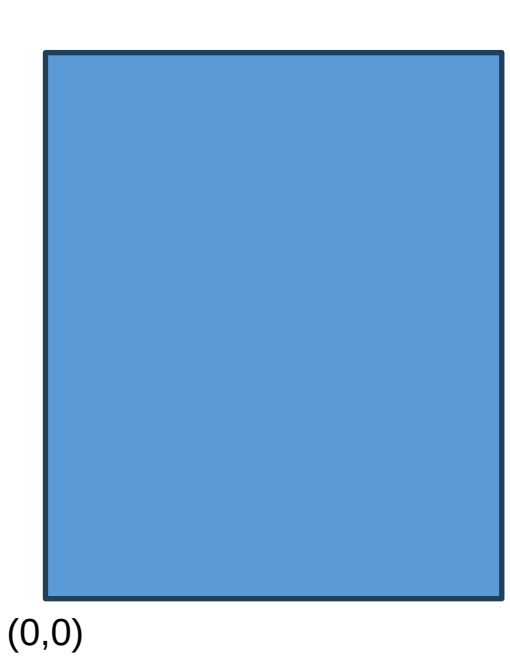
속도를 제어할 수 있게 해 보자



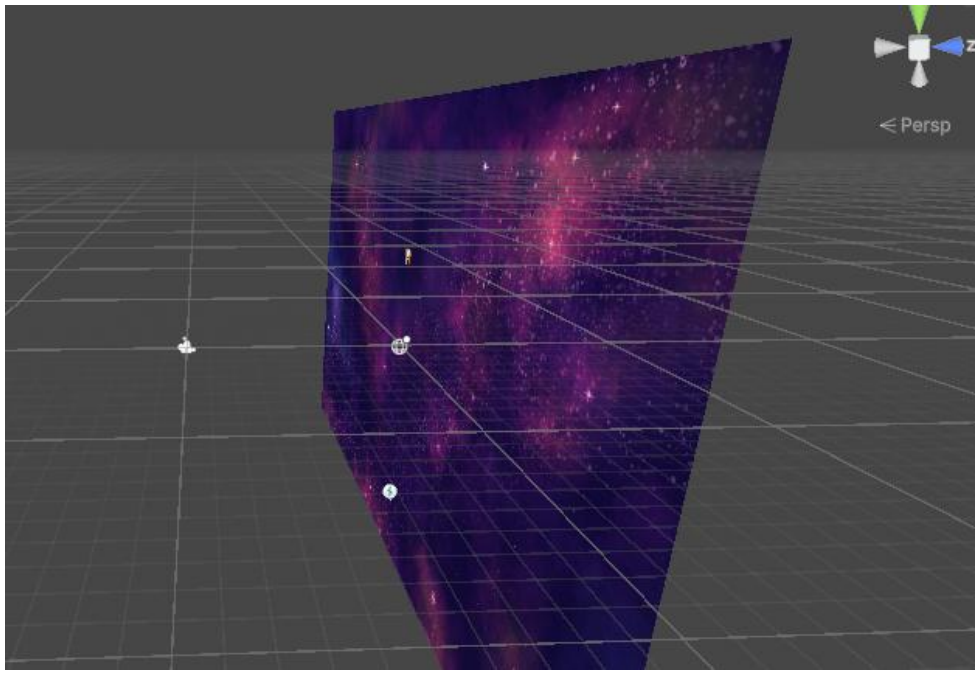
```
using UnityEngine;

using UnityEngine.InputSystem; // Required for Input System

public class Player : MonoBehaviour
{
    private Vector2 rawInput; // Store the raw input value
    private float speed = 5.0f; // speed of the player

    void Update()
    {
        transform.Translate(rawInput * speed * Time.deltaTime); // Move the player based on input and speed
    }

    void OnMove(InputValue value)
    {
        rawInput = value.Get<Vector2>();
    }
}
```

경계설정

Viewport: 표준화된 화면 좌표계

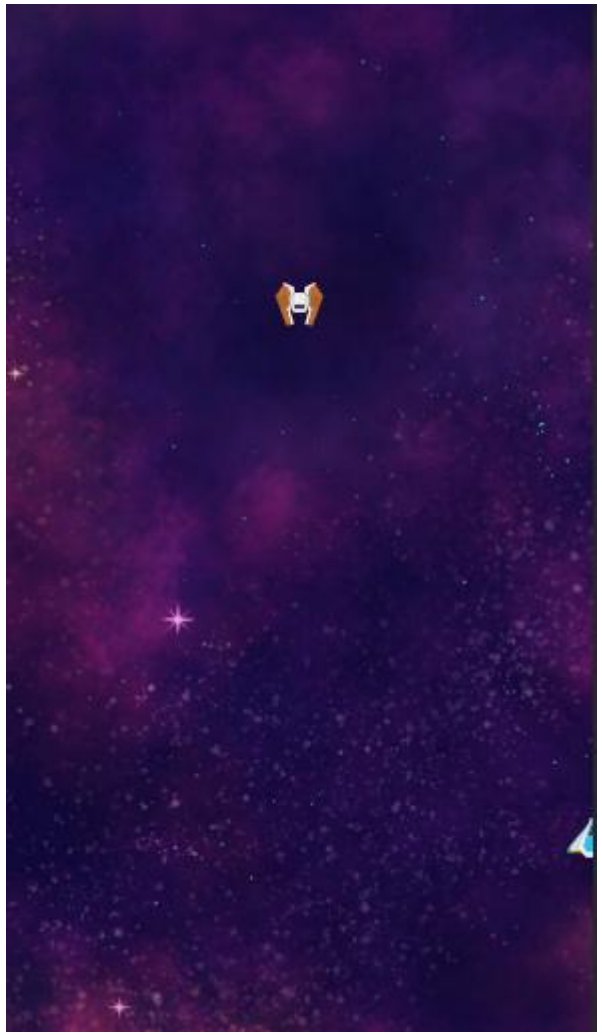


viewportToWorldPoint



World 좌표계 – 게임 콘텐츠를 담은 3차원 공간 내의 좌표

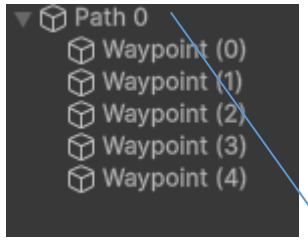
경계 안에 플레이어 움직임 제한하기

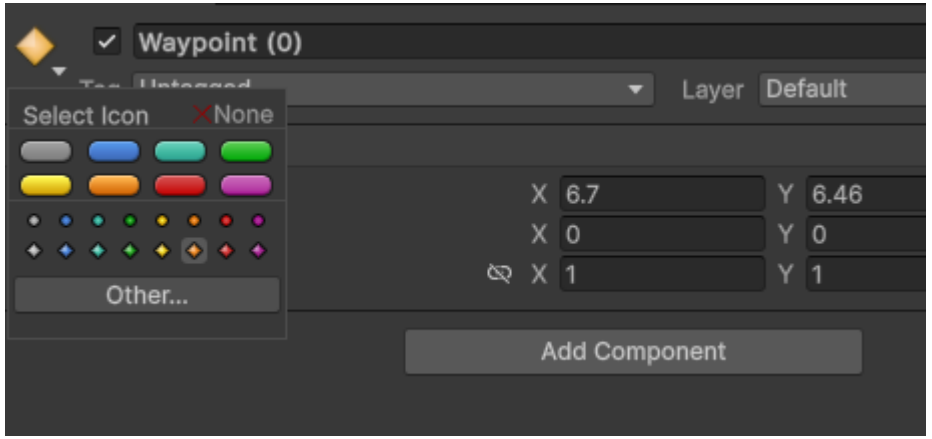


```
public class Player : MonoBehaviour
{
    private Vector2 rawInput; // Store the raw input value
    private float speed = 5.0f; // speed of the player

    Vector2 minBounds;
    Vector2 maxBounds;
    void Start()
    {
        // Set the bounds for the player movement
        Camera mainCam = Camera.main;

        minBounds = mainCam.ViewportToWorldPoint(new Vector2(0, 0)); // Bottom left corner of the screen
        maxBounds = mainCam.ViewportToWorldPoint(new Vector2(1, 1)); // Top right corner of the screen
    }

    void Update()
    {
        Vector2 moveDelta = rawInput * speed * Time.deltaTime; // Calculate the movement delta based on input and speed
        Vector2 newPosition = (Vector2)transform.position + moveDelta; // Calculate the new position
        // Clamp the new position to the screen bounds
        newPosition.x = Mathf.Clamp(newPosition.x, minBounds.x, maxBounds.x); // Clamp x position
        newPosition.y = Mathf.Clamp(newPosition.y, minBounds.y, maxBounds.y); // Clamp y position
        transform.position = newPosition; // Set the new position of the player
    }

    void OnMove(InputValue value)
    {
        rawInput = value.Get<Vector2>();
    }
}
```

적의 움직임 – Waypoints

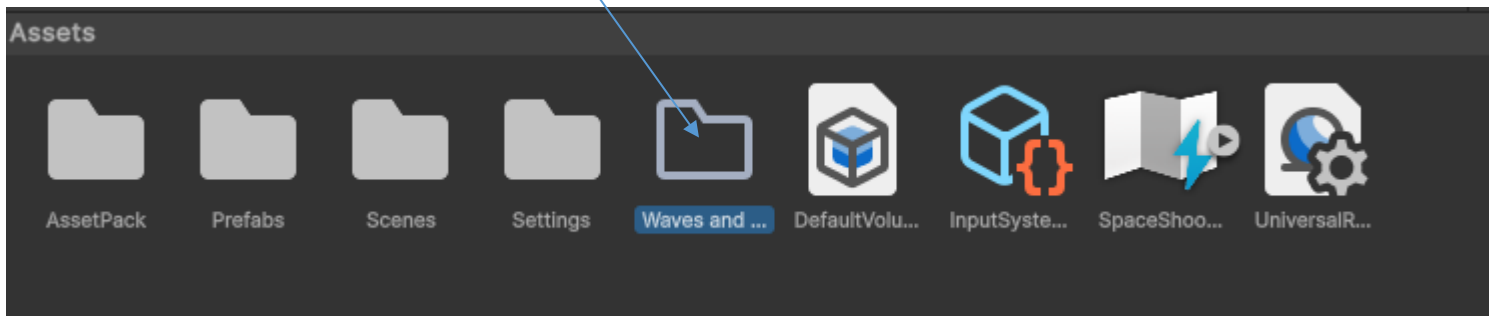
Empy Object로 생성

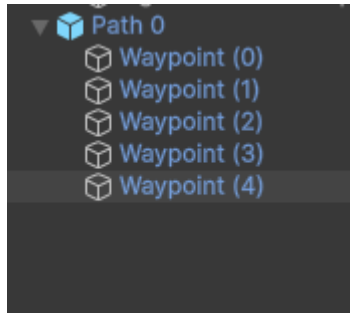




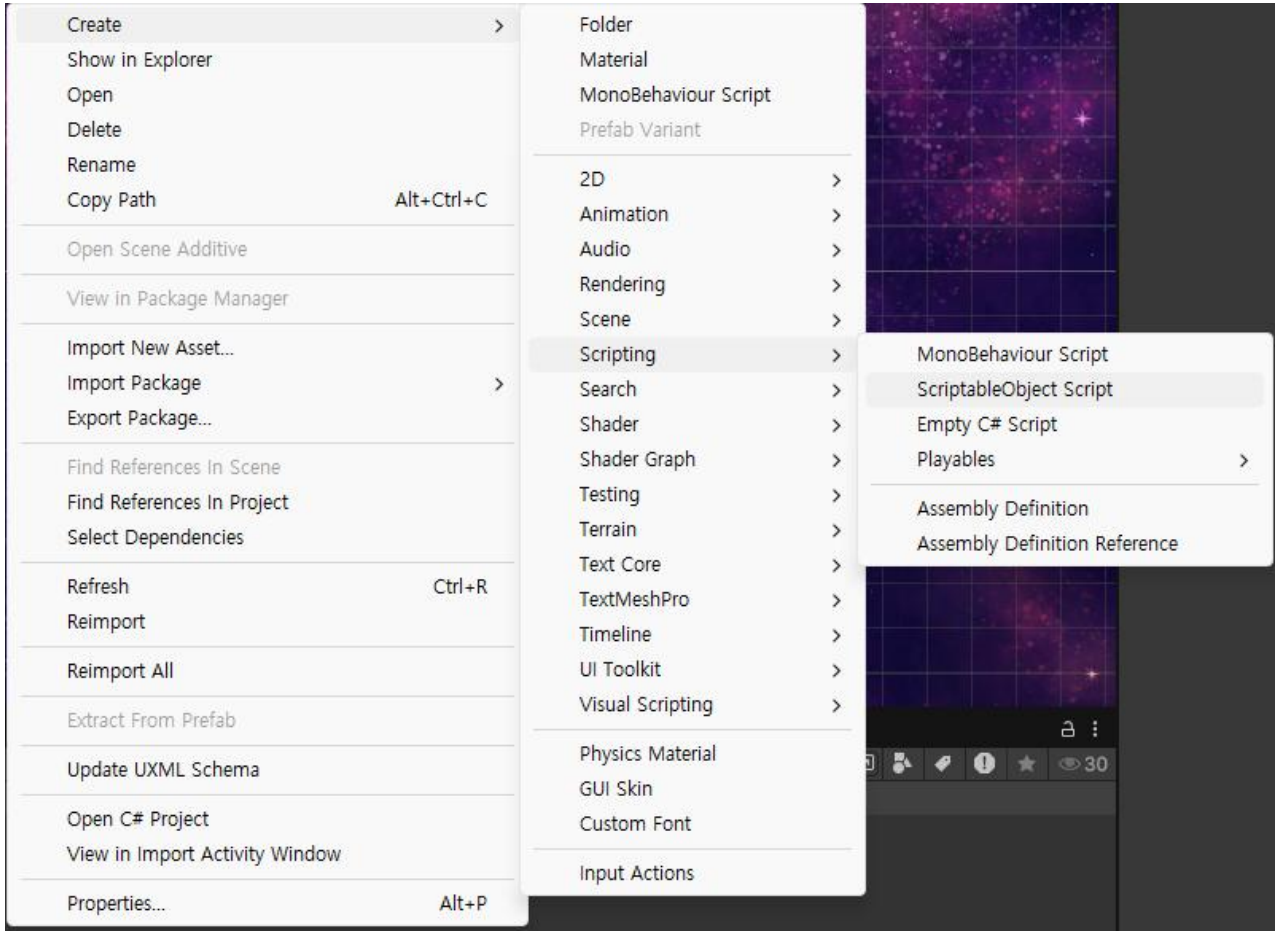
Waypoints 객체들은 보이게 설정

Waves and Paths 폴더를 만들고 여기에 Path0를 끌어다 놓음





적의 움직임을 제어하는 Scriptable Object



Scriptable Object 생성

Scriptable Object 변수

List를 쓰려면, System.Collections.Generic 필요

```
[CreateAssetMenu(fileName = "WaveConf_SO", menuName = "Scriptable Objects/WaveConf_SO")]
1 reference
public class WaveConf_SO : ScriptableObject
{
    2 references
    [SerializeField] Transform pathPrefab;
    1 reference
    [SerializeField] float moveSpeed = 5f;
```

Scriptable Object – Getters

```
0 references
public Transform GetStartingWaypoint()
{
    return pathPrefab.GetChild(0);
}
```

```
0 references
public float GetMoveSpeed()
{
    return moveSpeed;
}
```

Scriptable Object – Waypoints 얻기

List를 쓰려면, System.Collections.Generic 필요

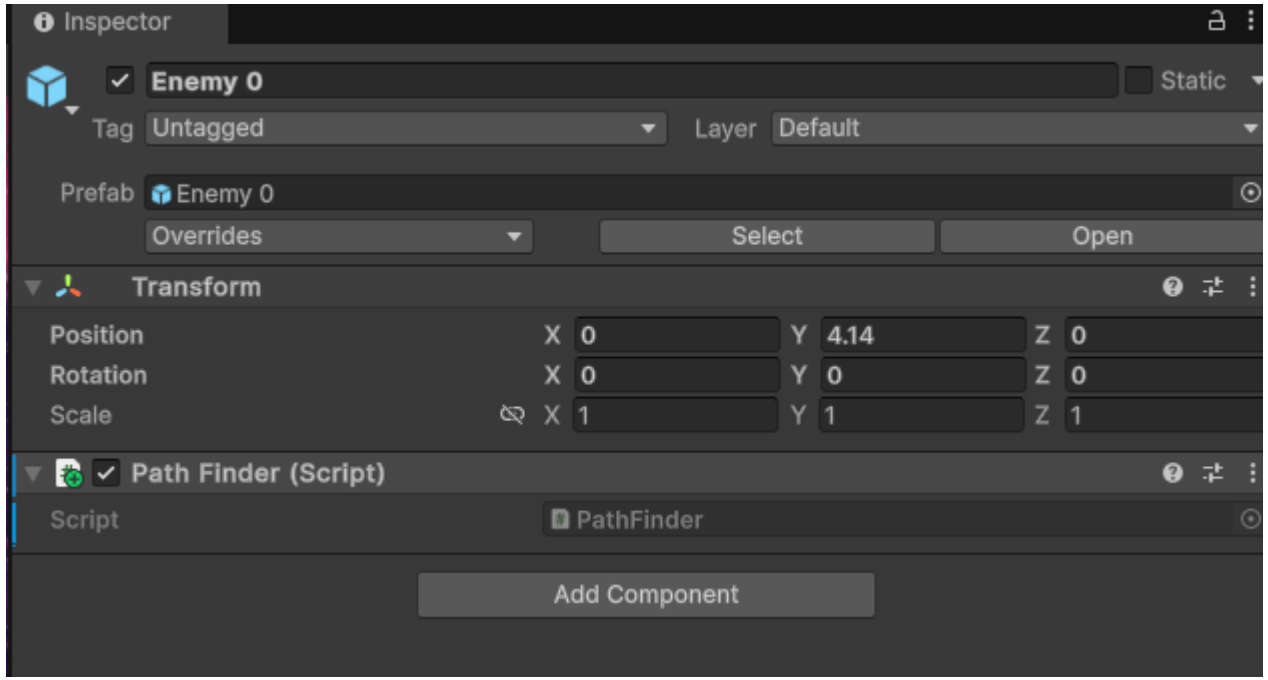
```
public List<Transform> GetWaypoints()
{
    List<Transform> waypoints = new List<Transform>();
    foreach (Transform child in pathPrefab)
    {
        waypoints.Add(child);
    }
    return waypoints;
}
```

Scriptable Object – Waypoints 얻기

```
public List<Transform> GetWaypoints()
{
    List<Transform> waypoints = new List<Transform>();
    foreach (Transform child in pathPrefab)
    {
        waypoints.Add(child);
    }
    return waypoints;
}
```

적의 움직임

PathFinder.cs 스크립트를 만들어서 Enemy에 연결



첫 WayPoint의 위치에 적을 옮겨 놓고, 리스트에 있는 다음 WayPoint들을 차례로 따라간 뒤 마지막 위치에서 사라짐

적의 움직임

List를 쓰려면, System.Collections.Generic 필요

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

Unity 스크립트 | 참조 0개
public class PathFinder : MonoBehaviour
{
    [SerializeField] WaveConfigSO waveConfig;
    List<Transform> waypoints;
    int waypointIndex = 0;
```

```
void Start()
{
    waypoints = waveConfig.GetWaypoints();
    transform.position = waypoints[waypointIndex].position;
}

0 references
void Update()
{
    FollowPath();
}
```

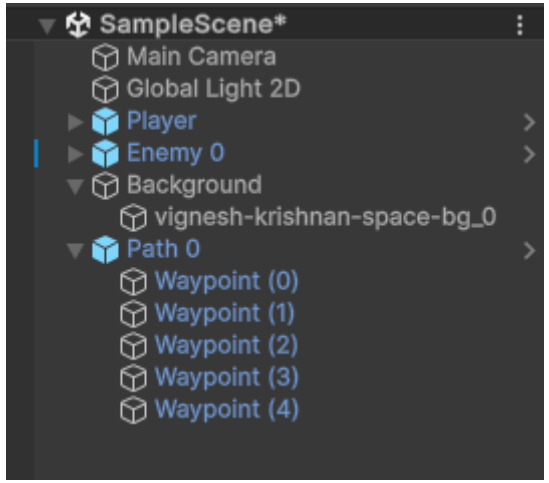
적의 움직임을 만들어내는 코드 – FollowPath

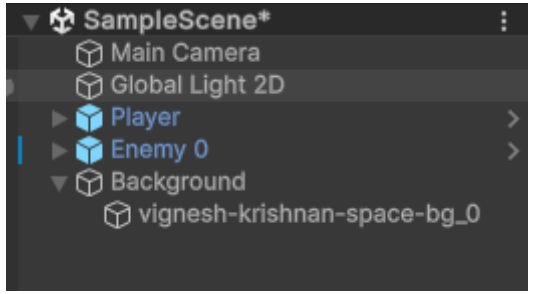
```
void FollowPath()
{
    if (waypointIndex < waypoints.Count)
    {
        Vector3 targetPosition = waypoints[waypointIndex].position;
        float delta = waveConfig.GetMoveSpeed() * Time.deltaTime;
        transform.position = Vector2.MoveTowards(transform.position, targetPosition, delta);

        if (transform.position == targetPosition)
        {
            waypointIndex++;
        }
    }
    else
    {
        Destroy(gameObject);
    }
}
```

적의 움직임을 만들어내는 코드 – FollowPath

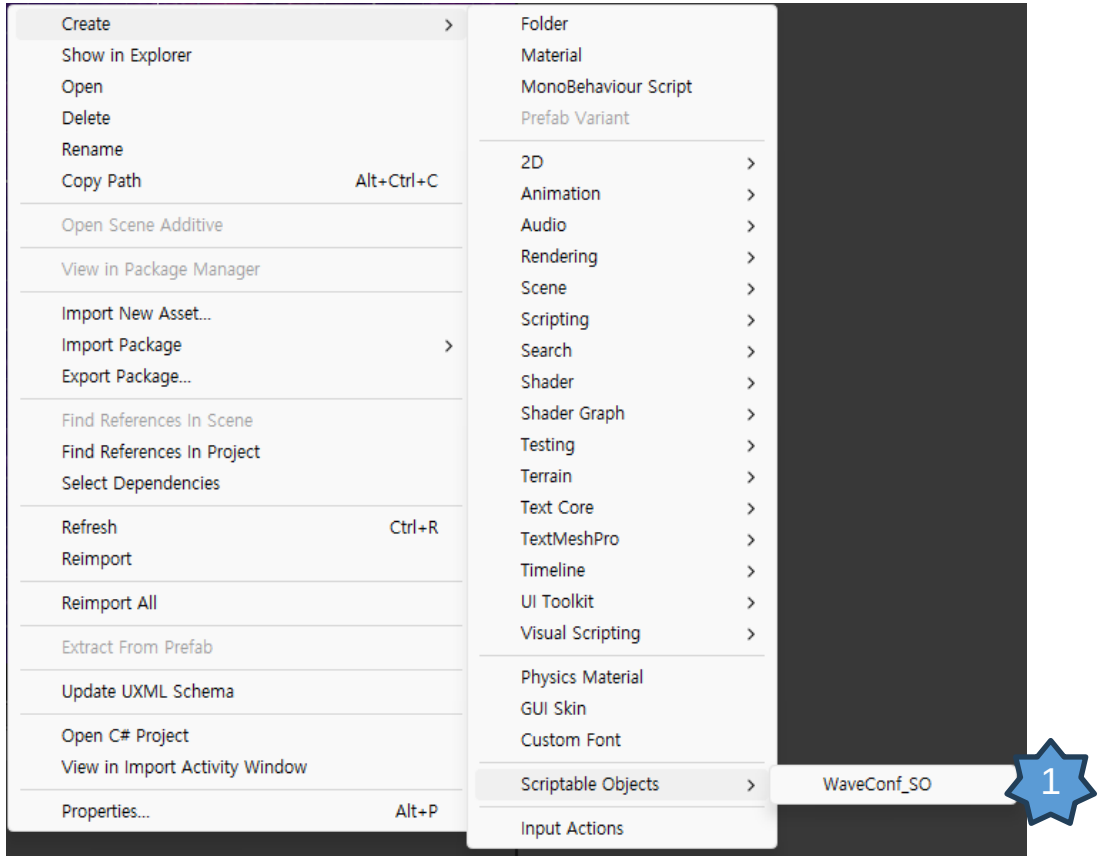
```
void FollowPath()
{
    if (waypointIndex < waypoints.Count)
    {
        Vector3 targetPosition = waypoints[waypointIndex].position;
        float delta = waveConfig.GetMoveSpeed() * Time.deltaTime;
        transform.position = Vector2.MoveTowards(transform.position, targetPosition, delta);

        if (transform.position == targetPosition)
        {
            waypointIndex++;
        }
    }
    else
    {
        Destroy(gameObject);
    }
}
```

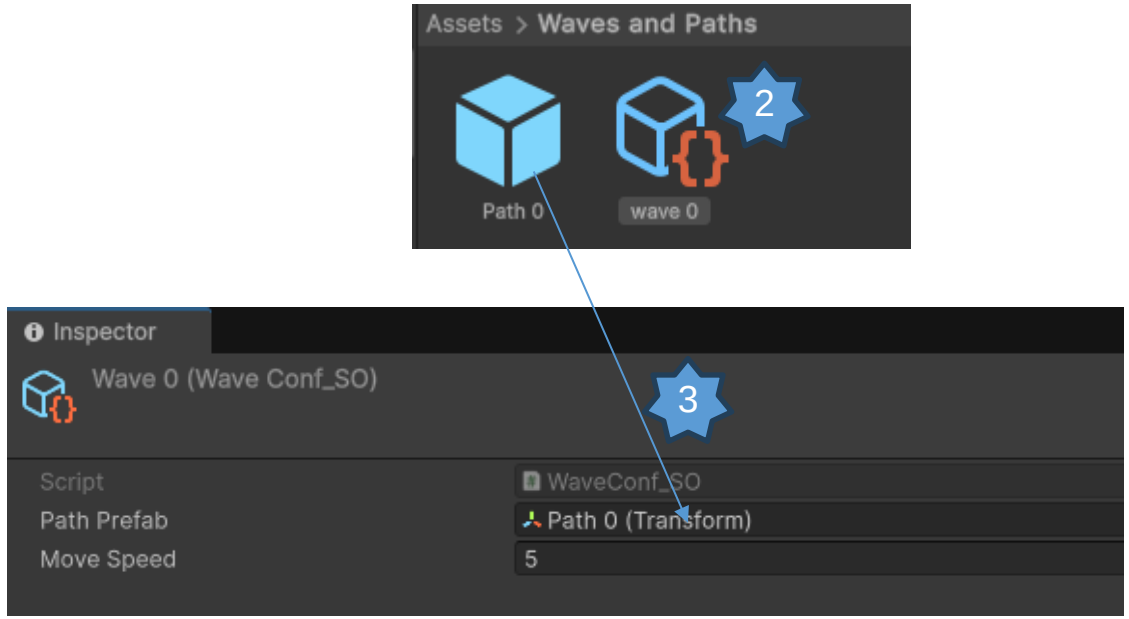
Prefab 된 객체는 없애도 됨



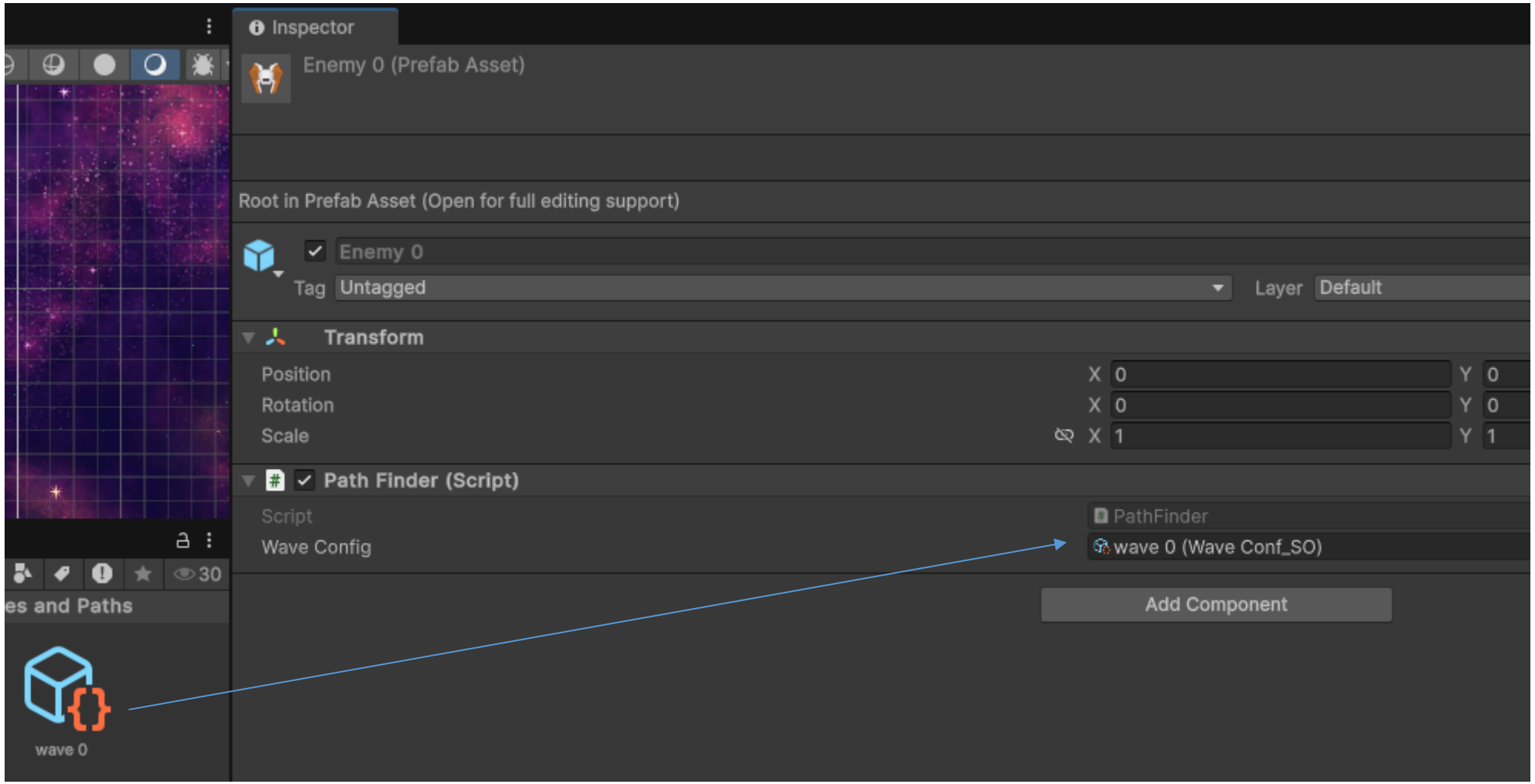


Scriptable Object 생성

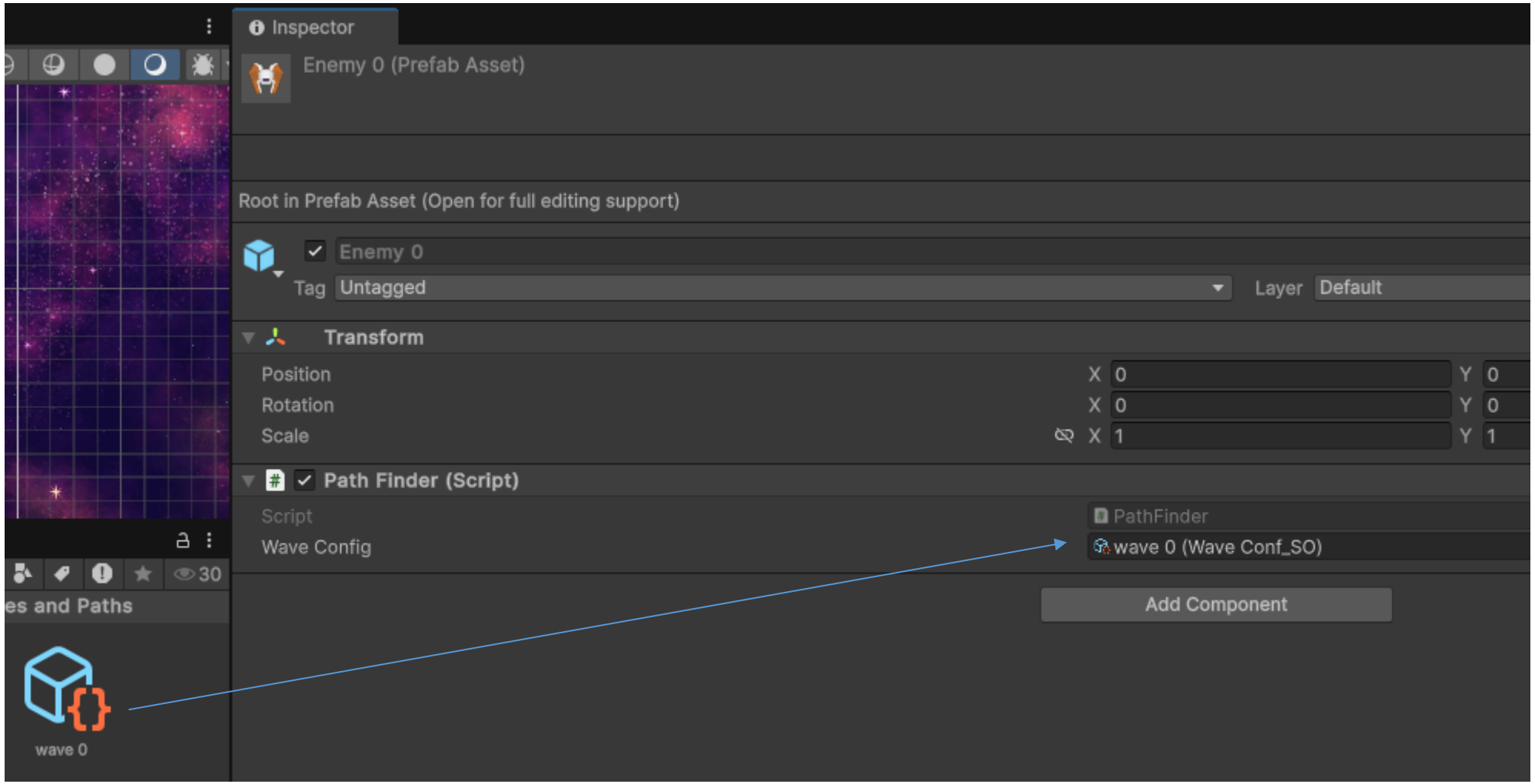




이 Wave 객체를 적 객체에 적용

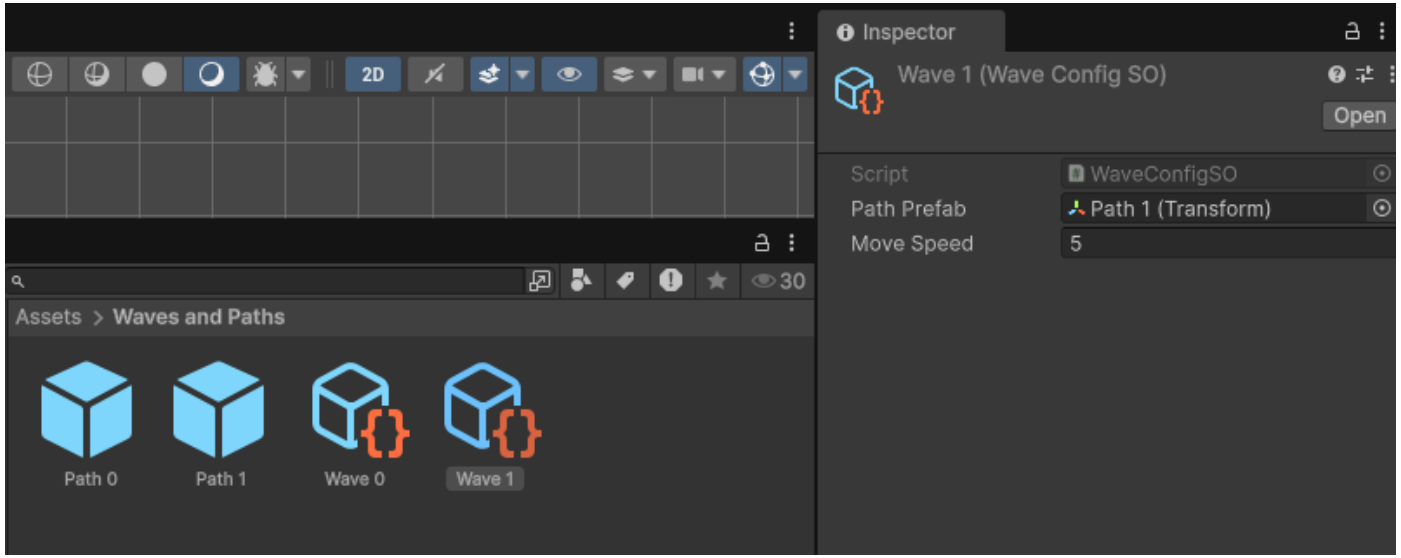


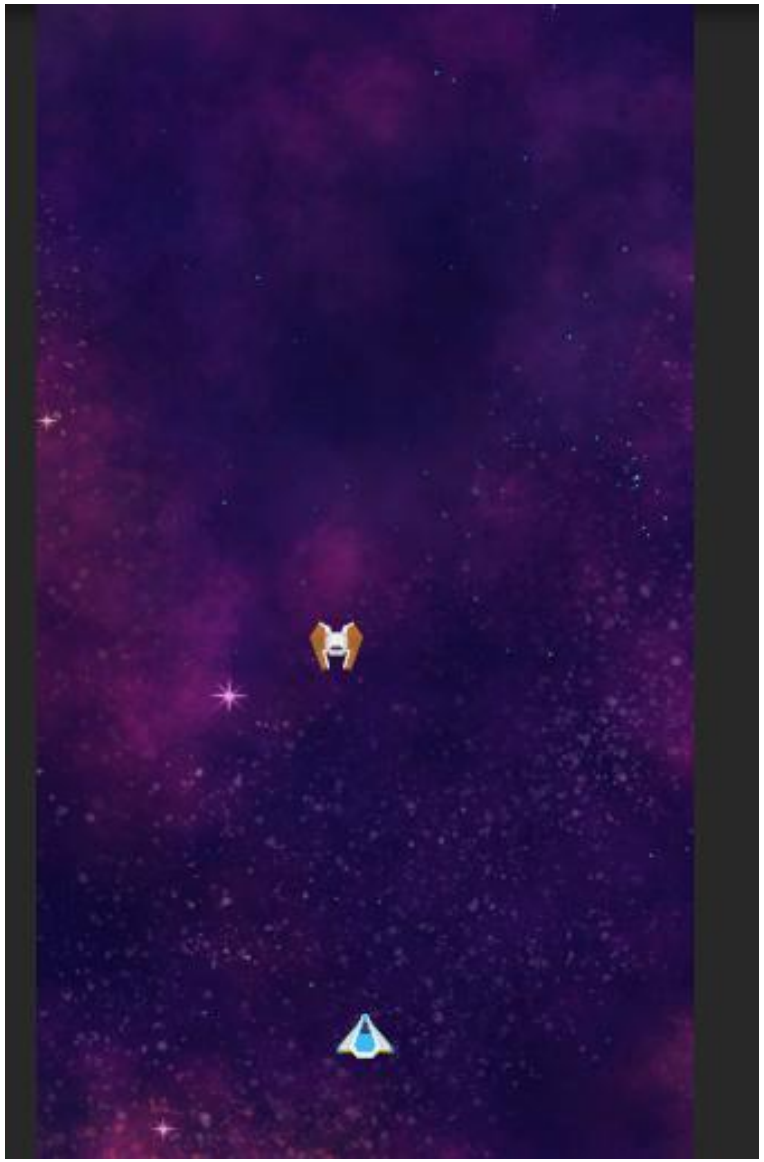
이 Wave 객체를 적 객체에 적용



움직이는 적

여러 Path를 만들 수 있음





적 인스턴스 생성

WaveConfig
PathFinder ← 수정

EnemySpawner ← 신규 생성

적 인스턴스 생성

WaveConfigSO.cs : 생성할 적들의 프리팹을 담아 놓고 이들을 Spawning!

```
public class WaveConfigSO : ScriptableObject
{
    [SerializeField] List<GameObject> enemyPrefabs;
    [SerializeField] Transform pathPrefab;
    [SerializeField] float moveSpeed = 5f;

    참조 0개
    public int GetEnemyCount()
    {
        return enemyPrefabs.Count;
    }

    참조 0개
    public GameObject GetEnemyPrefab(int index)
    {
        return enemyPrefabs[index];
    }
```

EnemySpawner.cs

```
public class EnemySpawner : MonoBehaviour
{
    [SerializeField] WaveConfigSO currentWave;

    Unity 메시지 | 참조 0개
    void Start()
    {
        SpawnEnemies();
    }

    참조 0개
    public WaveConfigSO GetCurrentWave()
    {
        return currentWave; //  PathFinder는 별도의 currentWave를 갖지 않고, 이를 불러서 쓰자
    }
    // Update is called once per frame
    참조 1개
    void SpawnEnemies()
    {
        Instantiate(currentWave.GetEnemyPrefab(0),
            currentWave.GetStartingWaypoint().position,
            Quaternion.identity);

    }
```

PathFinder.cs 변경

```
public class PathFinder : MonoBehaviour
{
    EnemySpawner enemySpawner;
    WaveConfigSO waveConfig; // Inspector에서 별도로 받지 못 하게 함
    List<Transform> waypoints;
    int waypointIndex = 0;

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        enemySpawner = FindFirstObjectByType<EnemySpawner>();
    }

    Unity 메시지 | 참조 0개
    void Start()
    {
        waveConfig = enemySpawner.GetCurrentWave();
        waypoints = waveConfig.GetWaypoints();
        transform.position = waypoints[waypointIndex].position;
    }
```

EnemySpawner.cs

```
void SpawnEnemies()
{
    for (int i = 0; i < currentWave.GetEnemyCount(); i++)
    {
        Instantiate(currentWave.GetEnemyPrefab(i),
        currentWave.GetStartingWaypoint().position,
        Quaternion.identity,
        transform);
    }
}
```

PathFinder.cs 변경

```
public class PathFinder : MonoBehaviour
{
    EnemySpawner enemySpawner;
    WaveConfigSO waveConfig; // Inspector에서 별도로 받지 못 하게 함
    List<Transform> waypoints;
    int waypointIndex = 0;

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        enemySpawner = FindFirstObjectByType<EnemySpawner>();
    }

    Unity 메시지 | 참조 0개
    void Start()
    {
        waveConfig = enemySpawner.GetCurrentWave();
        waypoints = waveConfig.GetWaypoints();
        transform.position = waypoints[waypointIndex].position;
    }
```

Awake는 뭘까?

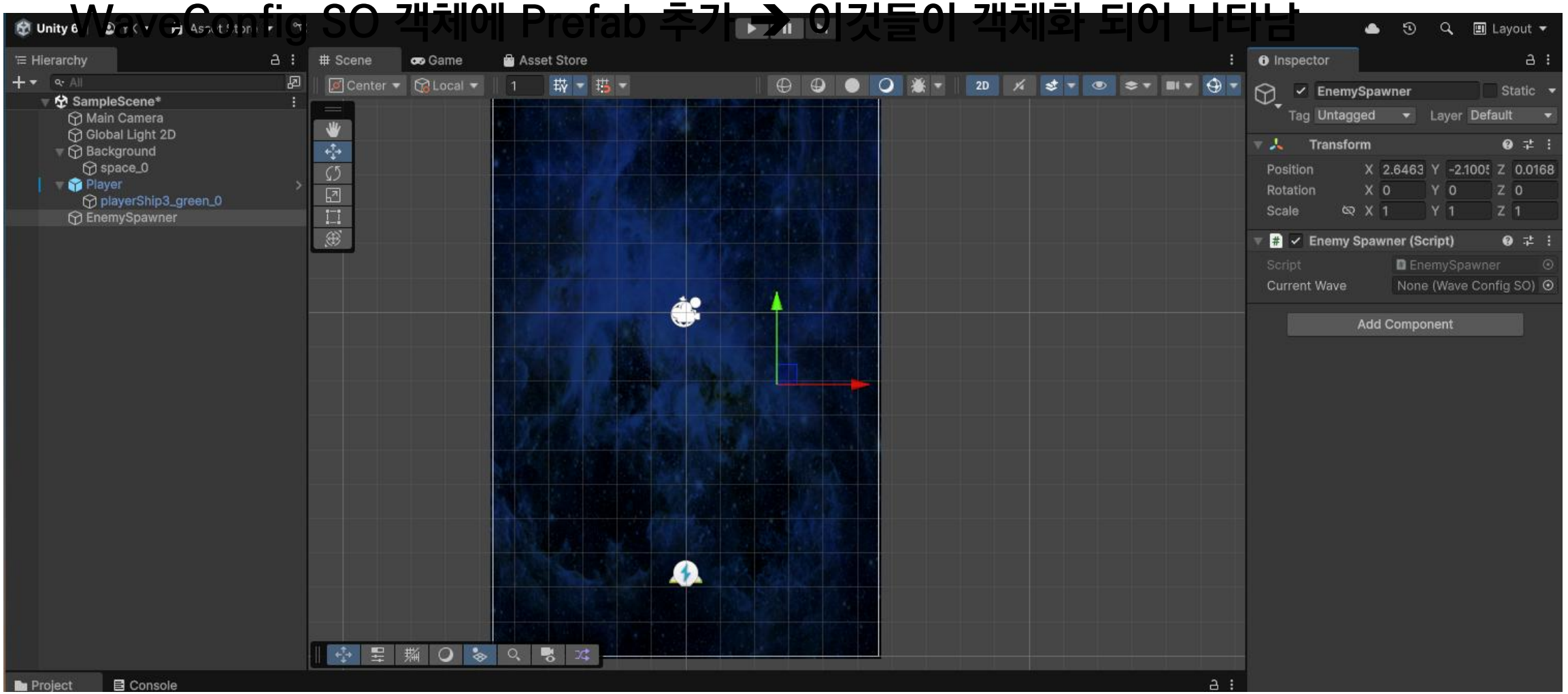
게임 객체의 동작으로 가장
먼저 호출되는 메소드

Start와의 차이는
- Start 게임 객체 활성화 이후
- Awake 코드 로딩시 바로

EnemySpawner 생성

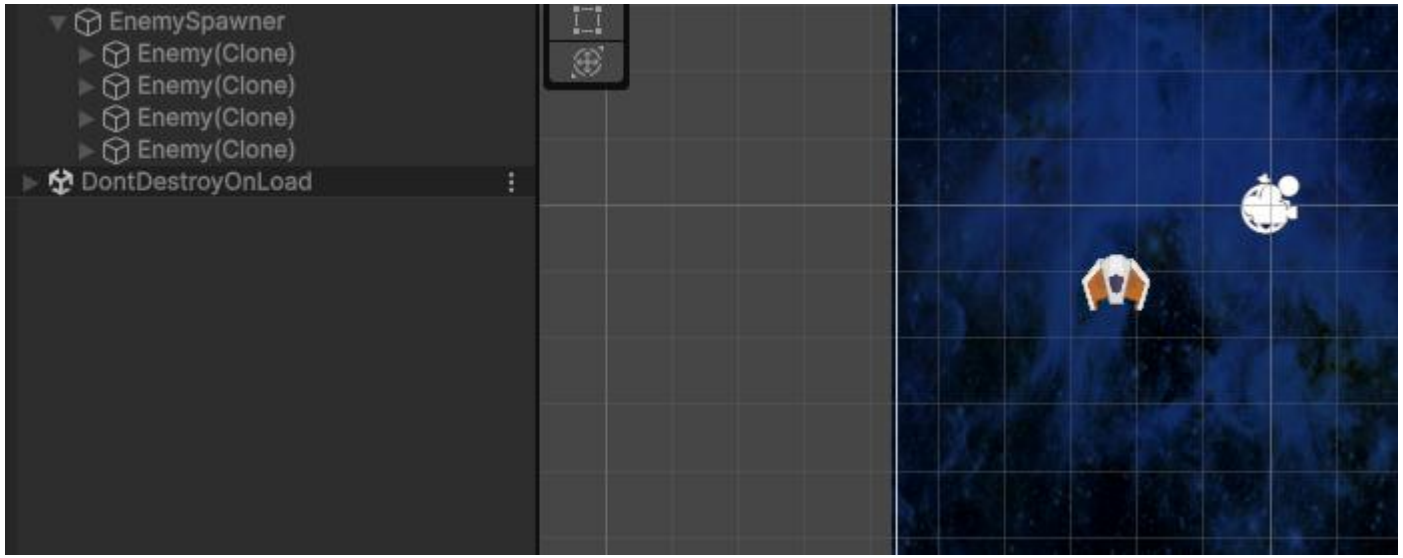
→ EnemySpawner.cs 연결 / Wave 설정

- WaveConfig SO 객체에 Prefab 추가 → 이것들이 객체화 되어 나타남



Hierarchy에 있는 Enemy 삭제 가능

WaveConfig SO 객체에 Prefab 추가 ➔ 이것들이 객체화 되어 나타남



그런데 하나만 보임
동일한 위치에 생성

Coroutine을 이용한 시차 생성

Coroutine
- 시간이 걸리는 작업을 중단하고 다시 이어서 실행할 수 있게 해주는 함수
- 잠시 멈췄다가 다시 실행되는 함수

```
IEnumerator WaitAndPrint()
{
    Debug.Log("Start");
    yield return new WaitForSeconds(2f);  // 2초 대기
    Debug.Log("2 seconds passed");
}

StartCoroutine(WaitAndPrint());
```

- 단순히 Sleep을 부르는 것과의 차이는
  Coroutine만 멈추지만, 함수의 Sleep은 다른 모든 동작을 멈춤
  비동기(Asynchronous) 동작이 가능 (UI 반응이 멈추지 않음)

Coroutine을 이용한 시차 생성

WaveConfigSO 스크립트에 새로운 변수들을 추가

```
public class WaveConfigSO : ScriptableObject
{
    [SerializeField] List<GameObject> enemyPrefabs;
    [SerializeField] Transform pathPrefab;
    [SerializeField] float moveSpeed = 5f;
    [SerializeField] float timeBetweenEnemySpawns = 1f;
    [SerializeField] float spawnTimeVariance = 0f;
    [SerializeField] float minimumSpawnTime = 0.2f;
```

Coroutine을 이용한 시차 생성

GetRandomSpawTime 추가

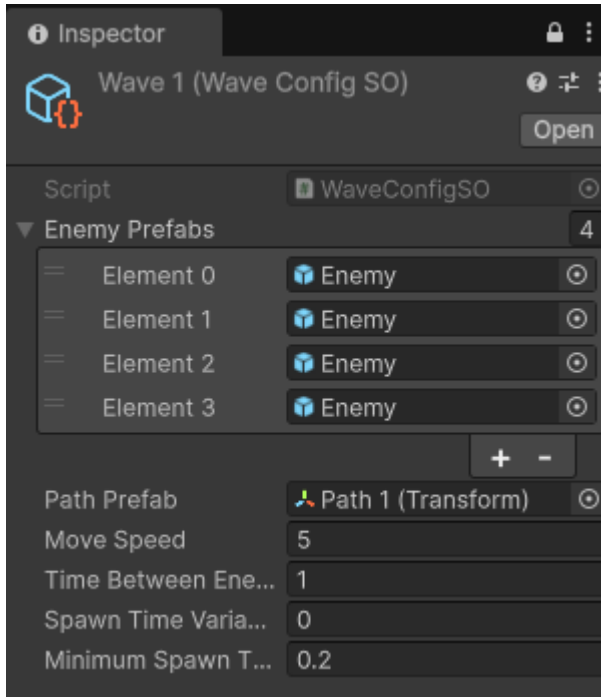
```
public float GetRandomSpawnTime()
{
    float spawnTime = Random.Range( timeBetweenEnemySpawns - spawnTimeVariance,
                                    timeBetweenEnemySpawns + spawnTimeVariance);
    return Mathf.Clamp(spawnTime, minimumSpawnTime, float.MaxValue);
}
```

Coroutine을 이용한 시차 생성

EnenySpawner 수정

```
// System.Collections 가 필요 (IEnumerator를 위해)
참조 1개
IEnumerator SpawnEnemies()
{
    for (int i = 0; i < currentWave.GetEnemyCount(); i++)
    {
        Instantiate(currentWave.GetEnemyPrefab(i),
        currentWave.GetStartingWaypoint().position,
        Quaternion.identity,
        transform);

        yield return new WaitForSeconds(currentWave.GetRandomSpawnTime());
    }
}
```

```
void Start()
{
    StartCoroutine(SpawnEnemies());
}
```



여러 Wave를 사용하자

```
using System.Collections.Generic;
using NUnit.Framework;

Unity 스크립트 | 참조 2개
public class EnemySpawner : MonoBehaviour
{
    [SerializeField] List<WaveConfigSO> waveConfigs;
    [SerializeField] float timeBetweenWaves = 0f;
    WaveConfigSO currentWave;   // [SerializeField] WaveConfigSO currentWave;
```

WaveConfig의 리스트를 인스펙터로 받고
현재 Wave는 이 리스트에서 얻어오게 함

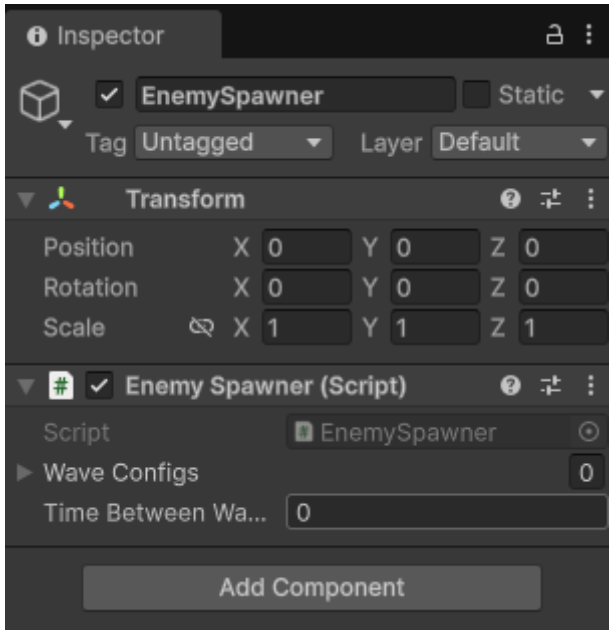
여러 Wave를 사용하자

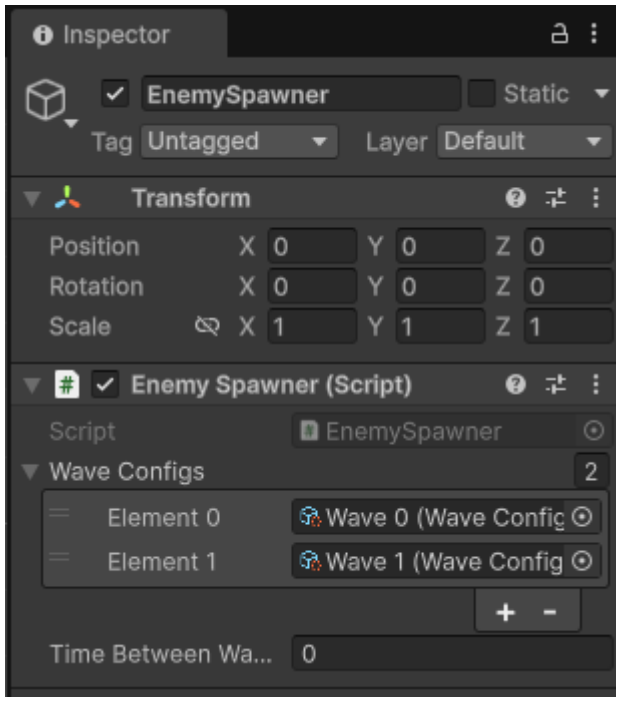
- WaveConfigSO 리스트에 대해 for loop
- 각 루프내에서
    * currentWave 정한다
    * currentWave 내의 적에 대해 루프를 돌며 생성 (이미 완성)
    * timeBetweenWaves만큼 기다리게 함: Coroutine으로

여러 Wave를 사용하자

```
IEnumerator SpawnEnemies()
{
    foreach (WaveConfigSO wave in waveConfigs)
    {
        currentWave = wave;
        for (int i = 0; i < currentWave.GetEnemyCount(); i++)
        {
            Instantiate(currentWave.GetEnemyPrefab(i),
            currentWave.GetStartingWaypoint().position,
            Quaternion.identity,
            transform);

            yield return new WaitForSeconds(currentWave.GetRandomSpawnTime());
        }
        yield return new WaitForSeconds(timeBetweenWaves);
    }

}
```

EnemySpawner에 Wave Config 리스트 설정





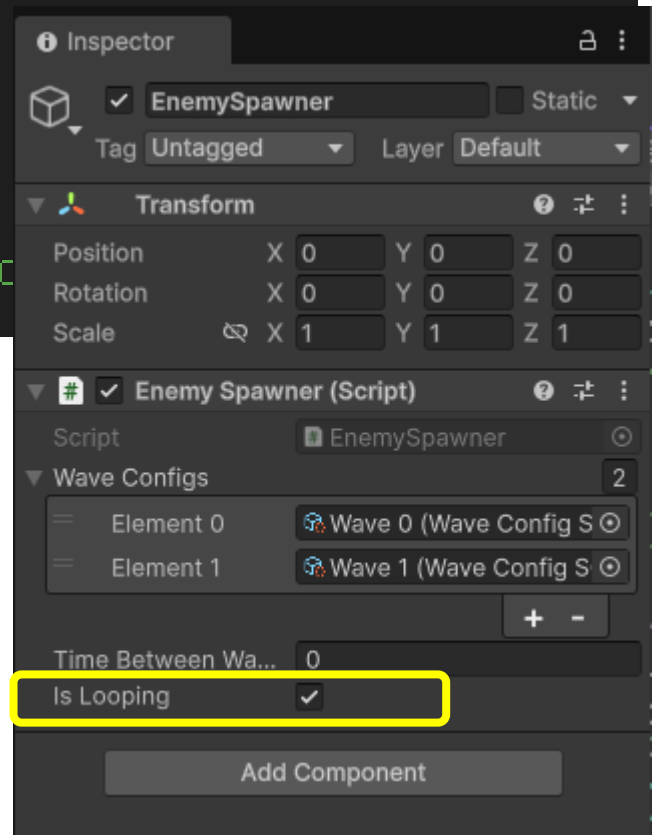
이렇게 고치면
어떻게 달라질까?

Coroutine의 특성을
이해해 보자

```
void Start()
{
    StartCoroutine(SpawnWaves());
}

참조 1개
public WaveConfigSO GetCurrentWave()
{
    return currentWave; //  PathFinder는 별도의 currentWave를 갖지 않고, 이를 불러서 쓰자
}
// System.Collections 가 필요 (IEnumerator를 위해)
참조 1개
IEnumerator SpawnWaves()
{
    foreach (WaveConfigSO wave in waveConfigs)
    {
        StartCoroutine(SpawnEnemies(wave));
        yield return new WaitForSeconds(timeBetweenWaves);
    }

}
참조 1개
IEnumerator SpawnEnemies(WaveConfigSO wave)
{

    currentWave = wave;
    for (int i = 0; i < currentWave.GetEnemyCount(); i++)
    {
        Instantiate(currentWave.GetEnemyPrefab(i),
        currentWave.GetStartingWaypoint().position,
        Quaternion.identity,
        transform);

        yield return new WaitForSeconds(currentWave.GetRandomSpawnTime());
    }


}
```

While Loop을 이용한 Wave 무한 생성

```
public class EnemySpawner : MonoBehaviour
{
    [SerializeField] List<WaveConfigSO> waveConfigs;
    [SerializeField] float timeBetweenWaves = 0f;
    [SerializeField] bool isLooping;
    WaveConfigSO currentWave;   // [SerializeField] WaveCo
```



isLooping 플래그 추가

While Loop을 이용한 Wave 무한 생성

```
IEnumerator SpawnWaves()
{
    do
    {
        foreach (WaveConfigSO wave in waveConfigs)
        {
            currentWave = wave;
            for (int i = 0; i < currentWave.GetEnemyCount(); i++)
            {
                Instantiate(currentWave.GetEnemyPrefab(i),
                            currentWave.GetStartingWaypoint().position,
                            Quaternion.identity,
                            transform);

                yield return new WaitForSeconds(currentWave.GetRandomSpawnTime());
            }
            yield return new WaitForSeconds(timeBetweenWaves);
        }
    }
    while (isLooping);

}
```

다양한 Wave를 만들어 담아 보아요

점점 게임처럼
변하고 있나요?