Unity를 이용한 2D 게임프로그래밍

Lecture 5
간단한 아케이드 게임을 위한 입력시스템

강영민

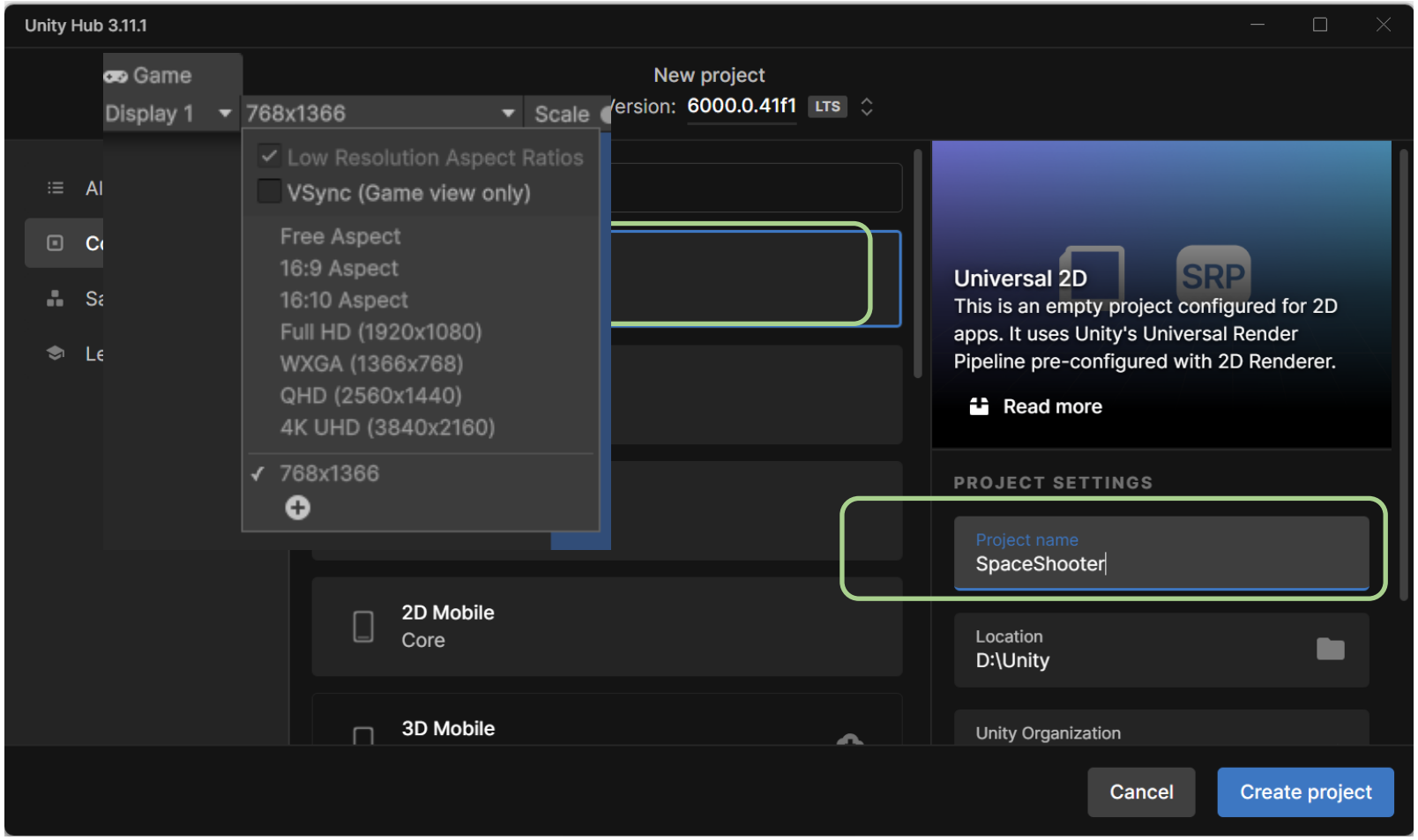
동명대학교 게임공학과

학습목표

- 전통적인 아케이드 게임 프로젝트를 만들어 본다
- 사용자 입력을 처리하기 위한 New Input System을 사용하자

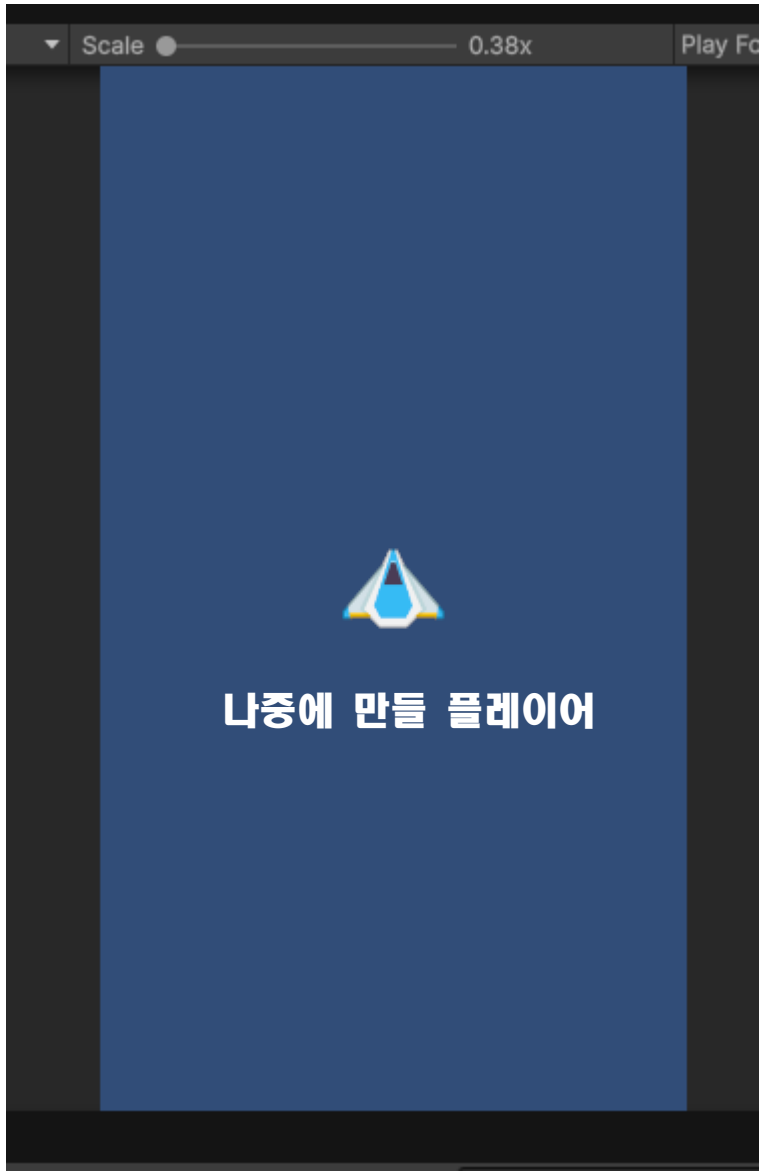
프로젝트를 생성하자

- 전통적 슈팅 게임
  - Space Shooter



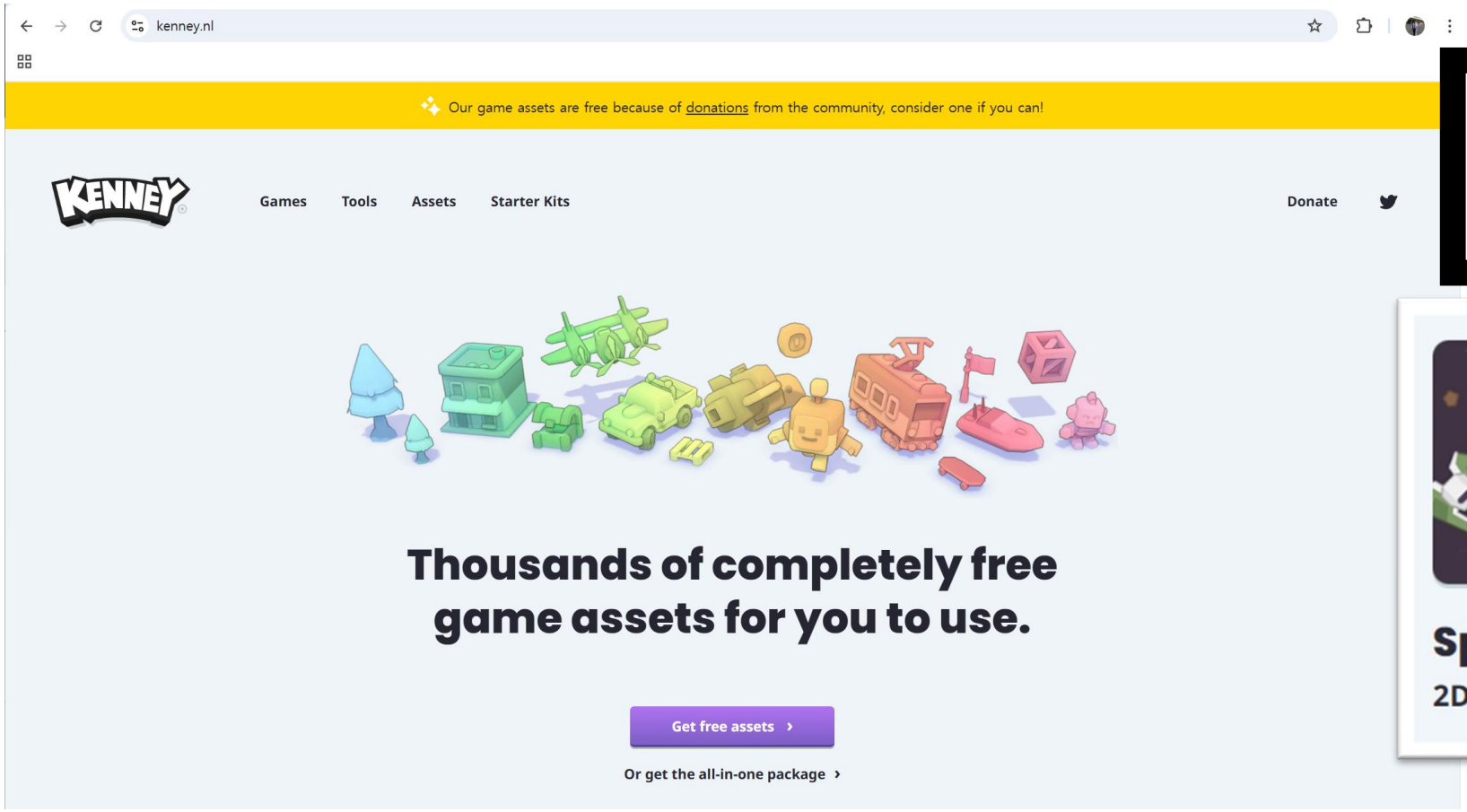
프로젝트 해상도 설정

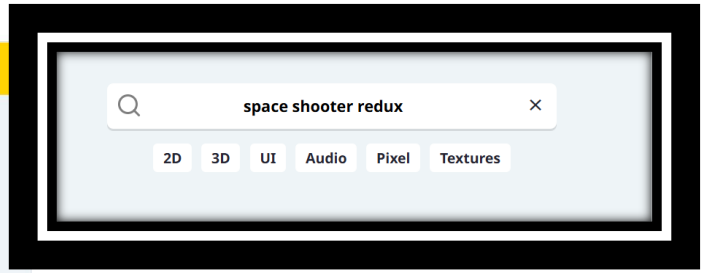
- 변경

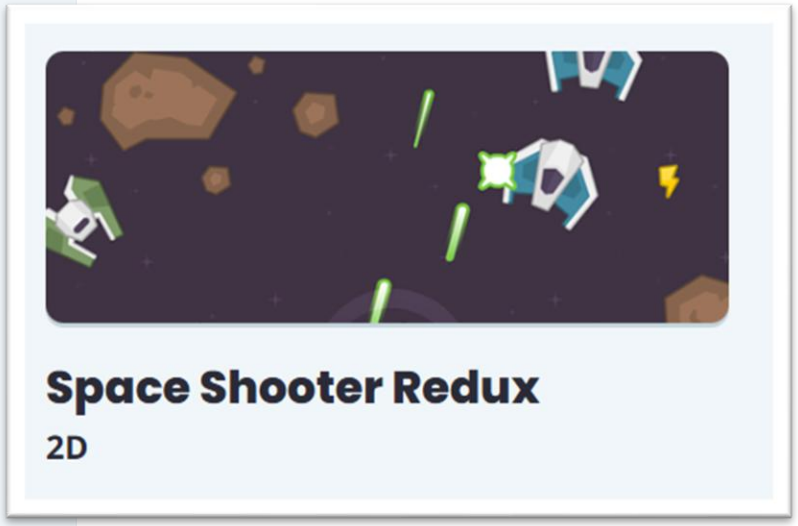


게임 리소스

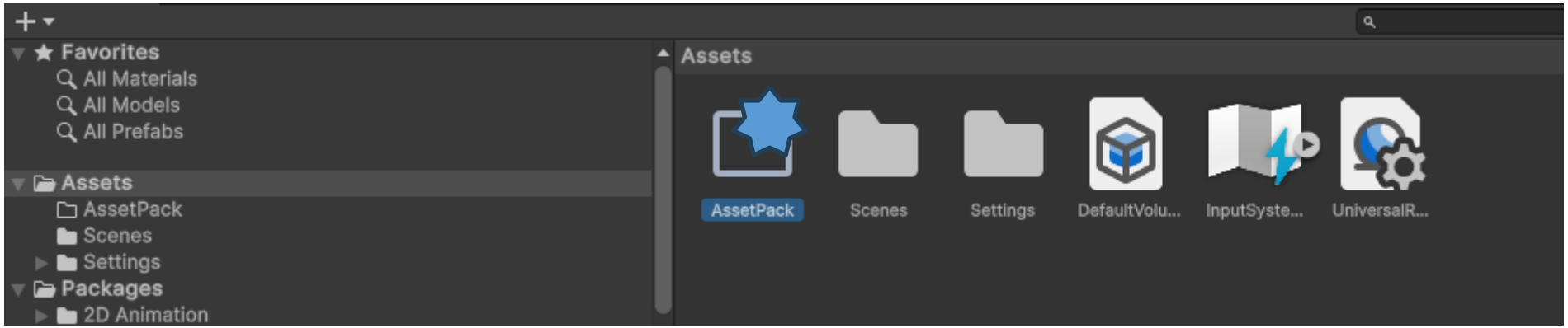
- Kenney is a good guy ☺ (kenney.nl)

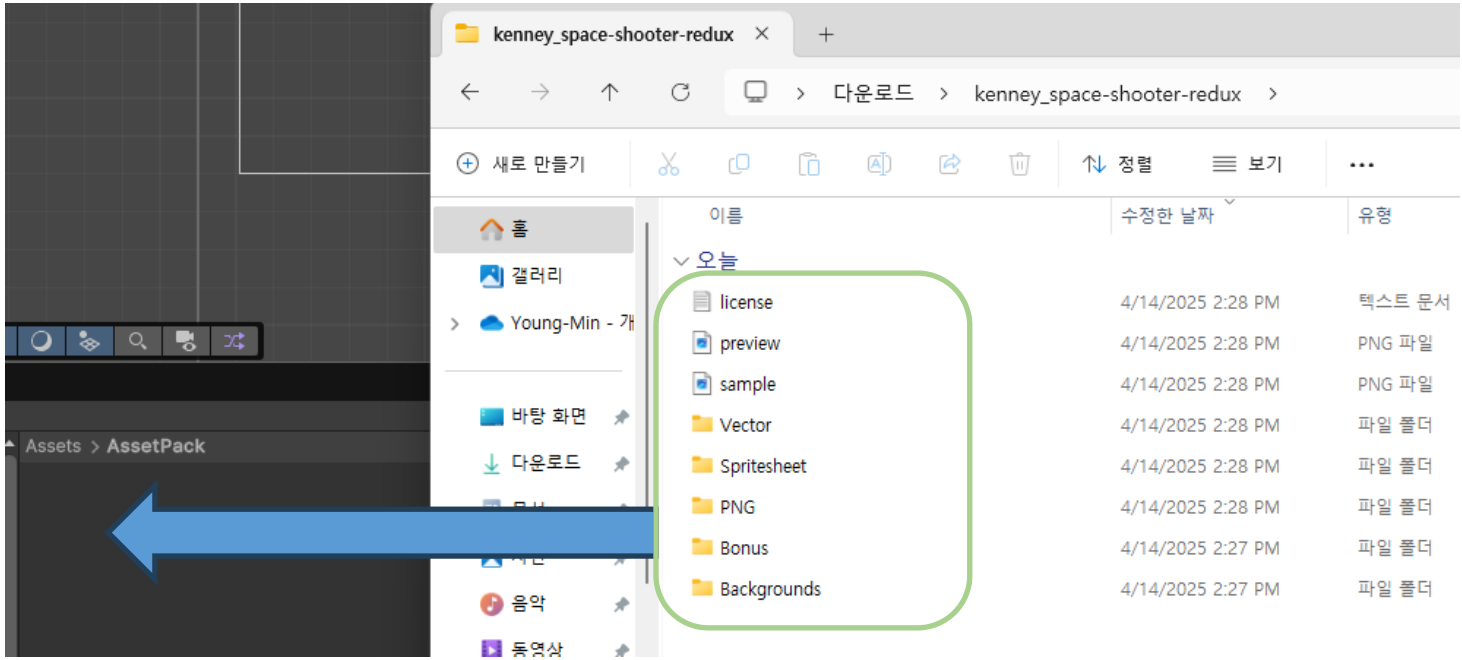






Import assets!

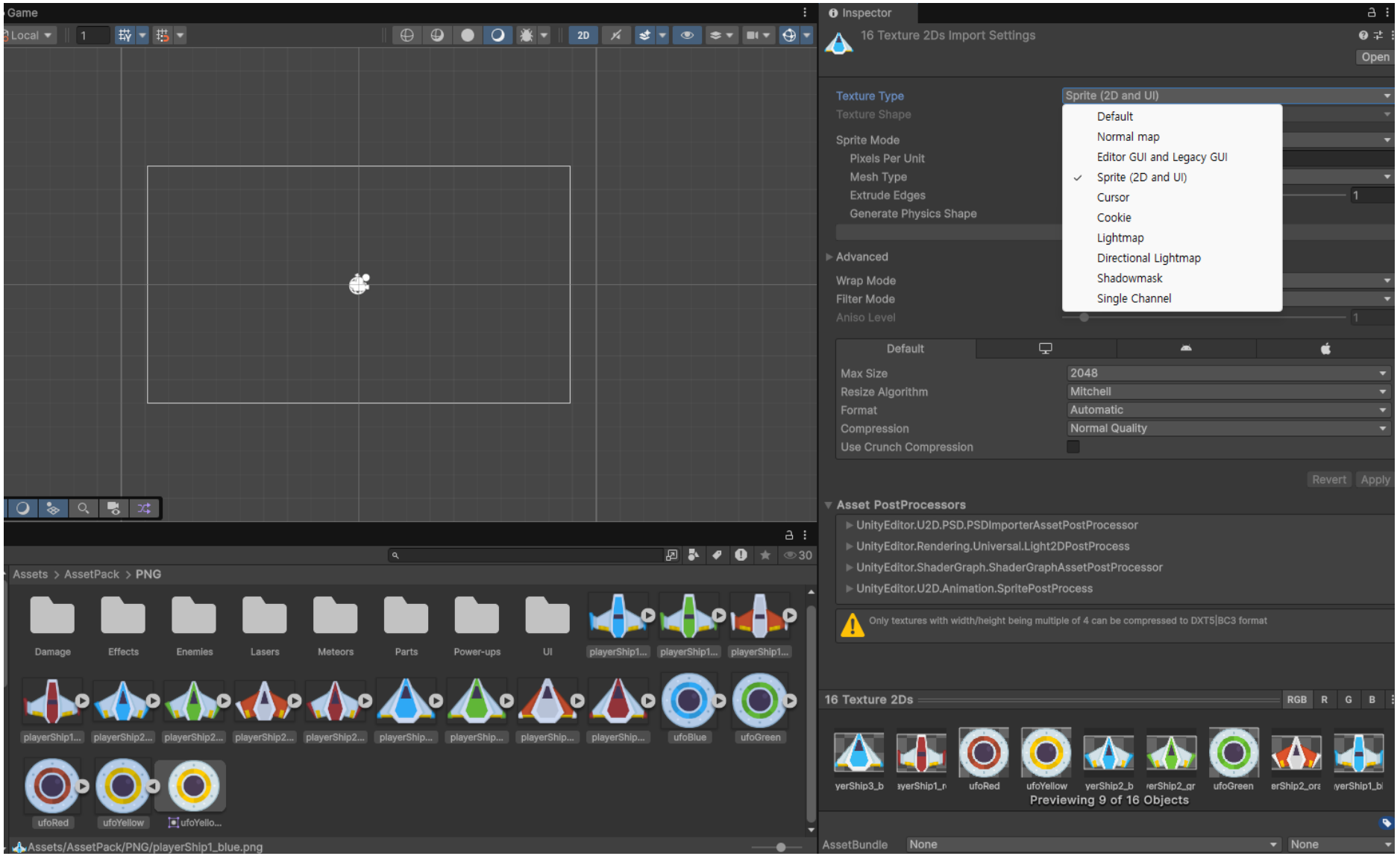




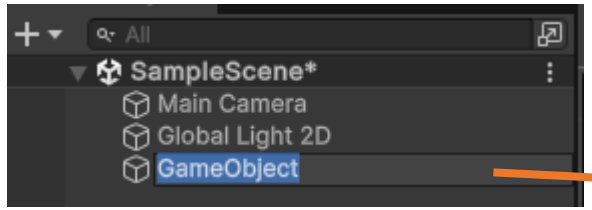
아주 Nice!


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

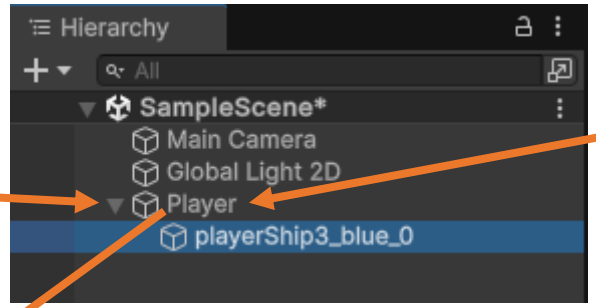
PNG 파일들이 Sprite Texture type임을 확인



Prefab 제작 1 – Player

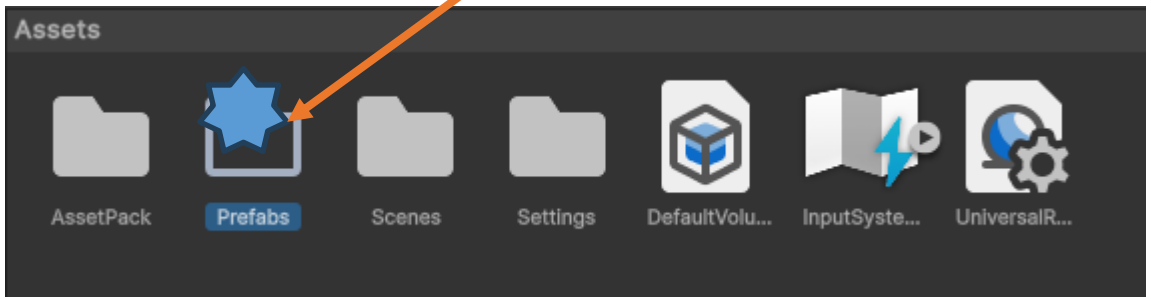


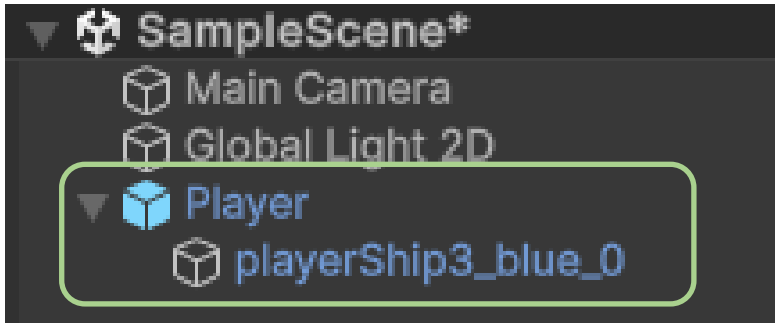
1. Rename



2. Drag & Drop

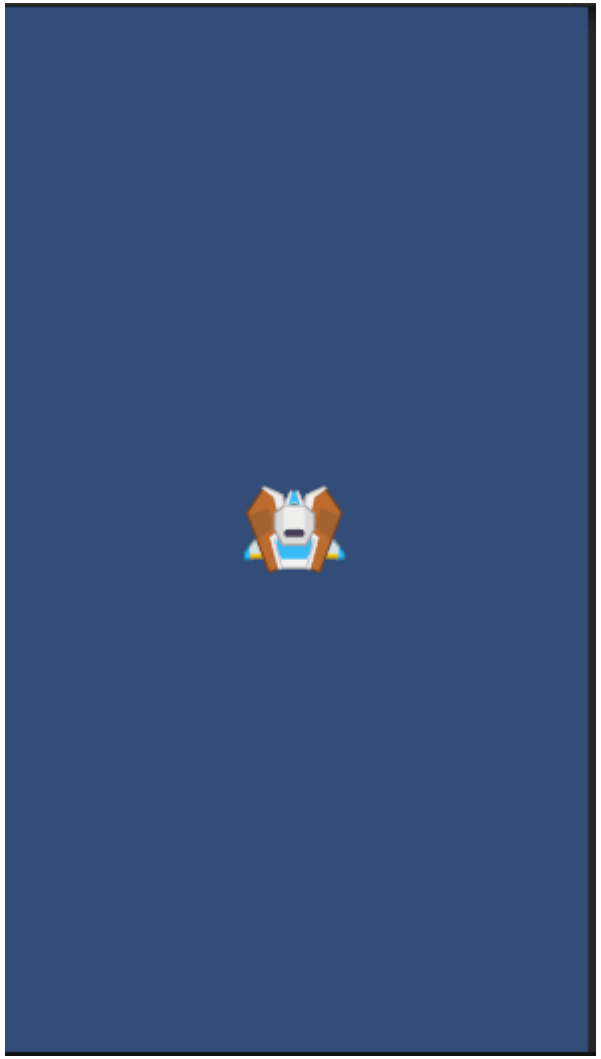
3. Create a folder for Prefabs and
Drag & Drop “player”

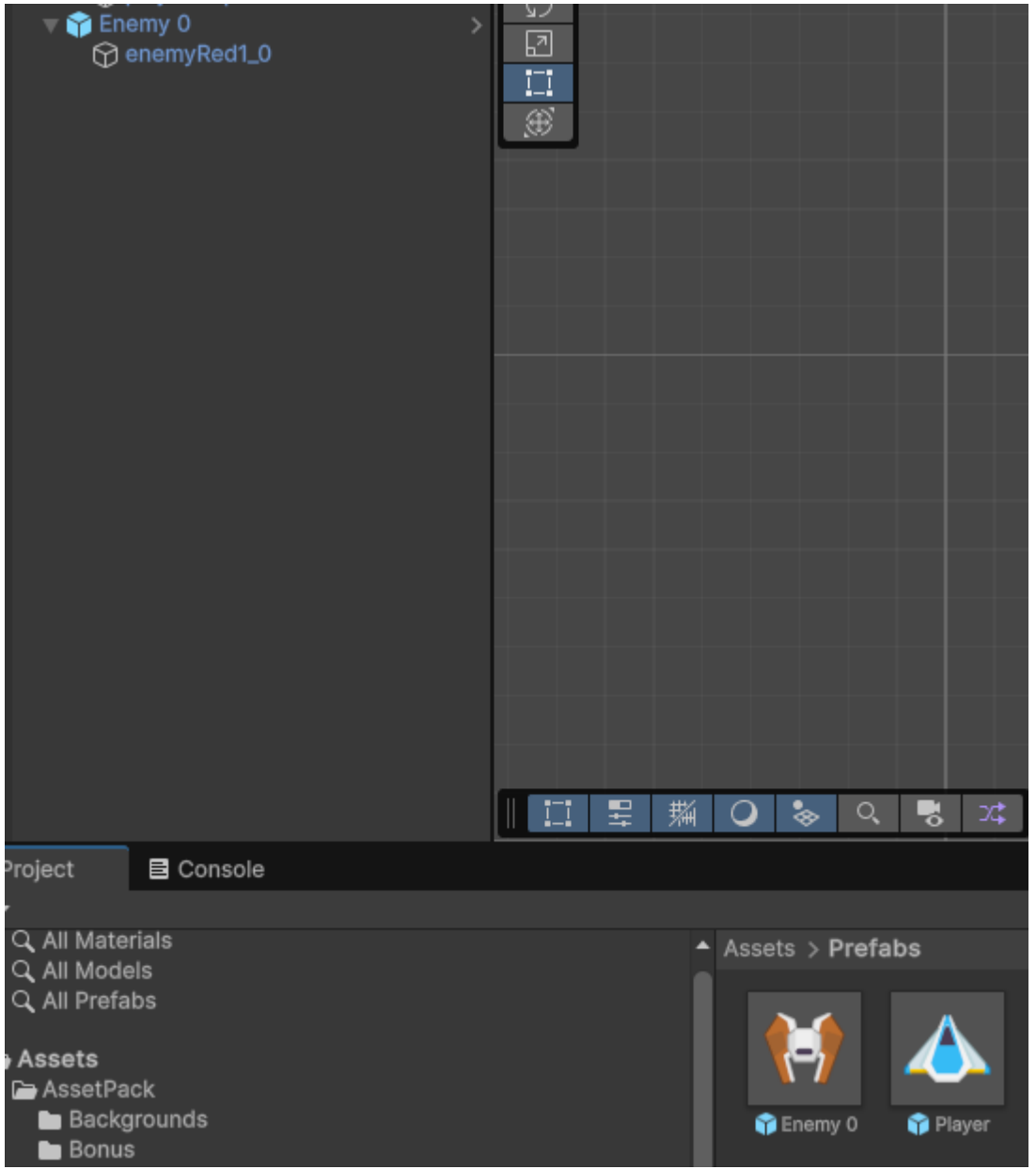




4. Prefab in Hierarchy

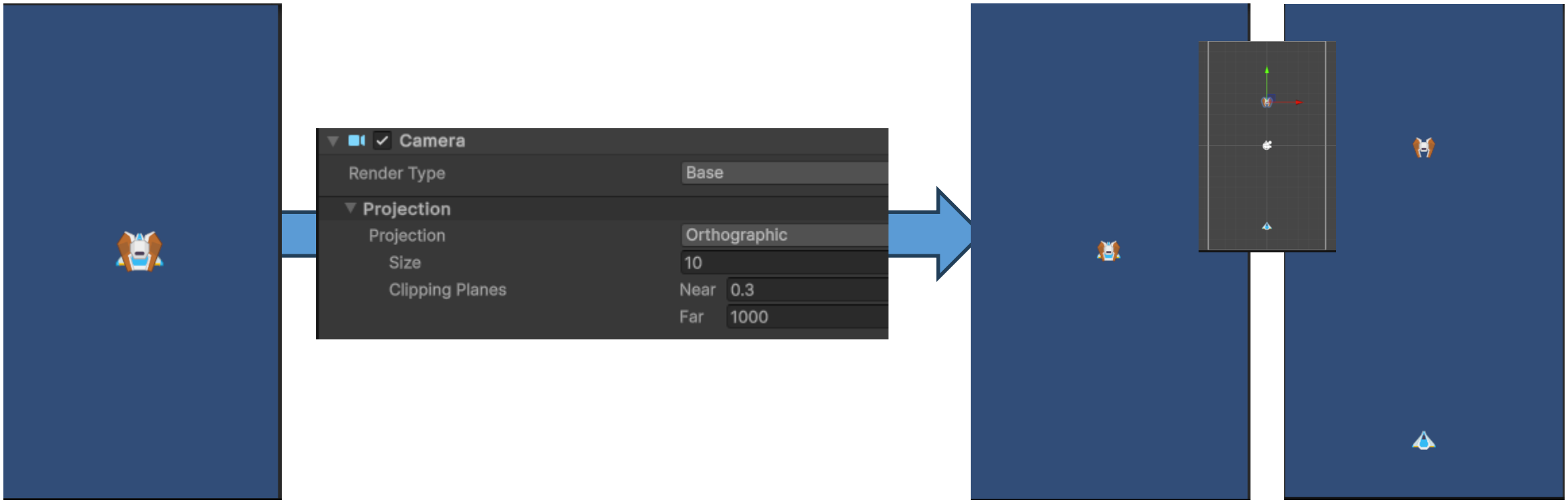
Prefab 제작 2 – Enemy 0





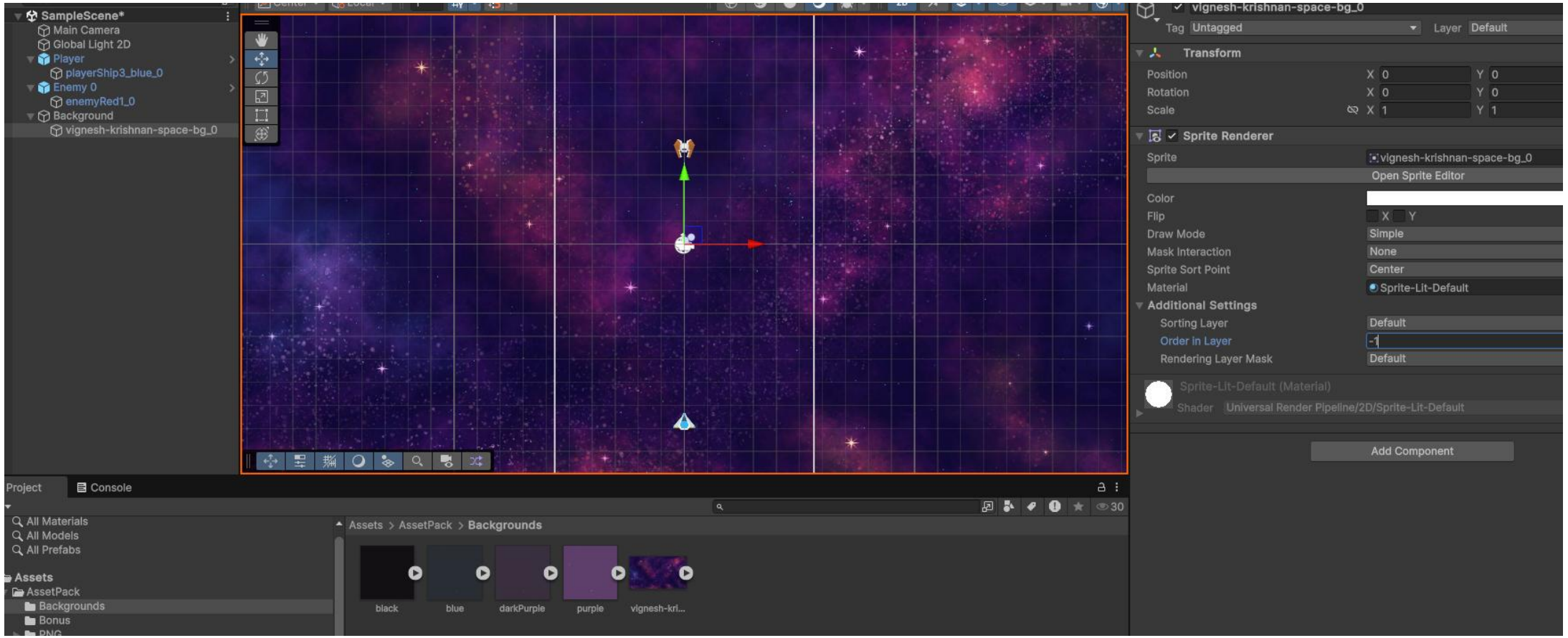
문제점

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



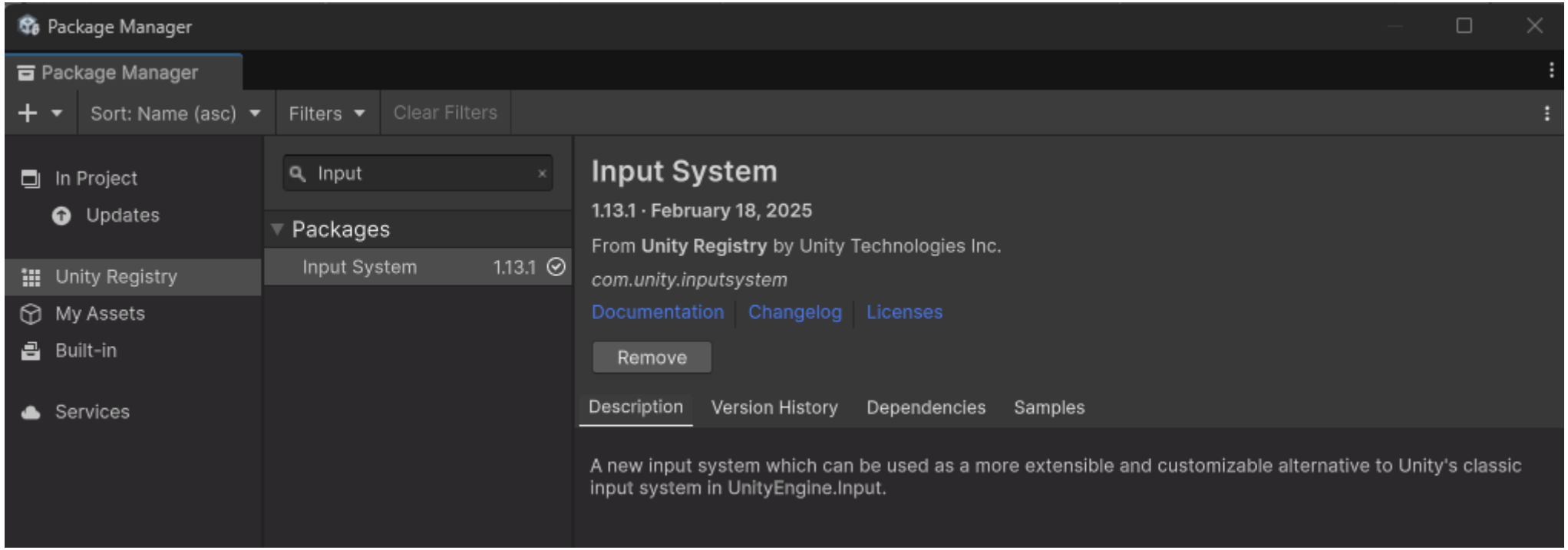
배경 넣기

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



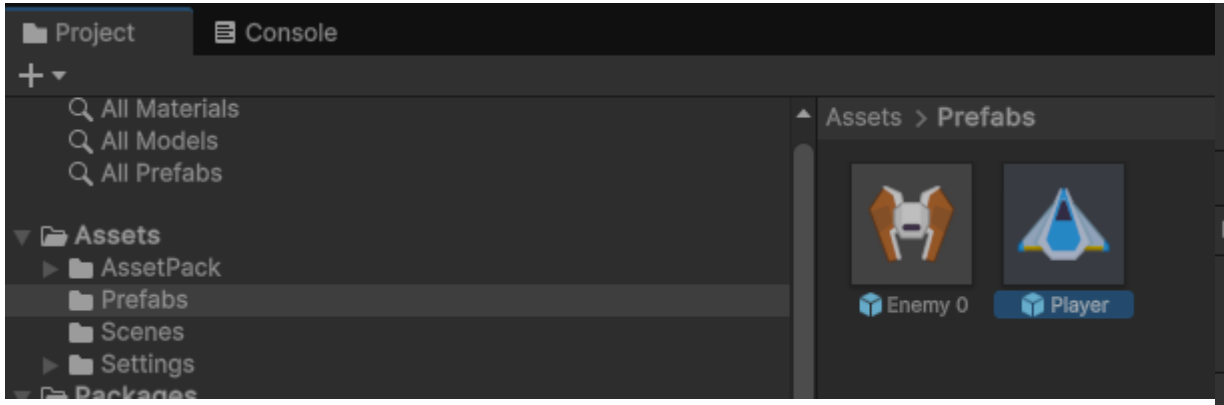
입력 시스템 – New Input System을 사용해 보자

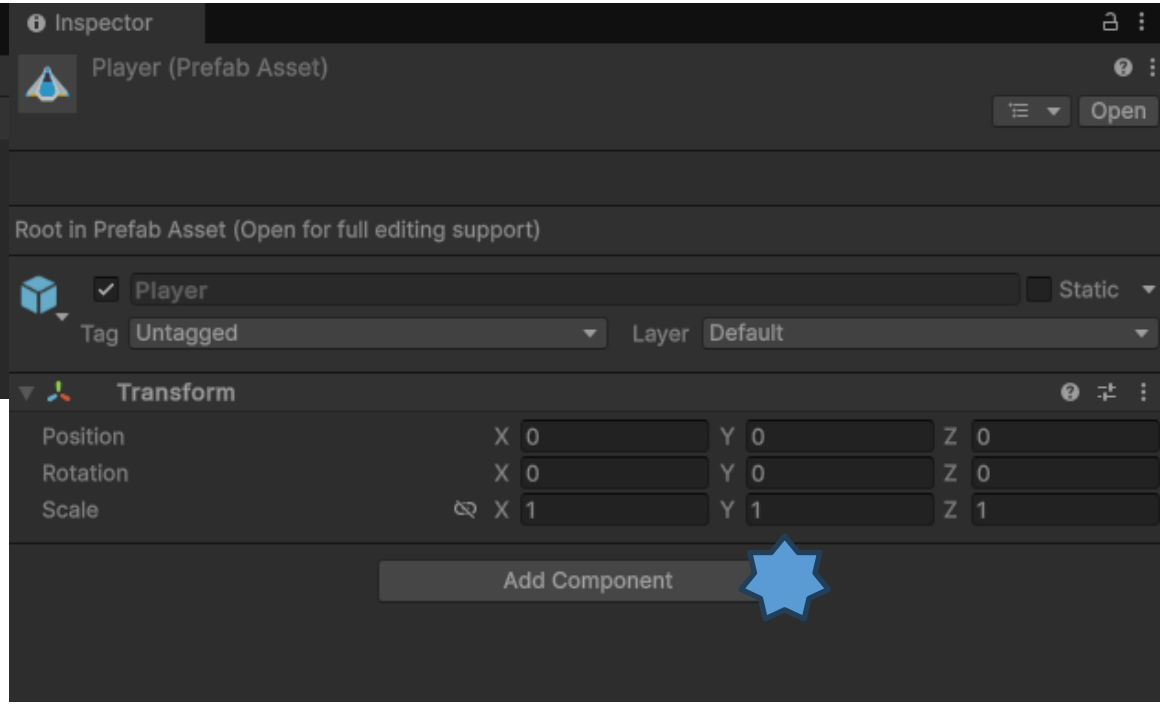
- Windows → Package Manager
  - Input System 패키지 설치를 확인 (없으면 설치)

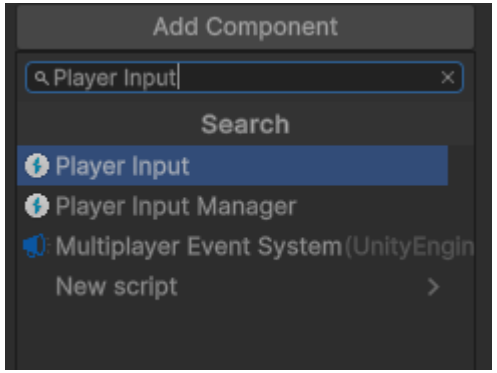


입력 시스템을 플레이어에 적용

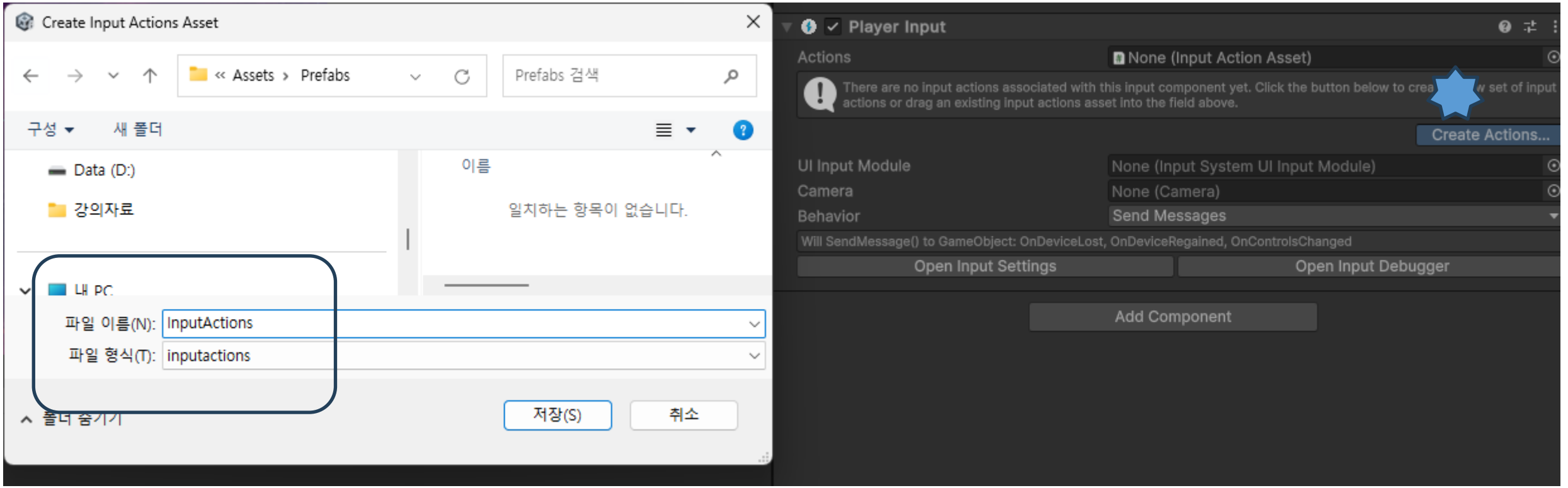
- Hierarchy에 있는 특정 객체가 아니라 Prefab을 변경



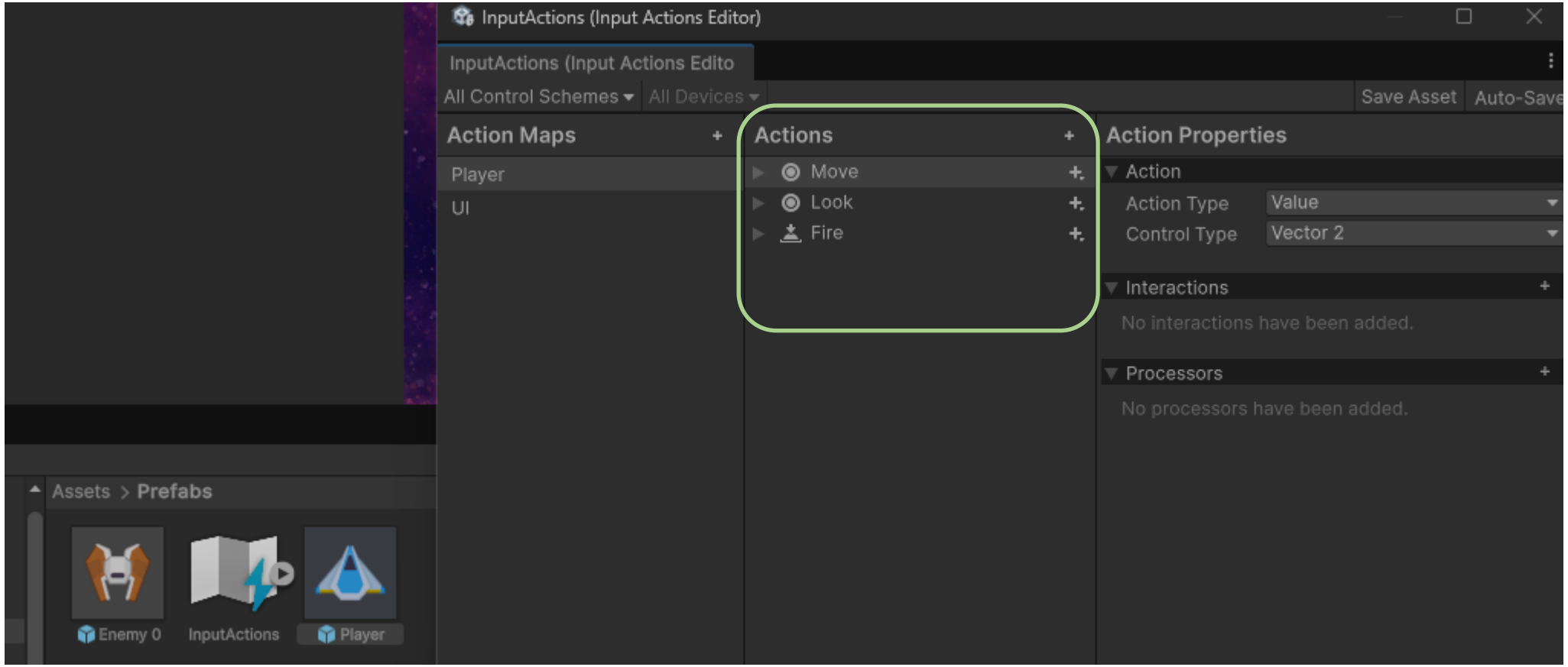




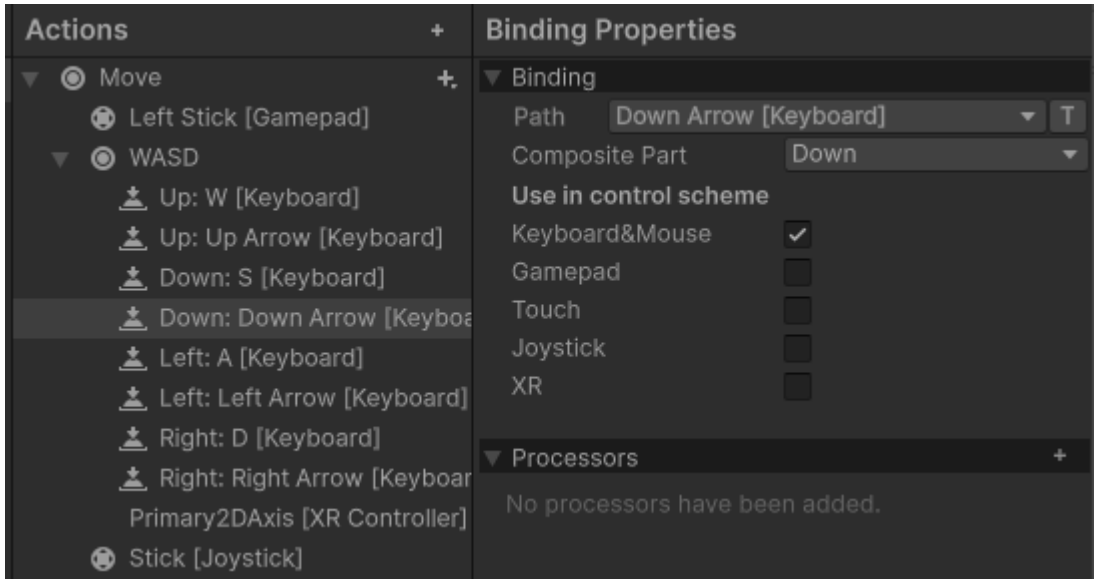
Player Input 설정



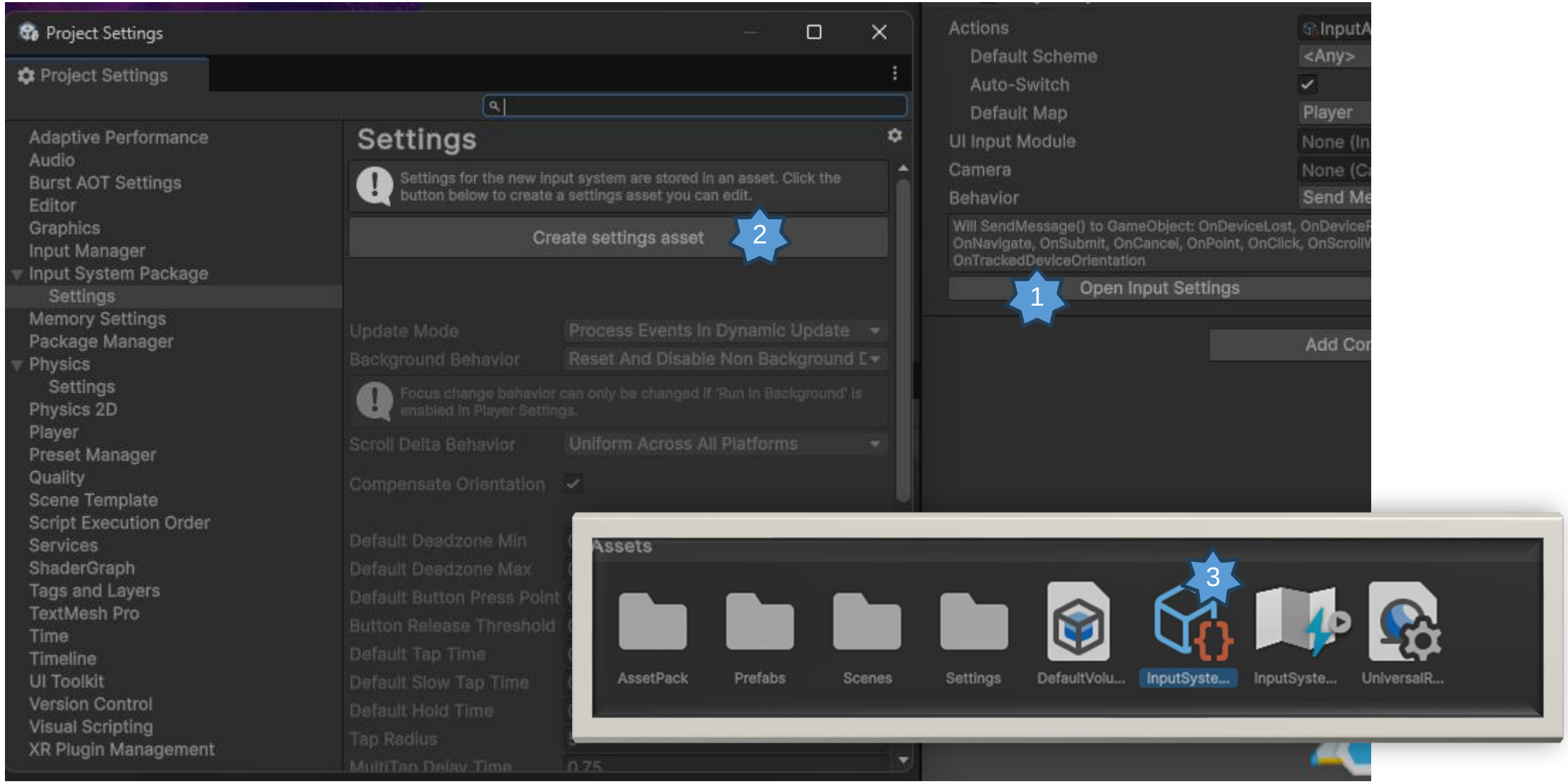
Input Actions



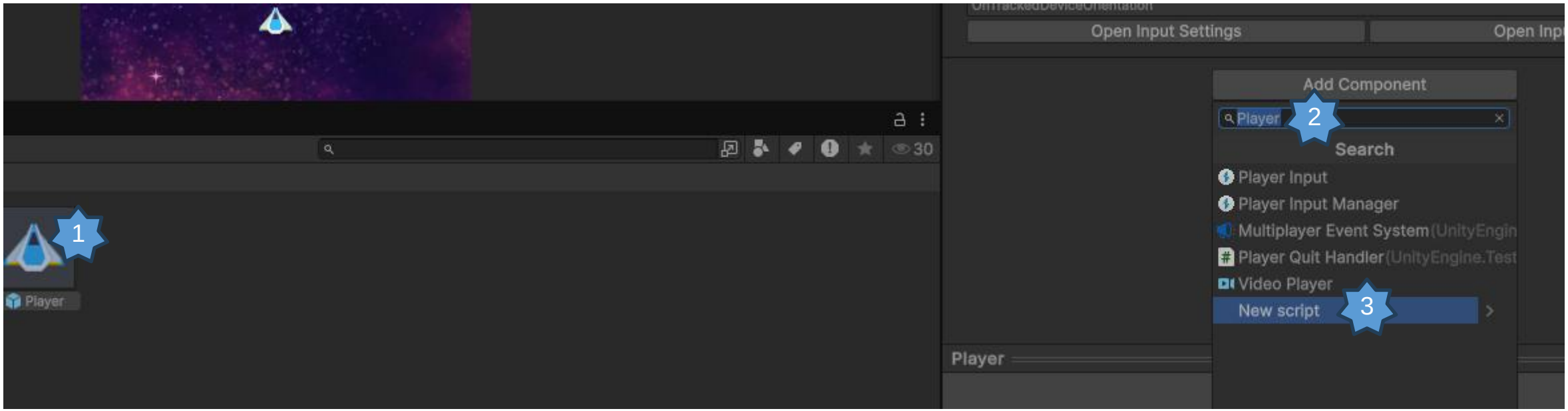
Key-bindings



Player Input이 동작하게 만들기

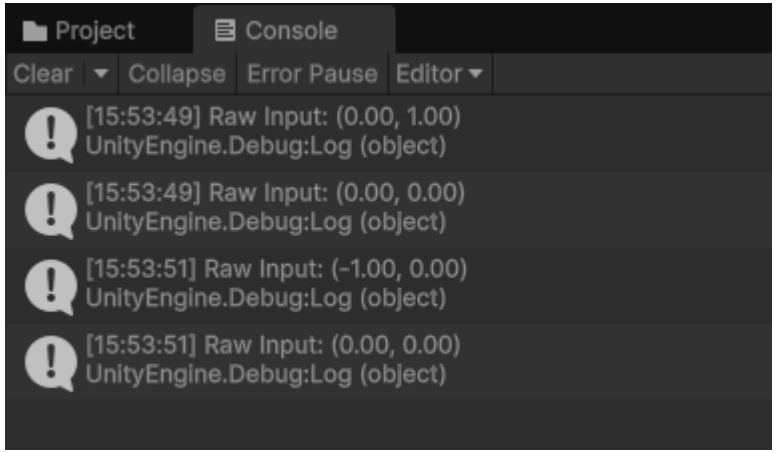


Message를 가로챌 코드 작성



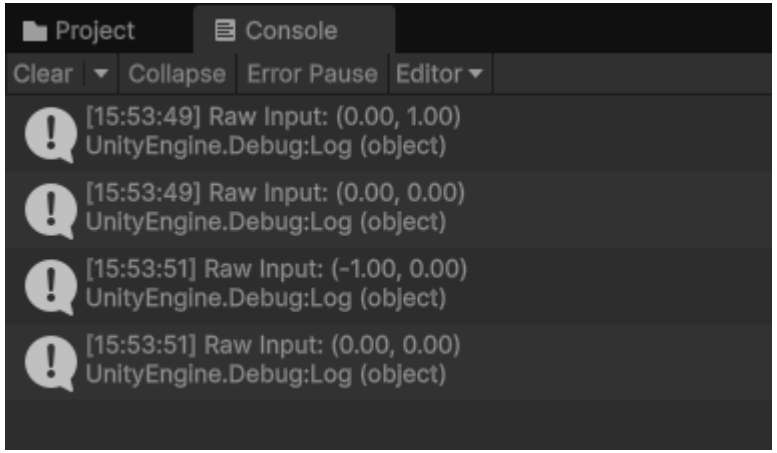
OnMove에 의해 실행될 내용 담기

```
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);
    }
}
```

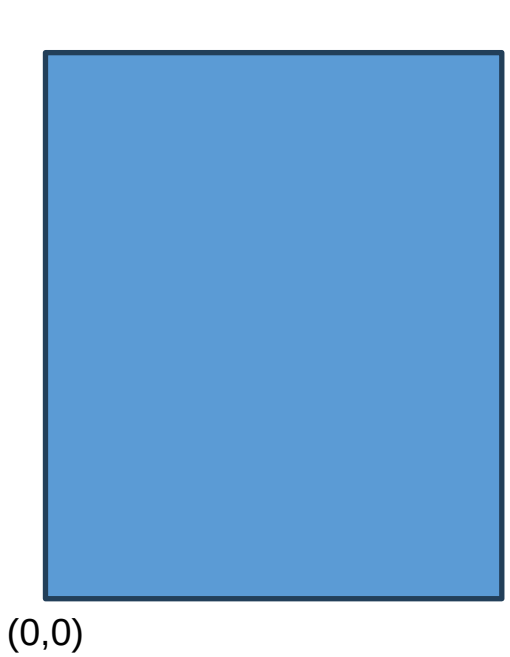


속도를 제어할 수 있게 해 보자

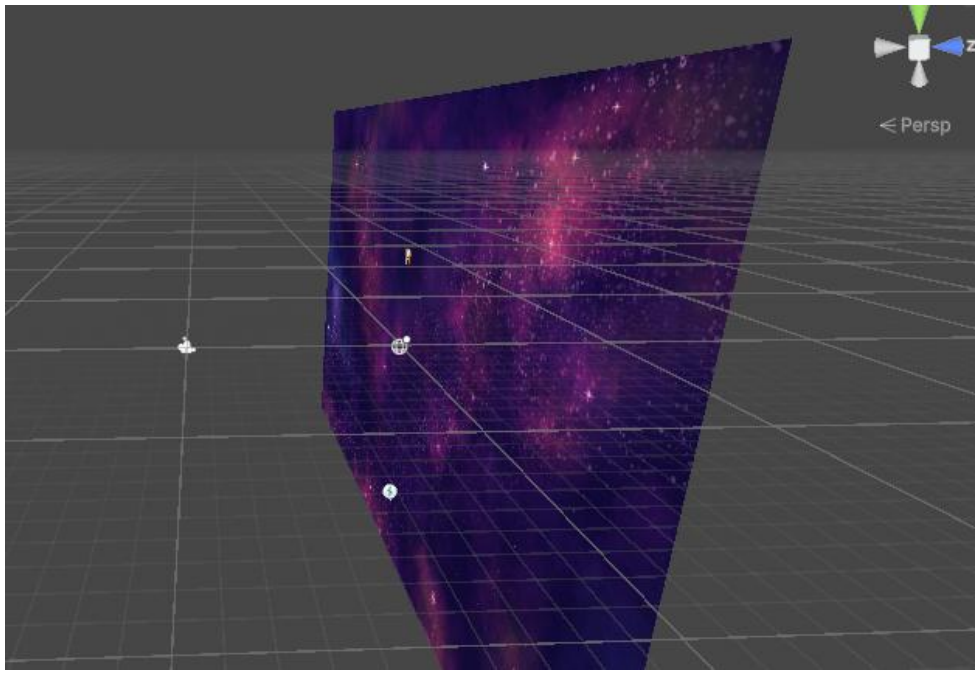
```
using UnityEngine;

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

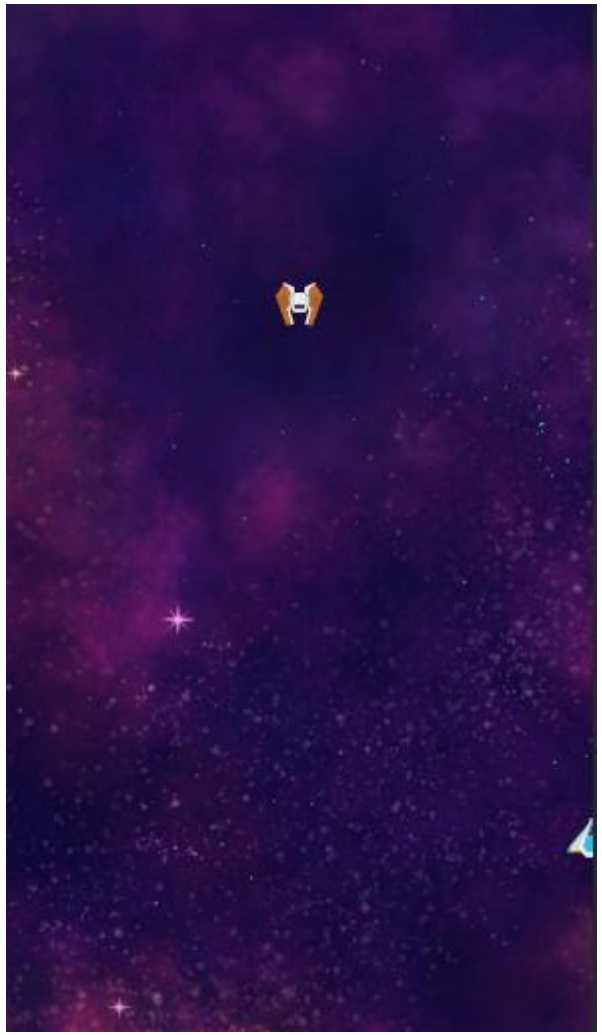


viewportToWorldPoint



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>();
    }
}
```

이제 좀 더 신나는
게임을 만들어 볼까요?