Unity를 이용한 2D 게임프로그래밍

Lecture 9
간단한 아케이드 게임
적의 동작과 각종 효과

강영민

동명대학교 게임공학과

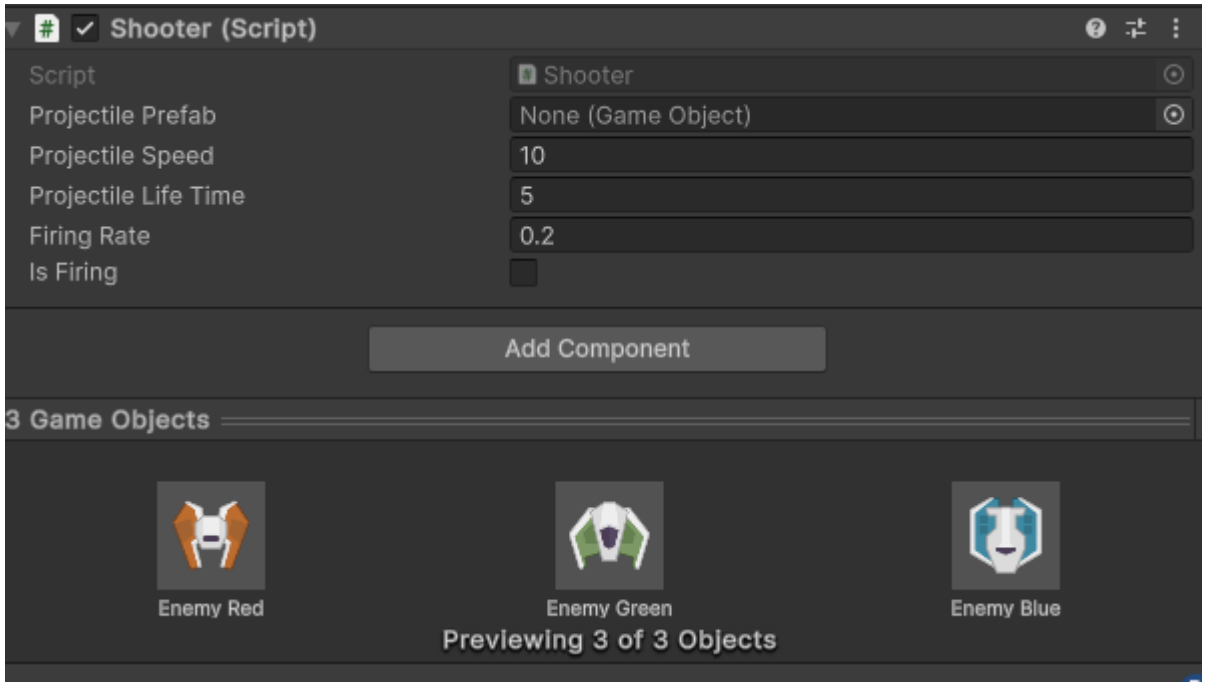
학습목표

- 적의 슈팅 제어
- 각종 이펙트 추가
  - 입자시스템
  - 음향 효과

적의 슈팅

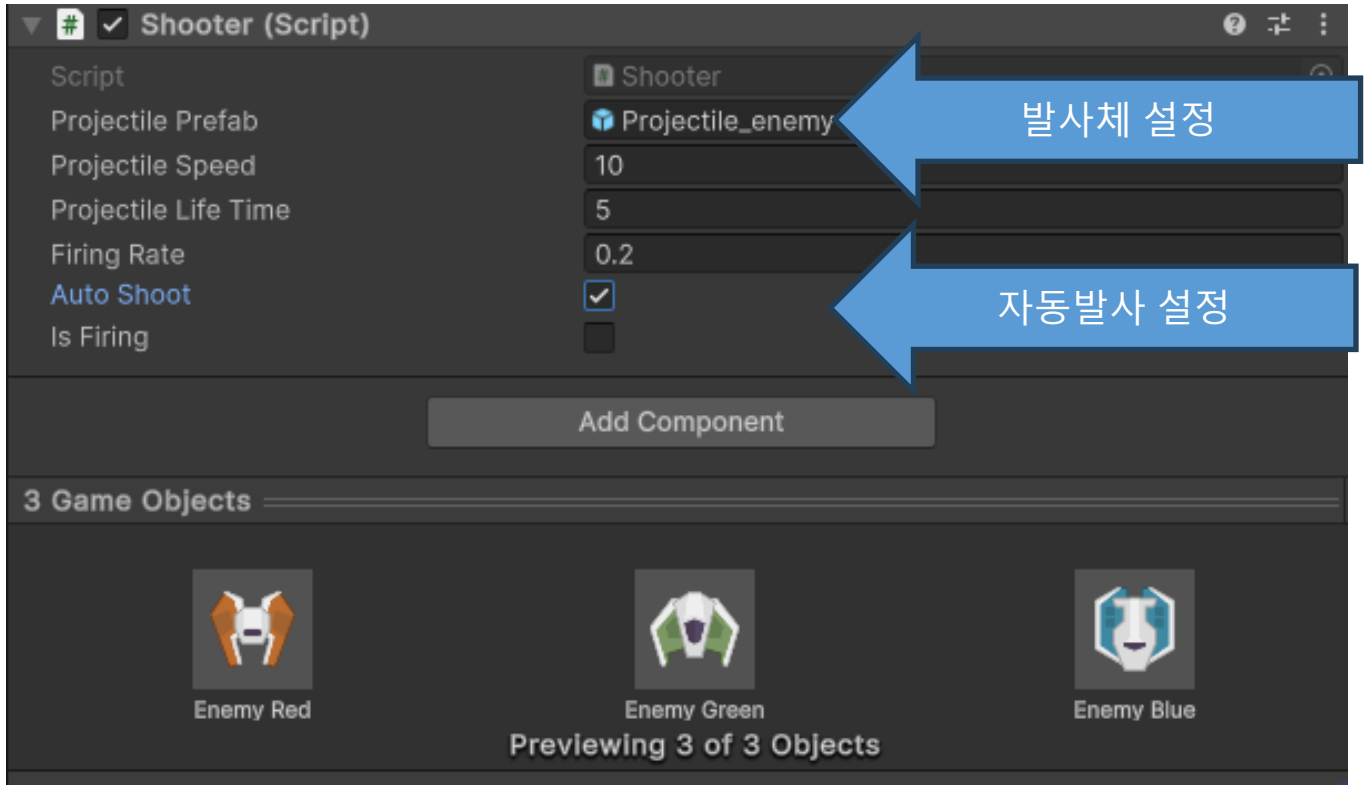
- 이미 만들어 놓은 스크립트 활용

Shooter.cs를 적에 적용



적은 입력이 아니라 자동으로 발사

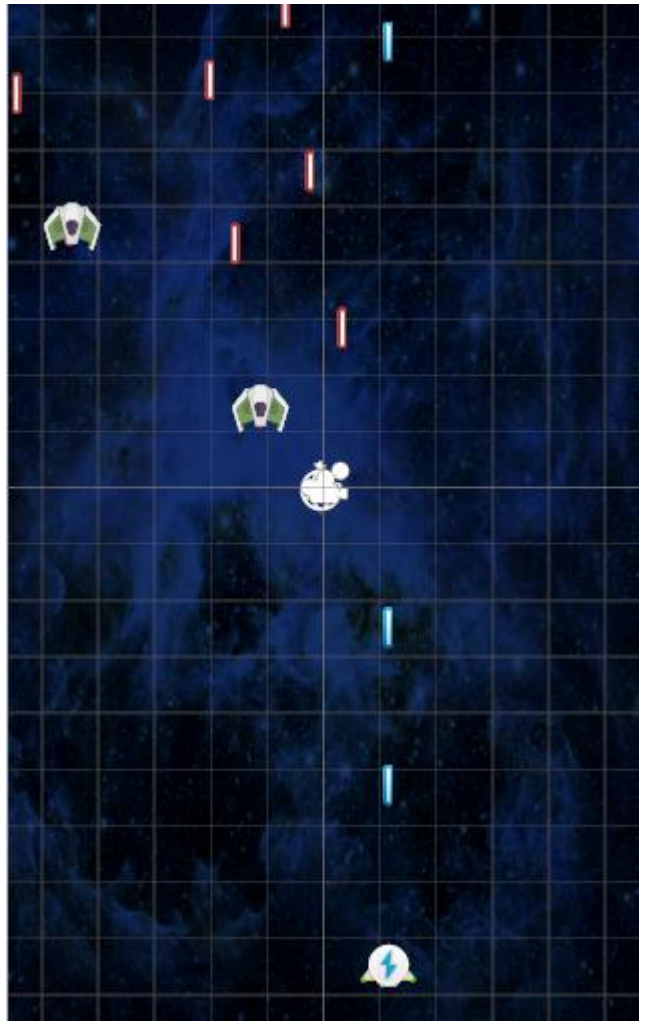
- Shooter.cs 수정: 자동 발사 옵션 추가




```
public class Shooter : MonoBehaviour
{

    1 reference
    [SerializeField] GameObject projectilePrefab;
    1 reference
    [SerializeField] float projectileSpeed = 10f;
    1 reference
    [SerializeField] float projectileLifeTime = 5f;
    1 reference
    [SerializeField] float firingRate = 0.2f;
    1 reference
    [SerializeField] bool autoShoot = false;

    5 references
    public bool isFiring = false;
    5 references
    Coroutine firingRoutine;

    0 references
    void Start()
    {
        if (autoShoot)
        {
            isFiring = true;
        }
    }
```

발사 방향 조정

- 거꾸로 쏘는 적기



자동 발사인 경우 반대 방향으로 속도 지정

- 무시무시하게 많은 적의 총알들

```
IEnumerator FireContinuously()
{
    while(true)
    {
        // firing action
        GameObject missile = Instantiate(projectilePrefab,
            transform.position, Quaternion.identity);

        Rigidbody2D rb = missile.GetComponent<Rigidbody2D>();
        if (rb != null)
        {
            if (autoShoot)
            {
                rb.linearVelocity = -transform.up * projectileSpeed;
            }
            else
            {
                rb.linearVelocity = transform.up * projectileSpeed;
            }

        }

        Destroy(missile, projectileLifeTime);
        yield return new WaitForSeconds(firingRate);
    }
}
```

적기의 발사 속도 조정

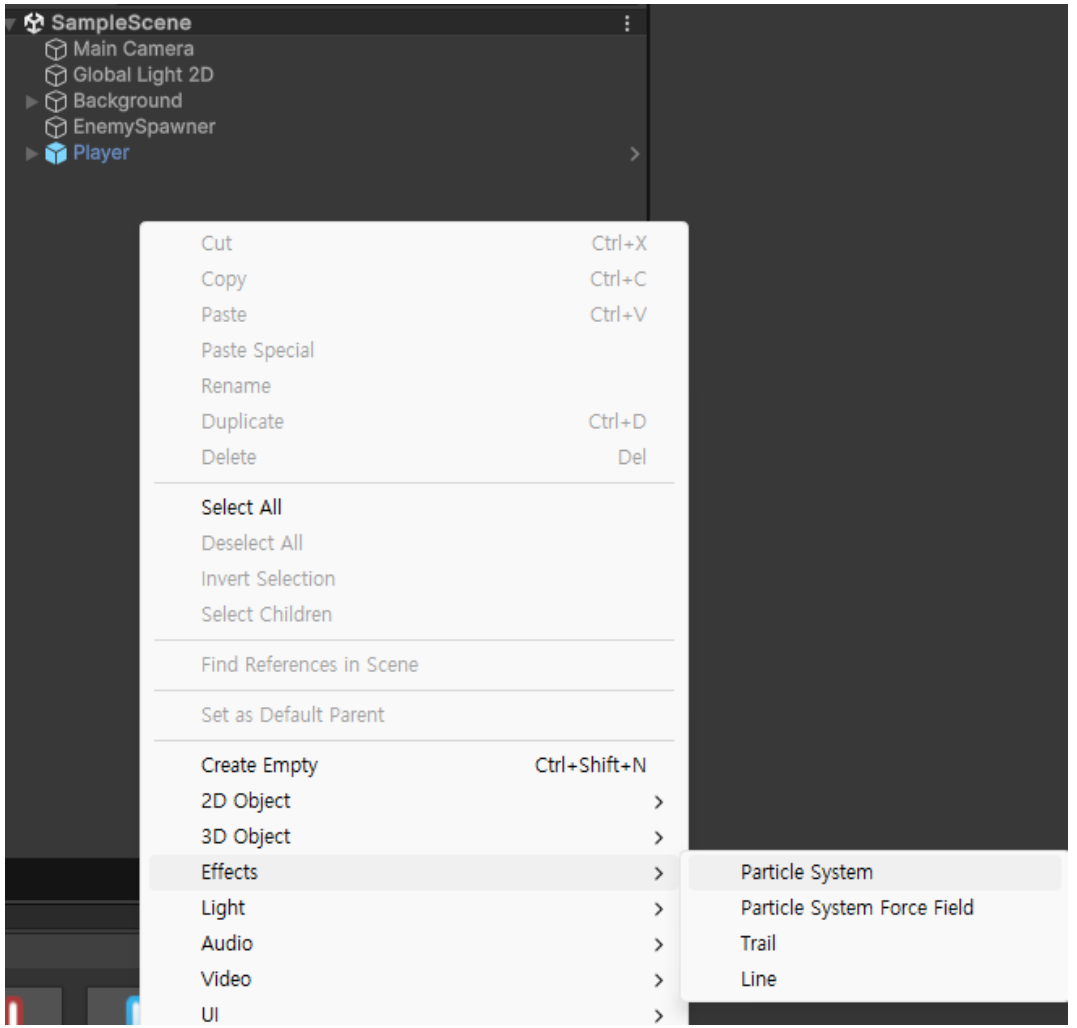
```
IEnumerator FireContinuously()
{
    while(true)
    {
        // firing action
        GameObject missile = Instantiate(projectilePrefab,
            transform.position, Quaternion.identity);

        Rigidbody2D rb = missile.GetComponent<Rigidbody2D>();
        if (rb != null) ···

        Destroy(missile, projectileLifeTime);
        float delay = autoShoot ? 4f : 1f;
        yield return new WaitForSeconds(delay*firingRate);
    }
}
```

폭발 효과 – Particle System

- Particle System 생성
  - 입자 효과 구현
- Prefab으로 만들기
- 필요할 때마다 Instantiate

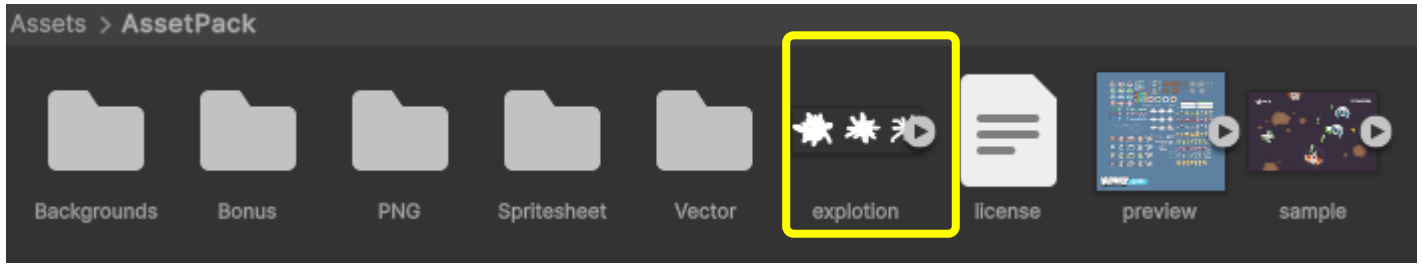
폭발 효과 – Particle System

- Particle System 생성



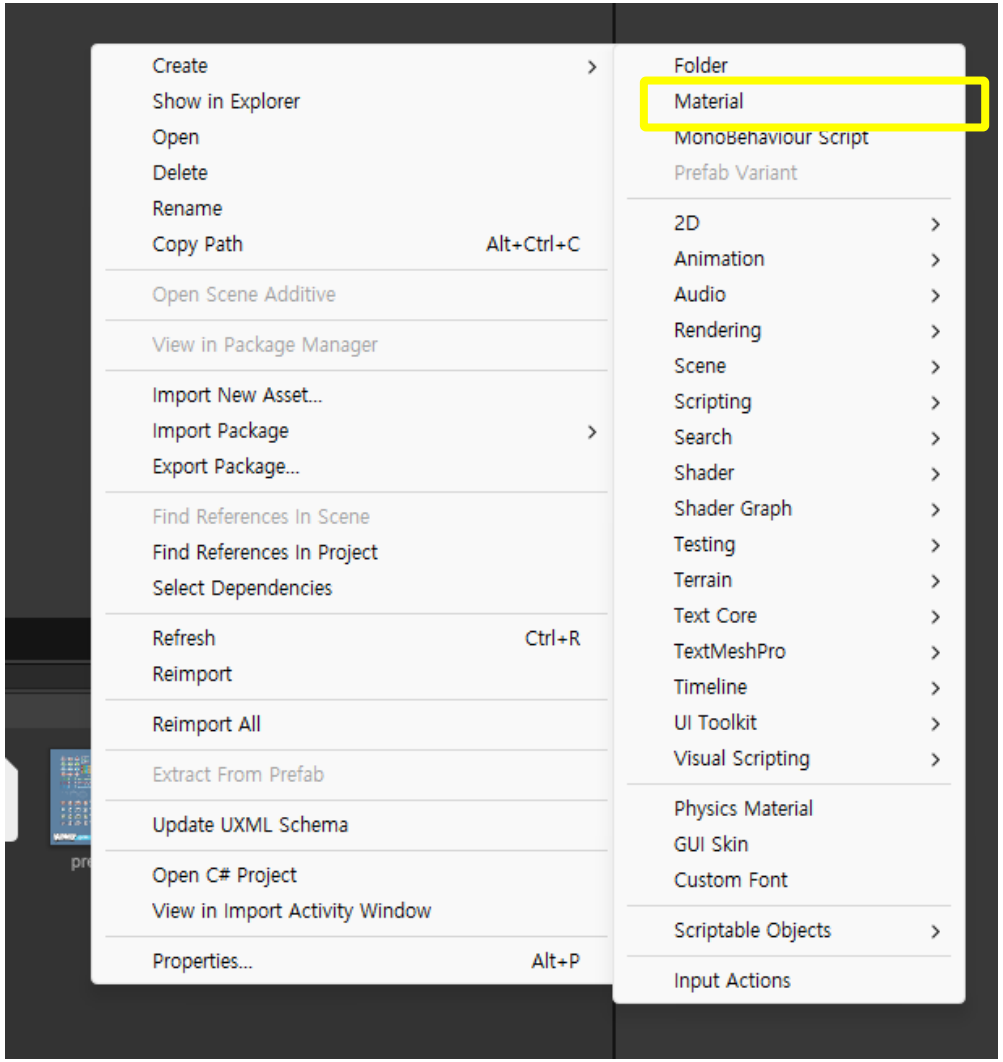
폭발 효과를 보일 스프라이트 다운로드

- 강의 홈페이지에서 다운로드
- AssetPack에 포함시키기



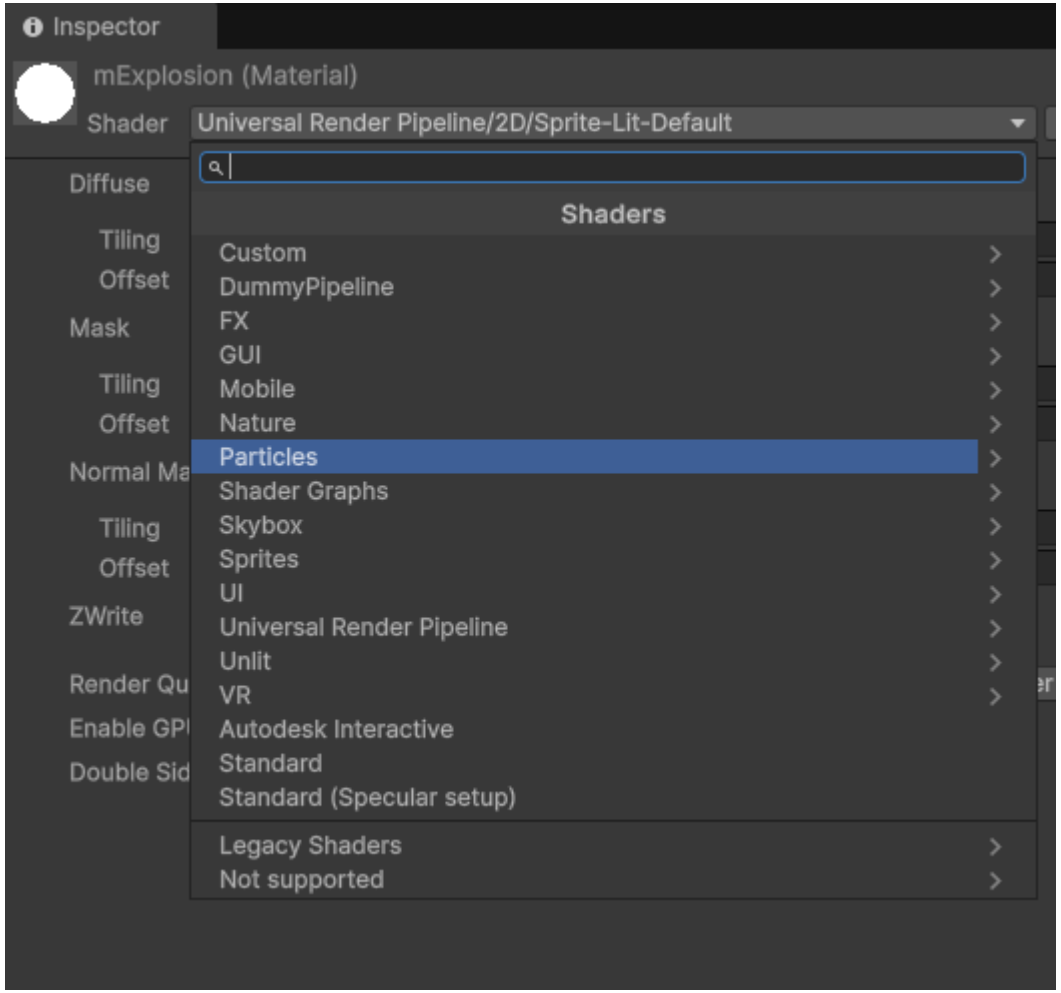
입자 시스템에 적용할 “재질-material” 만들기

- Asset 폴더에 Material 생성



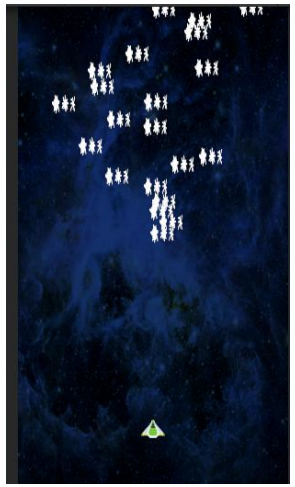
재질 설정

- Material 선택
  - Particle Shader로 변경
    - Standard Unlit
  - Rendering Mode
    - Additive (블렌딩 방법)
  - Map
    - Albedo (색상 결정)
      - 스프라이트 연결

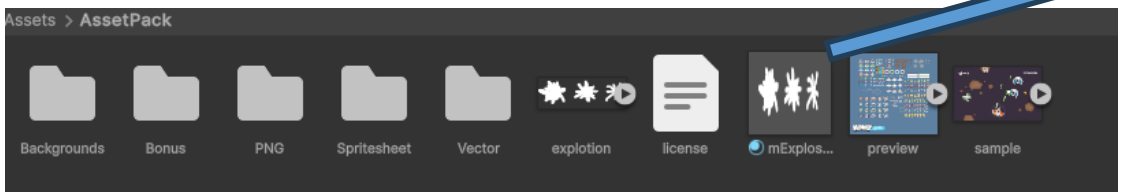


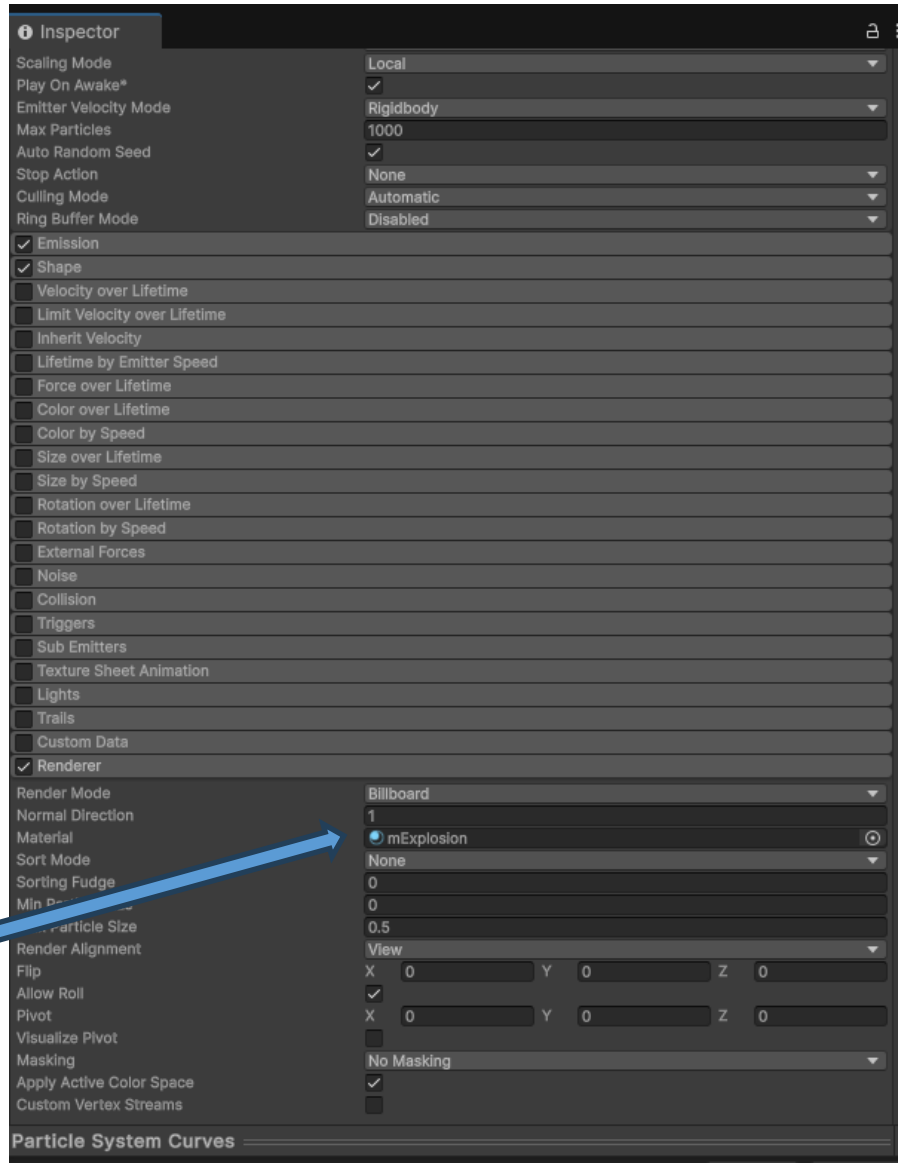
Particle System에 만든 재질을 적용

Particle System의 Inspector
➔ Renderer ➔ Material



만든 재질을 가져다 설정

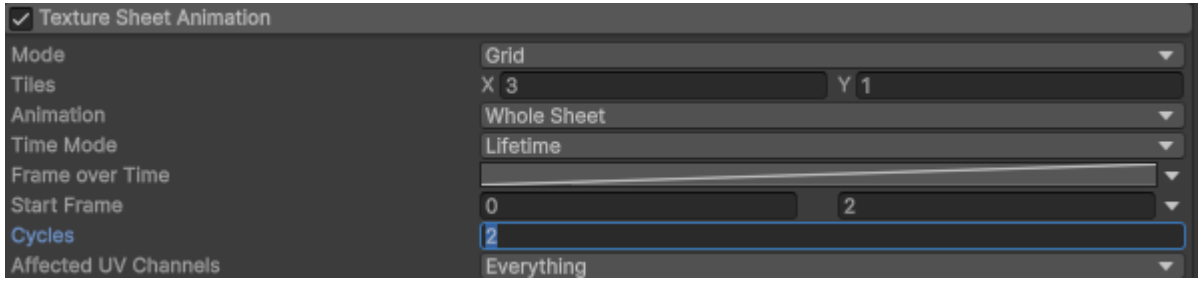




Particle System 설정 변경

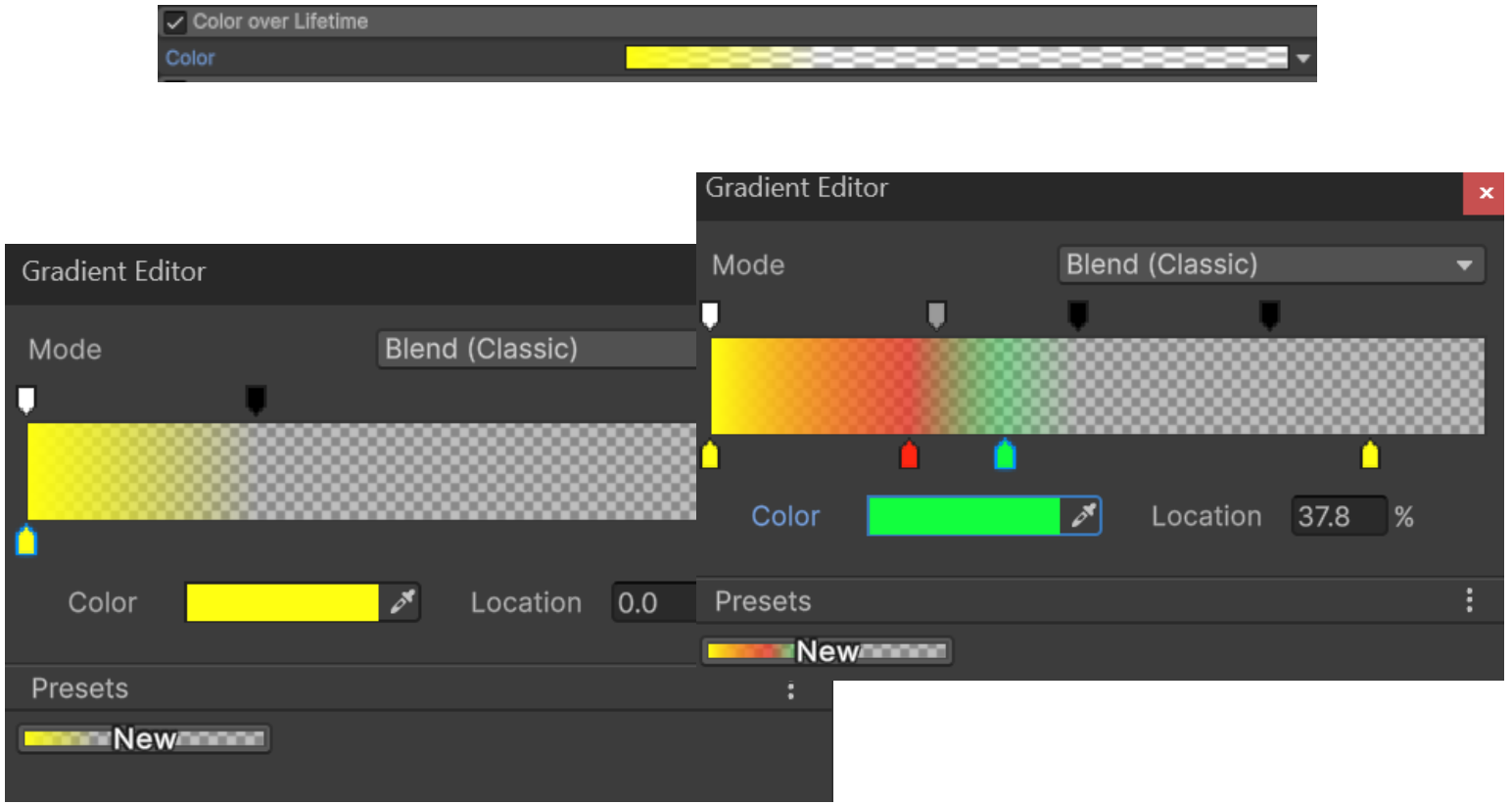
Sprite Sheet에서 한 영역만 가져오기: 스프라이트 시트를 3개의 타일로 변경

파티클이 존재하는 동안 두 번의 사이클로 스프라이트 애니메이션 실행 + 시작 프레임 랜덤화



Particle System 설정 변경

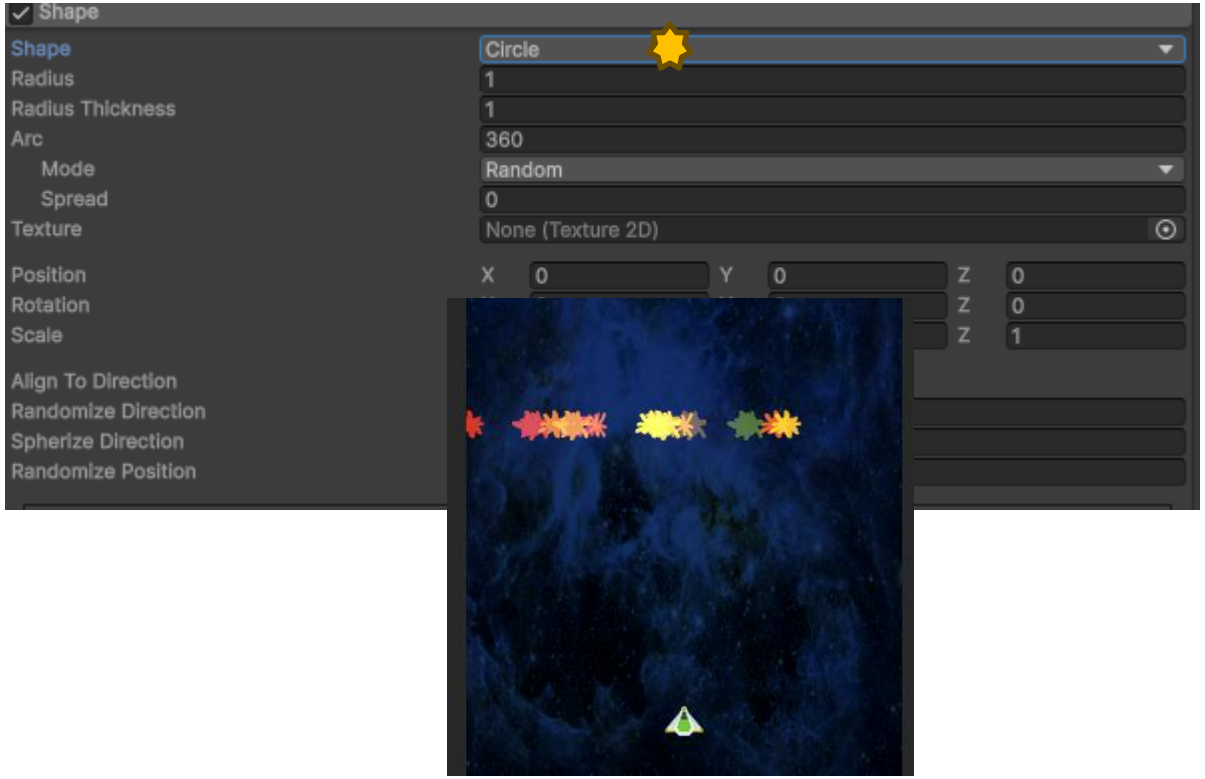
색상 변경

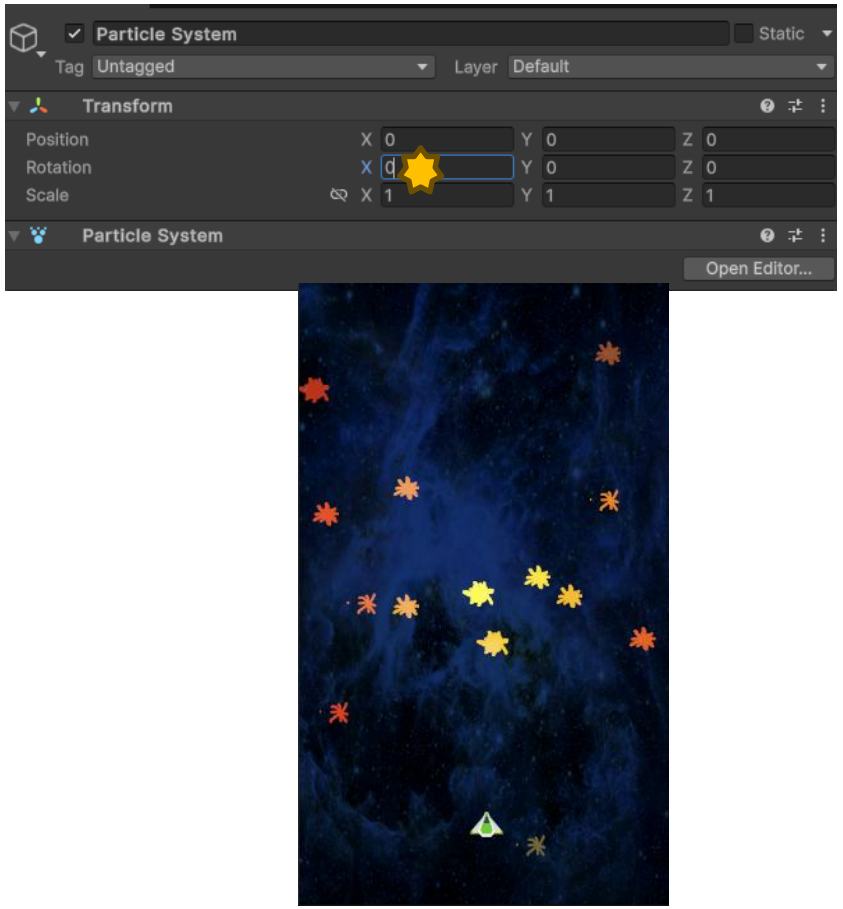


투명도와 색상 조정 가능

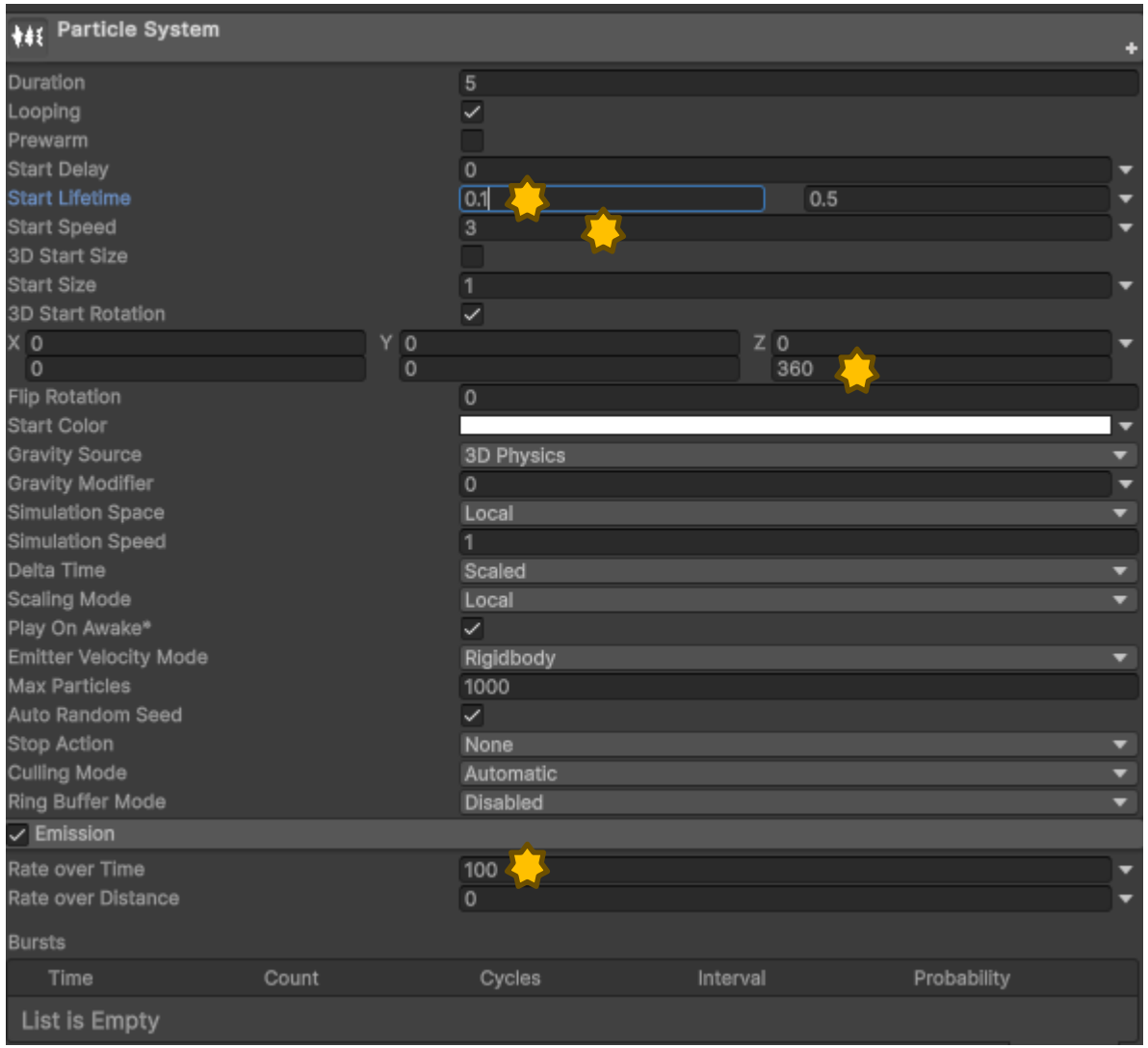
Particle System 설정 변경

분출 형태 (emission shape) ➔ Circle

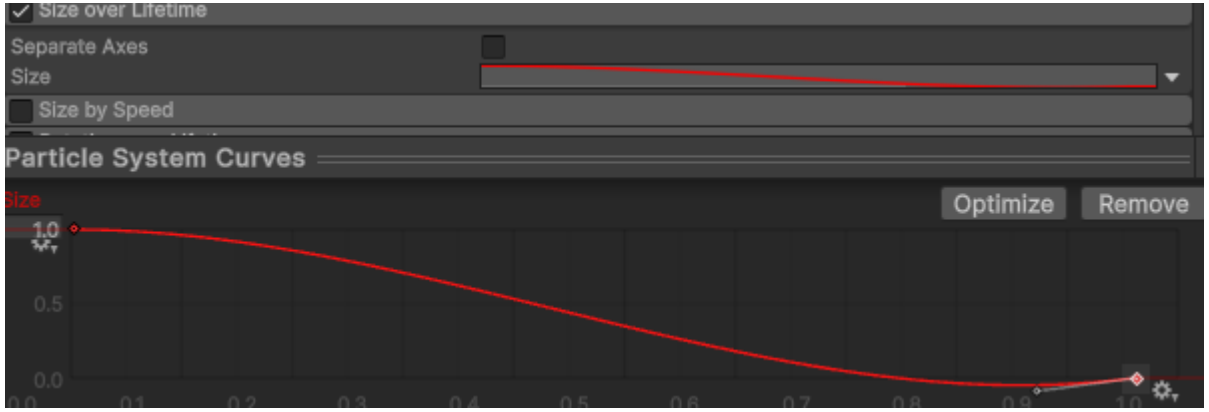


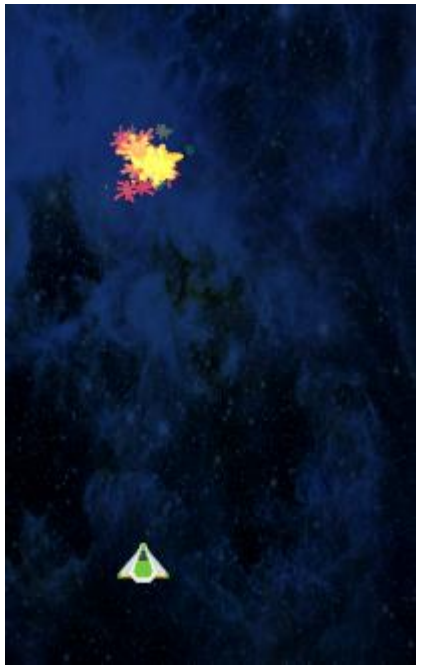


발사체 제작 – Enemy 발사체



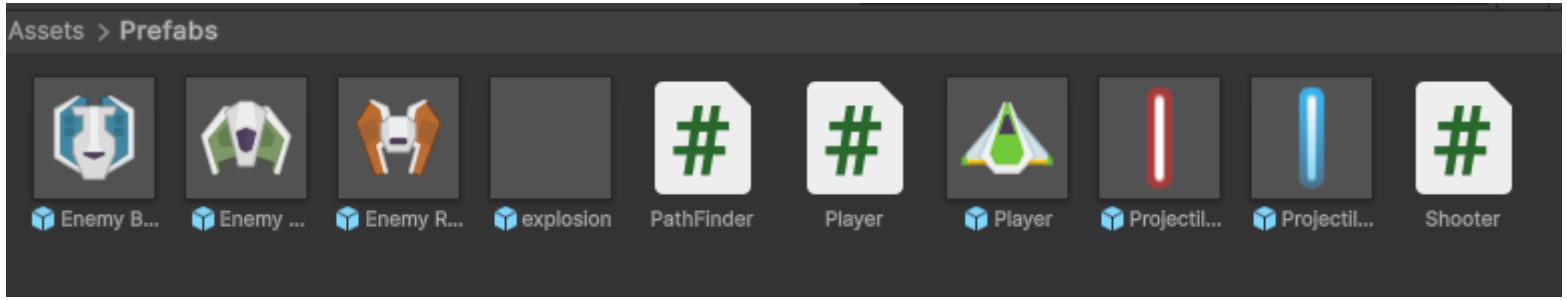
다양한 추가 조정


Size over Lifetime
Separate Axes
Size
Size by Speed
Particle System Curves
Optimize
Remove
1.0
0.5
0.0



입자 시스템을 프리팹으로

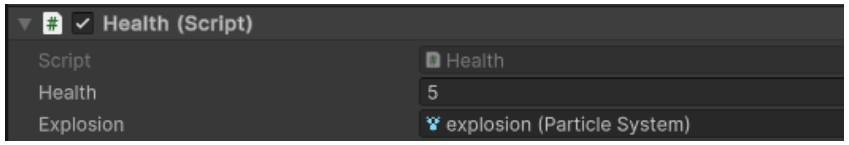
- 프리팹 폴더에 끌어다 놓고 Hierarchy에서 삭제
  - 적절한 이름으로 변경 (explosion)



Health.cs 수정 → 파괴될 때 효과

- Health.cs

```
public class Health : MonoBehaviour
{
    [SerializeField] int health = 50;
    [SerializeField] ParticleSystem explosion;
```

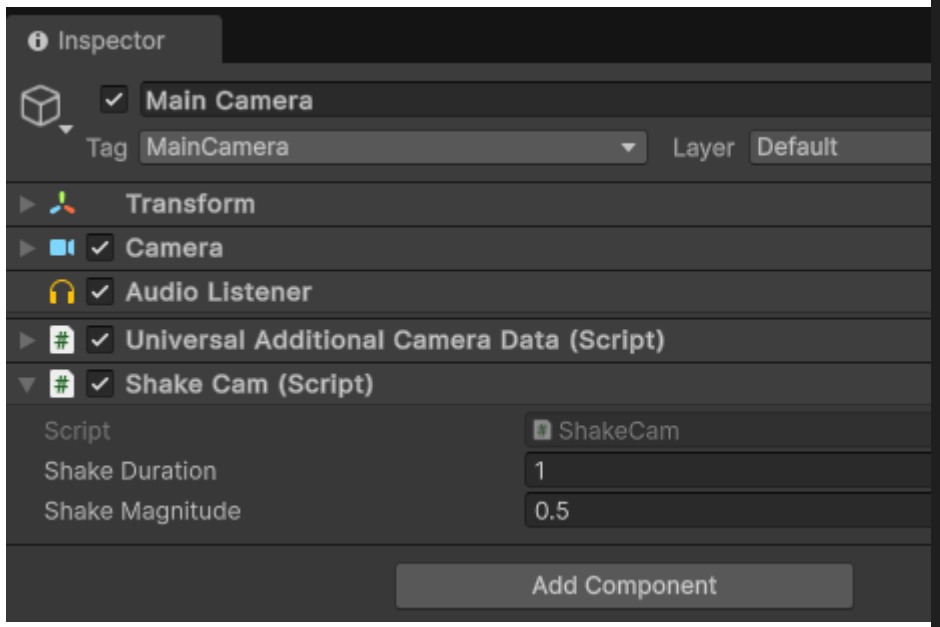


Enemy와 Player 모두 Explosion 등록

```
void TakeDamage(int damage)
{
    health -= damage;
    if (health <= 0)
    {
        if (explosion != null) ExplosionEffect();
        Destroy(gameObject);
    }
}

void ExplosionEffect()
{
    ParticleSystem effect =
        Instantiate(explosion, transform.position, Quaternion.identity);

    Destroy(effect.gameObject, effect.main.duration);
}
```

카메라 흔들기 효과

- ShakeCam.cs
  - 카메라에 추가




```cs
using UnityEngine;
using System.Collections;

public class ShakeCam : MonoBehaviour
{
    [SerializeField] float shakeDuration = 1f;
    [SerializeField] float shakeMagnitude = 0.5f;

    Vector3 initialPos;

    void Start()
    {
        initialPos = transform.position;
    }

    public void Play()
    {
        StartCoroutine(Shake());
    }

    IEnumerator Shake()
    {
        float elapsedTime = 0f;
        while(elapsedTime < shakeDuration)
        {
            transform.position = initialPos + (Vector3) Random.insideUnitCircle * shakeMagnitude;
            elapsedTime += Time.deltaTime;
            yield return new WaitForEndOfFrame();
        }
        transform.position = initialPos;
    }
}
```

Heath.cs에서 카메라 흔들기 필요시 실행

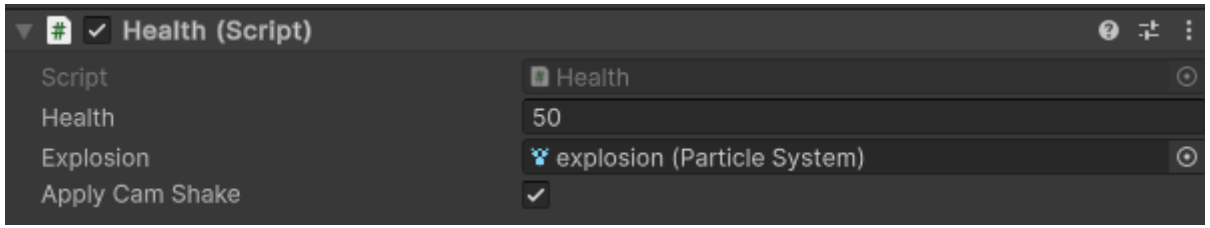
```
public class Health : MonoBehaviour
{
    [SerializeField] int health = 50;
    [SerializeField] ParticleSystem explosion;

    [SerializeField] bool applyCamShake;
    ShakeCam shakeCam;

    void Awake()
    {
        shakeCam = Camera.main.GetComponent<ShakeCam>();
    }
```

```
void TakeDamage(int damage)
{
    health -= damage;

    ShakeCamera();
    if (health <= 0)
    {
        if (explosion != null) ExplosionEffect();
        Destroy(gameObject);
    }
}
```

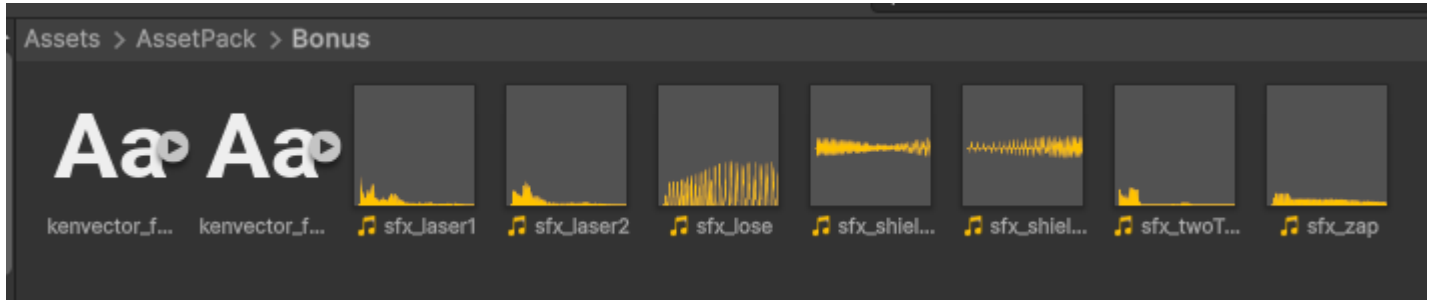
```
void ShakeCamera()
{
    if(shakeCam != null && applyCamShake)
    {
        Debug.Log("shaking");
        shakeCam.Play();
    }

}
```

플레이어 설정

- Apply Camera Shake 옵션을 체크



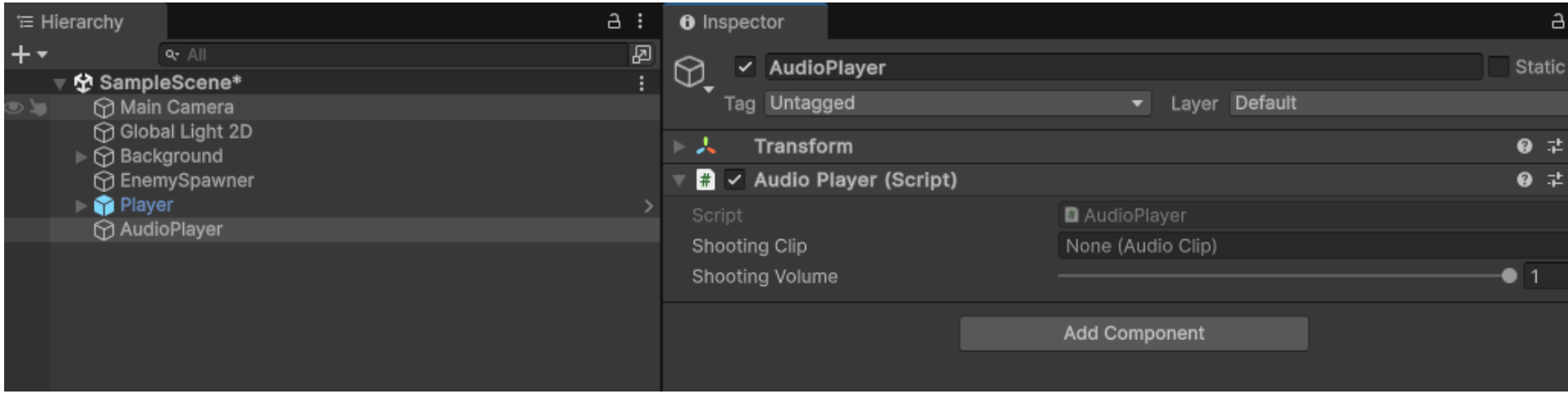
음향 효과 – 애셋 준비


Assets > AssetPack > Bonus
kenvector_f...
kenvector_f...
sfx_laser1
sfx_laser2
sfx_lose
sfx_shiel...
sfx_shiel...
sfx_twoT...
sfx_zap

AudioPlayer.cs

- Hierarchy에 AudioPlayer 생성
  - AudioPlayer.cs 연결

```cs
using UnityEngine;

public class AudioPlayer : MonoBehaviour
{
    [SerializeField] AudioClip shootingClip;
    [SerializeField] [Range(0f, 1f)] float shootingVolume = 1f;

    public void PlayShootingClip()
    {
        if (shootingClip != null)
        {
            AudioSource.PlayClipAtPoint(
                shootingClip,
                Camera.main.transform.position,
                shootingVolume
            );
        }
    }
}
```



Shooter.cs에서 음향 재생

```
public class Shooter : MonoBehaviour
{

    [SerializeField] GameObject projectilePrefab;
    [SerializeField] float projectileSpeed = 10f;
    [SerializeField] float projectileLifeTime = 5f;
    [SerializeField] float firingRate = 0.2f;
    [SerializeField] bool autoShoot = false;

    public bool isFiring = false;
    Coroutine firingRoutine;

    AudioPlayer audioPlayer;

    void Awake() {
        audioPlayer = FindAnyObjectByType<AudioPlayer>();
    }
```

Shooter.cs에서 음향 재생

```
IEnumerator FireContinuously()
{
    while(true)
    {
        if(audioPlayer != null) {
            audioPlayer.PlayShootingClip();
        }
        // firing action
        GameObject missile = Instantiate(projectilePrefab,
            transform.position, Quaternion.identity);

        Rigidbody2D rb = missile.GetComponent<Rigidbody2D>();
```

도전 과제 1

플레이어가 미사일을 맞아 데미지를 입을 때에 사용할 입자 시스템을 추가해 보라

도전 과제 2

적의 발사와 플레이어 발사가 서로 다른 사운드가 나오도록 해 보라

이제 게임 근처에 도달?