Object Pooling - це породжувальний патерн, який попередньо створює всі об'єкти, що знадобляться вам у певний момент, ще до початку гри. Це усуває необхідність створювати нові об'єкти або знищувати старі під час гри.

Цей патерн нестандартний, він здебільшого використовується в індустрії відеоігор. Тому що зазвичай саме там виникає проблема оптимізації при великої кількості об’єктів. По своїй структурі схожий на патерн Flyweight.

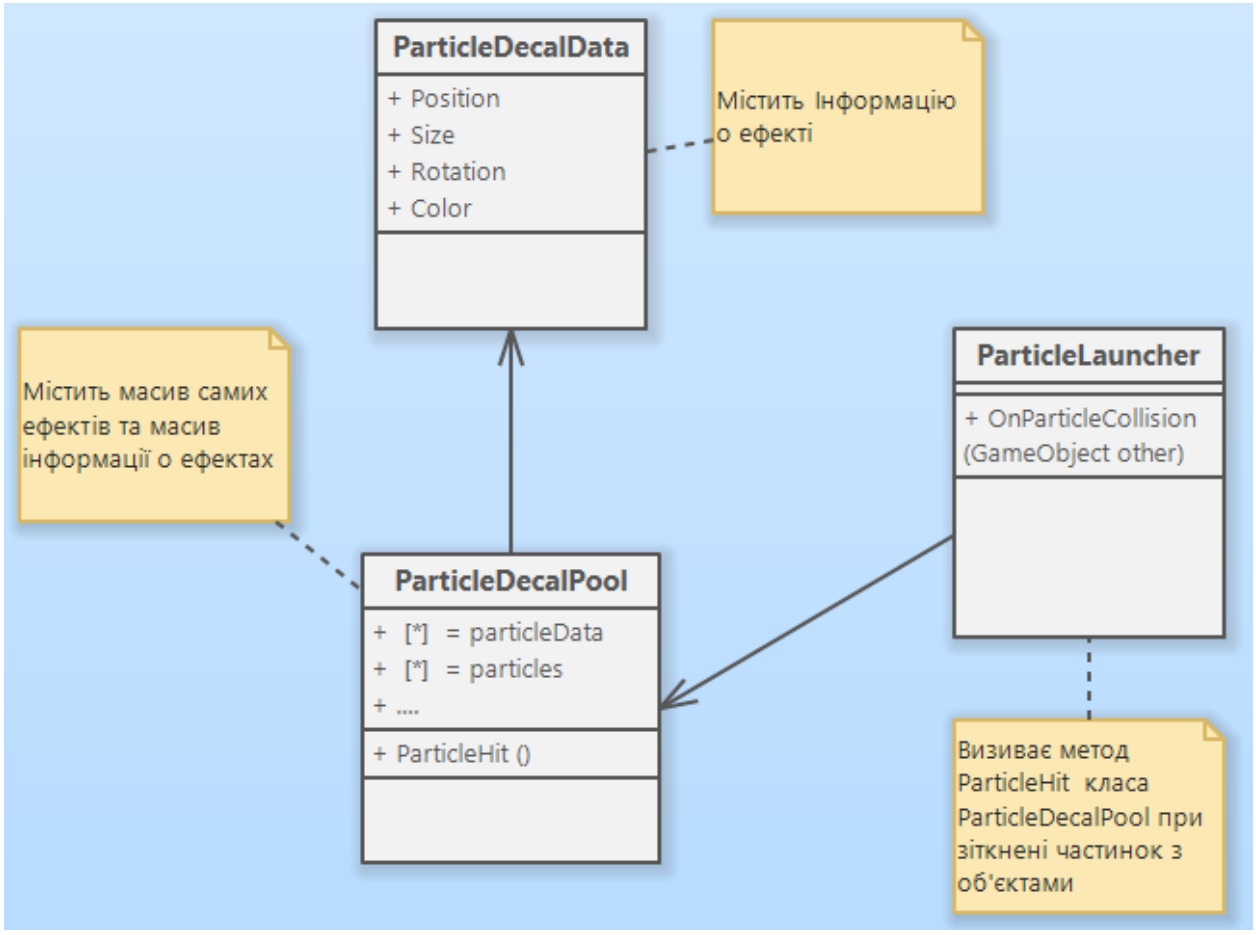
В моїй грі він також використовується багато де, але в основному для ефектів влучання. Тобто ці ефекти доволі такі “важкі” і при цьому їх потрібно багато, тому що кожне влучання по кожному об’єкту потребує свого ефекту. Тому звісно постійне створення та видалення таких об’єктів сильно впливає на продуктивність гри. Краще створити деяку купу таких об’єктів при старті гри та просто переміщувати їх в потрібну точку та в потрібний час.

Для цього в мене є три класа

- ParticleDecalData – клас, який містить деяку інформацію щодо ефектів.
- ParticleDecalPool – клас, який відповідає за створення об’єктів та переміщення їх з “басейну” на потрібне місце.
- ParticleLauncher – допоміжний клас, який визиває методи попереднього класу в потрібний час.

Клас ParticleDecalData містить інформацію щодо повороту, координат і т.д., наших ефектів.

Клас ParticleDecalPool містить масив ParticleDecalData та масив ефектів. Цей клас є основним в цій системі. На початку гри він створює певну кількість екземплярів ефектів, а потім відповідає за їх переміщення. Тобто в ньому є метод який переміщує вільні об’єкти на координати, які передаються параметром.



Клас ParticleDecalPool.

```
public class ParticleDecalPool : MonoBehaviour
{
    private int particleDecalIndex;//ітератор для прохождення по масивам
    public int maxDecals =100;//кількість об'єктів в басейні
    public float decalsMinSize = 100;//мінімальний розмір об'єкта
    public float decalsMaxSize =100;//максимальний розмір об'єкта
    public  ParticleDecalData[] particleData;//масив інформації, позиція, поворот і т.д.
    private ParticleSystem.Particle[] particles;//масив самих ефектів
    private ParticleSystem decalParticalSystem;//система, яка запускає ефект
```

Містить метод Start, який створює об’єкти (ефекти) при старті гри.

```
void Start()
{
    decalParticalSystem = GetComponent<ParticleSystem>();
    particles = new ParticleSystem.Particle[maxDecals];
    particleData = new ParticleDecalData[maxDecals];
    for (int i = 0; i < maxDecals; i++)
    {
        particleData[i] = new ParticleDecalData();
    }
}
```

А також метод ParticleHit, який приймає параметри координат (particleCollisionEvent) та градієнт для коліру.

```
2 references
public void ParticleHit(ParticleCollisionEvent particleCollisionEvent, Gradient colorGradient)
{
    SetParticleData(particleCollisionEvent, colorGradient);
    DisplayParticles();
}
```

Основний метод це SetParticleData. Він переміщую за потрібними координатами ефект та збільшує ітератор (за допомогою, якого вибирається ефект).

```
void SetParticleData(ParticleCollisionEvent particleCollisionEvent, Gradient colorGradient)
{
    if (particleDecalIndex >= maxDecals)
    {
        particleDecalIndex = 0;
    }
    particleData[particleDecalIndex].Position = particleCollisionEvent.intersection;
    Vector3 particleRotationEuler = Quaternion.LookRotation(particleCollisionEvent.normal).eulerAngles ;
       particleRotationEuler.z = Random.Range(0, 360);
    particleData[particleDecalIndex].Rotation = particleRotationEuler;
    particleData[particleDecalIndex].Size = Random.Range(decalsMinSize,decalsMaxSize);
    particleData[particleDecalIndex].Color = colorGradient.Evaluate(Random.Range(0f,1f));


    particleDecalIndex++;

}
```

Це простий патерн доволі сильно підвищує оптимізацію гри.

Також це не єдиний приклад використання цього патерну в моєму проекті. Аналогічно працює система, яка відповідає за дірки від куль (BulletHolesPool), замість постійного створення та видалення об’єктів, вона їх переміщує.

Скрипти можете знайти за шляхом:

Assets/Scripts/Controlling Particles/ParticleDecalData.cs

Assets/Scripts/Controlling Particles/ParticleDecalPool.cs

Assets/Scripts/Controlling Particles/ParticleLauncher.cs

Assets/Scripts/Weapon/ShootingWeapon/MainScripts/BulletHolesDataBase.cs