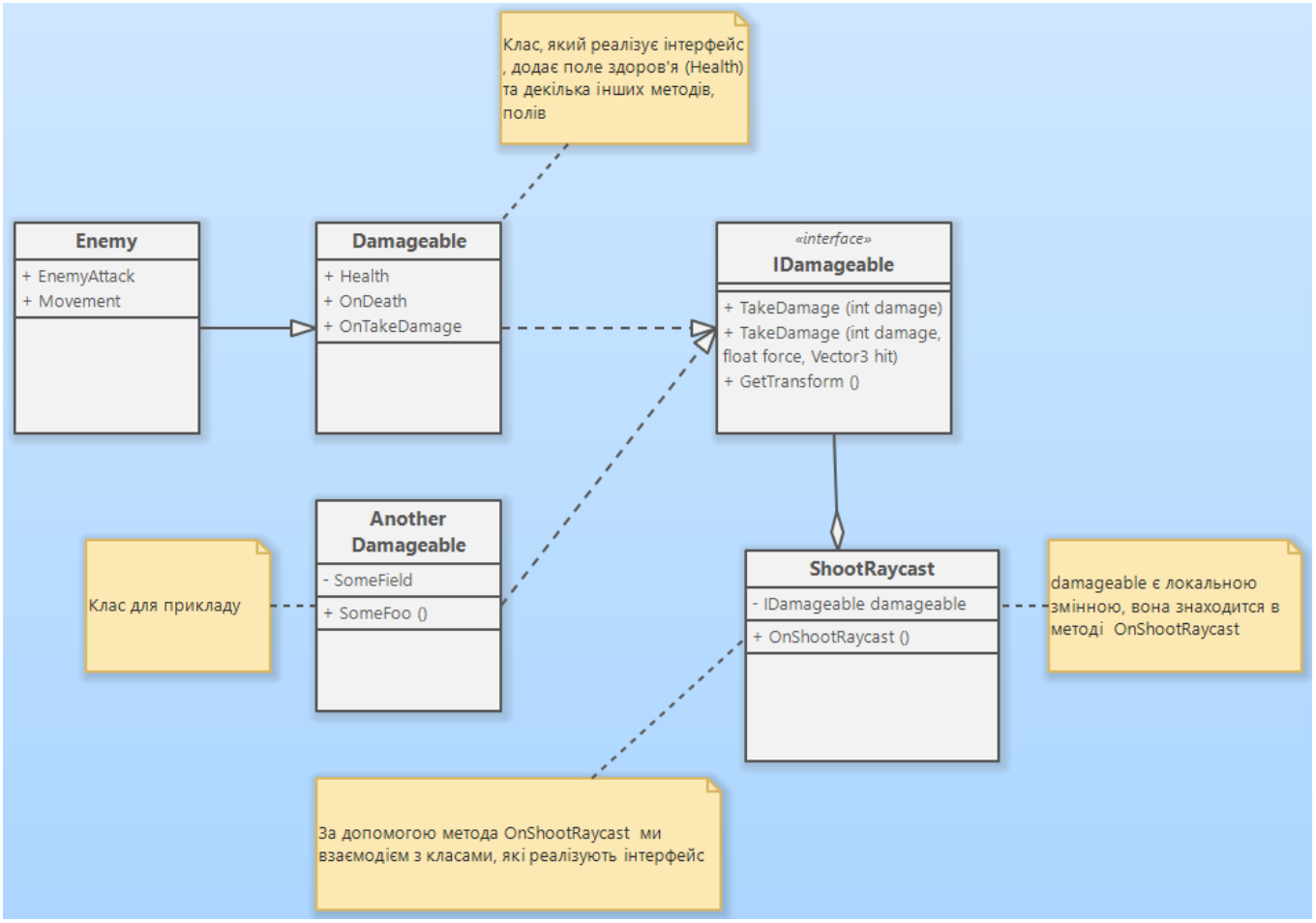
Міст - це структурний патерн проєктування, який розділяє один або кілька класів на дві окремі ієрархії - абстракцію та реалізацію, даючи змогу змінювати їх незалежно один від одного.

В моєму проекті мостом є інтерфейс IDamageable, Damageable та клас вазаємодії з ними ShootRaycast.

Клас Damageable реалізує інтерфейс IDamageable, а після чого всі об’єкти, яким потрібні, успадковують клас Damageable. Цей клас в мене містить поле зі здоров’ям, а також інші потрібні методи та поля. Тобто якщо мені буде потрібно зробити клас, який не буду мати поле здоров’я, але мати деякі методи, то я легко зможу це зробити реалізувавши ще раз інтерфейс.

Також клас ShootRaycast в мене відповідає за нанесення пошкоджень. Він наносить пошкодження всім класам, які реалізують інтерфейс IDamageable. Тобто це і є міст, паралельно я можу реалізувати, як хочу, клас ShootRaycast та інший клас Damageable. Але вони будуть зв’язані інтерфейсом IDamageable і я завжди зможу без проблем наносити пошкодження об’єкту.



Метод OnShootRaycast в класі ShootRaycast, виглядає так:

```
public virtual void OnShootRaycast(GunData gunData)
{

    Vector3 forwardVector = Vector3.forward;
    float deviation = UnityEngine.Random.Range(0f, maxDeviation);
    float angle = UnityEngine.Random.Range(0f, 360f);
    forwardVector = Quaternion.AngleAxis(deviation, Vector3.up) * forwardVector;
    forwardVector = Quaternion.AngleAxis(angle, Vector3.forward) * forwardVector;
    forwardVector = CameraOrigin.transform.rotation * forwardVector;

    if (Physics.Raycast(CameraOrigin.transform.position, forwardVector, out RaycastHit hitInfo, gunData.maxDistance, ~WhatIsRayCastIgnore))
    {
        //беремо з об'єкта по якому попали компонент IDamageable, та визиваємо у нього метод TakeDamage
        IDamageable damageable = hitInfo.transform.GetComponent<IDamageable>();

        if ((WhatIsEnemy | (1<<hitInfo.collider.gameObject.layer)) == WhatIsEnemy)
        {

            damageable?.TakeDamage(gunData.damage, gunData.force, hitInfo.point);

        }
        bulletHolesDataBase.MakeBulletHoleByLayer(hitInfo);
    }
}
```

Змінна damageable взаємодіє с будь-яким класом, який успадковує інтерфейс IDamageable.

Цей патерн зручний в таких ситуаціях та дозволяє уникнути “спагеті-коду”, що є дуже важливим аспектом при створенні великих проектів.

Скрипти можете знайти за шляхом:

Assets/Scripts/Enemies/Base/Enemy.cs

Assets/Scripts/CommonCore/Damageable/Damageable.cs

Assets/Scripts/CommonCore/Damageable/IDamageable.cs

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