Терещук С. О., ст. викладач,
Дзюбанчук Б. С., здобувач
Державний університет «Житомирська політехніка»

OBJECT POOL ЯК ЗАСІБ ДЛЯ ОПТИМІЗАЦІЇ ПРОЄКТУ

Під час розробки ігор часто зустрічаються моменти, коли нам потрібно створити велику кількість різного типу об’єктів. Все б нічого, але метод, що ми зазвичай використовуємо під час цього, а саме Instantiate, охоплює доволі важкі процеси, що часто може значно плинути на оптимізацію.

Задля вирішення даної проблеми було розроблено патерн програмування під назвою Пул об’єктів. Він є ефективним способом для оптимізації ресурсів та підвищення продуктивності програмного забезпечення. Це паттерн створення програмного забезпечення використовує набір ініціалізованих об’єктів, готових до використання (пул), замість їх виділення та знищення за запитом.

Паттерн об’єктного пула створює набір об’єктів, які можуть бути повторно використовувані. Коли потрібен новий об’єкт, його запитують з пула. Якщо попередньо підготовлений об’єкт доступний, він повертається негайно, уникнувши витрат на створення. Якщо в пулу немає об’єктів, створюється і повертається новий. Після використання об’єкта і коли він більше не потрібний, він повертається в пул, що дозволяє його використовувати знову у майбутньому без повторного витратного процесу створення. Важливо враховувати, що після використання і повернення об’єкта існуючі посилання стають недійсними.

У деяких об’єктних пулах ресурси обмежені, тому вказується максимальна кількість об’єктів. Якщо ця кількість досягнута і запитується новий об’єкт, може бути згенеровано виняток або потік буде заблокований до повернення об’єкта до пула.

Одною з основних переваг даного патерну є отримання об’єкта з пула, що відбувається за передбачуваний час, коли створення нових об’єктів (особливо через мережу) може займати чимало часу.

Для його реалізації потрібно мати узагальнюючий клас, що матиме в собі потрібні методи, а саме:

- Ініціалізація пулу;
- Взяття об’єкта з пулу: При потребі у використанні об’єкта, візьміть його із пулу, якщо немає доступного – створіть новий та поверніть, якщо не існують обмежень у кількості об’єктів.
- Повернення об’єкта в пул: Після закінчення використання об’єкта поверніть його назад у пул, де він буде готовий до наступного використання.

```
Frequently called  1 usage
private void Attack()
{
    if(Time.time < _nextAttackTime)
        return;

    _nextAttackTime = Time.time + 1.0f / _attackRate;

    Missle missle = _misslePool.Get();

    float flyTime = (transform.position - _target.position).magnitude / _missleSpeed;

    missle.Launch(_launchPoint.position, _target, flyTime, OnReachTarget);

    void OnReachTarget() => _misslePool.Return(missle);
}
```

Рис. 1. Лістинг використання пулу у проекті

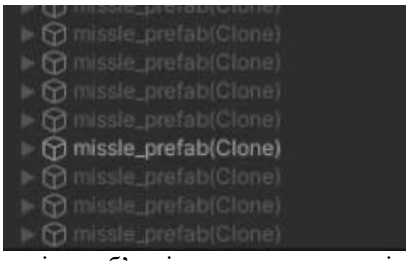

Рис. 2. Поведінка об’єктів при використанні пулу у проєкті

Приклад використання наведений на рисунку 1 та 2, де *_misslePool* є завчасно ініціалізований пул з відповідних об’єктів. При атаці замість звичайного створення об’єкту ми отримуємо уже існуючий, звертаючись до пулу. При дотику до цілі, викликається метод, що замість звичайного видалення об’єкту з сцени, повертає його у пул.

Отже, паттерн Object Pool є сильним інструментом для покращення продуктивності програмного продукту, адже він дозволяє отримувати доступ до готових до використання об’єктів, замість створення нового екземпляру, уникнувши витрат на їхнє створення.

Список використаних джерел

1. Object pool pattern [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Object_pool_pattern