УДК 001.894

ПУЛ ОБЪЕКТОВ В UNITY

ХАРХАСОВ ВЛАДИМИР ВИКТОРОВИЧ
студент
Гомельский государственный технический университет им. П.О. Сухого

Научный руководитель: Титова Людмила Константиновна
старший преподаватель
Гомельский государственный технический университет им. П.О. Сухого

Аннотация: в статье обсуждается подход для оптимизации проектов в Unity и способ разгрузить вычислительную мощность центрального процессора.
Ключевые слова: ЦП, пул объектов, шаблон проектирования, объект, код.

Объединение объектов – отличный способ оптимизировать проект и снизить нагрузку на ЦП при необходимости быстрого создания и уничтожения игровых объектов. Это хорошая практика шаблонов проектирования, о которых следует помнить, чтобы помочь разгрузить вычислительную мощность ЦП для выполнения более важных задач и не перегружаться повторяющимися вызовами создания и уничтожения. Это особенно полезно при работе с пулями в шутере с видом сверху.

Объединение объектов – это шаблон проектирования, который заранее создает экземпляры всех объектов, которые вам понадобятся в любой конкретный момент перед игрой. Это устраняет необходимость создавать новые объекты или уничтожать старые во время игры. Пулы объектов в основном используются для повышения производительности: в некоторых случаях пулы объектов значительно повышают производительность, когда проект многократно создает и уничтожает один и тот же игровой объект в быстрой последовательности. Он работает, создавая определенное количество игровых объектов перед запуском игры, и просто деактивирует или активирует необходимые игровые объекты, фактически просто перерабатывая игровой объект и никогда не уничтожая его.

Объединение объектов – важная концепция для понимания некоторых игровых объектов и того, как часто они будут создаваться или уничтожаться во время игры. При обработке множества вызовов инициализации и уничтожения одного игрового объекта. Следующий пример Space Shooter – идеальный кандидат для реализации объединения объектов (рис. 1), чтобы помочь оптимизировать время выполнения проекта.

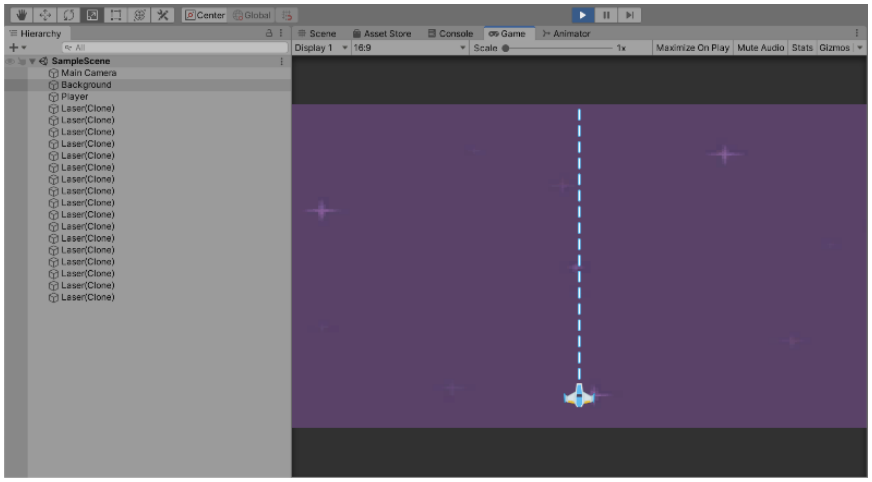

Рис. 1. Пример создания большого количества однотипных объектов

Когда космический корабль стреляет, создается несколько пуль, которые летят вверх по экрану, а затем уничтожаются, при столкновении с другим объектом или когда покидают экран. Это создает значительную нагрузку на ЦП, и, следовательно, пул объектов был бы отличным кандидатом для оптимизации в этой ситуации, поскольку в этом случае нет необходимости постоянно создавать и уничтожать объекты во время выполнения.

Создание сценария для реализации пула объектов:

1. необходимо создать новый скрипт и назвать “ObjectPool”;
2. необходимо прикрепить скрипт к игровому контроллеру;
3. необходимо открыть созданный скрипт и написать в определении класса (рис. 2).

```
public static ObjectPool SharedInstance;
public List<GameObject> pooledObjects;
public GameObject objectToPool;
public int amountToPool;

void Awake()
{
    SharedInstance = this;
}

void Start()
{
    pooledObjects = new List<GameObject>();
    GameObject tmp;
    for(int i = 0; i < amountToPool; i++)
    {
        tmp = Instantiate(objectToPool);
        tmp.SetActive(false);
        pooledObjects.Add(tmp);
    }
}
```

Рис. 2. Скрипт для пула объектов

Эта простая настройка позволяет указать GameObject для пула и число для предварительного создания экземпляра. Цикл For будет создавать экземпляр objectToPool указанное количество раз в amountToPool. Затем GameObjects переводятся в неактивное состояние перед добавлением их в список pooledObjects.

Также необходимо выбрать игровой контроллер, содержащий только что созданный сценарий. У него будет объект для пула и количество объектов для пула, где можно установить оба параметра соответственно. Перетаскивание префаба пули в Object to Pool сообщит скрипту, из какого объекта нужно чтобы пул состоял.

Необходимо установить для AmountToPool относительно большое число (рис. 3), например, 20. Причина этого в том, что нужно убедиться, что будет создано достаточно игровых объектов для работы.

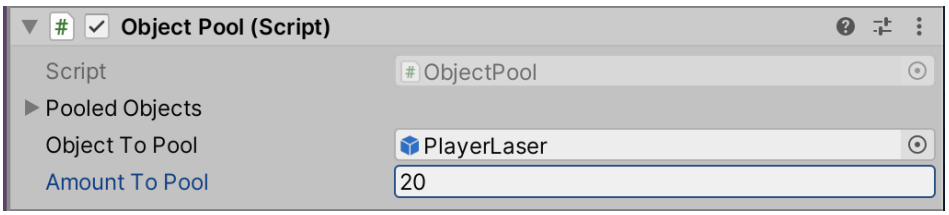

Рис. 3. Настройка сценария пула объектов

Теперь скрипт всегда будет создавать 20 PlayerBullets еще до того, как игра запустится. Таким образом, всегда будет набор предварительно созданных пуль для использования, но, чтобы полноценно воспользоваться этим, нужно сделать еще как минимум две вещи.

Необходимо дополнить созданный пул объектов (рис. 4), чтобы можно было создать новую функцию для вызова из других сценариев для использования пула объектов. Это поможет использовать идею отсутствия необходимости избыточного создания и уничтожения объектов во время выполнения. Это также позволит другим сценариям активировать объекты, в котором нужно придерживаемся дизайна пула объектов.

```
public GameObject GetPooledObject()
{
    for(int i = 0; i < amountToPool; i++)
    {
        if(!pooledObjects[i].activeInHierarchy)
        {
            return pooledObjects[i];
        }
    }
    return null;
}
```

Рис. 4. Метод для возможности вызова пула из вне

Также необходимо дополнить скрипт (рис. 5), который создает экземпляр пуль. Необходимо заменить код, который создает экземпляры пуль, например, Instantiate(playerBullet, turret.transform.position, turret.transform.rotation). Нужно использовать следующий код для замены вызовов Instantiate.

```
GameObject bullet = ObjectPool.SharedInstance.GetPooledObject();
  if (bullet != null) {
    bullet.transform.position = turret.transform.position;
    bullet.transform.rotation = turret.transform.rotation;
    bullet.SetActive(true);
  }
```

Рис.5. Код для создания объектов через пул

Последнее изменения, которое необходимо сделать, заменить метод по уничтожению пуль, и вместо этого деактивировать объект через пул объектов (рис. 6).

```
Destroy(gameObject);  →  gameobject.SetActive(false);
```

Рис. 6. Код для деактивации объекта через пул