#NVJOB Simple Pool

https://nvjob.github.io/unity/nvjob-simple-pool

This is a simple pool for optimizing object loading.

All objects in the pool are loaded during initialization, and then retrieved from the pool and returned back to the pool without sacrificing performance.

The pool allows you to completely abandon Instantiate and Destroy after initialization.

Prerequisites

To work on the project, you will need a Unity version of at least 2019.1.8 (64-bit).

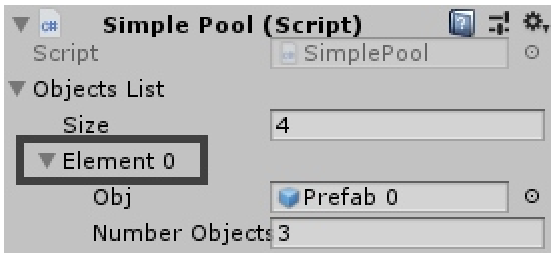
Information

Getting an object from a pool:

```
GameObject obj = SimplePool.GiveObj(0);
```

SimplePool.GiveObj() instead of using Instantiate(Object).

SimplePool.GiveObj(numElement)-> numElement - number of the item in editor (SimplePool).



Simple Pool (Script)

Script	SimplePool
Objects List	
Size	4
Element 0	
Obj	Prefab 0
Number Objects	3

After all the transformations of the object, activate it:

```
obj.SetActive(true);
```

Return object to the pool, remove from the scene:

```
SimplePool.Takeobj(obj);
```

SimplePool.Takeobj() instead of using - Destroy(Object).

SimplePool.Takeobj(GameObject)-> GameObject - is an object that to be returned to the pool.

The number of elements in the editor (SimplePool):

```
SimplePool.numObjectsList
```

Checking that the pool is not empty:

```
GameObject obj = SimplePool.GiveObj(0);
if (obj != null)
{
}
```

If the pool is empty it will return null.

Example script:

```
using UnityEngine;

public class Example : MonoBehaviour
{
    GameObject obj;

    void Start()
    {
        if(SimplePool.numObjectsList == 0) obj = SimplePool.GiveObj(0);
        else obj = SimplePool.GiveObj(Random.Range(0, SimplePool.numObjectsList));
        if (obj != null)
        {
            obj.transform.SetPositionAndRotation(transform.position, transform.rotation);
            obj.transform.parent = transform;
            obj.SetActive(true);
            Invoke("DestroyObject", 5);
        }
    }

    void DestroyObject()
    {
        SimplePool.Takeobj(obj);
    }
}
```