**尽量避免使用foreach**

Posted on 2013年04月17日 by U3d / [Unity3D 基础教程](http://www.unitymanual.com/category/manual/unity3d-%e5%9f%ba%e7%a1%80%e6%95%99%e7%a8%8b)/被围观 208 次

为什么要尽量避免使用foreach。GC Alloc是造成Unity游戏偶尔卡帧的罪魁祸首。第一个循环大概会造成2.3K的GC Alloc。第二个循环则没有任何GC Alloc。所以，不要途方便，到处哪都用foreach循环。

foreach 语句为数组或对象集合中的每个元素重复一个嵌入语句组。foreach 语句用于循环访问集合以获取所需信息，但不应用于更改集合内容以避免产生不可预知的副作用。此语句的形式如下：

foreach (type identifier in expression) statement

其中：

type

identifier 的类型。

identifier

表示集合元素的迭代变量。如果迭代变量为值类型，则无法修改的只读变量也是有效的。

expression

对象集合或数组表达式。集合元素的类型必须可以转换为 identifier 类型。请不要使用计算为 null 的表达式。

而应计算为实现 IEnumerable 的类型或声明 GetEnumerator 方法的类型。在后一种情况中，GetEnumerator 应该要么返回实现 IEnumerator 的类型，要么声明 IEnumerator 中定义的所有方法。

statement

要执行的嵌入语句。

举一个代码列子：

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| 01 | **using** UnityEngine; |
| 02 | **using** System.Collections; |
| 03 |  |
| 04 | **public** **class** ForeachTest : MonoBehaviour { |
| 05 |  |
| 06 | **protected** ArrayList m\_array; |
| 07 |  |
| 08 | **void** Start () |
| 09 | { |
| 10 | m\_array = new ArrayList(); |
| 11 | **for** (**int** i = 0; i < 2; i++) |
| 12 | m\_array.**Add**(i); |
| 13 | } |
| 14 |  |
| 15 | **void** Update () |
| 16 | { |
| 17 | **for** (**int** i = 0; i < 1000; i++) |
| 18 | { |
| 19 | **foreach** (**int** e **in** m\_array) |
| 20 | { |
| 21 | *//big gc alloc!!! do not use this code!* |
| 22 | } |
| 23 | } |
| 24 |  |
| 25 | **for** (**int** i = 0; i < 1000; i++) |
| 26 | { |
| 27 | **for** (**int** k = 0; k < m\_array.Count; k++) |
| 28 | { |
| 29 | *//no gc alloc!!* |
| 30 | } |
| 31 | } |
| 32 | } |
| 33 | } |