【Unity基础教程】重点知识汇总

（二十）

Unity C#常用算法入门之冒泡排序

冒泡排序算法（概念）

概念：冒泡排序算法（Bubble Sort）是一种简单的**排序算法**，它通过**多次遍历**待排序的元素，比较相邻的元素，并根据大小顺序**交换**它们的位置。每经过一轮遍历，**未排序部分中最大的元素会被“冒泡”到序列的末端。**因此，冒泡排序的基本思路就**像水中的气泡向上冒一样**，最大的元素会逐渐“冒泡”到正确的位置。**其核心思想就是每一轮遍历都将最大的元素移到末尾，接着再进行下一轮比较。重复这个过程，直到整个数组（或列表）排好序。**

具体步骤：

① 比较相邻的两个元素，如果第一个元素比第二个大，则交换它们的位置。

② 对每一对相邻元素进行此操作，从数组的起始位置一直遍历到结尾。

③ 每次遍历后，最大的元素被移动到数组的末尾（因此排好序的元素不再参与下一轮比较）。

④ 重复步骤1-3，直到整个数组排序完成。

冒泡排序算法（具体实现）

```
void BubbleSortArray(int[] arr)
    {
        int n = arr.Length;
        bool swapped;

        // 外层循环控制排序的轮数
        for (int i = 0; i < n - 1; i++)
        {
            swapped = false;  // 每一轮开始时，假设没有发生交换

            // 内层循环进行相邻元素的比较和交换
            for (int j = 0; j < n - 1 - i; j++)
            {
                // 如果当前元素比下一个元素大，就交换它们
                if (arr[j] > arr[j + 1])
                {
                    // 交换元素
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;  // 发生了交换
                }
            }

            // 如果一轮比较没有发生交换，说明数组已经有序，提前结束
            if (!swapped)
            {
                break;
            }
        }
    }
```

语句解释：

- 外层循环（for (int i = 0; i < n - 1; i++)）

控制排序的轮数。由于每次遍历后，最大的元素已经被排到末尾，所以每完成一轮排序，就**不需要再检查末尾的元素**。因此，i的循环次数从0到n-1。

- 内层循环（for (int j = 0; j < n - 1 - i; j++)）

用于比较相邻的元素并进行交换。每轮排序结束后，已经排序的元素被放到了末尾，因此在内层循环中，我们**不再比较这些已排序的元素**（n - 1 - i部分）。

- 交换操作（int temp = arr[j]; arr[j] = arr[j + 1]; arr[j + 1] = temp;）

交换是通过一个**临时变量**temp来完成的，temp存储 arr[j]，然后把 arr[j + 1] 赋值给 arr[j]，最后将temp赋值给 arr[j + 1]。

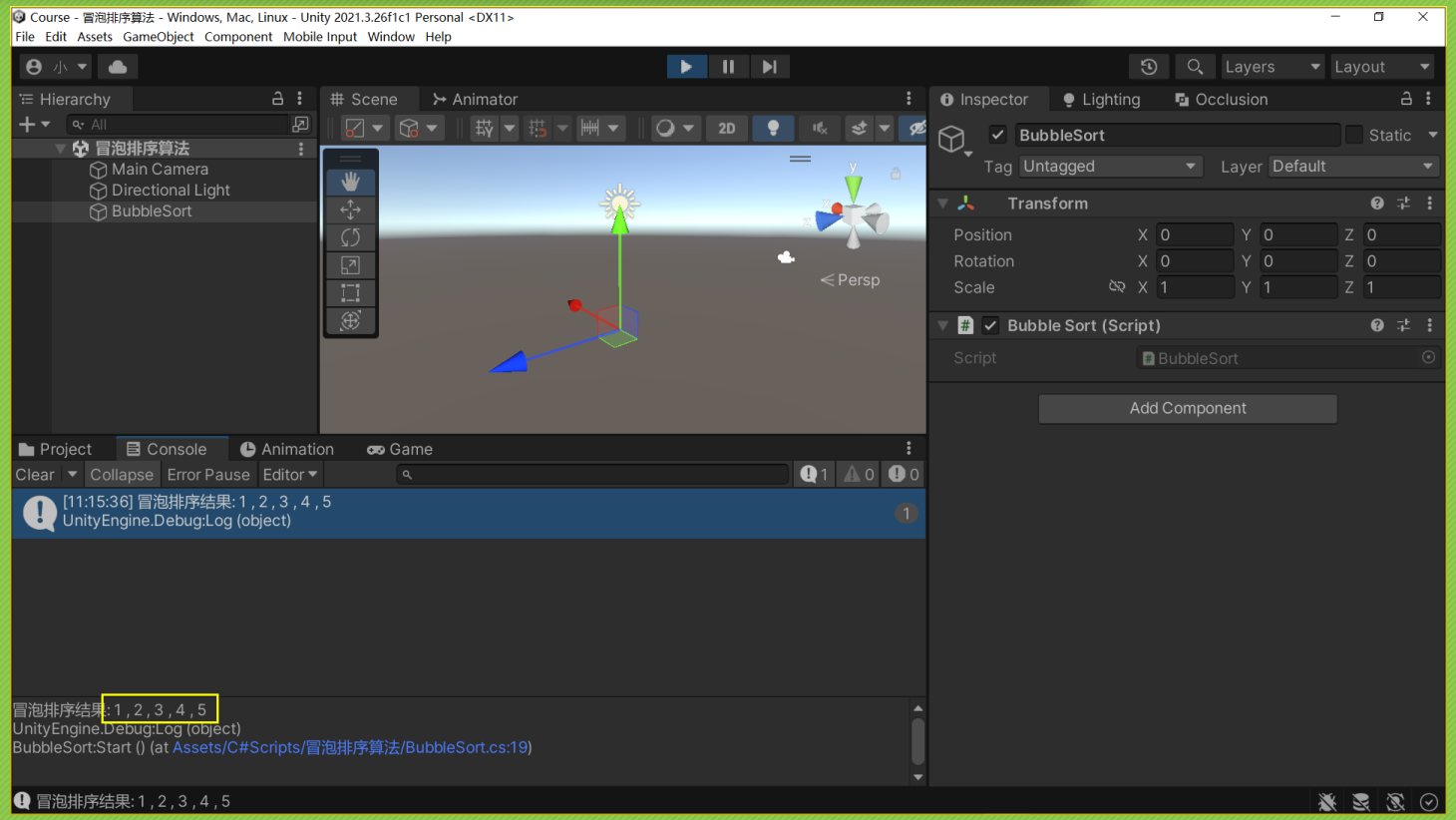
冒泡排序算法（动画演示）

Num	5	2	1	4	3
第一轮	5	2	1	4	3
第二轮	2	5	1	4	3
第三轮	2	1	5	4	3
第四轮	2	1	4	5	3
第五轮	2	1	4	3	5
第六轮	1	2	4	3	5
第七轮	1	2	4	3	5

冒泡排序算法（运行结果）

```
private int[] num = { 5, 2, 1, 4, 3 };

void Start()
{
    BubbleSortArray(num);
    // 使用 string.Join() 输出排序后的数组
    Debug.Log($"冒泡排序结果: {string.Join(" , ", num)}");
}
```



*如果你想进行**降序排列**（即5,4,3,2,1），只需要在比较相邻元素时**改变比较的方向**，也就是说交换条件需要从arr[j] > arr[j + 1]改成arr[j] < arr[j + 1]。这样较大的元素会被“冒泡”到数组的前面。

冒泡排序算法（总结）

冒泡排序的优化：虽然冒泡排序本身是简单的，但是它的**效率较低**。通过优化可以减少不必要的比较。例如，如果在某一轮遍历中没有进行交换，说明数组已经是有序的，这时可以**提前结束排序**，直接进入下一轮。

冒泡排序的时间复杂度：

- 最坏时间复杂度：$O(n^2)$，当数组是逆序时，每次都需要交换。
- 最好时间复杂度：$O(n)$，如果数组已经有序，则只需要一次遍历（优化版本）。
- 平均时间复杂度：$O(n^2)$，通常情况下，冒泡排序的比较次数是平方级别的。

冒泡排序的空间复杂度：

冒泡排序是**原地**排序算法，它只需要**常数级**的额外空间，因此空间复杂度是$O(1)$。

【Unity基础教程】重点知识汇总

（二十）

Unity C#常用算法入门之冒泡排序