Fixing Performance Problems

1，Optimizing scripts in Unity games

Writing efficient code

Calculating a square root takes a CPU more time to execute than multiplying two numbers.

Move code out of loops when possible

Consider whether code must run every frame

Only run code when things change

Use caching

Use the right data structure

Minimize the impact of garbage collection

Use object pooling

Avoiding expensive calls to the Unity API

SendMessage() and BroadcastMessage()

Find()

Transform：

If our code makes frequent use of Transform.position and we can use Transform.localPosition in its place.

Update()：

Empty Update() calls can be particularly wasteful.

Vector2 and Vector3

Float calculations instead of Vector2 and Vector3

If our game makes extensive and very frequent use of magnitude or Distance, it may be possible for us to avoid the relatively expensive square root calculation by using Vector2.sqrMagnitude and Vector3.sqrMagnitude instead.

Camera.main

2，Optimizing garbage collection in Unity games

Unity has access to two pools of memory: the stack and the heap.

The stack is used for short term storage of small pieces of data, and the heap is used for longer term storage and larger pieces of data.

Three things can cause the garbage collector to run:

The garbage collector runs whenever a heap allocation is requested that cannot be fulfilled using free memory from the heap.

The garbage collector runs automatically from time to time (although the frequency varies by platform).

The garbage collector can be forced to run manually.

In Unity, value-typed local variables are allocated on the stack and everything else is allocated on the heap.

GC alloc. This column shows heap allocations that are being made in that frame. .

Reducing the amount of garbage created

Caching

Don’t allocate in functions that are called frequently

Clearing collections

Creating new collections causes allocations on the heap. If we find that we’re creating new collections more than once in our code, we should cache the reference to the collection and use Clear() to empty its contents instead of calling new repeatedly.

Object pooling

Common causes of unnecessary heap allocations

Strings

Removing calls to Debug.Log()

Unity function calls

Every time we access a Unity function that returns an array, a new array is created and passed to us as the return value. For example, Mesh.normals

Use GameObject.CompareTag(), no longer generates any garbage

Input.GetTouch() and Input.touchCount in place of Input.touches

Boxing

Boxing usually occurs when we pass value-typed variables, such as ints or floats, to a function with object parameters such as Object.Equals().

Coroutines

Calling StartCoroutine() creates a small amount of garbage, because of the classes that Unity must create instances of to manage the coroutine.

使用yield return null;代替yield return 0;

WaitForSeconds delay = new WaitForSeconds(1f); yield return delay;

代替yield return new WaitForSeconds(1f);

Foreach loops

Function references

LINQ and Regular Expressions

Manually forcing garbage collection

System.GC.Collect();