# Emma! V1.1 Optimization Technical Review

This report compares the project’s performance before optimization with the performance after optimization and explains the optimization implementation process.

## Debug.Log

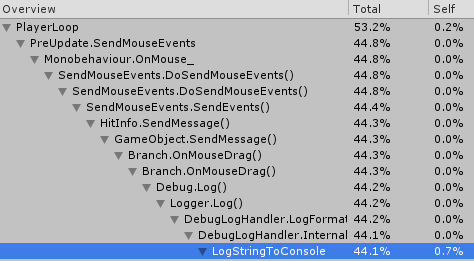


Figure 1 Logging to console cpu usage

There are several Debug.Log calls under Update for temporary test use. These calls are resource-consuming since they are being called every single frame. They are removed to increase in-editor performance.

Performance after optimization was not necessary to record since Debug.Logs are completely removed.

## Memory Usage

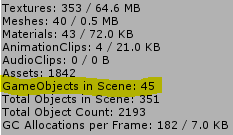


Figure 2 Gameobjects count increase abnormally

According to the memory usage analysis, the number of gameobjects in the scene gradually increases. Every 2 times the scene switches (every 2 usage peak), the gameobject increases by 1. According to this, there is one gameobject that was not destroyed with the scene on scene switching.

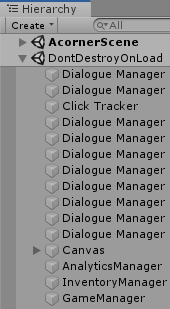


Figure 3 Multiple non-singleton managers

Multiple instances of dialogue manager are created, which is a memory leak caused by logic problems. This gradually slows down the running speed. It must be optimized. The optimization method is to make the dialogue manager into a singleton.

After implementing the method above, the ***Gameobjects in Scene*** maintains at a reasonable count. The memory leaking problem is solved. Performance after optimization was not necessary to record since this issue only causes significant performance drop after a long-time use of the program.

## Polygon Colliders



Figure 4 Unoptimized collider on prefab

The polygon collider auto-generated by Unity is detailed but unoptimized, this can cause significant performance drop during physics simulation.

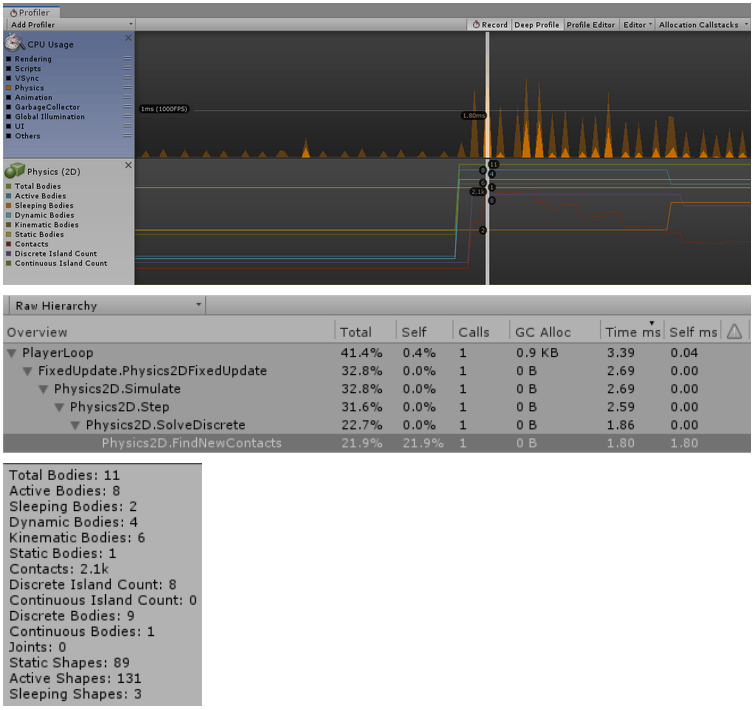


Figure 5 Acorn(dynamic rigidbody) polygon collider without optimization performance

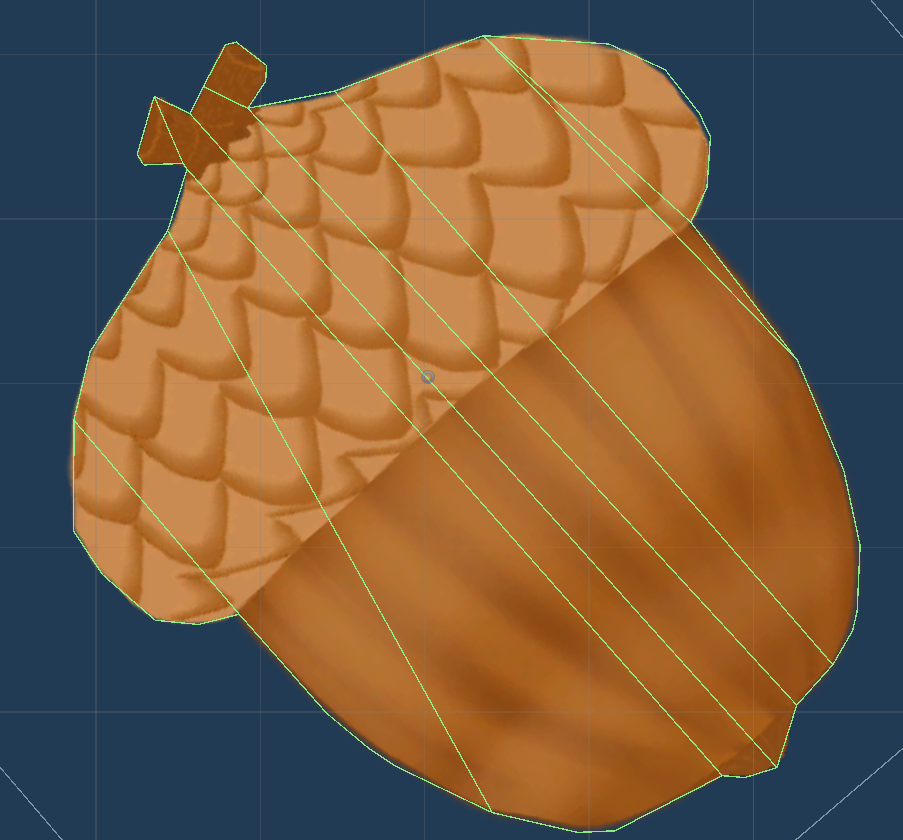


Figure 6 Optimized collider on prefab

Colliders on gameobjects(especially non-static ones) are optimized by manually reducing the vertexes.

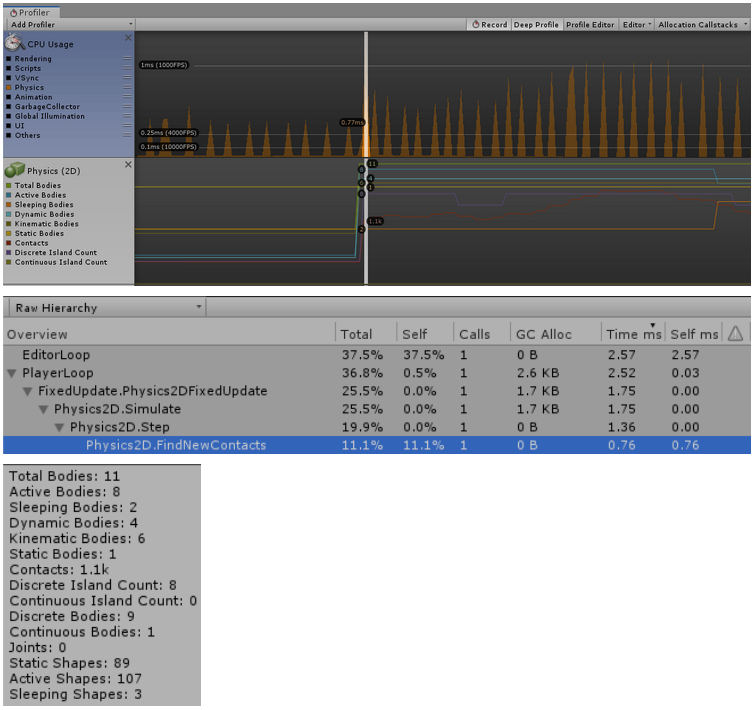


Figure 7 Acorn(dynamic rigidbody) polygon collider after optimization performance

Contacts before optimization: 2.1k After: 1.1k

Active Shapes before optimization: 131 After: 107

According to the graphs above, the optimization causes a significant performance boost.