OVERVIEW OF .NET IN UNITY

SCRIPTING BACKENDS

Mono uses just-in-time (JIT) compilation and compiles code on demand at runtime.

IL2CPP uses ahead-of-time (AOT) compilation and compiles your entire application before it is run.

MANAGED CODE STRIPPING

When you build your application, Unity scans the compiled assemblies (.DLLs) to detect and remove unused code.

This process reduces the final binary size of your build, but increases build time.

GARBAGE COLLECTION (GC)

Garbage collector only runs for a limited period of time and does not necessarily collect all objects in one pass. This spreads the time it takes to collect objects over a number of frames and reduces the amount of stuttering and CPU spikes.

.NET SYSTEM LIBRARIES

Unity tries its best to support as much of the .NET ecosystem as possible, there are some exceptions to parts of the .NET system libraries that Unity explicitly does not support. You should use the .NET Standard 2.0 API Compatibility Level for all new projects

THIRD-PARTY .NET LIBRARIES

You should only use third-party .NET libraries that have been extensively tested on a wide range of Unity configurations and platforms.

You should profile the usage of your .NET system libraries on all target platforms.

C# REFLECTION OVERHEAD

GC continuously scans the cached C# reflection objects, which causes unnecessary overhead.

To minimize the overhead, avoid methods such as Assembly.GetTypes and Type.GetMethods()

UNITYENGINE.OBJECT SPECIAL BEHAVIOR

UnityEngine.Object is a special type of C# object in Unity, because it is linked to a native C++ counterpart object.

MonoBehaviour/ScriptableObject override the equality (==) and inequality (!=) operators.

AVOID USING ASYNC AND AWAIT

The Unity API is not thread safe and therefore, you should not use async and await tasks

Async tasks often allocate objects when invoked, which might cause performance issues if you overuse them

LEARN MORE AT

HTTP://BIT.LY/OVERVIEW-OF-DONET-IN-UNITY