How FixedUpdate Works

Many Unity programmers misunderstand FixedUpdate because it seems so basic. *Usually* it behaves in the simplistic way it’s thought to work, but there are important exceptions that are often forgotten. Today we’ll take a closer look at FixedUpdate to get a better handle on how it works and learn to use it correctly.

Let’s start simple with Update. This “message” function of our MonoBehaviour classes is called every frame, no matter what. Consider this code:

**class** MyScript : MonoBehaviour

{

**void** Update()

{

Debug.Log("Update called on frame " + Time.frameCount);

}

}

Running the above code will produce an output like this:

Update called on frame 1

Update called on frame 2

Update called on frame 3

Update called on frame 4

Update called on frame 5

This is true regardless of how long each frame takes. A frame could take one millisecond or one hour and Update would still be called once for that frame, the frame before, and the frame after.

Now let’s look at FixedUpdate:

**class** MyScript : MonoBehaviour

{

**void** FixedUpdate()

{

Debug.Log("FixedUpdate called on frame " + Time.frameCount);

}

}

Running this version of the code *might* produce a log like this:

FixedUpdate called on frame 1

FixedUpdate called on frame 3

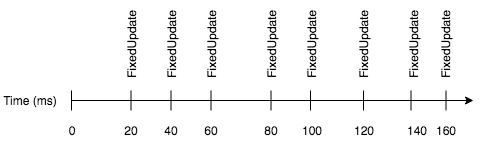
FixedUpdate called on frame 3

FixedUpdate called on frame 3

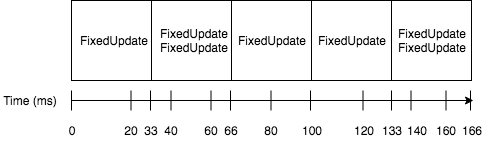
FixedUpdate called on frame 5

Without knowing more about the game, there’s no way to say for sure what the output would look like. Why? Because FixedUpdate is much more complicated.

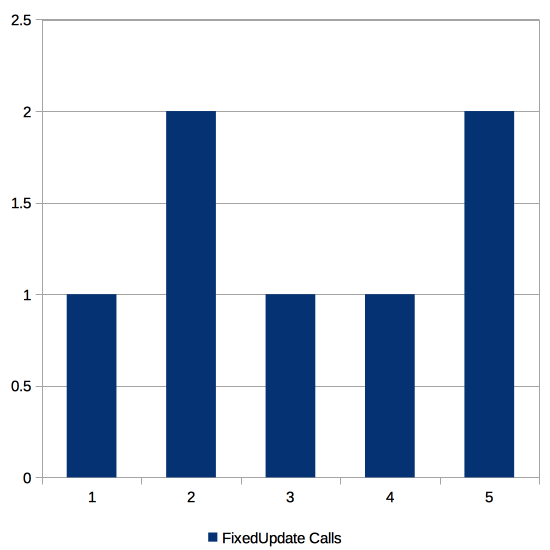
Let’s look at a diagram showing a super simple explanation for when FixedUpdate would be called if we set it to 50 Hz in Edit > Project Settings > Time:



In this diagram we see that FixedUpdate is called every 20 milliseconds, which lines up with 50 Hz. This explanation completely leaves out the concept of frames. When we add in frames, we immediately see that the even spacing can’t be maintained. Let’s look at the “normal” understanding of how a 50 Hz FixedUpdate rate works with a 30 FPS game:

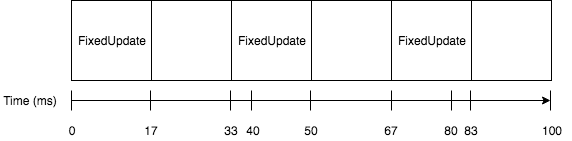


Notice how sometimes there are two FixedUpdate calls in a single frame and sometimes only one. The following graph shows how many calls occur per frame:



So even with this “normal” situation, the number of FixedUpdate calls per second is only *averaging* out to the target 50 Hz. At the individual frame level, FixedUpdate is being called at either 30 Hz or 60 Hz.

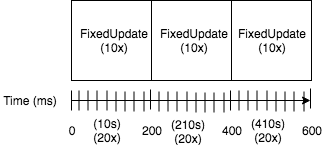
Now let’s look at what happens if the game were to run at 60 FPS but set the FixedUpdate rate to 25 Hz (40 milliseconds):



Now we have the opposite issue as above. FixedUpdate is supposed to be called less frequently than the FPS, meaning that not every frame will have a call to FixedUpdate. As above, this averages out to the desired 24 Hz, but the calls are happening on ≅16.66 millisecond intervals rather than 40 millisecond intervals.

It’s important to note that in both of these situations we are not having our FixedUpdate function called on a “tick-tock” basis. That is, we’re not necessarily seeing a 1, 2, 1, 2, 1, 2 pattern or a 0, 1, 0, 1, 0, 1 pattern. The call rate is much less regular than that and, as we’ll see below, can be even stranger.

Now let’s deviate a little more from the norm. Let’s say that our game is performing very poorly for some reason and we’re running at 5 FPS for a total of 200 milliseconds per frame. Then we set the desired FixedUpdate rate to 100 Hz, which should mean calls every 10 milliseconds. Finally, we configure the maximum update interval for FixedUpdate to 50 milliseconds in Edit > Project Settings > Time. Here’s how this scenario would look:



In this case, notice how we’re *not* having our FixedUpdate called at the desired 100 Hz. The maximum interval of 100 milliseconds is being hit half way through each 200 millisecond frame, limiting the number of FixedUpdate calls to only 10 when it should have been called 20 times.

We can draw some conclusions with this expanded understanding of how FixedUpdate works. First, Unity will indeed try to call FixedUpdate at the desired rate. However, those calls need to fit within discrete frames. This means there may be zero, one, or more than one calls on any given frame as Unity dynamically adjusts to the actual frame rate. This also means that calls to FixedUpdate are not necessarily on the specified interval.

One particularly nasty situation to be aware of occurs when the FixedUpdate function itself is the cause of a frame rate slowdown. This will very likely result in Unity attempting to “catch up” with additional calls to FixedUpdate so that the desired rate is averaged out. Those additional calls will further slow down the frame rate since FixedUpdate is expensive in this scenario. The situation may compound until the maximum interval is hit. At this point the frame rate may have completely tanked.

Exactly what you should and shouldn’t use FixedUpdate for is beyond the scope of this article, but this deeper understanding should help in making those per-game decisions.