

Other Udon optimization tips

Update events, like Update() or FixedUpdate()

It is generally not recommended to execute code every frame, the performance impact can be quite noticeable.

`FixedUpdate()` is even worse because the refresh rate is generally higher. It is actually locked to your current screen's or headset's refresh rate (this behaviour is not documented in the VRChat documentation so it could change in the future...), you can get the fixed delta time with `Time.fixedDeltaTime`.

In Unity, the default fixed update rate is 50 updates per second, that value can be changed in the project settings, but VRChat overrides that value and it is not possible to change the update rate.

So if you're playing VRChat on a 200 Hz monitor, the `FixedUpdate` event will be executed up to 200 times per second!

Benchmarking your Update events

Merlin once published on Gist a Udon Profiler that you can use to benchmark your Updates, it can be downloaded here : <https://gist.github.com/MerlinVR/2da80b29361588ddb556fd8d3f3f47b5>

The profiler shows the execution time of your Udon code for each frame, it works for `Update()`, `FixedUpdate()`, `PostLateUpdate()`, and `LateUpdate()`, it does not benchmark `SendCustomEventDelayedSeconds/Frames()`

Alternatives to Update() #1 : SendCustomEventDelayed

A small Update event loop is generally fine, but if you want to execute an Update loop for each player in the instance, ask yourself the question if you really need that.

Also ask yourself the question if you really need to update something every frame. If you don't need to, you could use `SendCustomEventDelayedSeconds()` or `SendCustomEventDelayedFrames()` and build your own custom Update loop. Those methods take as parameter the event name (you need to use your custom event method), and the delay.

```

private void Start()
{
    CustomUpdateSeconds();
    CustomUpdateFrames();
}

public void CustomUpdateSeconds()
{
    SendCustomEventDelayedSeconds(nameof(CustomUpdateSeconds), 1.0f);
}

public void CustomUpdateFrames()
{
    SendCustomEventDelayedFrames(nameof(CustomUpdateFrames), 10);
}

```

Alternatives to Update() #2 : Animators

If you want to move an object smoothly from a point A to a point B, you could use an animator instead.

But also be careful, too many animators can also affect the performance, in certain big VRChat events with 60+ players you may have noticed heavy FPS drops even after hiding all avatars, that's because many avatars have terribly optimized animators, this also applies to worlds : Optimize your animators, but animators are still better than doing animations in Udon.

Alternatives to Update() #2 : Shaders

VRChat is heavily CPU bound, so sometimes it could be interesting to execute some "code" on the GPU somehow...

Well, shaders are executed on the GPU!

For instance here is a very basic vertex Shader function I wrote that scales a mesh based on the distance of the player, without any Udon !

```

v2f vert(appdata_base v)
{
    v2f o;

    #if UNITY_SINGLE_PASS_STEREO
        //The user is in VR
        //To avoid getting a weird 3D effect in VR, since there are two cameras in VR,

```

we'll basically say that the camera is between both eyes...

```
float3 cameraPositionWS = (unity_StereoWorldSpaceCameraPos[0] +
unity_StereoWorldSpaceCameraPos[1]) / 2;

#else

//The user is on Desktop

float3 cameraPositionWS = _WorldSpaceCameraPos;

#endif

float cameraDistance = distance(mul(unity_ObjectToWorld, float4(0.0,0.0,0.0,1.0)),
cameraPositionWS);

o.position = UnityObjectToClipPos(v.vertex * cameraDistance / _SizeReducer);
o.texcoord = TRANSFORM_TEX(v.texcoord, _MainTex);
return o;
}
```

- If you move further away from the mesh, the mesh scales up.
- If you get closer to it, the mesh gets smaller

The code above could probably be optimized even further, I am not a Shader programmer and I don't know any shader best practice...

If you want to know what amazing things the VRC community created with shader, here are a few worlds I highly recommend to check out:

- VRCLinux by _pi_ : https://vrchat.com/home/world/wrld_8126d9ef-eba5-4d49-9867-9e3c4f0b290d an entire operating system coded inside a shader
- pix2pix by SCR_N : https://vrchat.com/home/world/wrld_014d362b-164b-4b91-b628-a68f5ea0d5b7 an algorithm coded inside a shader that can detect what you're drawing.
- ShaderFes : https://vrchat.com/home/world/wrld_b526c3e0-7c32-4774-931e-5dcbbd125794 A showcase of some really amazing shader creations.

There are many, many other amazing worlds, and it is impossible for me to list them all.

Disabling scripts if they are not used

Let's say you have a fancy clock in your world, you might not need to update the clock if the player doesn't look at it.

Unity has two events OnBecameVisible/OnBecameInvisible, you could use those events to only enable the script if the renderer is visible by the player.

As of January 2023, OnBecameVisible/OnBecameInvisible does not work in Udon, but

`_onBecameVisible()` and `_onBecameInvisible()` works!

```

public Clock ClockInstance;

///It is important to note that those events only works on renderers!
///So that script needs to be attached to a MeshRenderer
public void _onBecameVisible()
{
    □ClockInstance.enabled = true;
}

public void _onBecameInvisible()
{
    □ClockInstance.enabled = false;
}

///Once those events get fully exposed in Udon, you can use the lines
///bellow
void OnBecameVisible()
{
    □ClockInstance.enabled = true;
}

void OnBecameInvisible()
{
    □ClockInstance.enabled = false;
}

```

Side note concerning triggers

Player triggers could also be used to disable scripts, but as of January 2023, stations can cause issues (PlayerTriggerEnter/Exit events might not get fired if someone sits on a station, even avatar stations). So make sure your triggers won't cause any issues because of a few stations.

- If you're **inside** a trigger, and you sit on a station that is located **inside** a trigger, `OnTriggerEnter` is called twice.
- If you're **inside** a trigger, and you sit on a station that is located **outside** a trigger, `OnTriggerExit` is not called.
- If you're **outside** a trigger, and you sit on a station that is located **inside** a trigger, `OnTriggerEnter` is not called, but gets called one you exit the station.

In certain cases, the safest way would be to use `OnTriggerStay`, but `OnTriggerStay` is called every fixed update rate (so it is again linked to the screen's refresh rate), so I would not recommend to

use it!

Revision #5

Created 2023-02-15 21:13:49 UTC by MyroP

Updated 2023-02-16 21:05:01 UTC by MyroP