Mobile depth water shader

Hello, my name is Paul and I'm in unity for few years now. For the last few month I was dealing with a lot of optimization routines on work while making mobile game. One of the problem was about a lot of overdraw which is critical for mobile hardware. We couldn't just turn off effects and simplify game graphics, so I needed to find a solution.

After few days of research I got it. The main problem was in transparent textures, so I decided to refuse them and write a shader that is making pretty much the same but without transparent texture. It makes all calculations right in the main geometry drawcalls. And it works!

That effect was a bit specific for our project and I can't put it into asset store. But I made a mobile depth water by the same way and it works really fast.

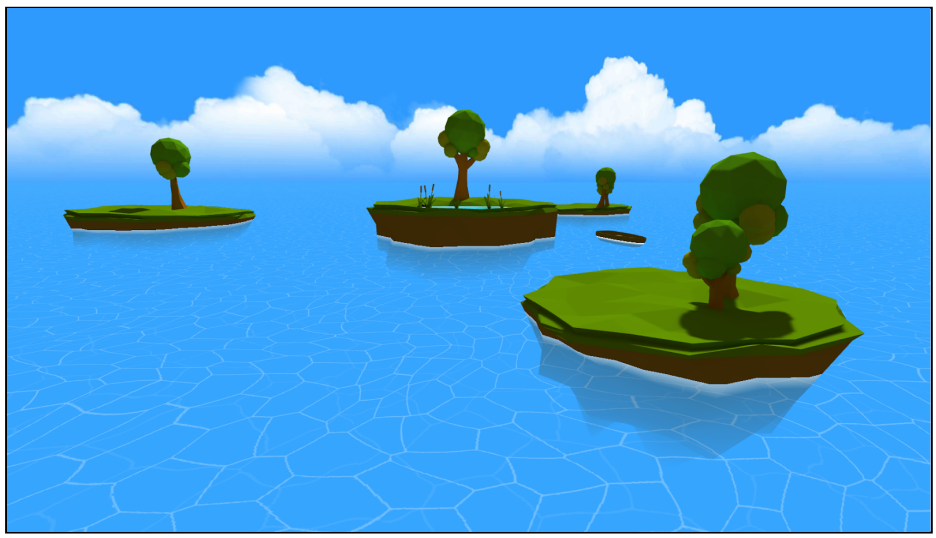
This demo scene was tested on several devices and here are results:

- Xiaomi Mi 4i: 58-60 fps
- Samsung Galaxy S4 mini GT-I9195: 56-60 fps
- Sony Xperia Z3 Compact: 47-48 fps
- Sony XPERIA S: 28-30 fps

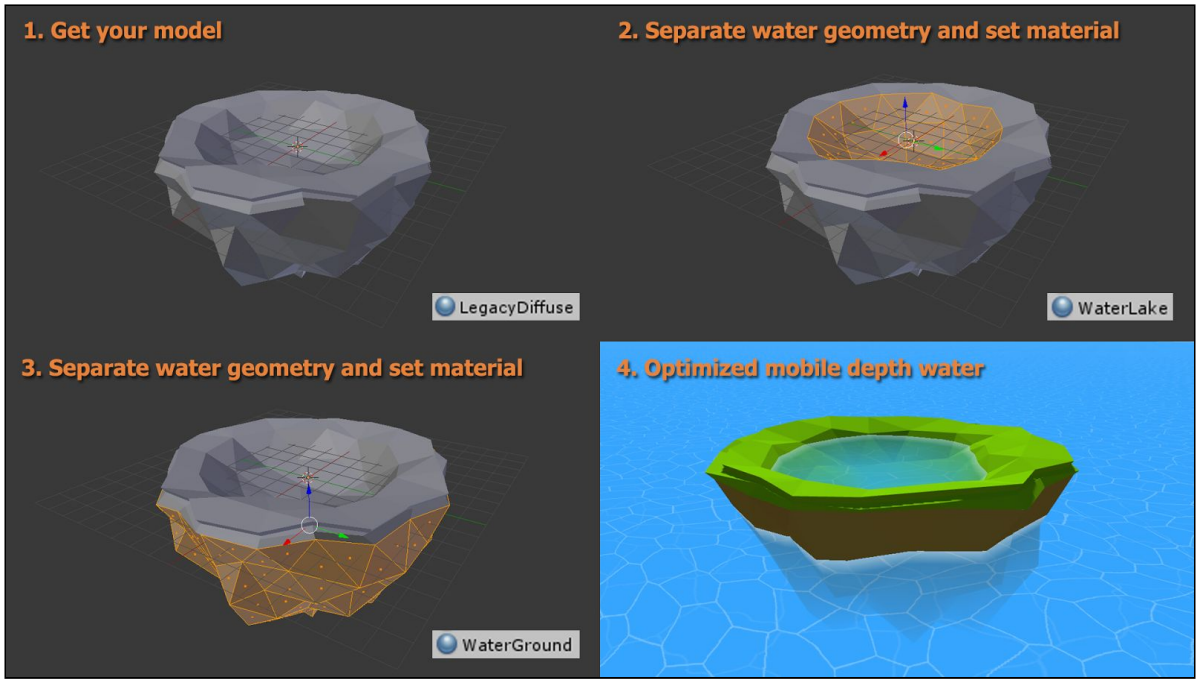
Work process

Ordinary water shader uses transparent texture, so if there are a lot of water on the screen, performance is going down to hell. Another problem of usual depth shader is using depth texture, that is rendering by camera. It's very heavy operation for mobiles.

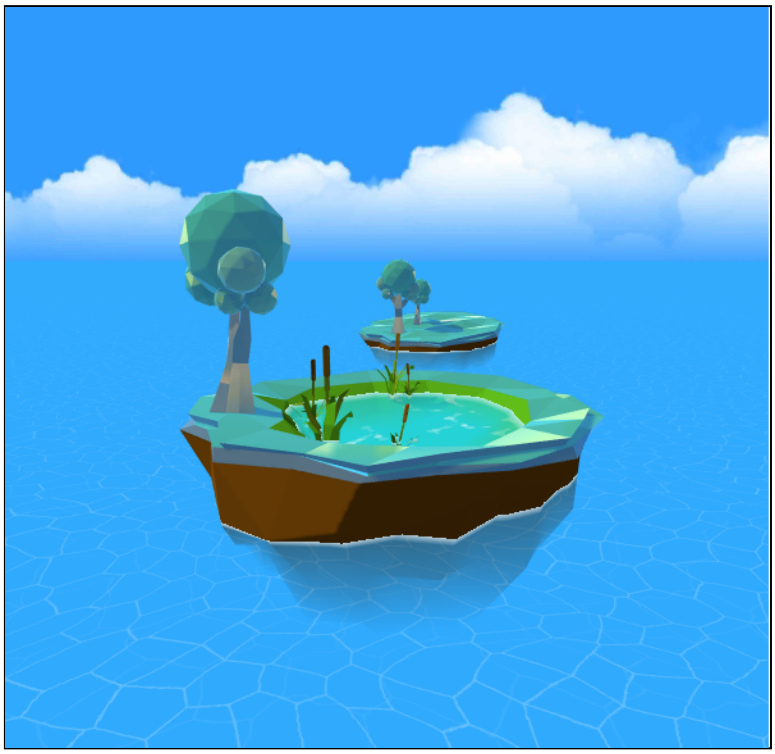
As I said, the main idea is drawing water right in main geometry drawcalls. So while drawing it we can get depth of each pixel without using any depth textures. Moreover we don't need transparent texture anymore! All two problems are solved.



But because of such way of rendering water, it's a bit tricky to setup scene. Water height became global parameter for the whole mesh and there is no good way to make it working with just one separate part. Or not? To achieve it we need to separate geometry that will be in the water.



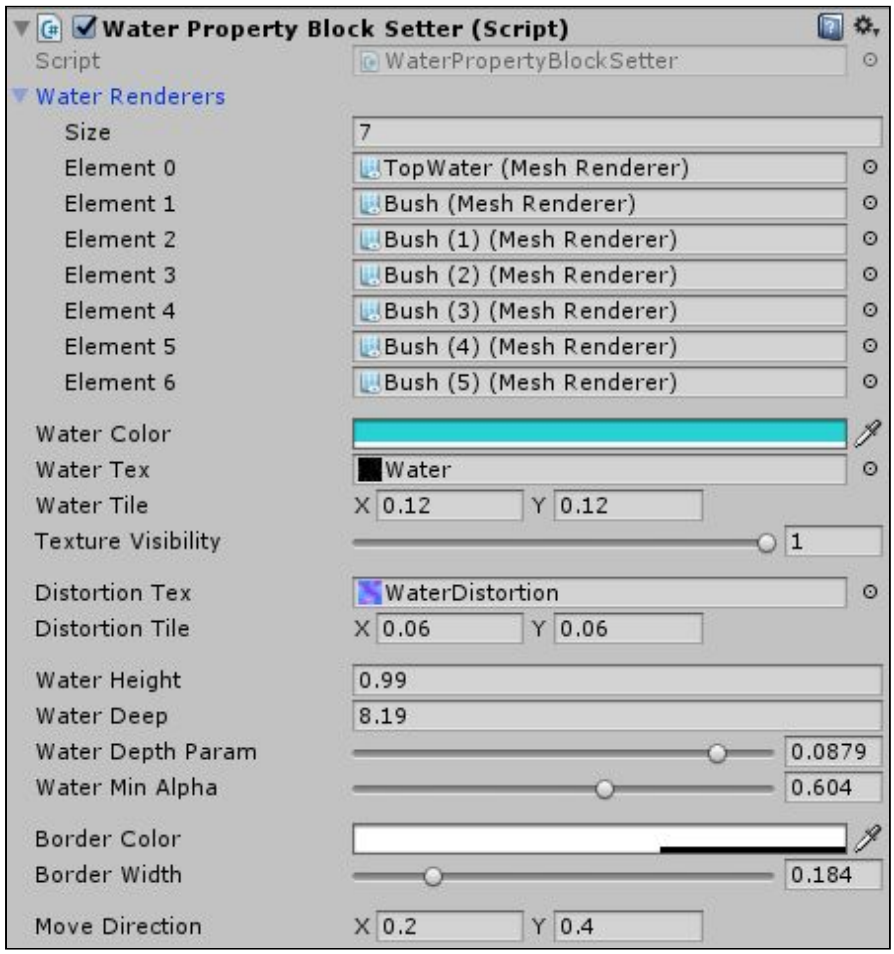
As you see, this shader is drawing not only water, but the main geometry too. So this is another bottleneck: you can't use it with custom shaders. Water uses legacy diffuse shader as a base. So for your main geometry you can use materials with the same coloring and lighting model. In another case geometry in water will look differently like here (main geometry uses standard specular shader):



Scripts

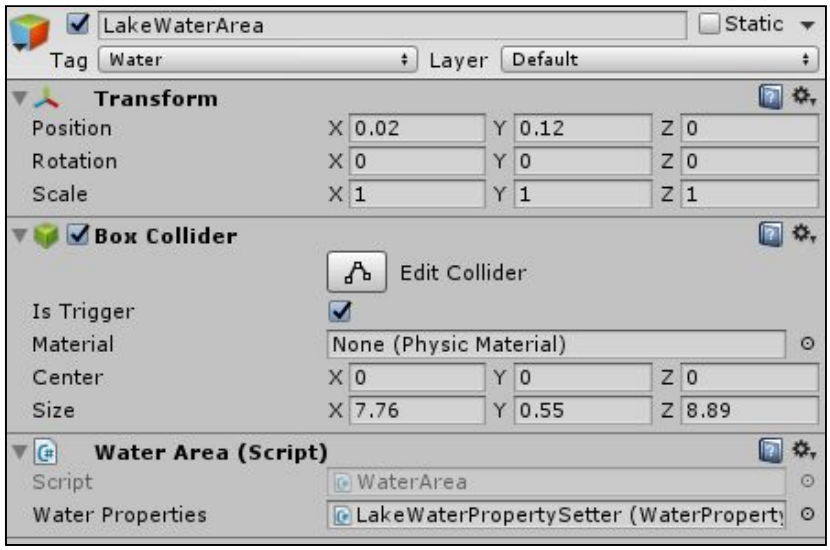
I made few scripts that will help you to manage all these properties and dynamic objects, that can enter and exit water.

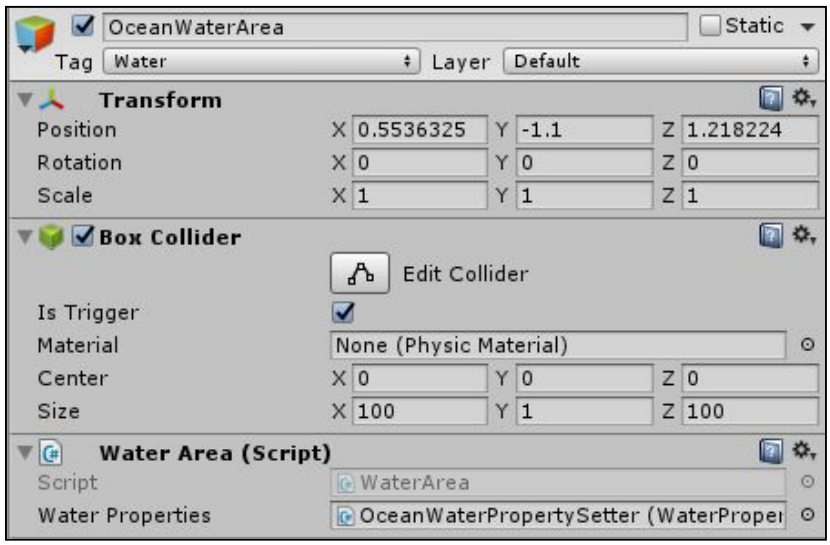
The first one is **WaterPropertyBlockSetter.cs**. Put it on the scene and add all renderers that use one type of water to **Water Renderers** property. Then you can set up water settings inside **WaterPropertyBlockSetter** game object and it copies them to all renderers by itself. So you don't need to set up all renderers separately.



Another script **WaterMaterialSwitcher.cs** will help you to manage dynamic objects in your scene. Add it to your dynamic object. You need to have two materials for it: **water material** and **default diffuse material**. Link them to the script. Also link your dynamic object's renderer.

Then you need to define your water areas, where dynamic object will be in water. **WaterArea.cs** script will help you. There are two areas in demo scene: **OceanWaterArea** and **LakeWaterArea**. These areas need to have **Colliders** to catch **TriggerEnter** and **TriggerExit** events. There is a link to **WaterPropertyBlockSetter** in every **WaterArea**. When dynamic object enters them, it gets water properties from that **WaterPropertyBlockSetters**. Here are two water areas:





Notice! *Water area game object should has tag* ***"Water"****!*

Also don't forget to add **Rigidbody** and **Collider** to dynamic objects, so they can catch **TriggerEnter** and **TriggerExit** events.

For large water planes that render only water, use **DiffuseWaterOpaque** shader. It's much faster. Also put this plane under all geometry of the scene.

All examples you can find in **DemoScene**.

Notice! *Without managing dynamic objects and switching their material, you can get into situation, when your object is far from water, but if it is below water level (that is setted up in material) it will rendere water anyway.*

Supporting

For now shader doesn't support orthographic camera and it wasn't tested on iOS. But it supports lightmaps, light probes and unity default fog.

If there will be a lot of request for supporting another lighting and coloring models, I'll do my best to bring it to build.

Conclusion

I know, there are a lot of limitations of using this technique for the first view, but it's really easy to use it if you use suitable shader for the main geometry. Also the result and performance are awesome. Hope you will find it helpful.

If you have any questions, feel free to write me: **valakhp@gmail.com**