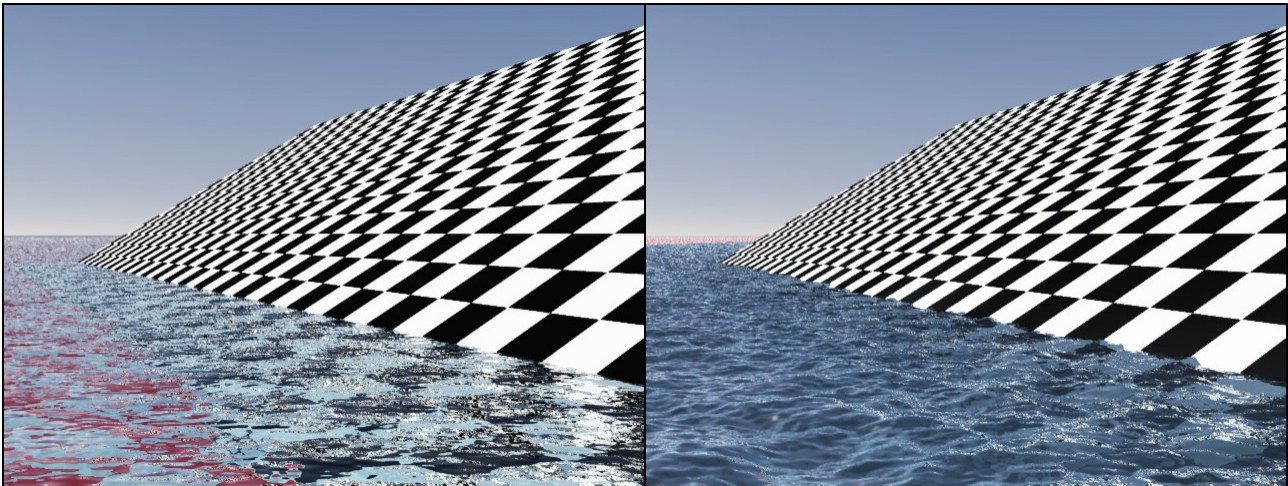


Tutorial

Real Waves in Vue6

After having looked around the internet and found not a single tutorial covering this topic I thought, either nobody ever thought about that or that it's *that* easy that everybody can do it. Anyway, I tried around a bit, and I think I came up with a solution myself. However, this tutorial is in no way complete nor does it come up with *the* definitive approach for creating real waves.

Here's where we start, and where we'll eventually end up with:



Okay, how to get there? The trick is, to use a procedural terrain which is large enough that it won't show anything of the underlying ground layer close to the horizon.¹

Step 1: Creating the water terrain

I suggest that you start with the default water plane, and load a material that you like – in this case I used the “Default Water” mat with the bumps scaled down to 1.0 instead of 1.5. That just helps to make things a little easier, and in the case of the “Default Water” mat produces better results.

Now, go to the material editor and click on the “Bumps” tab. There you find the function preview. Right click on it and save the bumps function. You may want to have a look at the bump depth – the “Default Water” is set to 0.200. Okay, let's leave the material editor.

Select the water plane and delete it. Then create a flat terrain with the resolution of 512x512. You should end up in the terrain editor. Switch the flat terrain from standard to procedural, and make it “Skin only”. In case you're being asked:

Don't let Vue add nodes to preserve the current terrain's geometry!

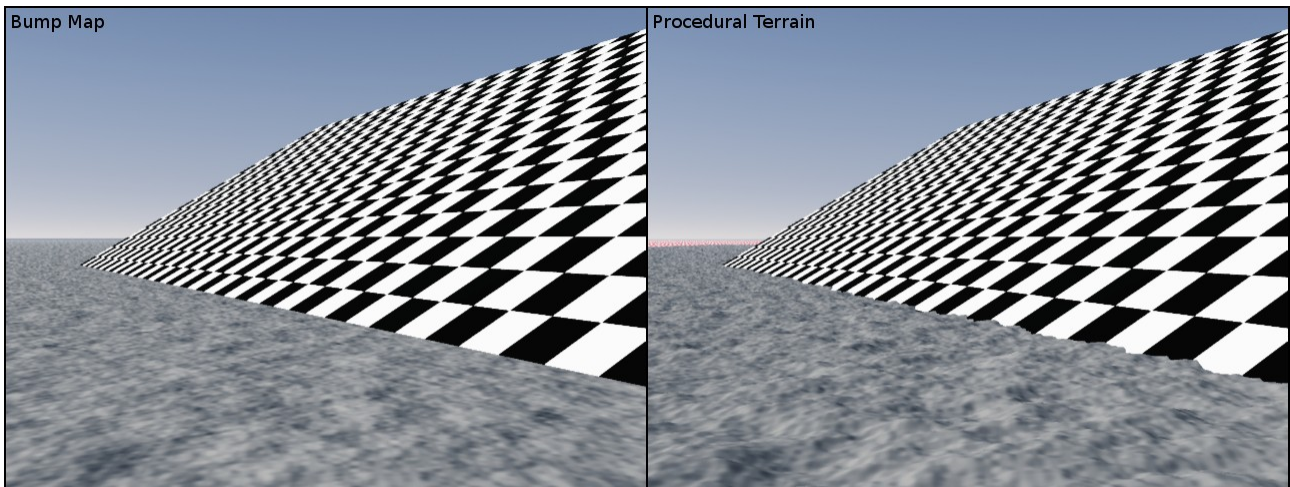
Now, go to the “Proc.” tab and first check if the scaling of the function is set to 1.0, and if the “Quality Boost” slider is set to zero. Enabling “Fast Shadows” is also a good idea. Next, right click on the function preview and load the saved bumps function. Change the mapping mode to “World” and again:

Don't let Vue add nodes to preserve the current terrain's geometry!

¹ I didn't pay attention to this, since how large the terrain has to be depends on various settings in your scene, for example how much fog and haze there is in the atmosphere, or if there's a distant terrain which hides (parts of) the horizon line.

When you leave the terrain editor your newly created water terrain will appear. Only that the waves are way too high. So, go to the “Numerics” tab and set the height of the terrain to the same number as the bumps scale was (in meters; 0.2m in case of the “Default Water”).

The following comparison shows our results so far:

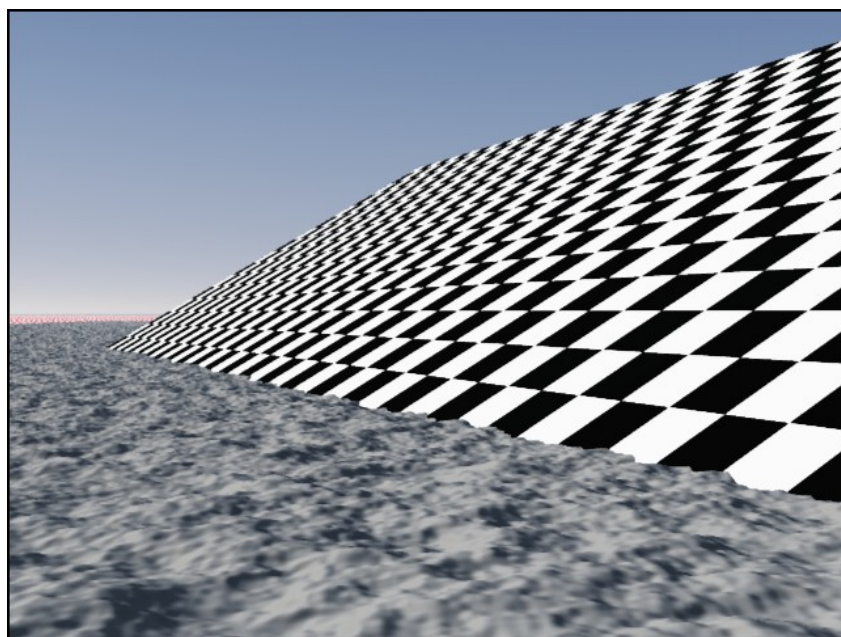


I reset the material here, so the effects are more visible. On the left is the infinite water plane, with the previously saved bumps function loaded and set to 0.2, on the right you see the water terrain.

Note: when you need to resize the terrain, only do so in one direction at a time, otherwise you'll spend a lot of time resetting the terrain height back to 0.2, or whatever height you started with. The “World” mapping mode of the terrain function takes care, that the waves remain in the same place, while the terrain's getting bigger.

The water terrain already looks pretty good. But it could need a little more structure. So, why not use the bumps function with the – at the moment solid gray – material? “Bumps” tab, load bumps, set depth to 0.200.

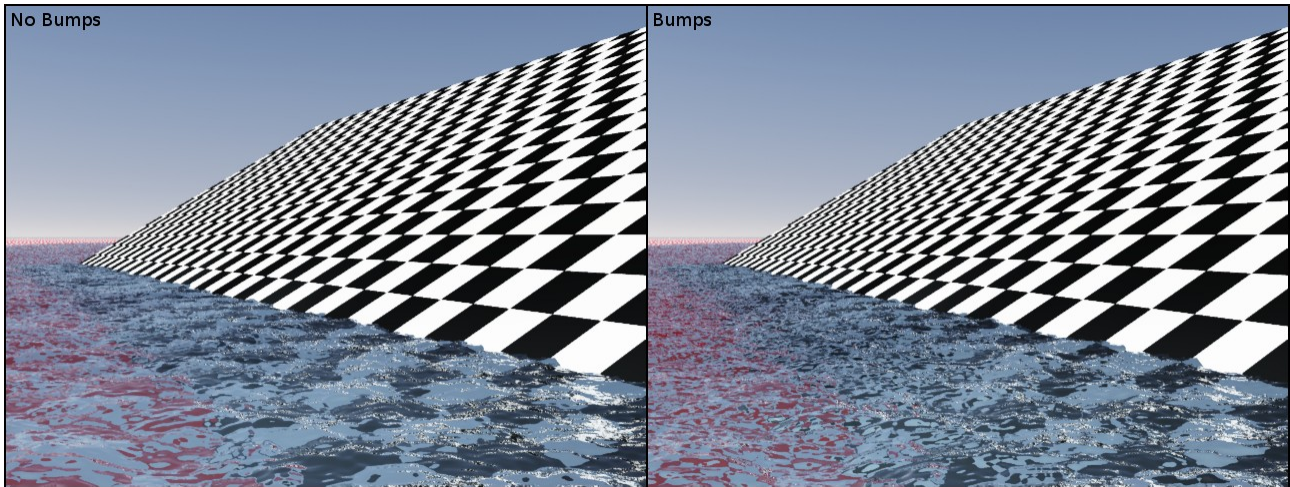
Voilà!



Step 2: Apply the water material

That's a fairly simple step. Just load the water material and adjust the bumps scaling (if needed).

Depending on your decision, whether to use an additional bump map or not, the result is as follows:



Just a word on one setting you made before. The water terrain was created as “Skin only”. This produces just the shape of the waves, instead of a solid filled terrain with a maximum thickness of, say, 0.2 meters.

Step 3: Subsurface Scattering

As you notice, the ground is shining through very much. Since this might be appropriate for some purposes, this step is not obligatory, but it helps improving realism.

Go to the material editor. First set “Fading out” to zero in the “Transparency” tab. Then go to the “Translucency” tab and use the following settings²:

Average depth: 3.2m
Balance: 100% (full absorption – no multiple scattering)
Refraction Index: 1.33 (IOR of water, if not already set)
Color: same hue as the basic water color, with much more luminosity.
Anisotropy: 0.24

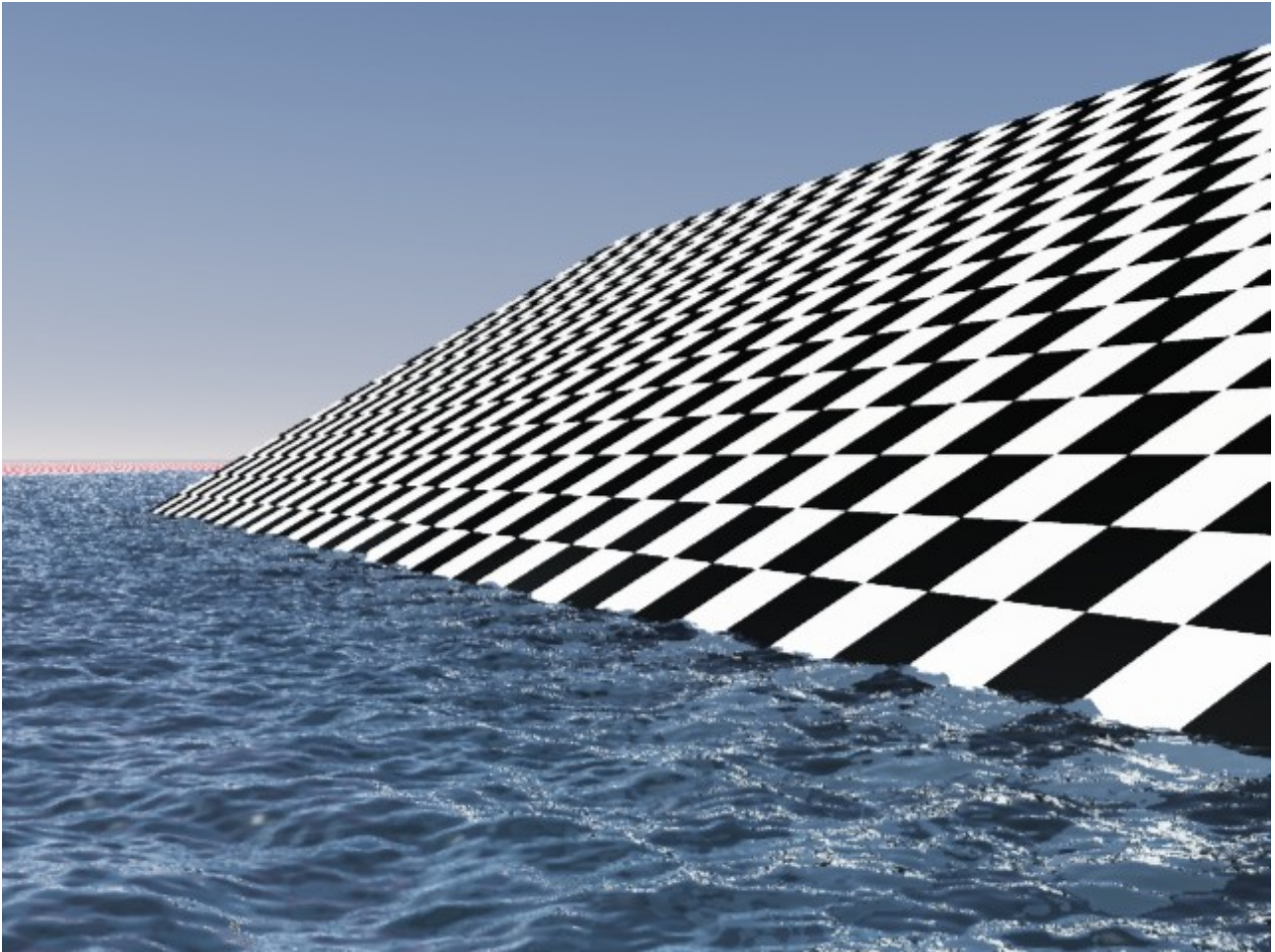
And in the “Effects” tab set the “Color transmitted light” to 50%

You may want to play with these settings. Especially the “Average Depth” is usefull to control how far light travelles, and thus how transparent or how opaque the water is – for this tutorial, I set the Depth to 3m.

See next page for the final result!

² These settings come from **Rutra** at **Renderosity**
http://market.renderosity.com/mod/forumpro/showthread.php?thread_id=2747422

Final image:



We now have achieved a water surface with real waves and, in this case, just a very subtle transparency close to the surface. The result is the effect of a slightly deeper water. Actually it's just about 2m high. So, you could use subsurface scattering to control the depth of the water – usefull if you're not working in a scale of 1:1.

Concluding remarks:

You don't necessarily have to use the same water material that you've used to get the bumps function from. Other combinations, such as using “Default Water” for the procedural function and “Faked Caustics” for the water material can yield good results, as well.

You can use basically every bump function from Vue's default water materials. Just make sure that the scaling is correct. I'd suggest to control the scale from inside the function editor rather than with the bumps scaling in the material editor, because you can leave the funtcion of the terrain set to 1.0.

Of course you're not restricted to using only the bumps function found in the water materials. If you're really eager or if you don't find a suitable bump function, you can always build the terrain function and the water material from scratch.

**Thanks for reading through this,
and have a lot of fun with real waves in Vue.**