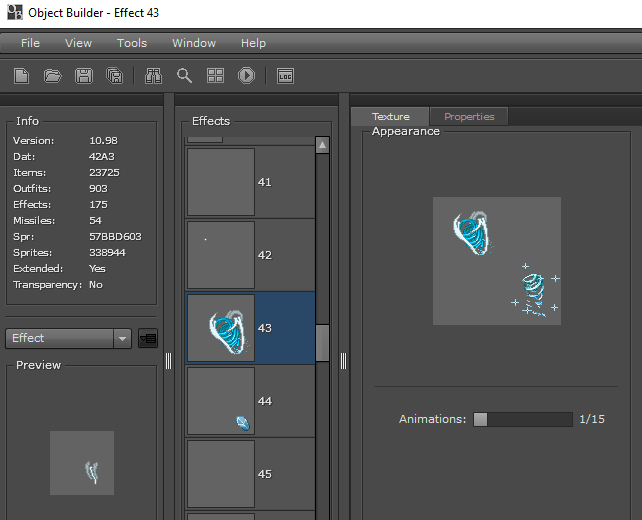
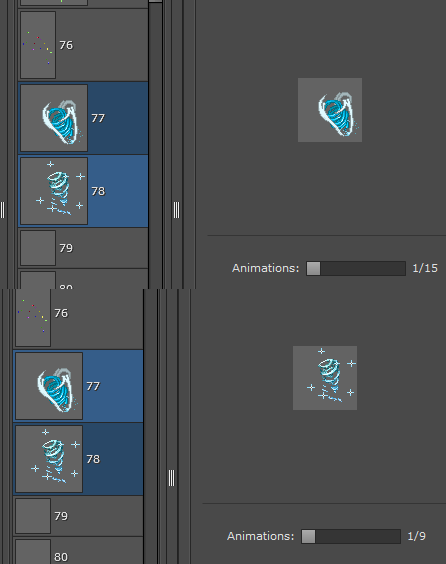
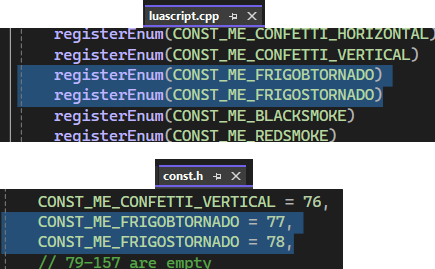
I’ll start by saying that I’m not sure if I reproduced the Frigo spell in the way the lead developer wanted, but I did my way and I’m satisfied with the result.

The first thing I tried was to replicate the area of the spell, and testing it I realized I don’t know how to force the pattern of the frigo spell that I wanted, so the fastest solution I could think of was to separate the effects into two effects of their own, and I did it by editing the spr and dat of the cliente 1098 I was using as a base.



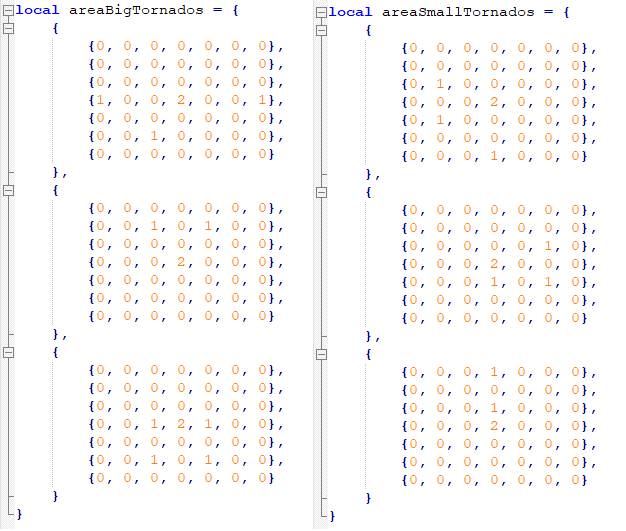
The image above turned to the image below, and I compiled, saved and went to the server source to implement those two new effects. By just adding the effects all that I changed was the Tibia.dat, not the sprites.





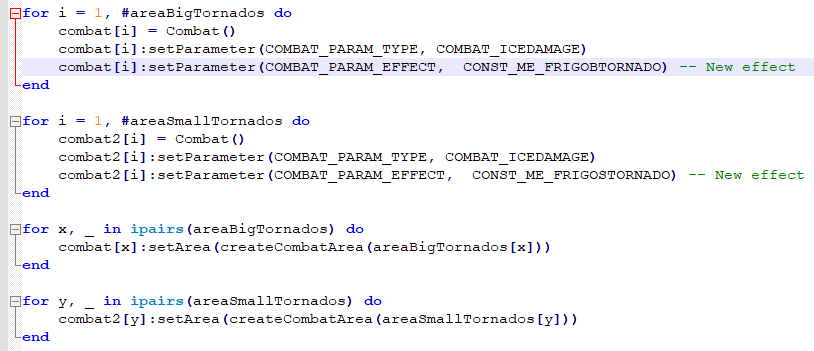
The image above shows how easy it was to list the new effects on the server side, I just had to list the effects and put their Ids referente to their number on the object builder. Then all that was left was to create the spell Frigo and set it’s area.

I first analyzed the reference vídeo, and discovered that it had 3 patterns, that looped 3 times to create the spell, so I started by creating two areas, one for the big tornadoes, and one for the small ones.

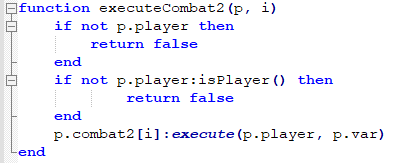
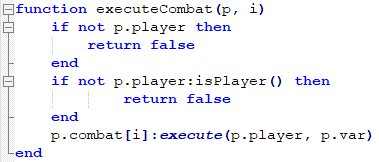


It’s two vertex with 3 areas each, and I had to write them reversed and upside down, because, I’m not gonna lie, I don’t know why they were happening reversed, maybe it’s reading the vertex from the end to the beggining, so I just adapted and wrote it backwards.

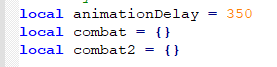
After I set the effects and damage types, and create the combatAreas, all checking with for in pairs.

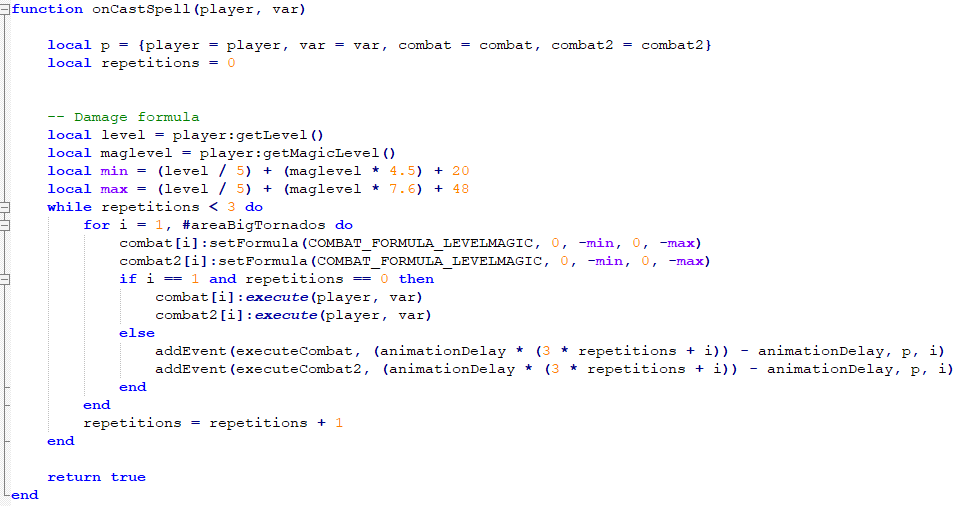


I declare the functions that execute the combat, and I have two of them because some effects of the small tornadoes appear separately from the big tornadoes, so I used 2 so I create events for them separately.



After all this come the function onCastSpell, that calls everything and makes it happen. To make it loop I used a local variable that starts at 0 and goes up to 2, and I use it to count each loop it must do. I could’ve used a for inside a for but I opted to use a for inside a while and count manually the repetitions, since I found it easier to manipulate the tests and adjust the delay between loops. The interval between each round of spell effect was taken by the eye watching the vídeo, so it’s not exact but it’s very similar. It’s declared at the very top of the spell function, and I manually calibrated it.





The result you can see in the video, and I’m very fond of result, hope it reaches your expectations and that I was clear in the explanation, the codes are also commented.