Fantasy Grounds Effects Processing Performance Improvements

Author: Ryan Hagelstrom
Date: March 19th, 2023

Problem

There is considerable inefficiency in the Fantasy Grounds (FG) Effects processing system, specifically in the functions getEffectsByType and hasEffect. Both of these functions are CoreRPG functions but are reimplemented for each ruleset to provide extra checks specific to the ruleset. Most effect tags, shown in Figure 1, are processed within these functions, with the exception of some extensions that aren't following the FG paradigm. While this inefficiency is largely unnoticeable, the problem can manifest itself as sluggishness when the Combat Tracker (CT) actors have a significant number of effects, extensions are enabled that, in totality, add a large number of new tags, or extensions are enabled that enable overloading of the CT with a large number of actors.

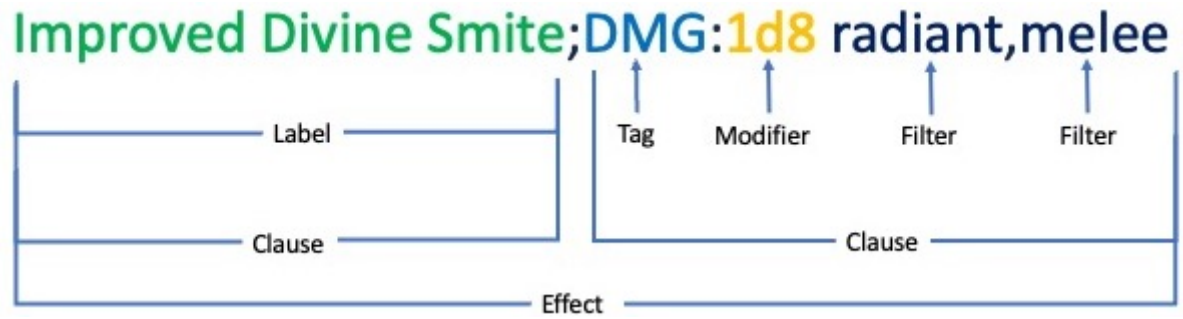


Figure 1 Example Fantasy Grounds Effect

Hypothesis

A reduction in the number of processing loops in the two main effects processing functions, getEffectsByType and hasEffect, would reduce the overall processing time for a tag and provide better FG performance.

Prediction

A reduction in processing loops is expected to be significantly more efficient, with a 100% or more improvement, and to eliminate the sluggishness when the Combat Tracker (CT) actors have a significant number of effects, extensions are enabled that, in totality, add a large number of new tags, or extensions are enabled that enable overloading of the CT with a large number of actors.

Experiment

A solution was implemented as an extension, Turbo.ext, that reduces the processing loops in getEffectsByType and hasEffect. Observation was done by implementing a timestamp when getEffectsByType was entered and when it returned, in the Better Combat Effects Gold extension. The difference in those two timestamps is the time getEffectsByType spent processing a single tag. The value was then converted from milliseconds to microseconds and saved in one of two CSV strings, one for normal (vanilla) FG functionality or another which the hypothesis is tested. The code for experiment observation is below. The data was output to file on FG tabletop close for later analysis.

```
function customGetEffectsByType(rActor, sEffectType, aFilter,
rFilterActor, bTargetedOnly)
    local timeStart = os.clock();
    .
    .
    .
    local timeProcess = os.clock() - timeStart;
    if TurboManager and OptionsManager.isOption('TURBO', 'on') then
        nTotalTimeTurbo = nTotalTimeTurbo + timeProcess;
        sTurboTimes = sTurboTimes .. ',' .. timeProcess * 1000;
    else
        nTotalTimeOriginal = nTotalTimeOriginal + timeProcess;
        sOriginalTimes = sOriginalTimes .. ',' .. timeProcess * 1000;
    end
    -- RESULTS
    return results;
end
```

An initial test was performed with a party of four with twelve total effects looping the CT ten rounds, which provided encouraging results. Playtest data was then gathered from multiple sources over a couple of months from DMs running Windows, OSX, and Linux consisting of at least two games run without Turbo.ext enabled in order to gather baseline data, and then at least two games with Turbo.ext enabled to test the hypothesis and gather performance improvement data. All data gathered was with the 5E ruleset.

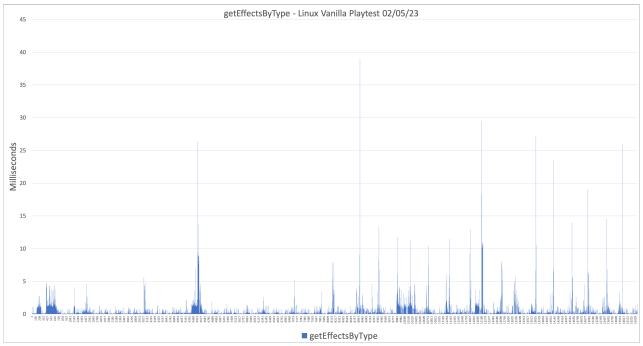
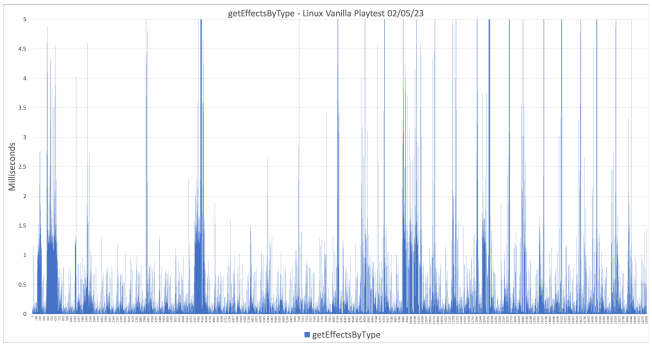
Results

The following graphs display the results of game playtesting on Linux, OSX, and Windows. Due to the imprecise nature of os.clock() in Windows, useful data gathering was abandoned for this platform. The left-hand graphs depict the raw data, with occasional spikes that may be due to processing interruption, whereas the right-hand graphs' Y-axis is capped at 5ms for easy visual comparison. Each graph contains 16382 data points, which is the limit for Excel import, but additional missing or present data points do not affect the results. The average time spent in getEffectsByType during the 16382 data points is shown below the graphs. The total game time in function getEffectsByType is listed but cannot be compared with each other, as that number is dependent on the length of a session and the number of times getEffectsByType is called. Total game time, however, is a measure of how much time was spent waiting for Fantasy Grounds to process effects.

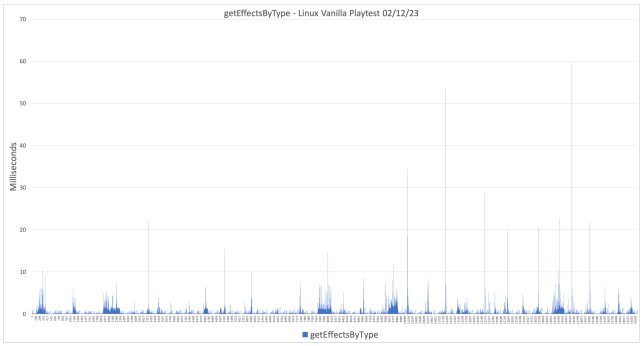
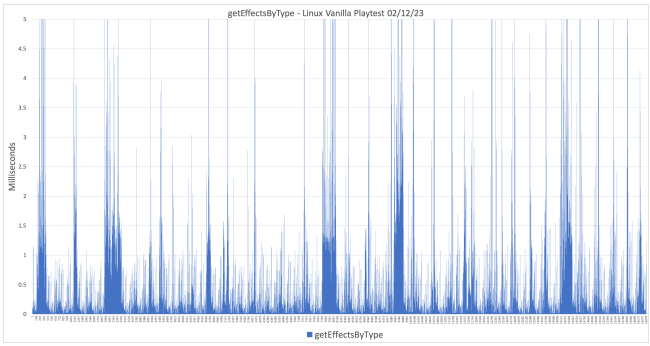
The results indicate that the enhanced data, which includes Turbo.ext, consistently outperforms the data collected without Turbo.ext enabled on both Linux and OSX. On Linux/OSX, an average performance increase of **590%** was observed for getEffectsByType.

In addition, GMs reported a feeling of *"I don't notice any difference."* While the data indicates significant improvement, the differences are so small that they are hardly noticeable. Therefore, any performance *"lag"* experienced by the GM and players is likely caused by another source and not effect processing.

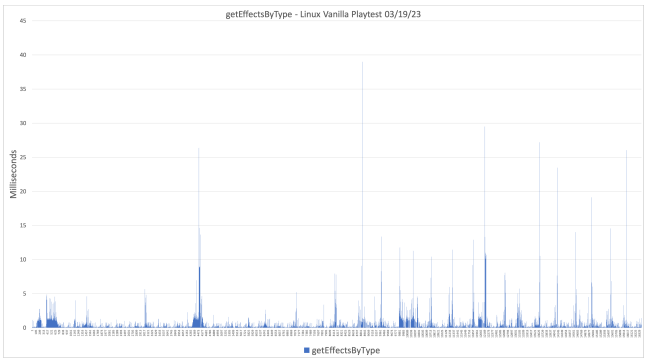
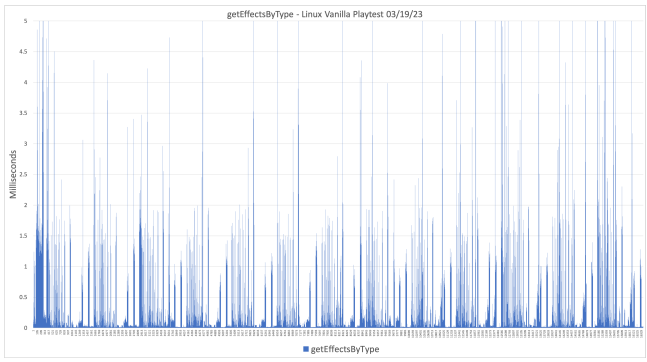
Linux Vanilla Playtest 02-05-23	**Linux Vanilla Playtest 02-05-23 (5ms Scale)**

Linux Vanilla Playtest 02-05-23	Linux Vanilla Playtest 02-05-23 (5ms Scale)
	

- Average Time Per Function Call: 0.315030397 ms
- Total Game Time in Function: 32.75429 seconds

Linux Vanilla Playtest 02-12-23	Linux Vanilla Playtest 02-12-23 (5ms Scale)
	

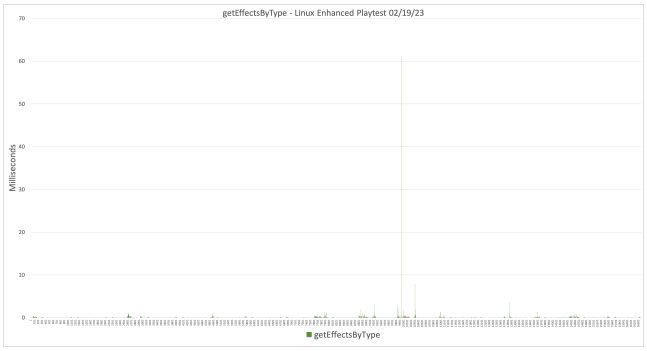
- Average Time Per Function Call: 0.406323587 ms
- Total Game Time in Function: 104.365166 seconds

Linux Vanilla Playtest 03-19-23	Linux Vanilla Playtest 03-19-23 (5ms Scale)
	

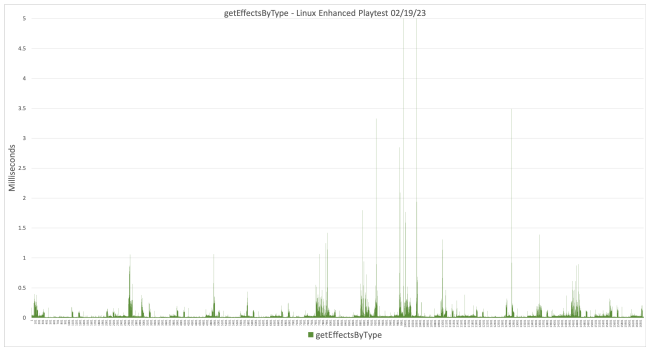
- Average Time Per Function Call: 0.285552863 ms
- Total Game Time in Function: 84.422962 seconds

Linux Enhanced Playtest 02-19-23	Linux Enhanced Playtest 02-19-23 (5ms Scale)

Linux Enhanced Playtest 02-19-23

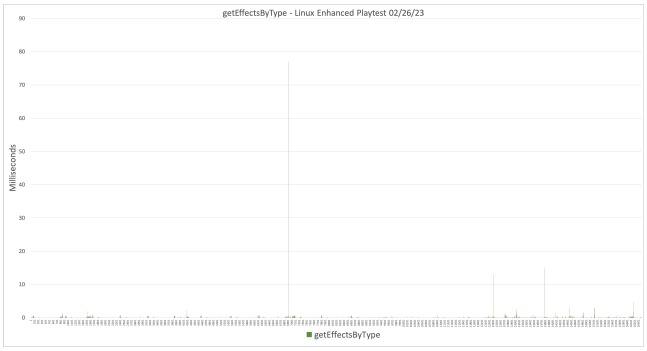


Linux Enhanced Playtest 02-19-23 (5ms Scale)

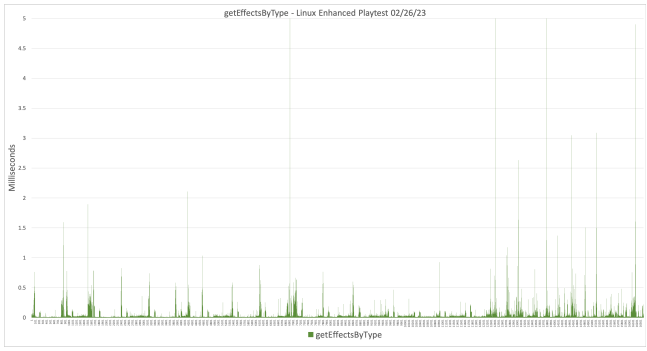


- Average Time Per Function Call: 0.067710353 ms
- Total Game Time in Function: 1.4895 seconds

Linux Enhanced Playtest 02-26-23

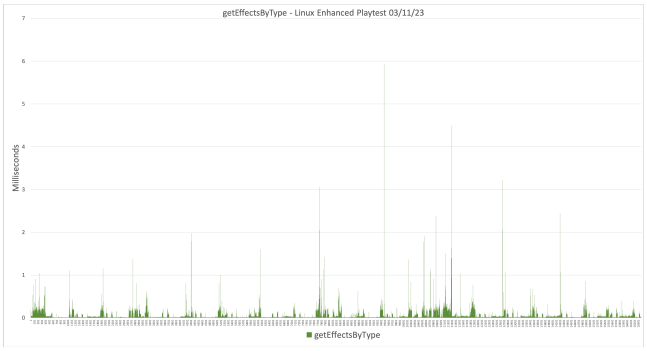


Linux Enhanced Playtest 02-26-23 (5ms Scale)

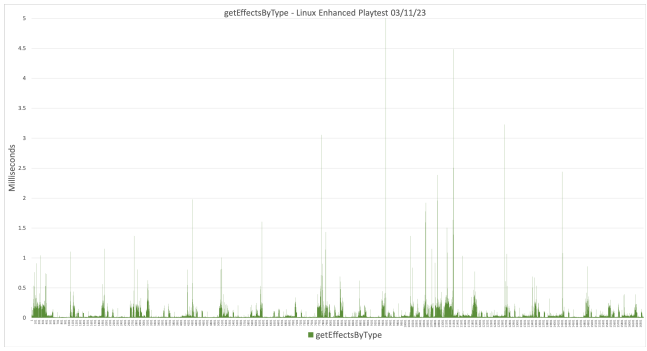


- Average Time Per Function Call: 0.052520571 ms
- Total Game Time in Function: 2.18681 seconds

Linux Enhanced Playtest 03-11-23



Linux Enhanced Playtest 03-11-23 (5ms Scale)

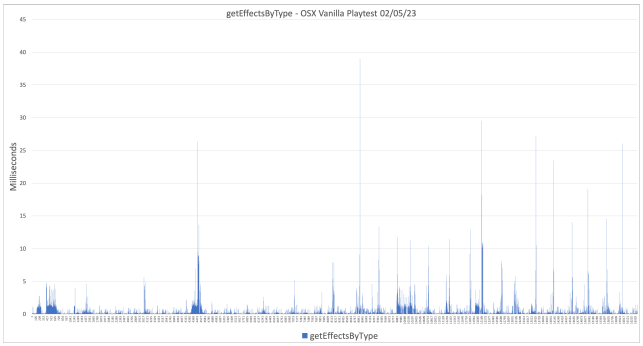


- Average Time Per Function Call: 0.054786839 ms
- Total Game Time in Function: 3.356106 seconds

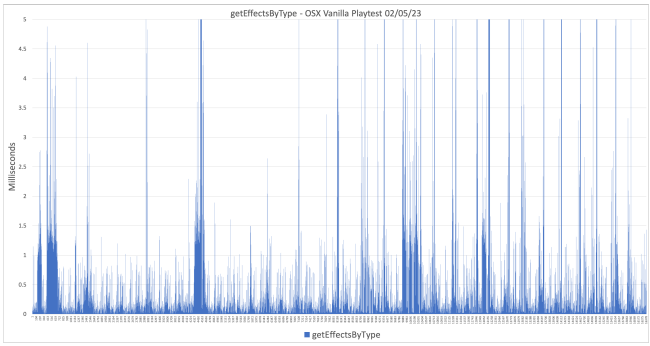
OSX Vanilla Playtest 02-05-23

OSX Vanilla Playtest 02-05-23 (5ms Scale)

OSX Vanilla Playtest 02-05-23

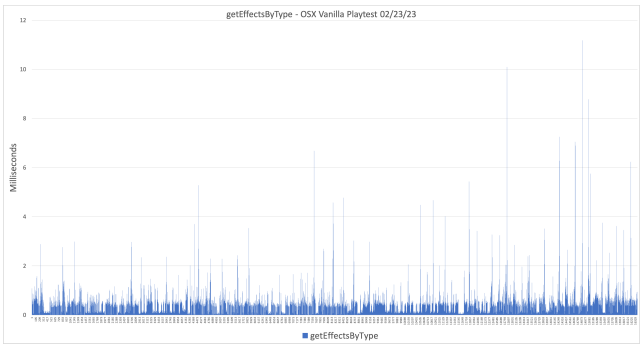


OSX Vanilla Playtest 02-05-23 (5ms Scale)

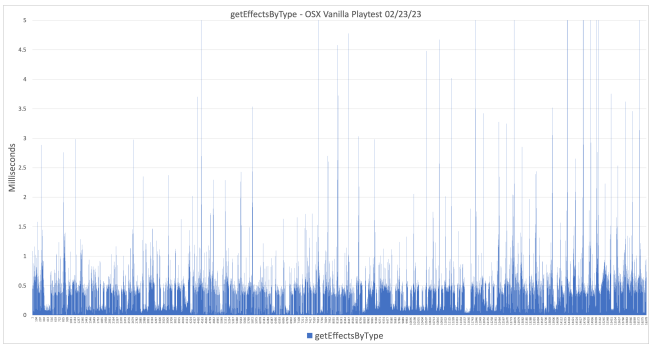


- Average Time Per Function Call: 0.325219631 ms
- Total Game Time in Function: 6.683059 seconds

OSX Vanilla Playtest 02-23-23

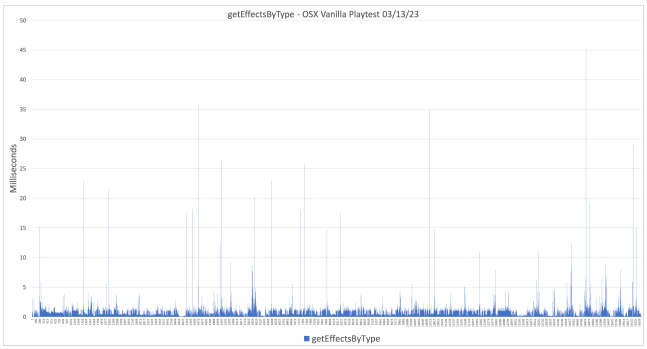


OSX Vanilla Playtest 02-23-23 (5ms Scale)

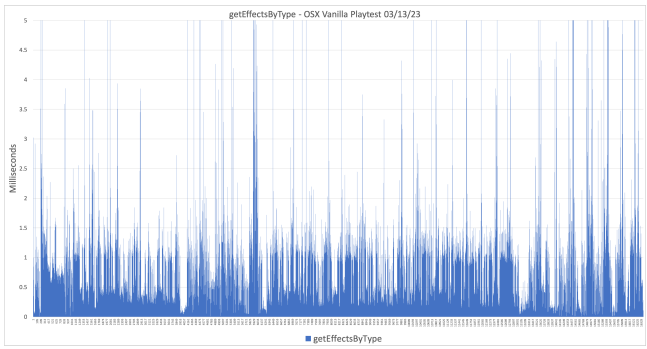


- Average Time Per Function Call: 0.315030397 ms
- Total Game Time in Function: 32.75429 seconds

OSX Vanilla Playtest 03-13-23



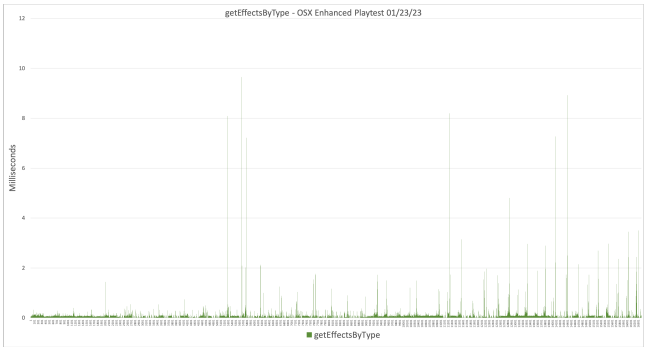
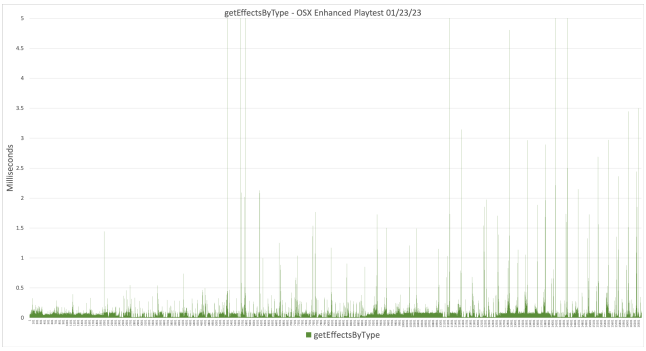
OSX Vanilla Playtest 03-13-23 (5ms Scale)



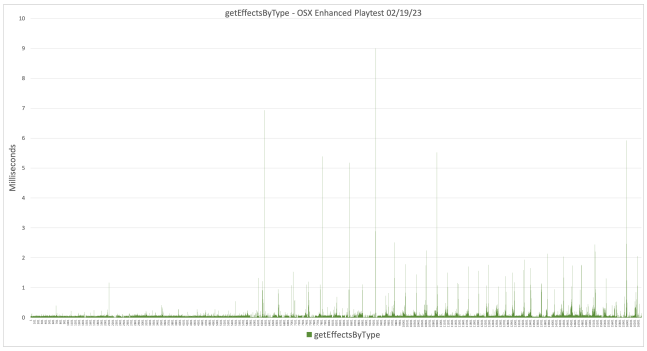
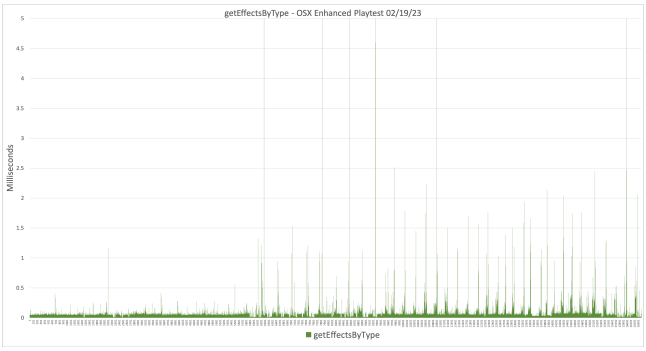
- Average Time Per Function Call: 0.707113295 ms
- Total Game Time in Function: 21.586727 seconds

OSX Enhanced Playtest 01-23-23

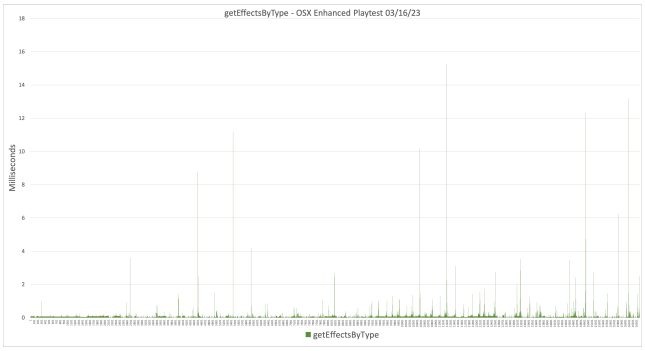
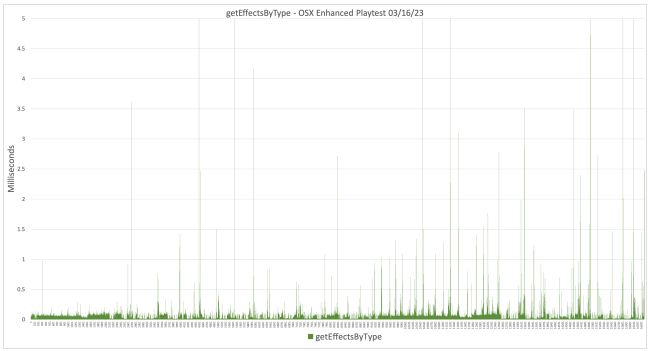
OSX Enhanced Playtest 01-23-23 (5ms Scale)

OSX Enhanced Playtest 01-23-23	OSX Enhanced Playtest 01-23-23 (5ms Scale)
	

- Average Time Per Function Call: 0.069346844 ms
- Total Game Time in Function: 1.454021 seconds

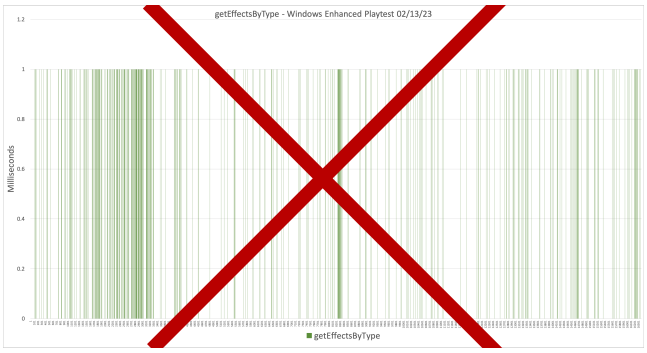
OSX Enhanced Playtest 02-19-23	OSX Enhanced Playtest 02-19-23 (5ms Scale)
	

- Average Time Per Function Call: 0.067710353 ms
- Total Game Time in Function: 1.4895 seconds

OSX Enhanced Playtest 03-16-23	OSX Enhanced Playtest 03-16-23 (5ms Scale)
	

- Average Time Per Function Call: 0.067418996 ms
- Total Game Time in Function: 1.717316 seconds

Windows Enhanced Playtest 02-13-23	Windows Enhanced Playtest 02-13-23 (5ms Scale)

Windows Enhanced Playtest 02-13-23	Windows Enhanced Playtest 02-13-23 (5ms Scale)
	

Windows Enhanced Playtest 02-13-23	Windows Enhanced Playtest 02-13-23 (5ms Scale)
	

Summary

Turbo.ext has shown an average performance improvement of **590%** in the 5E ruleset's getEffectsByType function. Although hasEffect's performance gains were not measured, it is also expected to improve, albeit to a lesser extent than getEffectsByType, since it is a simpler function. Turbo.ext has been adapted to support various rulesets, including 5E, 4E, 3.5E/PFRPG, 2E, PFRPG2, and SFRPG, and can be customized to support additional rulesets. Performance gains in rulesets other than 5E are expected to be similar. While performance gains with Turbo.ext may be mostly imperceptible, any improvement will help reduce the perceivable performance impacts caused by other sources.