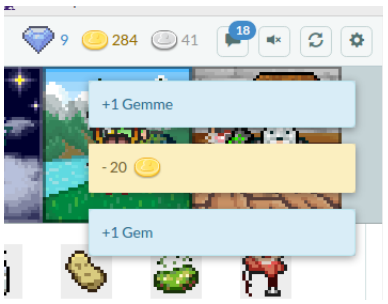
Open Source Project

I began this semester with a few open source projects of my own that I've been maintaining. I had had some open source experience before; my capstone project is completely open sourced as well as a repository full of work I've done throughout my college career. Much of the open source work I've done, however, has not been collaborative. At the beginning of the semester, we read over *The Mythical Man Month* which outlined various open source ideas. One of the ideas perpetuated by the author is that open source projects fall under Brook's Law, which states adding more manpower to a project delays the ultimate outcome of the project. In the open source world, however, this phenomenon seems to be absent. The project that we worked on, Habitica, is a prime example of an application making great strides towards completion while having hundreds of contributors. One of the reasons that open source projects grow so quickly towards their goals might actually be due to the lack of communication. Large groups having to communicate often leads to miscommunication; additionally, it takes a lot of time to meet, talk about what needs to be done, and ultimately get around to doing it.

Midway through the semester, my Apple laptop died, so I was forced to buy a new laptop. Since the newest models of MacBook Pros have only PCIe SSD's in them, I went with a Windows machine; since I don't care for windows, I decided to install Linux; since I was in the middle of the semester and didn't have the time to configure a version like Arch, I decided to install Ubuntu. I've noticed subtle bugs within the Ubuntu distribution that I have, such as the Dock getting stuck and the system not always obeying the Super key. I've been considering looking into the Ubuntu source to see if these issues are fixable by someone with my experience. Linux overall is an incredible example of open source in its prime. The idea that an entire operating system can be driven by an open source community is extraordinary. The main reason that I haven't looked into the source of Ubuntu or Linux is that I wouldn't have anywhere to test my changes and the source is not in a neat place such as Github (the kernal is on Github, but I'm nowhere near experienced enough to mess with it). The Linux operating system is not very different from OS X, so making the switch was not too hard. Originally, I wanted to use a fork of Ubuntu called Elementary OS. The user interface of Elementary OS is very close to that of OS X; ultimately I was not able to make it past the installation onto my computer.

Habitica was a very large project; I had never dealt with a project on the magnitude of it before. It was hard to find a bug that I though I could easily fix, much less actually fix it. I ultimately settled on an issue with notifications when purchasing gems.

> If you set your language to some of the non-English languages and then go to the Market to buy a gem with gold, you receive two notifications for `+1 gem`, one in your chosen language and then after a short delay, another one in English:



> I've tested a few languages and it happens with only those that have a translation for the `plusOneGem` text that is different than the English version. For example, it happens with German and French, but not with `en_GB`, `en@pirate`, or `zh_TW` all of which still use the default text: `"plusOneGem": "+1 Gem"`

To fix this bug we had to first reproduce it. In the local build for Habitica, there was already a button that would grant the user 10 gems, so I managed to wire it to make a purchase without authentication. The goal was to have the notification fire. Once we reproduced the bug, we needed to locate the place where the notification was being fired twice; it turned out to be in the `userServices.js` file. The way that Habitica determined whether or not a notification had fired was to compare the contents of the notification that should have fired to the contents of the notification that was being created. If they were the same string value, then the program would not create and fire the second notification; if they were different, however, then the second notification would fire. This was a problem only in languages that had translations for the word *Gem* within Habitica. Since, for example, the French word *Gemme* is not the same as the English translation *Gem*, the notification fired again with *Gem*. To fix this, we simply added a boolean that checked whether or not a notification had already been fired. If it had, then we wouldn't fire the second one; otherwise we would.