Edward Kennedy
Dr. Pulimood
CSC 415
18 October 2016

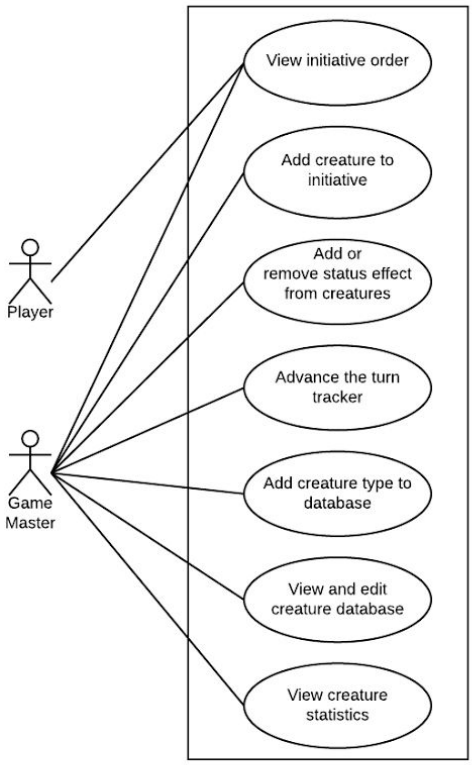
Improved Initiative Tracker

Github repository: https://github.com/agilaz/improved_initiative_tracker

The project will be an Improved Initiative Tracker -- a system to help Game Masters track combat for a tabletop turn-based role playing game (RPG), focused on the game Pathfinder. Programs like this already exist, but where many of these systems fail is in their project scope; developers tend to try to make one app that can do everything a Game Master wants help with, and in doing so end up leaving individual features in less than fully useful states. However, with proper planning and focus, the Improved Initiative Tracker will do its one job and do it well. In addition, several aspects of combat are frequently ignored in such systems. One such aspect that is particularly useful is managing status effects, where one or more creatures are inflicted with statuses (such as blindness) for particular amounts of time. For an individual effect on one creature, tracking such an effect is relatively simple; however, for multiple creatures under various effects for different amounts of time, it can be easy to forget some of the effects and lose track of their durations. So, the system will strive to track not only turn order but status effects and their duration.

A major algorithm that will differentiate this tracker will be the algorithm for managing status effects. At the most basic level, this involves allowing the user to add a status effect, giving it a name and duration, and applying it to various creatures. It would then involve tracking the duration, and removing the effect once it runs out. However, the algorithm can also help with the more complicated aspects of statuses in two ways: handling stat modifications and handling effects that allow saving throws. When a status is applied, it could include extra information about what it does to a creature (for example, decrease their strength by 2 or their movement by 5 feet) and have the system track that. In addition, the algorithm would automatically make saving throws for creatures every turn (a saving throw involves rolling a die, adding a modifier based on the creature's statistics, and seeing if the result is high enough for them to remove the effect from themselves). So, as a whole, the algorithm will run every turn. It will decrease the duration of the effect, and attempt a saving throw for every affected creature. Then, it will remove the effect if it ends; otherwise, it will note the new duration and maintain the stat modifications.

A data structure that will be necessary for this project would be a list of all creatures involved in the combat, sorted by their initiative (which determines turn order). For all the necessary information to be stored, a Creature class needs to be created. This will store various attributes of a creature - mostly their statistics, which are used in determining initiative and the modifiers used for various saving throws.



Above is the use case diagram for the system. Users would be either a Player, who would only be able to see the turn order and nothing else, and a Game Master, who can control all elements of the system. For the scope of this class project, I will focus on the Game Master view and not the Player view. Additionally, I will first work on features essential to the tracker (viewing, adding a creature, adding or removing an effect, and advancing the tracker) and will not extend my work to include a database of creatures unless all other features function as intended, as this feature would be convenient but is not strictly necessary for the system to function well.

This will be implemented as a web app, using Ruby on Rails. It will be written from scratch (option 2), as few existing trackers are free and open source. The project will require various skills that I have yet to fully develop. For one, I have not yet worked with Rails, nor do I have much experience with Ruby, so I will need to learn how to use them. In addition, my skills with writing web apps is very limited; my only experience was a small project using nodejs that was essentially a small modification of someone else's project. So, I will need to learn proper web development techniques (and become more comfortable with the MVC architecture) while also learning Ruby on Rails. I will also, time permitting, create a new database or integrate an existing one (most likely using MySQL or Postgres) that stores all creature types for easy setup so that the user would not have to manually add creatures that already exist.

To learn Ruby on Rails, I plan on using various online tutorials. One option would be a Lynda tutorial: Kevin Skoglund's Ruby on Rails and Ruby training series would both be worth watching. I could also supplement this with other tutorials from sites like Codecademy or Rails for Zombies. I intend to start directly with Skoglund's Ruby on Rails tutorial, hoping that through

exposure to the code in that series I can become more familiar with Ruby itself; however, if I continue to find my lack of understanding of Ruby to be an obstacle, I would break from the Rails tutorial to review the basic Ruby tutorial first. While I work on this tutorial, I would also either follow through other tutorials or try the examples from other tutorials. Given that there are multiple students who intended to work on web apps, I would probably be able to find someone else learning Ruby on Rails with whom I could share resources and practice.

Open source licenses

GNU GPL is a copyleft license. This means that the project can be modified by others, but such modifications will also be under GPL. In addition, the project's source code must be kept publicly available, with any changes logged. While the license allows for others to distribute paid versions of the software, it also requires that the source code and the original source be noted (essentially showing users that they could otherwise get the software for free). The reason for using a license like this is to ensure that future use of the code remains as openly available, so that no one uses it in proprietary software.

The MIT license is a very open license. It states that anyone can copy, modify, or distribute the source code, and they can sublicense it, but the only restriction is that the MIT license also has to be included. Like GPL, code can be copied and modified, but the MIT license does not carry the stipulation of needing it to be strictly under the same license (although the license and copyright notice do still need to be included). This license is very permissive, allowing users to have great control over how they distribute their versions of the code.

Another common license is the Apache license. This is similar to the MIT license in that it allows distribution and modification of the code as long as the original copyright and license are included. However, it has the added conditions that changes to the original code must be noted. In addition, the Apache license also has stipulations regarding patents, granting the original owner patent rights and protecting them against patent litigation. This license is very permissive like the MIT license, but requires a bit more with noting the original program and (unlike the MIT license) mentions patents.

For this project I will use the GNU GPL. This license is well suited for this project because my goal is to make a tool that is freely available, for use and modification, no matter who picks up development on it. I hope to expand upon the tool myself, but if other people work on it I would like their versions to also have publicly available code so that the system can be collaboratively improved as time goes on and not adapted into a proprietary and paid application that ends up going nowhere.