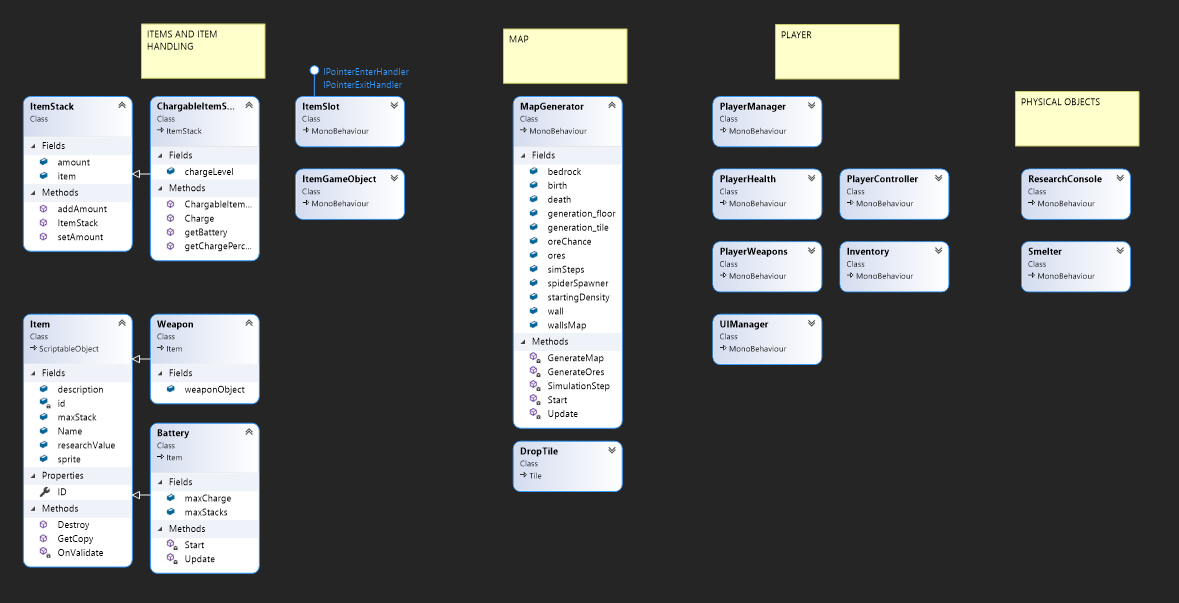
MINING STATION ALPHA POSTMORTEM



* I have only had experience making more complex games by having a million different managers for all the different parts of the game. I do not think this game is different. I split up the player into more manageable classes, but they are still handled by a PlayerManager class. The spiders’ sight is intrinsically tied to the existence of the player, but I think that would be simple to refactor. The GameObject workflow in Unity is both a blessing and a curse. With it I can do independent things like the entire map not relying on anything else besides the generator, but it makes pushing around data weird.
* My design worked out quite well. There is a lot of code with Item handling and different classes, but I think it is okay because it gives more customization on how you want certain things implemented. I know components can be split up more, but I definitely wanted this game playable and kind of fun instead of worrying about splitting components into a million little things.
* My code is…. already reused! I really like the item system with static Items and the ItemStack basically handles the state. It was directly inspired by code from a popular mining and crafting game with blocks (cough). I wrote most of the item code in the summer while working on a personal project where you manage a shop in a fantasy RPG setting, so the complex item handling was already there. I think that part is quite reusable and extensible because I was able to add item charge to it for this project. The map generator is generic too, as it just uses Unity tilemaps, nothing game specific – it can easily be changed to generate different things. I would say around 40% of the code is specific to the game – a lot is the item system and the map generator. Unity specific on the other hand – 80%. The item system is not *exactly* unity specific, as it is based on an item system written in a different language completely, but the implementation of Items are Unity ScriptableObjects. This would be very easy to just replace with static data (like that game) or programmatically load them from a .json file or something. Sadly, the map generator is tuned exactly to the busted and terrible Unity tilemap setup so that is firmly Unity only.
* I think the code is a little rough but maintainable. Things I patched up to finish up the project are easily removed and replaced by something far more complex and better without breaking the entire game. I had to change from a band aid approach to batteries to Inventory handled batteries for the final game and that went smoothly. It would be easy to add more enemies as their code is quite simple, but if they were split more with handling state/movement it would be even easier. Map changes are EXTREMELY easy as you just plop it down on the map. If you want stuff generated you can use the green “GEN PLZ” sprite and it will make cool caves for you.
* The game is PLAYABLE. Complete? No.
  + My scope was wildly ambitious for the class. Even with four weeks I had some crunch (I went home from work early to finish it up). I even had reused parts from previous projects that I wrote, and it was still ridiculously long and stressful.
  + While it is “finished”, a lot of the features I wanted are not really there or implemented poorly.
    - Researching is gutted to what I originally wanted, along with upgrades
    - No crafting
    - No building
    - Enemies are just cannon fodder and don’t work in any meaningfully complex ways
  + If I could do the proposal again, I would try to focus more on directly interactable gameplay such as more enemies or more interesting ores/challenges than worrying about wrangling systems like the item system. I think it was too complex even with four weeks. Something better suited for the project would be something akin to a hyper casual mobile game instead of a deep game.