# Starfleet Mine Clearing Exercise Evaluator

Externalized Thought Process

Dax Garner

October 2016

## 1 First Thoughts

- I read through the entire problem statement and looked at the examples.
- The firing patterns are not immediately obvious to me how the coordinates work. I am looking at the examples to figure them out.
- So the firing patterns are the coordinates relative to the ship from where the torpedoes are sent along the z-axis, and then immediately destroy a mine
- Decided to choose python as the language: easier to get started, test and deploy to customer.

## 2 Getting Started

- Utilizing github for deployment requirement.
- Created repository named mineclear
- Sketched out a procedural and class diagram of initial thoughts in Figure 1
- Starting writing top-level function
- python argparse package will create a basic user-interface

## 3 Implementation

- I built the basic main() that parses and validates user arguments
- I've decided to build the game class, but at the moment think I can handle the actions and field as lists, primarily because a 2D field can be distilled into just holding the z-distance value
- I am debating about whether to have an absolute field and ship location and then the relative field is printed to the screen, or manage a relative field...
- Random thought: for checking whether a field input file is valid, I am thinking minor errors such that the field isn't correctly input as rectilinear could be assuaged by assuming there is just no mines in unspecified areas that would make it rectilinear
- Any character that defines the field that is not alphabetic is considered empty to give the benefit of doubt to the student
- I want to validate the game inputs early for quick failure detection
- I am working on whether it would be faster to maintain a 2D array of the mines, or just a short list of the mines and their location

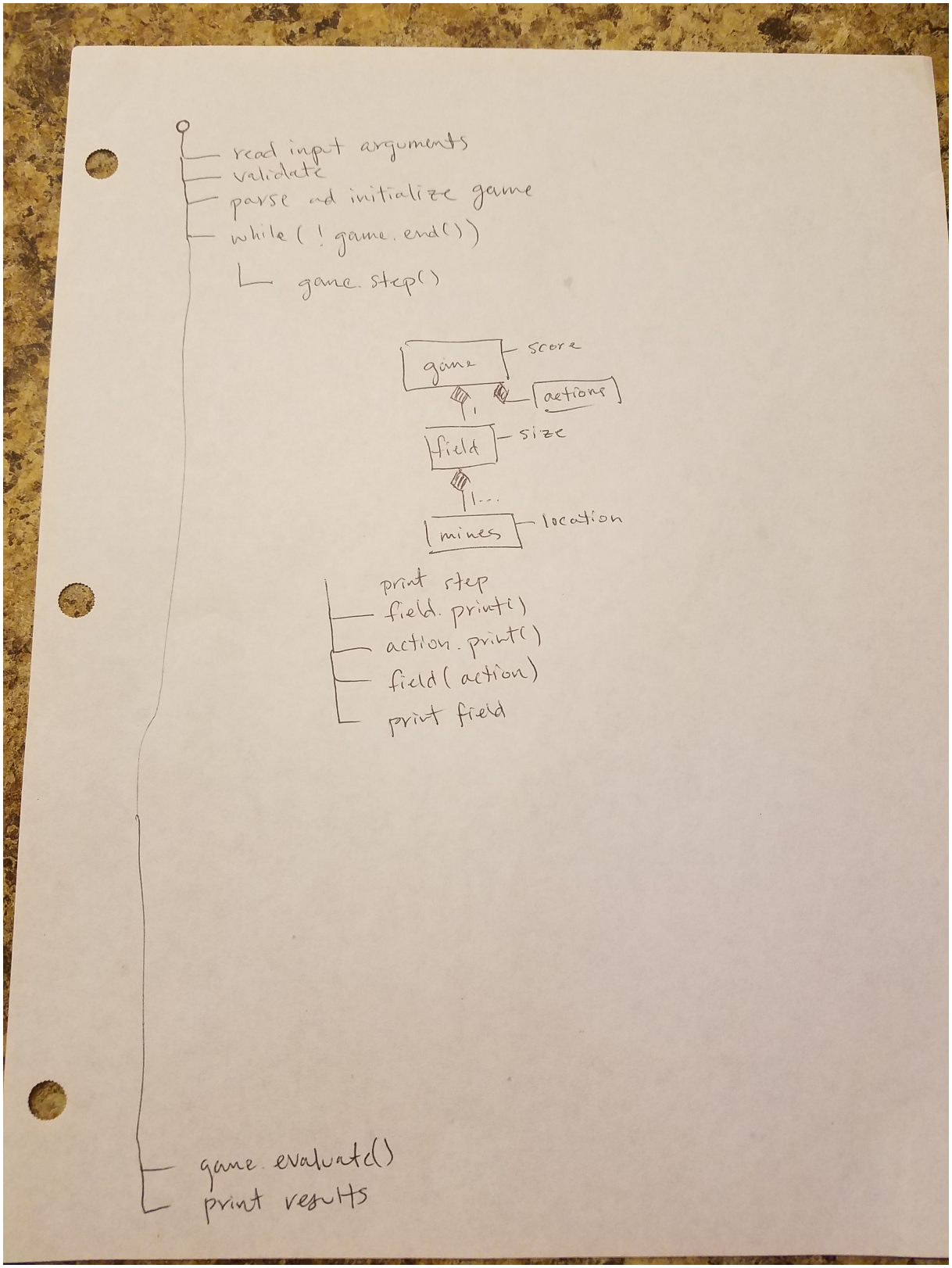


Figure 1: Thoughts from 30,000 ft

- If I just have a short list of their location, then I have to search through all of the each time I want to evaluate against them.
- I decided on a 2D array, and I think managing it as a relative 2D array will make it easier to evaluate interactions with mines
- Maintaining a relative 2D field will also make it faster to evaluate shooting the mines as the indicies are directly computed
- The hardest part about this problem is maitaining a "balanced" field with the ship in the middle, so it can be printed easily
- I am reverting back to maintaining the mines as a list of locations, such that I can quickly compute distances from the ship and print the field based on those calculations
- Ah, the ship should be set at the origin after the field is read in
- Wow, I was making it harder than it needed to be
- Now the calculation for printing the field is straight-forward
- All the operations for moving and shooting are already computed based on the distance from the origin
- Testing some additional input failure cases, and other examples that I think may stress the code
- Found some base case issues of a field initially having no mines
- I had to guess on the best result/score given the inputs if a field has no mines. If you have no mines but do have actions, then you pass with 1 point per the third ending criteria. If you initially have no mines or actions you pass with zero points due to the 4th condition
- If a field of mines comes in such that the rows and columns are an even number, then my program makes an assumption and rounds down. So a 2x2 field would put the origin/ship at (0,0), and a 4x4 field would put the ship initially at (1, 1)
- I didn't see anything else specify what the "middle" would be in these cases

# 4 Retrospect

- I purposely choose to develop in a single file because I figured I'd be the only developer on this single script. Otherwise, I'd be more likely to develop various classes/functionality in different files for ease during collaborative team development
- I also felt the point was more to make a script that was user-friendly, than extensible
- If I was expected to configure the game for various characters, depths, rates or point systems, I wouldn't have hardcoded values in the code and made them more easily configurable