Yang Hu

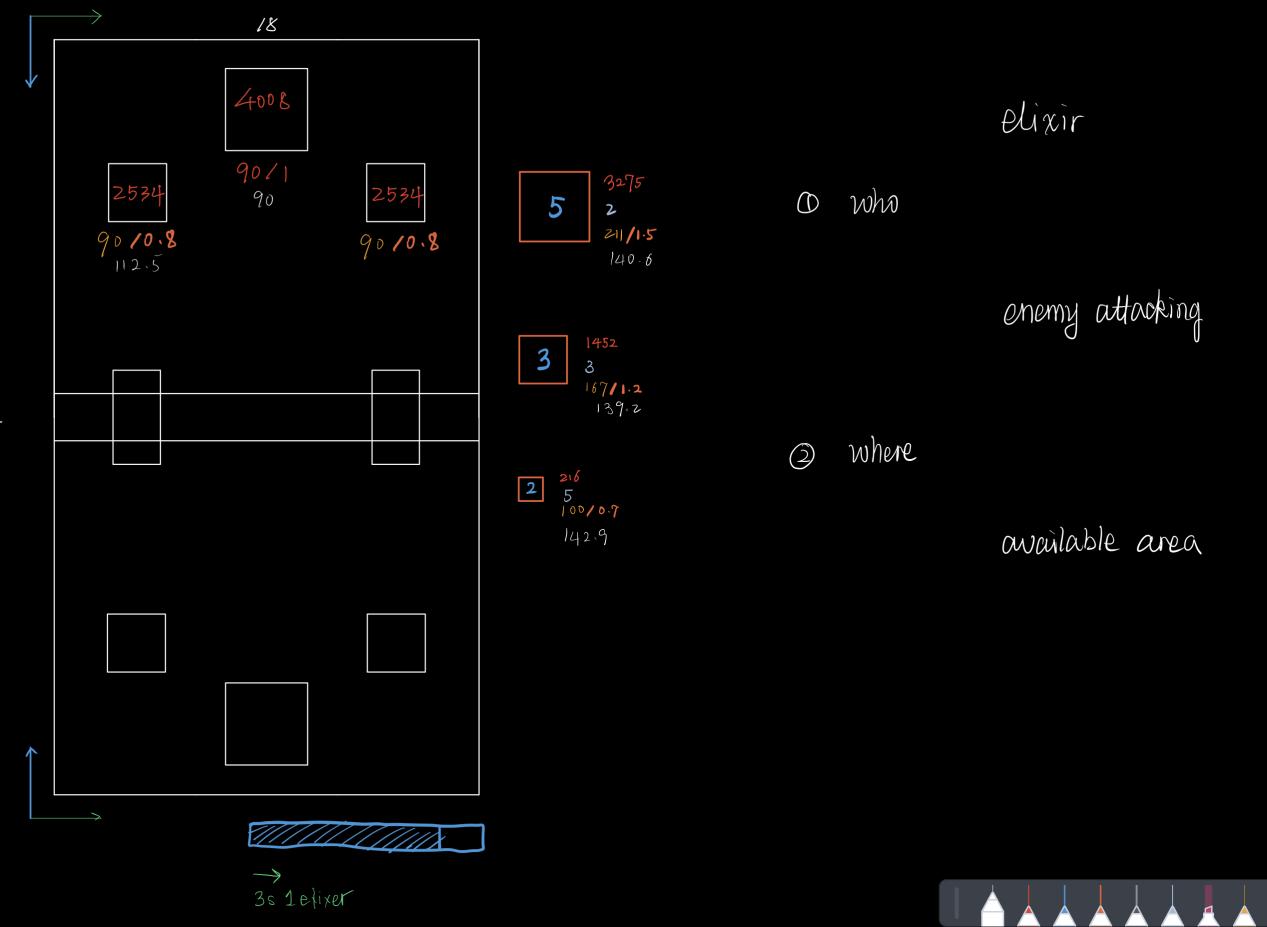
Professor Kevin Dill

CS 4150/5150: Game Artificial Intelligence

March 29, 2021

Game AI Development Note

**Initial Brainstorming:**

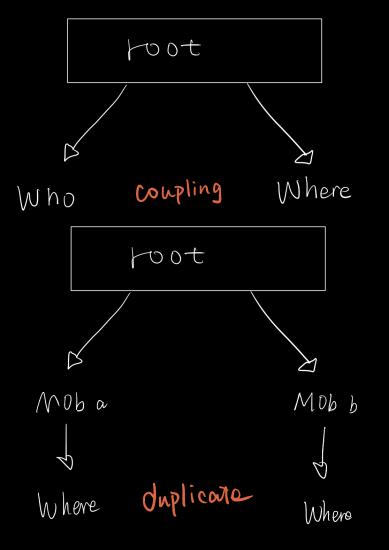


At the brainstorming stage, I listed all the useful information, and listed two questions:

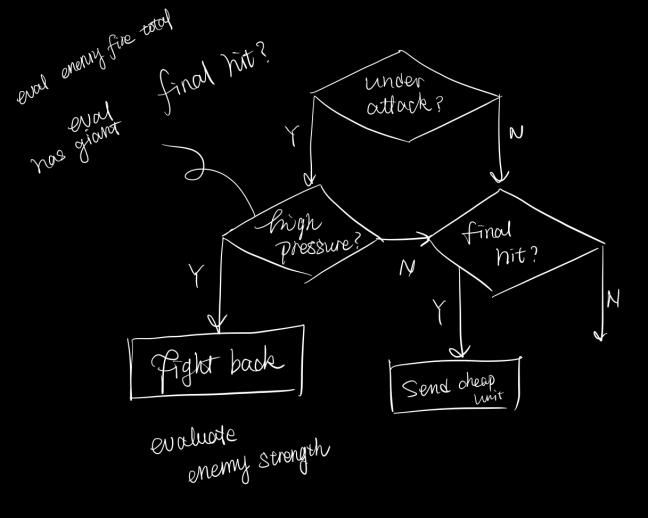
1. What decisions can the AI make?
2. What are the variables that determines those decisions?

And, my initial answer was, the AI needs to decide which mob to place and where to do it. Accordingly, the variables are:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Elixir | Enemy | Available area |
| Who | Elixir availability | Counter certain mob | - |
| Where | - | Place near target mob | Area availability |

Meanwhile, I was also planning to implement a behavior tree to run the AI logic. However, in this case, determining “who” and “where” could conflict with the tree structure.

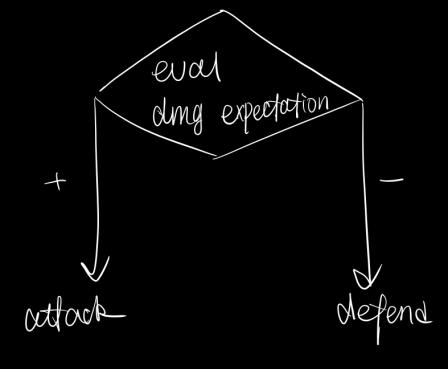
As showed in the graph, putting “who” and “where” at the same level would cause coupling (passing the mob type around) and seems not logical, as they are not parallel choices like “left or right”. Putting them at different level could cause duplicate. Different “where” behaviors are needed as we need different placing strategy due to mobs’ different speeds, while most of the logic are similar.

**Revised Tree Structure:**

After abandoned the first version, I tried to re-structure the tree, staring with binary choices. One of the ideas is to see if the AI is under attack. The following decisions include:

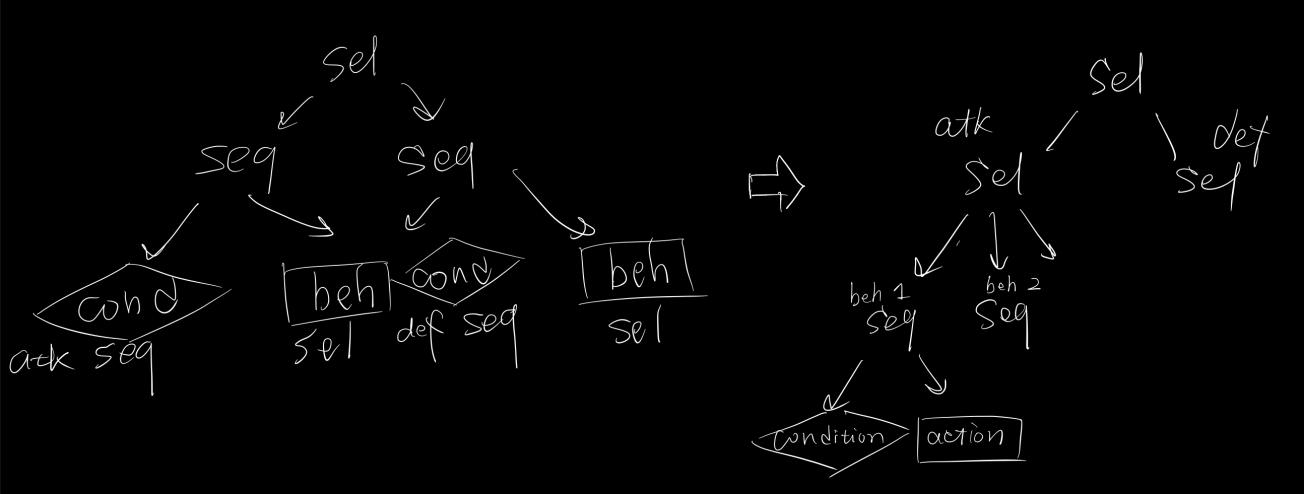
1. If we are under high pressure, try to defend. This involves a strength evaluation;
2. If the enemy is about to lose, send cheaper attacking unit to finish the battle.

However, as in the graph, coupling still happens (The N choice of “high pressure?”). If we are not under high pressure, or we are not doing a final hit, we need to do normal attack under both circumstances. Duplicate behaviors happen.

**Simplify the structure:**

After tried several versions, I realized that the core of playing the Crash Loyal is about **choosing between Attacking and Defending**. An easy formula to make such choice is to simply compare between the expected damage we might take and the expected damage that we could deal with current elixir.

After realized this, I re-designed the behavior tree, starting with a binary choice-should we attack or should we defend? Two of my designs are showed below:

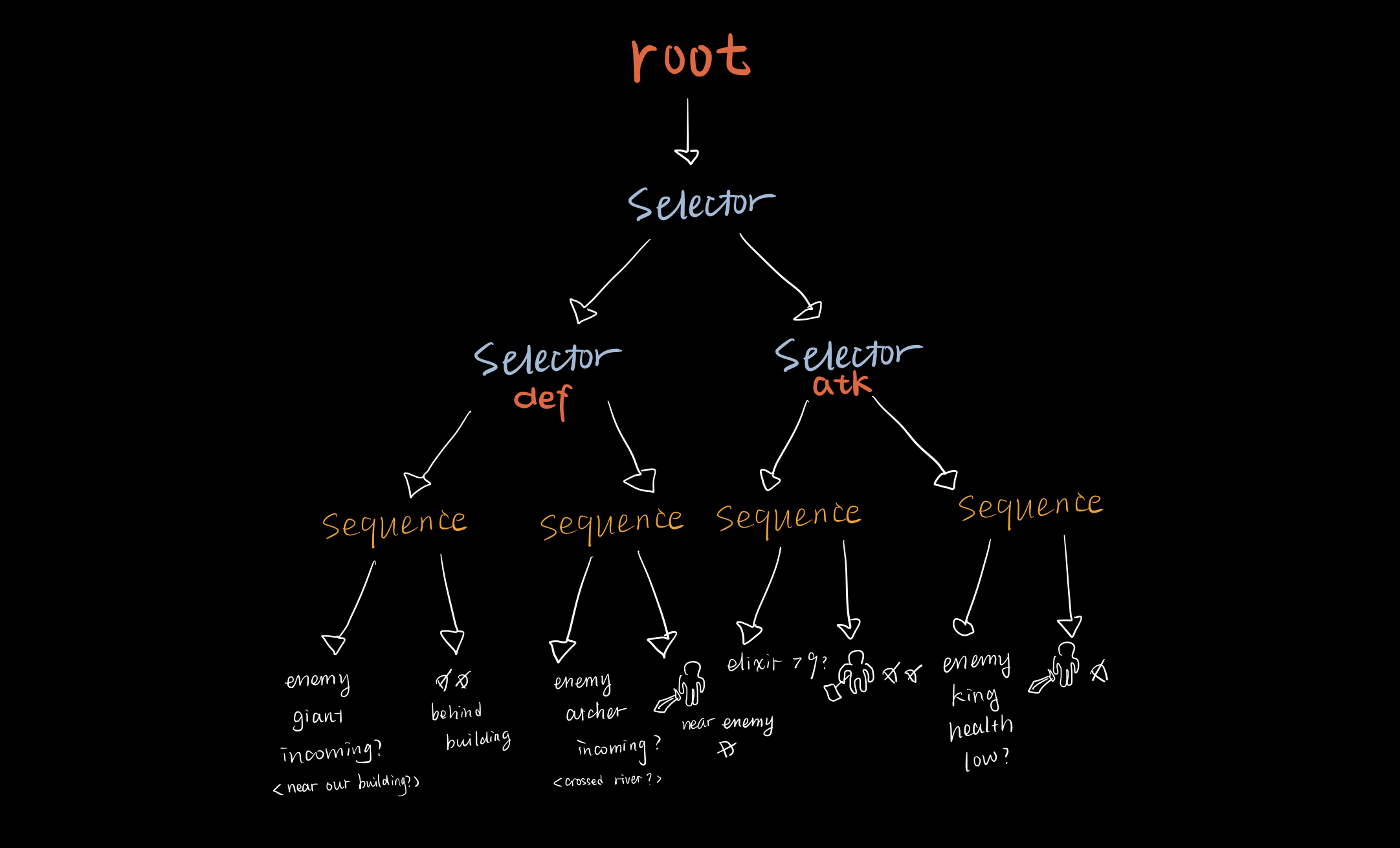


The one on the left is the initial design, where the BHTree select (return SUCCESS when any of the child behavior succeeds) between attack and defend sequence (return SUCCESS when all children succeed). Each sequence contains a condition and an action, which is a typical behavior design. The action would only be made when the condition succeeds.

Then, I added another layer for more pairs of conditions and actions.

**Final version:**

As the AI only decides under which circumstances should it attacks or defends and how, the only work left is adding (condition, action) pairs to respective branch. Here is the final version I created:



I also added a simple personality to the AI (see Controller\_AI\_YH:: buildBHTreeDefensive() and Controller\_AI\_YH:: buildBHTreeAggressive()). The graph above shows a defensive AI, which prioritize defensive actions (defensive branch on the left). Also, the aggressive AI won’t check enemy giant (defend decision 1), and only kill the archers behind. In this way, the AI will sacrifice some building health in order to maximize troop attacking force and hopefully deals more damage on opponent buildings.

**Extensibility:**

The AI structure can be easily extended by adding more (condition, action) pairs under selectors. It is also possible to add more personalities to the AI, by swapping the defending and the attacking selectors or adding some defensive/aggressive behaviors to the front of respective selectors.