Love Almgren
Kirk Easterson
Nan Yang
Github Repo

Feedback Document

Design Choices Made

There are interesting choices in endpoints. ‘endTurn’, ‘attack’, and ‘useHeroPower’ look good, but ‘playMinionCard’ and ‘playSpellCard’ appear to be quite narrow. They take almost identical parameters with the exception of the position in ‘playMinionCard’. It may have been a better design choice to generalize this to ‘playCard’ and then have a ‘:cardType’ parameter. This may add complexity to the ‘playCard’ function but it will make it easier to add new card types (although this is very unlikely to happen since it is a large change in design for the game itself).

Level of Abstraction

There is a strong level of abstraction. In core.clj and construct.clj, none of the functions are character or hero specific (in regards to specific minions or heroes). The implementation of triggered events could be abstracted further though. :on-summon and :on-damage could be abstracted to :ability where the trigger is specified. That way if the implementation is ever to change, it has to be done in less places. This can also be used with battlecries.

Tests of the Engine

There are no missing tests in adapter.cljc. They are quite complete. There are no tests for api.cljc, endpoints.clj, and api.cljc though.

Tests of the Cards

There are tests missing for some of the cards (Ancestral Spirit, Archmage Antonidas, Backstab, Entomb, etc). There are a lot of tests which are good. But some tests yield a false positive. For example, the test for darkscale-healer calls the do-battlecry function. But this shouldn’t have to be called manually every time. Ideally the battlecry would happen automatically. The tests also don’t edge cases into account. But apart from that, the tests look quite complete.

Misplaced Logic

Most of the logic is in card.cljc and core.cljc. The division of responsibilities between these two files is blurred though. For example, there is a get-damage-taken function in card.cljc but there is a get-health function in core.cljc. It would make more sense to have both of these functions in the core.clj file since they involve the core logic of the game.

Modelling of the State

The state is modelled as suggested by Tomas. It is similar to the example below. It was taken from a test for ‘create-game’ in construct.clj. Minions can also have their abilities (battlecry, end turn, rush, etc) in state.

```
{:player-id-in-turn "p2"
      :players   {"p1" {:id      "p1"
      :deck [{:entity-type :card
             :id        "c3"
             :name      "Injured Blademaster"
             :owner-id  "p1"}]
      :hand [{:entity-type :card
             :id        "c4"
             :name      "Silver Hand Recruit"
             :owner-id  "p1"}]
      :minions [{:damage-taken             0
             :attacks-performed-this-turn 0
             :added-to-board-time-id    2
             :entity-type               :minion
             :name                      "Boulderfist Ogre"
             :id                        "m1"
             :attack-gain               0
             :health-gain               0
             :position                  0
             :cannot-attack             nil
             :owner-id                  "p1"}]
      :mana 10
      :fatigue-counter 0
      :graveyard []
      :taunts  []
      :hero {:name          "Uther Lightbringer"
             :id        "h1"
             :entity-type  :hero
             :damage-taken 0
             :hero-power-available true
             :hero-power "Reinforce"
             :attacks-performed-this-turn 0
             :health-gain  0}}
      "p2" {:id       "p2"
```

```
    :deck []
    :hand []


    :minions []
    :mana 10
    :fatigue-counter 0
    :graveyard []
    :taunts  []
    :hero {:name          "Anduin Wrynn"
           :id        "h2"
           :entity-type  :hero
           :damage-taken 0
           :hero-power-available true
           :hero-power "Lesser Heal"
           :attacks-performed-this-turn 0
           :health-gain  0}}}
    :counter                        5
    :minion-ids-summoned-this-turn []
    :action-index                   0
    :event     {:name  "start-of-game"}
    :seed 888}
```

Responsibilities of the Functions

The functions have explicit responsibilities which can be inferred from the function name. But as mentioned previously, some methods are too narrow (get-random-*). This leads to a lot of functions with very narrow responsibility. As suggested, further abstraction can help with future maintainability of the program and less risk of it breaking.