Fluxx Project Proposal

Coen D. Needell
Computational Social Science, University of Chicago
(Dated: November 1, 2019)

Implementing a card game in python to be used as an interface for game-theoretic simulation study. The card game, Fluxx, will be implemented as it exists in its fifth print edition. The basic goal is to define all of the cards and classes and a game engine. There are stretch goals to implement a couple of basic strategies and a system for analyzing those strategies. An additional stretch goal is to use external machine learning tools to implement a simple adversarial system.

I. BACKGROUND

Fluxx is a mildly popular card game.[1][2] It has achieved cult status among table-top game players including myself and a number of my close friends. It has spawned almost 30 variants and licensed versions which implement genuinely different rules. For this project, we'll refer only to the fifth edition of the vanilla variant. The core mechanic of Fluxx is that it has a mutable list of rules. These rule changes can effect everything from how many cards can be played or drawn per turn, what actions are available to the players, they can even have more exotic effects like adding one to all of the numerals (like '1' or '2') in the game. The innovation to the table-top gaming world that this game implemented was this mechanic where the game can alter it's own rules. This is why I think it will be a good python[3] project.

The game is played entirely with decks and hands of cards. The cards are split into four categories, which are color coded. These are keepers, which enter play permanently (until removed) and are owned by a player, new rules, which augment the rules of the game, they enter play permanently (until removed) but are not owned by a player, there are goals, which enter play until another goal is played, and there are actions, which cause an effect and are played directly to discard. The actions are generally exotic, and avoid classification, but new rules can be play rules, which change the number of cards played per turn, draw rules which change the number of cards each player draws per turn, instant effects, which change or add some rule immediately after they're played, limits, which limit the amount of cards in hand or keepers owned by a player while *it is not that player's turn*. In addition, there are free actions, which provide a new option to the player which the player may perform each turn. Finally, there are the goal cards, if a player satisfies the condition written on a goal card (usually having two specific keepers, but there are three exotic goals), then that player wins the game immediately.

II. IMPLEMENTATION

As of right now, I have a basic conception of how this game should be implemented. Most games are implemented in three parts, the assets, the objects, and the engine. Since this implementation will be text based, the assets will just be text, and perhaps some basic images. The objects consists of the actual mechanics, cards, and rules of the game. The engine consists of a driver which cycles through turns, and does all of the background calculations needed to actually play the game. In short, the assets are essentially artistic flourishes, to help the player understand the game, the objects define the rules and mechanics of the game, and the engine helps the player interact with the objects. I intend to implement these as two modules, since the assets are just text, they can be folded in with the objects module. As for external packages, the only one I can see myself using is numba[4] for it's Just In Time compilation feature, which is designed to drastically speed up simulations.

A. Objects

I intend to implement a base class for the cards, which has method that the engine can use to run the game from an abstract level. Below that there will be classes for each major type of card, and I will define sub-classes below that for other groupings of cards as I see fit.

These objects will need methods which can be activated by the engine and attributes which tell other objects the game state.

B. Engine

I intend to implement the engine using a few basic classes to store the game state and the game record. Then a few functions which move the turns along. Then there will be a controller function which allows a player to interact with these functions. The turn-moving functions will probably be methods of the game-state object.

III. STRETCH GOALS

I have a few stretch goals for this project, which either will be started by the time the project is presentable for class, or I will add to as a follow up to the project after the class is over. The first stretch goal is to create a

module containing strategies for the game, like optimizing for hand size, optimizing for keeper accumulation, or optimizing for making it difficult for the other players to act freely.

The second stretch goal is to use an external machine learning tool to analyze the game. This will likely not be finished for this class.

---

[1] K. L. Andrew Looney, *Fluxx* (2014), URL `https://www.looneylabs.com/games/fluxx`.

[2] *Fluxx-board game geek*, URL `https://boardgamegeek.com/boardgame/258/fluxx`.

[3] G. van Rossum, *Python tutorial* (1995).

[4] S. K. Lam, A. Pitrou, and S. Seibert, *Numba: A llvm-based python jit compiler* (2015), URL `http://doi.acm.org/10.1145/2833157.2833162`.