Motivation

Our motivation for this project was to make an ai that could help players of the Pokemon trading card game learn to play better. Many players do not have access to players of equal or greater skill to practice against. Our project allows a player to enter a game state and have the program tell them what it determines to be the optimal game move. This will allow the player to practice on their own or analyze previous games they have played afterwards.

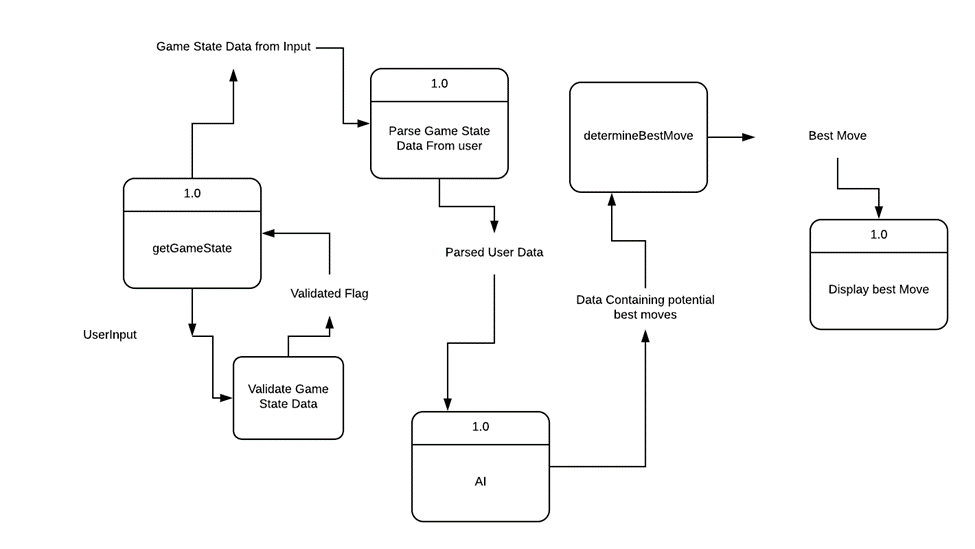
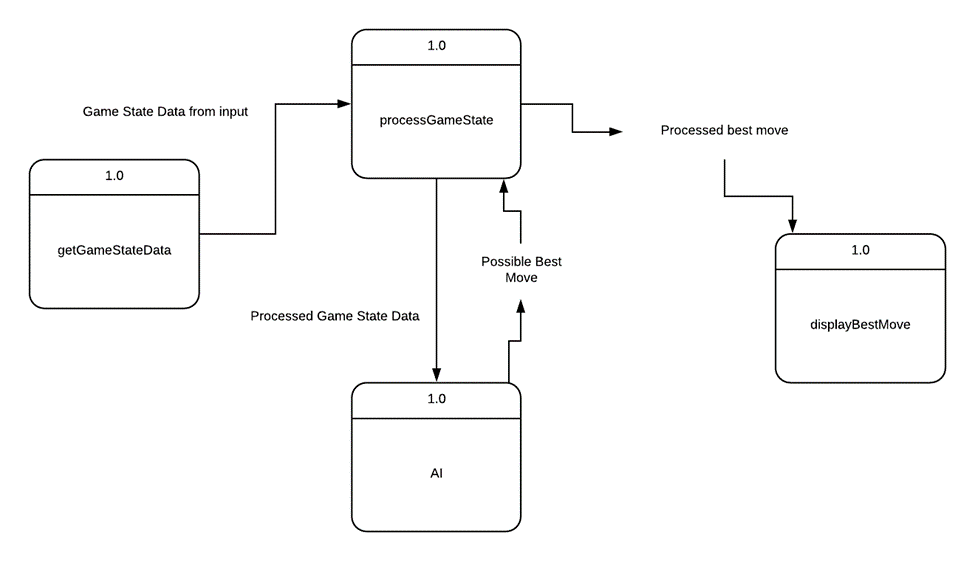
Comparison to Similar projects

A project called open-MTG we found is similar in that it uses the Montey Carlo tree search algorithm to play a trading card game. While our project analyses the Pokemon trading card game, Open-MTG has two ai controlled players play Magic: the gathering against each other. While our project is intended to help players learn to play the game better, Open-MTG sought to solve the “meta-game”. The meta-game in competitive trading card games is determining what the best decks are.

Community and Social Implications

The impact we hope to have on the Pokemon trading card game community would be to help players improve at playing the game. We believe players new to the competitive scene would find the project most helpful. Since it is a game of skill and variance we hope it will help players to understand that making the best game actions does not guarantee success. This is also true of life. This is one of the most important lessons we believe can be learned through playing games. Having involvement in any person’s journey to understandoing and internalizing this concept will have us consider the social impact the project has both successful and positive.

Project block diagram



Project implementation details

Difference from design document

There are a few differences from the design document. We decided not to use a database like mySQL or MongoDB to hold card information. Instead it was coded into the python files themselves do to the ease of access to the information. This allowed us to make changes to the Card class easily and quickly. With the limited cards available in this version a separate database would have been overkill. Another difference is the ability to create and choose different decks. This came down to not having enough time to create a parser to build the attack functions. This meant we had to code each attack with its special actions for each card we included. In a future version we would continue adding cards to the card pool and implement a “deck management” feature.

Challenges during implementation

We struggled with the game engine itself much more than we believed we would. One error we saw more than we care to remember was getting index out of bounds errors when moving card objects from one game location to another. We finally found a fix close to the due date and implemented it where we were still getting errors. Working as a team is always a challenge. This was mostly due to communication issues.

Use of software engineering principles

We used object oriented programing principles to realize the game engine. We created classes for the gameboard and the cards. The attacks and items were separated from the game engine and imported to make the code more readable and easier to update with new attacks and items. The Montey Carlo tree search algorithm uses a tree structure of nodes. Each node represents a gamestate and holds data on the usefulness of the game action that got it there. It is a depth first search with a threshold to keep the branching factor from getting out of control. This scaling allows it to be increased on more powerful hardware getting more accurate results or decreased on less powerful hardware to allow it to be used at all.

User’s manual (usage, etc)

Thank you for your interest in Project Vulpix!

This manual will guide you through running and operating the program.

First you will need to make sure you have python 3 installed.

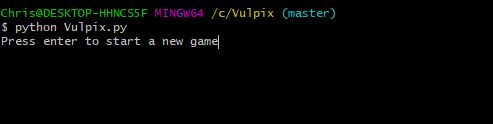
To do this open your command prompt or bash shell.

Simply type ‘python’ (no quotes!) and press enter. If python is installed you will see some text providing you with the version of python you have installed. If it provides something like ‘Python 3.X.X …’ you are all set and can move on to running the program. If you get text like ‘Python **2**.X.X …’ you have Python 2 installed. You can then try entering ‘python3’. If this is successful it means that you have both Python 2 and Python 3 installed on your system. When this guide tells you to type ‘python’ into the prompt type ‘python3’ instead. If neither of those were successful you will need to install Python 3. It is available to download at no cost from ‘www.python.org’. Go to the downloads tab and select your operating system. Follow the instructions on the website to install Python 3. Follow the above steps to make sure your installation has been installed correctly.

To run the program you need to locate the directory the Vulpix.py file is contained within. There also needs to be the files mcts.py and GameManager.py.

Once you are in the proper directory type ‘python Vulpix.py’ and press enter. For those of you who had both Python 2 and Python 3 you will type ‘python3 Vulpix.py’.

You should get a prompt something like the following picture.

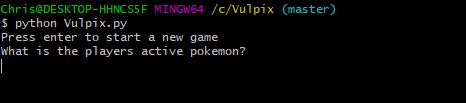


You will follow the prompts on screen to start a game and enter the gamestate.

Project Vulpix is intended to be used with either physical Pokemon cards or with a representation of a gamestate for the user to enter and the program to analyze.

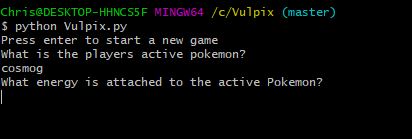
Here we will go through entering the gamestate one step at a time.

The first prompt you will see will be asking you to enter the player’s active Pokemon.

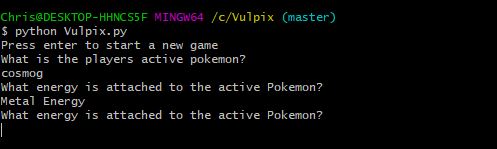


For this prompt you simply enter the name of the Pokemon in the player’s active Pokemon slot. If the Pokemon is a stage 2 or stage 3 Pokemon the Pokemon that they evolve from will be accounted for automatically. For example, Solgaleo is a stage 2 Pokemon that evolves from the stage 1 Pokemon Cosmoem, who in turn evolves from the basic Pokemon Cosmog. If Solgaleo is entered cosmoem and cosmog will be taken from the deck and attached to solgaleo appropriately. (See the Appendix: Technical Glossary for more information) For purposes of this guide we will enter ‘Cosmog’.

The next prompt you will see is asking what energy cards are attached to the active Pokemon.

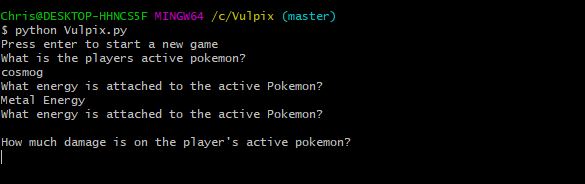


If there are energy cards attached to the active Pokemon you enter the name of it here. For this example we will enter ‘Metal Energy’.



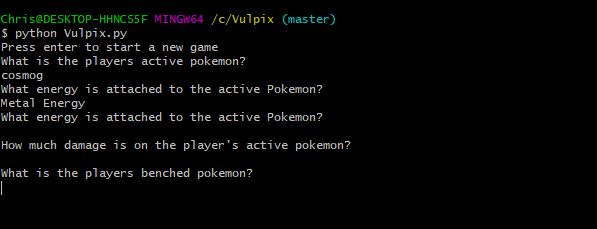
Notice here that the prompt is repeated. This is so you can enter all of the Energy Cards that are attached to the Pokemon on e at a time. Once you have entered all energy cards attached simply press enter to continue.

The next prompt will ask for the amount of damage the Pokemon has taken.



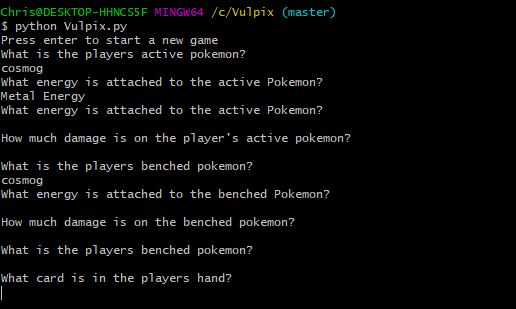
You may enter a number here or if there is no damage on the Pokemon you can just press enter.

The program will now prompt you for the Pokemon on the player’s bench.



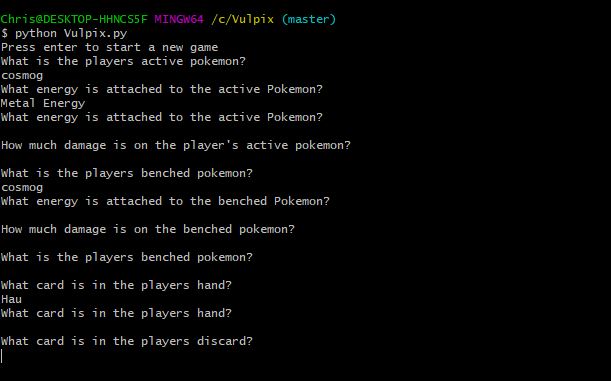
This will follow the same process as entering the details for the active Pokemon. The only difference is that it will repeat this same prompt again until you enter no value. This will tell the program you have finished entering Pokemon from the bench.

This next prompt will ask you to enter the cards in the player’s hand.



Similarly to how we entered Energy Cards attached to the Pokemon, we will enter the cards in the Player’s hand one at a time. Once we have entered all of the cards press enter without entering anything to move on.

The next prompt asks for the cards in the player’s discard pile. The steps are the same as entering the player’s hand.



Next you will be prompted for the stadium card in play. The stadium card can be played by either player so it will prompt for the player who owns the stadium card as well. Enter ‘p’ for player or ‘o’ for opponent.

