Design Document 3: Pokemon Server Query Program
by: Gabriel Racz 101181470

Intro

The functionality of this iteration of the query program remains the same as the first specifications with the exception that the work is offloaded to a server. This is useful as it allows for all of the heavy computation to be done on the server where each of the client only recieves the results without much overhead. It also has the benefit of removing the need to download a specific .csv file to access the Pokemon data. Finally the server searching algorithms and database can be easily altered without having to push client updates because the client only works with Pokemon records in the form of strings.

Design

For this application, the TCP communication protocol is better suited than UDP. It is important for the server to communicate reliably with the client as well as delivering the search results in the order they are found. With UDP, data transmission is fast but not as reliable. Packets of data arrive in an arbitrary order, and sometimes not at all. The TCP also ensures that not only is the server only sending data to one client at a time, but clients can queue search requests in a line-up to be handled in the order that the requests are received.

The server's purpose is to connect to a client socket, receive a type request, and search the Pokemon database for Pokemon with matching type1. The server reads the .csv file and parses each line, storing the data as PokemonType structs in memory. This is useful in the case of maintainability as new features that search, sort, and manipulate the Pokemon in different ways benefit from the data organized in structs. The client, however, is only interested in the Pokemon records that match the requested type. To minimize the complexity and memory that needs to be sent over the network, only a string representing the Pokemon's data is sent to the client. Once all of the matches have been sent, the server closes the connection to that client socket and begins listening for a new requests.

The client receives this string from the server and saves it to a masterList containing all of the previous query results. To write the Pokemon results to a file, the Pokemon records are printed to a specified output file.

To allow for the menu to always be responsive, multi threading is used in both the search and saving processes. In both cases, a thread array is used to store a number of thread pointers. When a search or save request is made, a thread at the current index in the array is created and sent to the function. When more than the max amount of threads have been passed, the oldest thread is joined with the main and a new one is created in its place. For searching, each thread is its own client when talking to the server. This allows for the queuing of searches because the server accepts a lineup of clients and handles them in order. This is where TCP shines because it ensures all of the correct results are delivered to the right client instead of having a connection less stream of data coming out.

More information on the Pokemon Query functions can be found in the previous design document.