Abstract

Colton Pulliam, Chase Saba, Caleb Neumann, Yo'an Sanchez

The goal of this project is to create a runner code and two algorithms (per person) that are all compatible in competing with one another in Rock, Paper, Scissors. The primary processes we are trying to include are initialization of the game(rules), number of rounds, the wins/losses/ties and how these algorithms will adapt to their results, and the overall results after the game is over.

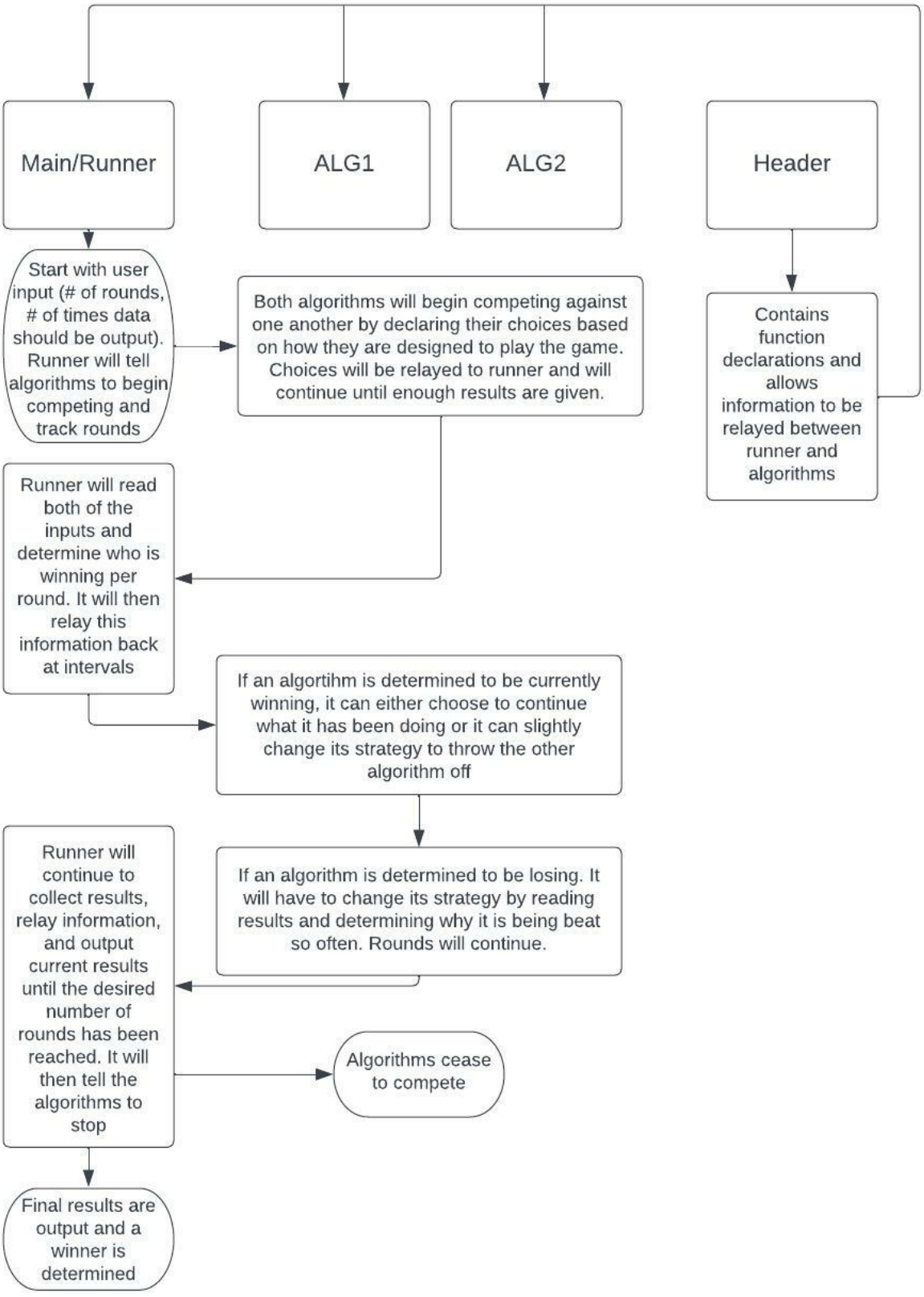
These processes can be broken up between the runner code and algorithms, with all of the declarations being in the header file so that they can all share information. The runner code would contain the initialization of the number of rounds and would keep track of the algorithm's wins, losses, and ties as the game progresses. The main function will also output the results as many times as desired during the game and will output final results after the game. The algorithms will contain the processes such as game rules and analysis of their results so that their playstyles will adapt to their opponents.

Data required for the runner mostly involves integers to keep track of the number of rounds, times outputting, and results. It is worth noting that most results will be outputted as a percentage rather than just a number. The algorithms will likely run using both data from strings and integers in order to read its opponents choices and determine what it must change based on its numerical results.

Almost all functionality will occur with respect to other components. The runner will immediately give information to both of the algorithms when beginning the program and will collect information from the algorithms for output. The competing algorithms will almost always be relaying information between one another and the runner code so that the game can be tracked and the algorithms can learn.

Any competing algorithm will need to gather data from our components. This means we need to make a universal runner that can make almost any two algorithms compete against one another as long as the data required for both is similar.

Visual Representation: Functionality



*idea is that each section is its own .h/.cpp file

Visual Representation: Data Storage