Vector Domination Design Document

Zane Emerick

This is a very complicated game that will likely involve many methods. It will require a system to create the board itself, a system to draw the moves from a vector line into a coordinate system, a system to determine if a move is valid or not, a system to take input files; and many more.

To start simple and get a foundation, the board methods should be created first. This would simply create a new board object with the options to return the contents of a given space and print the board itself. This would require the use of f-strings, or something similar, to get the spacing correct. The board needs to be of size (N x N) and would be read in from a file so a system for file reading needs to be implemented as well in the main method.

Next a set of methods to accept input and validate those inputs would be required. The system would first need to read an input from the file, access the game board at that location to test if the spaces were already taken, as the only open spaces need to be the start and end. After that, a way to test the lines themselves for validity needs to be created. A vector object with a slope variable would need to be created for each input and saved to an array that could then be iterated over to check for validity. The vector object would be created as follows:

Text

Description automatically generated

To check the vector for validity, a function similar to the following would need to be created that could iterate over every vector played and compare the slopes. The vectors array would also be updated to be shortened to include only the last K rounds per the input file. As stated in the rules, the game ends when both players have made an invalid play so this method would also be used in a counter for each player to track when the game would end.

Text

Description automatically generated

A system would also need to be crated to represent the lines on the actual board and flip the necessary boxes to the player’s icon. This would likely involve some way to map the line to the board, and using a simple method to flip a box if it lines up. All of the calculations would be done in one method, and the second would simply be called when a box needs flipped.

Diagram

Description automatically generated

To accomplish this, a system would need created that can take the slope of the line in a coordinate system, turn it into a ratio, and then add it to the starting point to get a set of locations corresponding to the table. The flip method could then be called on every box that the vector passes through to create the second image.

Finally, the method to take the input would need to be able to accept a line of inputs and split that into the separate coordinate systems, before returning a vector. As the file would be opened in the method, a counter outside would be needed as well to track line numbers. This would likely look similar to the following:

Text

Description automatically generated

For time management, creating the board and the vector objects would take the least amount of time so those would be completed first, followed by the system to read the input and then the system to compute the math and display the actual game each turn. Finally, the loop to run the game itself would be implemented last once the rest of the pieces are completed and the project is nearing completion.

The program would be tested using multiple input files created by the user, as well as using the main loop of the game to accept inputs. Major sources of difficulty that will likely need the most testing include the portion using math to place the icons along the lone in the grid system. It will be difficult to map the line exactly to a grid system, and compute which boxes the line passes through. After that system is completed, however, the rest of the project should require minimal testing.