Chris Cordaro

Project Milestone

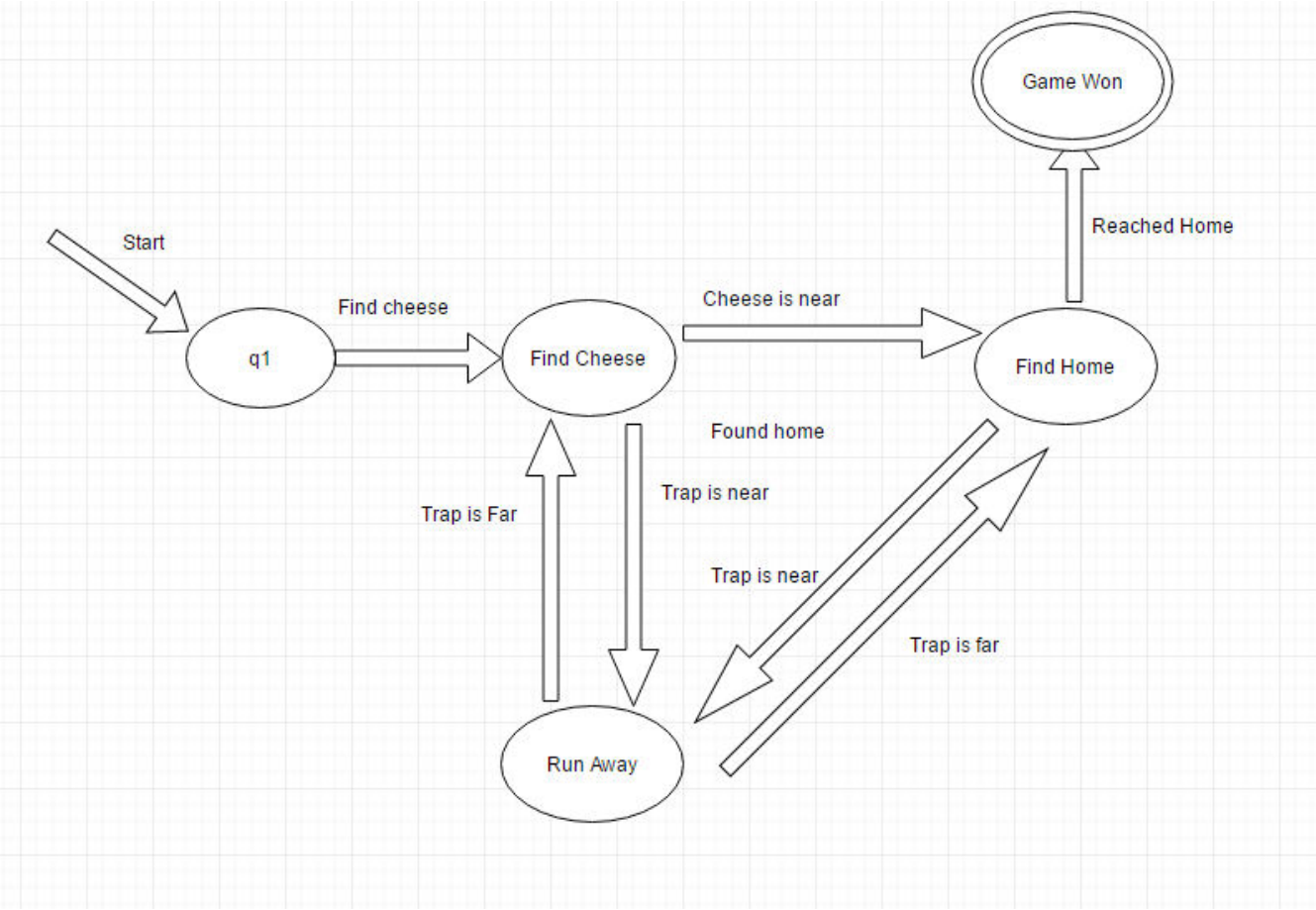
Cmpt440

Abstract: This project is a game simulation that implements a DFA in order to create a simple game AI. In the game world there is a mouse who wanders the map in search for his piece of cheese, and upon finding it, must return back to his mouse-hole safely while avoiding mousetraps that are placed around the map. This paper will demonstrate how the implementation of a DFA can create a simple AI that can make basic decisions while moving across the multiple states of the DFA.

Introduction: The motivation behind this particular project was to explore basic game development through the use of a DFA. I have been interested in the create aspect of programming and have not really had a proper opportunity to explore it throughout my time at Marist. Creating such a game will allow me to gain technical programming skills in regards to creating a DFA, while also allowing me to explore new aspects of software development. The DFA implemented will be relatively simple, as it mostly handles the AI movement, along with some basic user interaction, such as making sure the user enters in proper values in a variety of fields. If CPU speed and technical knowledge allows it, the user will be able to decide how many traps, mice, and cheese are placed on the game map in order to demonstrate how multiple DFAs can run at a time and complete the same task. Implementing this functionality will expand the DFA in regards to user interaction, as proper values must be inputted in these fields.

Detailed System Description: This system uses a DFA in the game loop in order to complete the task of the mouse finding the cheese and successfully bringing it back to the start point. The implementation of the DFA will be through a function list, or tree. Users will be able to choose how many mouse traps

will be present on the game map in order to adjust the difficulty of the game. Essentially, the AI doesn't 'know' how to avoid the mousetraps, as that would mean the mouse would successfully complete his task every time, however, it may be interesting to implement a "smart mode" where the mouse will be able to detect and avoid the traps. The time of completion can be tracked and a simple model could be shown that displays the relationship between the amount of traps, and the time of competition; such a model would be a way to represent the efficiency of the DFA along with the AI functions. The basic AI DFA looks like this:



The mouse will start at Q1 and attempt to find the cheese by wandering the map. While he is attempting to find the cheese, if he recognizes that a trap is near he will enter the Run Away state, once he has gone far enough away from the trap he will begin his search for the elusive cheese. When he finds it he will enter the Find Home state. While searching for home if he recognizes a trap he will have

to enter the Run Away state again, however, now there are two “Trap is Near” and “Trap is Far” transitions for the Run Away state, but this can be mitigated by a if statement. If the mouse has picked up the cheese, transition on run away towards find home, else, transition on run away towards find cheese. Once the mouse has found the cheese, the game has ended and enters an accepted state. The user input DFA is simply a text validation through the use of regular expressions.

Requirements: The requirements to run this system is a PC that has java installed. Even with the addition of multiple mice, a standard processor should be able to handle the CPU load.

User Manual: Users will be able to load the java file and have the game open up to a title screen that will lay out the rules of the game and what it is hoping to accomplish. The user will press the ‘start’ button and the game will begin. The user will be able to press ‘p’ in order to pause the game, presenting the user with the options to restart and change the rules and number of enemies, mice, and cheese, present on the game map.

Conclusion: This project demonstrates the ability for a state machine to implement a relatively simple AI system. Through a graph/tree implementation of a DFA I am able to visualize a the larger picture of the game states and more easily change them and optimize them to my liking. I am curious to research how large games from real game studios implement their AI, and if it is any similar to a DFA implementation or something more complex and sophisticated. It’s interesting to demonstrate how a relatively simple DFA can be used in order to create a game AI that has basic decision making capabilities. Although the DFA above may not be strictly a DFA due to the issue laid out above in the Run Away stage, the issue is mitigated through a “cheese check” which allows for a pseudo DFA classification. Through the implementation of a DFA this project allows me to explore both the technical aspect of DFA creation along with the creative aspect of game design.