DIJKSTRA'S ALGORITHM FOR SUPER MARIO

COEN 4650 - Dr. Henry Mederiros

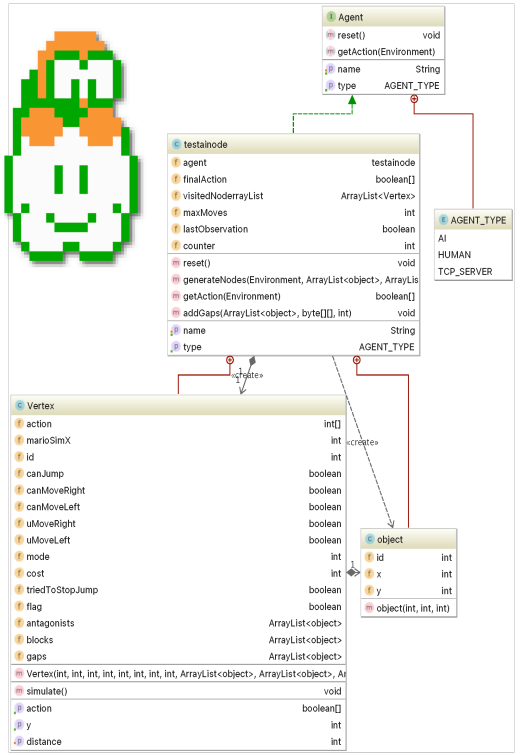
Yungwoo Ko David Helminiak Drew Vanderwiel

OPUS College of Engineering
MARQUETTE UNIVERSITY

INTRODUCTION

The goal of this project was to develop an algorithm based on Dijkstra's to play a clone of a Super Mario game. This implementation also has applications for real world problems, such as self driving vehicles and GPS navigation. Objectively, the final program should be able to plan its way around basic obstacles using Dijkstra's algorithm, make compromises in pursuit of its goals and complete levels at levels similar to, or surpassing human players, as well as other artificial agents.

PROGRAM STRUCTURE



NODE GENERATION AND SEARCH

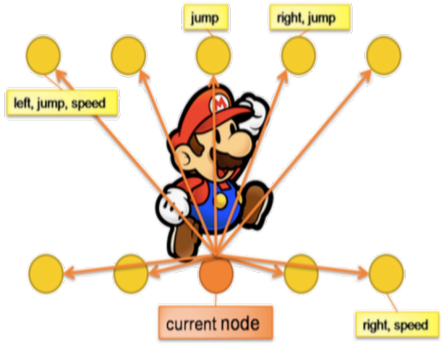
The algorithm is implemented using a 3rd party framework, which includes a Super Mario game and tools to evaluate AI agents. The framework provides Mario's current state and the environment to the active agent once every frame. The environment is comprised of various objects, each with unique identification IDs that may be interpreted by the retrievable from the framework. During each frame the program generates a graph containing all of the possible actions Mario may make. The nodes then contain a cost based on their propensity to either cause harm to Mario, or reach the end of the level. Children containing all subsequent possibilities are then continually generated. In order for an action to be taken in real time, a timer is used to stop generation after 20 ms and return an action. However, in practice this generates far too many nodes for system memory, so an additional limitation, a maximum number of selected actions is also in place. Dijkstra's is then applied to the graph in order to determine Mario's optimal path.

ALGORITHM SELECTION

The original plan for this project was the implementation of the popular and optimized path finding algorithm A*. The predecessor to this was Dijkstra's algorithm, which provides a simple, yet powerful solution for finding the shortest path between nodes in the graph.

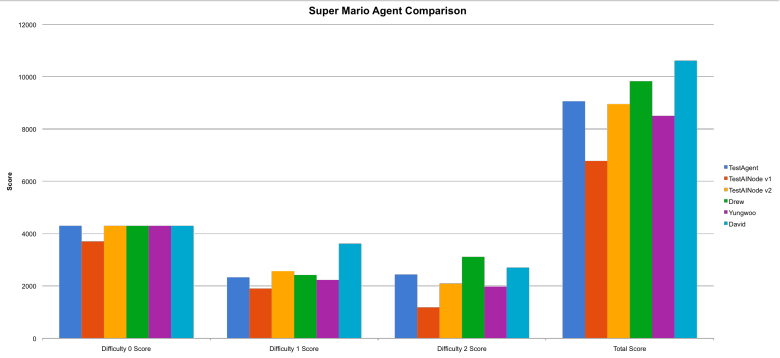
ALGORITHM DESCRIPTION AND ANALYSIS

Dijkstra's shortest path algorithm starts from the source node. From that point, it calculates the shortest path to all other nodes. All vertexes are initialized with an infinite distance from the source, and this distance is updated as the algorithm progresses. The worst case running time of this algorithm is O(E + VlogV), where E is the number of edges, and V is the number vertices in the graph.



J. Togelius, S. Karakovskiy and R. Baumgarten, "The 2009 Mario AI Competition," *IEEE Congress on Evolutionary Computation, Barcelona, 2010, pp. 1-8.*

DATA COLLECTION AND RESULTS



The testing methodology comprised the average of 5 randomized Mario levels at 3 levels of difficulty. Seen in the performance evaluation, the original artificial agent performed below the average human scores. However, updating the costs assigned to environmental factors greatly improved its performance, to the degree that it matched and even slightly surpassed some human agents in levels of difficulty 0 and 1.

Table 1: Super Mario Agent Comparison

Agent	Difficulty 0 Score	Difficulty 1 Score	Difficulty 2 Score	Total Score
TestAgent	4297.6	2326.662	2435.0844	9059.3464
TestAINode v1	3699.8438	1898.9137	1179.0117	6777.7692
TestAINode v2	4297.6	2565.1111	2089.6215	8952.3326
Drew	4297.6	2417.315	3111.1518	9826.0668
Yungwoo	4297.6	2227.7553	1977.1199	8502.4752
David	4297.6	3614.7027	2699.6548	10611.9575

CONCLUSIONS

Dijkstra's shortest path algorithm was successfully implemented to control an agent, so the primary objective of this project was complete. However, there are still some unsolved problems when Mario encounters enemies or pits. We weren't able to consider all possible situations in the game, nor anticipate all combinations of environmental conditions. Nevertheless, Mario reaches the destination point reasonably well when the difficulty level is 0 or 1. Overall, the original and updated AI agents didn't perform as well as the human players, yet Dijkstra's algorithm was able to choose optimal moves based on the provided information, within the allotted time frame.