**Computational Modeling of Intelligent Behavior**

**by**

**Lindsey Brianna Harris**

**An Honors Capstone**

**submitted in partial fulfillment of the requirements**

**for the Honors Diploma**

**to**

**The Honors College**

**of**

**The University of Alabama in Huntsville**

**May 18, 2015**

**Honors Capstone Director: Dr. Daniel Rochowiak**

**Associate Dean, College of Science**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Director Date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Department Chair Date**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Honors College Dean Date**

**Table of Contents**

Dedication……………………………………………………………..………………….. 2

Abstract……………………………………………………………….…………………… 3

Introduction……………………………………………………………………………….. 4

Project Description……………………………………………………………………….. x

Iteration 0: Preliminary Structures……….…………………………………………….. x

Iteration 1: The Simple Case..………………………………………………………….. x

Iteration 2: One-Step Memory………....…..……..…………………………………..… x

Iteration 3: Story Generator Application……………………………………………….. x

Demo………………………………………………………………………………………. x

Conclusion………………………………………..………………………………………. x

Bibliography…………………….…………………...……………………………………. x

**Dedication:**

This Capstone Project is dedicated to my mother, Angela Harris, who inspired me to become a Computer Science major and to explore programming and other such computer science skills.

I dedicated this also to the love of my life, Matthew Pittenger, who inspires me to work hard and solve problems on a regular basis. Without whom, my problem-solving skills would not be nearly as cultivated. Thank you for solving so many challenges alongside me.

Lastly, I dedicate this to my fellow Computer Science majors, who hopefully will find this matter to be useful and/or inspirational as they continue in their studies.

**Abstract** //To Be Redone

The problem of modeling intelligent behavior deals with varied probabilities of actions as well as patterns of action distribution in behavioral trees. It is simple to model a perfectly rational entity, which makes the rationally correct choice all of the time. However, as humans do not live perfectly rational lives, a perfectly rational model is not very realistic. Modeling human-like behavior accurately, therefore, must also take into account the irrational things that humans do that still seem to make sense. Some of the “irrationalities” are based on social norms, while others are based on habits. Utilizing a number of iterative steps, the branch of action that the model takes could change based on monitoring that occurs during the previous steps that the model has taken. With that in mind, the purpose of this project is to research the probabilities and behavior structures necessary to produce a reusable software component that realistically models human-like, artificial intelligence.

**Introduction**

//To be done after the rest of the thesis is written (below)

**Project Description**

The purpose of this project is to explore the usability of a probability tree structure for behavioral AI and to demonstrate the structure’s effectiveness in an artificial intelligence setting. This will be accomplished through a series of iterative program designs that will build on each other and will eventually result in a finalized proof of concept program that will demonstrate the probability tree structure’s usability. As a proof of concept, I will design and implement a random story generator program, which will use the probability tree structure I design to select options from a given story line. The iterations of this project will include a preliminary overview and exploration of the structures to be used (iteration 0), an implementation of the “simple case”, which will be a time-independent implementation that will select options from the same choice repeatedly in order to test the basic structure (iteration 1), and an implementation which will include a one-step memory (iteration 2). Finally, I will create a demo story tree that can be used by the generator to demonstrate the iterations of the project.

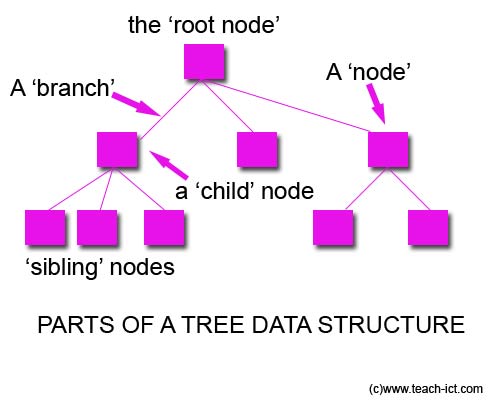
**Preliminary Structures**

To understand the implementation of this project, it is crucial to first understand the structures that are being discussed throughout the text, namely the object of our study: the probability tree.

**The Basic Tree**

Anyone who has much experience with programming data structures should be aware of the tree data structure. Trees are hierarchical structures consisting of branches and nodes, where a node is a structure that holds data and a branch is a connection between two nodes. The “root” node is the base of the tree, and all of the other “leaf” nodes derive from it. The nodes directly connected to the root node are called the root’s children, and each of these “child” nodes can have child nodes of their own. A node with children connected to it is called a “parent” node. For a visual example of what a tree looks like, see Figure 1 below.

**Figure 1 (Google Images)**



Trees are used in a variety of applications, and are often used to store data that can be sorted in some hierarchical format. Often times, this makes it much easier for programmers to search for specific pieces of data more quickly, and it also allows users to make varying paths from the root to an ending leaf node. The path application is what will be used here to create the different stories in the story generator program. Each parent node will be a “choice” that the program can make, while the child nodes will each be an option that results in another choice. While this is quite useful, it is not enough to simply have a basic tree of choices to create a believable AI. To add some variability that will make the AI more believable, a special type of tree called a probability tree will be used for the story generator program.

**The Probability Tree**

A probability tree is a tree with multiple children whose nodes contain probability data. This probability data can be utilized by the probability tree’s extra functionality to determine what percentage of the time a given child node ought to be selected. For example, let us look at a simple tree where parent node P has branches to its child nodes A, B, and C. Let us also say that the user wants to randomly select one of the children. Using a normal tree, random selection would have to be implemented outside of the tree structure, and would likely give equal opportunity to each of the child nodes because there is nothing in the tree that conveys any sort of priority. In a probability tree, however, the functionality allows the user to select nodes based on the priority (probability) of the given node.

For example, looking at a tree with parent node P, let us say that node P is the choice of what to do when you are inside of a burning building and you see someone trapped behind a beam than you are capable of moving. Let us say that choice A is to help the person get out from behind the beam and then leave, choice B is to leave the building and to tell a firefighter that you saw a person trapped and where you saw them, and choice C is to leave the building and say nothing because the firefighters have it handled. If attempting to design an AI that selects one of these options using a normal tree structure, there is no way to determine which path should be chosen if one should be chosen more often than another. The only way to implement the selection process with a normal tree is to have a selector implemented outside of the tree. This selector then has two options: it can either select randomly, giving equal opportunity to each of the child nodes, which is highly unpredictable, OR it can select unevenly, giving weight to each of the options. However, the weighted approach would have to be based on the order of the nodes, and could not be based in the value of the node itself. In other words, a weighted implementation might select node A 50% of the time, node B 30% of the time, and node C 20% of the time, but it would give those same probabilities to every choice that was made in that order, which is not as useful as the probability tree implementation.

With a probability tree, we can say that node A (choice A) has the probability value of 0.7, node B (choice B) has the probability value of 0.2, and node C (choice C) has the probability value of 0.1. When the user goes to select a child of P, now there is data that can be used such that you rescue the person 70% of the time, leave and tell a firefighter about the person 20% of the time, and you ignore the person only 10% of the time. Given another choice, you would be able to set node A’s probability value to 15%, node B’s probability value to 40%, and node C’s value to 45%. As you can see, this enables the tree to determine the value of each choice, and makes it much simpler for implementing a useful and adaptable artificial intelligence. Case in point, with the firefighter example, a normal tree would make all characters utilizing the tree approximately the same, whereas with the probability tree, you could have some characters more likely to be a hero, or some to be more of an oblivious person. Therefore, a probability tree can be used to add personality to different AI characters, which helps make the characters realistic.

**Probability Tree Code**

Conceptually, this sounds fantastic, but how is it implemented in code? The probability tree used for the purposes of this project is laid out into three classes as follows: ProbabilityNode, ProbabilitySelector, and ProbabilityTree. The ProbabilityTree structure is comprised of ProbabilityNode structures and utilizes the ProbabilitySelecetor to pick the child node. The layout of each of these classes is as follows.

**ProbabilityNode**

For more implementation details, see Appendix A (To be added later).

**The Simple Case**

**One-Step Memory**

**Story Generator Application**

**Demo**

**Conclusion**

**Bibliography/References**

[1] D. Mark, *Behavioral mathematics for game AI*. Boston, MA: Course Technology Cengage Learning, 2009.

**References**

<http://vivin.net/2010/01/30/generic-n-ary-tree-in-java/> -- This is where I viewed example tree code for Java.