CSC 584/484 Spring 24 Homework 4: Decision and Behavior Trees, Learning

Due: 04/22/24 by the start of class

Overview

Your task for this assignment is to explore **decision trees**, **behavior trees**, and **learning decision tress**. Working alone using C++ and `SFML`, your task is to implement the algorithms described below and to analyze your results in a 2–3 page writeup. Note that for some of these tasks you will be integrating your new solutions with your code from Assignments 2 and 3.

This assignment is worth 30 points on your homework grade.

First Steps

For this assignment, you will be using your movement and pathfinding code from Assignments 2 and 3 to demonstrate higher-level decision making. To start, ensure that you have functioning pathfinding algorithms and basic movement algorithms including seek/arrive and pathfollowing.

Additionally, make sure that your environment from the "putting it all together" section of homework 3, or one substantively similar in complexity and form, is implemnted and supports pathfinding/pathfollowing.

Decision Trees (6 points)

First, you are to incorporate decision making into your movement and pathfinding system. Your task is to devise a parameterization of your environment that will enable you to build a decision tree model to control changing of targets or movement behaviors. For example, you may decide that whenever your character reaches maximum velocity, it's behavior should change to wander. Or whenever it wanders near a wall, its behavior should change to pathfind/pathfollow to a particular target location (like the center of the screen). Your decision tree will be responsible for the decisions made about when and where to update target locations, when to pathfind to those targets, when to change movement behaviors, *etc*. Use your imagination. And more importantly, as always, write about it!

The big challenge in this part of the homework is not the algorithm itself, but the choice of how to ***represent the state of the environment*** (*i.e.*, what parameters comprise the observations you expose to the AI). Make sure you include the variables you will need for your decision tree to encapsulate the choices you want it to make. Make sure you devote a significant portion of your writeup on this section to addressing this point. It is crucial! Be sure to also include a representation of your decision tree in your writeup (a photo of a hand-drawing in a notebook is sufficient if you don't want to use a diagramming tool).

Note: While there are open source decision tree implementations, they are not allowed for this assignment. You must implement your own decision trees.

Behavior Trees (8 points)

Your task for this section of the assignment is to implement a behavior tree algorithm. You must have at least three types of composite nodes (sequence, selector, decorator, random, parallel, or something else you come up with). Using your behavior tree and movement algorithms from your previous assignment, incorporate

a "monster" into your environment. Your monster's task should be to move around the environment and try to collide with your character. Also, have your monster do something else interesting (perhaps a little dance or following a scripted path). Use your behavior tree to encode all of this behavior. If your monster is successful at "eating" your character, have them both return to neutral starting positions and begin again.

Make sure in your writeup you describe the behavior in both words and also present the behavior tree (again, a photo of a hand-drawing is sufficient). If you invent your own type of behavior tree composite task, make sure to clearly explain it as well. Was it hard to author the behavior tree? Was it hard to implement? Did your monster behave as you expected? Write it up!

Note: While there are open source behavior tree implementations, they are not allowed for this assignment. You must implement your own behavior trees.

Decision Tree Learning (8 points)

For the last part of the assignment, you will be using decision tree learning to clone the behavior of the behavior tree. By recording data about your monster's movement and then learning a decision tree from those data, you should have a decision tree that results in behavior similar to the behavior produced by the behavior tree that controls your monster.

In order to facilitate learning a decision tree, you will need to parameterize the state space. You can use a binary- or nominal-valued vector as your state representation. Examples of attributes include the room the character is in, the number of other characters in the room, distance to the closest obstacle in a particular direction, time since the last collision with a monster, *etc.* The values of these parameters will be the input to your decision tree learning algorithm. Lastly, *make sure you also include the action the monster is currently taking as part of the state information*.

Run your behavior tree-controlled monster and record the parameter values at each time step to a file. Each of these recorded set of values will be used as input to your decision tree learning algorithm. The parameter values will be evaluated at the interior nodes of the decision tree, and the actions will be the leaf nodes of the tree you are learning. Using these data, run the decision tree learning algorithm to build a decision tree. How much data did you collect? Why?

Now execute that decision tree. How do the two monsters (the behavior-tree-controlled and the learned-decision-tree-controlled) compare to each other? Are they as effective at eating your character? Do they look qualitatively different. *Devise a scheme to quantify performance and report on any differences between the decision tree and the behavior tree.*

It is completely fine to have your decision tree learning code run separately from your main SFML project. You do not need a single executable that will learn the decision tree and then deploy it to control the character's behavior. You may record the learned decision tree to a file, and then integrate it into your SFML project manually.

Please note: while there are plenty of freely-available implementations of decision tree learning algorithms, to **earn full credit** ***for this assignment you must implement the algorithm yourself.***

Make sure you clearly define your attributes in your writeup, as well as depict your learned-decision tree (again, hand-drawn will suffice). Take screenshots or make videos of your character moving through the space and changing behaviors according to the decision tree you constructed and the attributes that evolve in the environment. Write about it!

Writeup (8 points)

Now that you have implemented and evaluated a number of algorithms, write a 2–3 page single-spaced paper summarizing your findings. That is at least **two full pages**. It is **strongly** suggested that you do not

limit yourself to only answering those questions posed in this assignment. Think creatively about what you have done. What other parameters can you tweak and what effect do they have on the results? The most successful writeups will contain evidence that you have thought deeply about these algorithms and what they can produce and have gone beyond what is written in the assignment.

As an appendix to your paper, please include all relevant screenshots to support your analysis. The appendix does not count toward your 2–3 page requirement.

What to submit

By the start of class on 04/22/24, please upload a .zip archive to moodle. This archive should contain all of your source files from each part of the assignment, one or more Makefiles, a README file, plus your writeup in .pdf format. Make sure your README contains ***ALL*** of the instructions necessary for compiling and running the different versions of your code.