CS540 Project #1
Spring, 2018
Due Thursday, February 15th

Team Formation

Projects for this class are team based. Most teams will have four members, and are expected to stay together for the duration of the course. However, there may be a few three member teams, and *with the permission of the instructor* there may be changes in teaming between assignments.

All teams will include at least one distance student and at least one on-campus student. This has two purposes: to give on-campus students the experience of working in distributed teams, and to make sure there is at least one on-campus student in every group to give in-class presentations. Project #1 is due two weeks after teams are assigned, so in addition to the intellectual items below, teams should quickly coordinate schedules, communication mechanisms (e.g. skype, zoom, email…), and distributed tools (git, svn, overleaf, google docs…)

Teams are assigned by the instructor, but pairings may be suggested by the students. If you want to work with someone else in the class, email draper@colostate.edu with the name of that person by the end of the day on Wednesday, Jan. 31st. Do not provide more than one name. I will try to accommodate pairing requests, but make no guarantees. Teams will be assigned on Thursday, Feb. 1st.

Task

The task has four components. The first is to build a *drone world* simulator (described below). This simulator will be used for this assignment and future assignments, so it's worth putting time into, even though this part of the task is not directly graded. The second component is to write a five-page (no longer[1]) paper in 2-column IEEE format that describes the performance of two or more search algorithms on drone world problems, comparing and contrasting their performance and showing the results. Enhancing at least one of the algorithms is recommended. The paper should include a brief literature review (based on what we have discussed in class), and follow the professional paper outline discussed in class. In general, a 'B' paper has a professional structure, including correct grammar and spelling, and confirms for me something I already know about search; an 'A' paper has the same structure and teaches me something new. (A paper with identifiable deficiencies receives a grade of 'C' or worse.)

The third components of the assignment is to prepare and give a five minute (no longer) oral presentation with slides in class. These presentations will be strictly timed, and will be graded by (1) how well you were able to present the most important aspects of your paper and (2) the quality of the visuals and accompanying descriptions. Slides must be emailed to the

[1] Page limit includes all figures and tables, but not references.

instructor 24 hours before the lecture in which they are to be presented, so that all slides can be preloaded onto a single laptop for rapid-fire presentations.

The fourth component is individual, not team. Every student will write a brief paper one IEEE column long (no longer) describing their contribution to the team's project.

Every team will receive a grade. 2/3 of the grade is based on the paper; 1/3 on the presentation. The grades assigned to individual members of the team may then be adjusted from the team grade based on the contributions (or lack thereof) of the team member.

Drone World

Drone world is a simulated desk-top world populated by blocks and drones. It is 3D and grid-based. The middle of the table has coordinates (0,0,0), and the table extends 50 units in the X and Z directions, so table-top coordinates range from -50 to 50 in X & Z. (In other words, the table top has 101 x 101 grid cells.) Grid world also has 50 units of open space above it in the Y direction, so (50, 50, 50) is one corner of its 3D world. Negative coordinates in Y are not allowed.

Blocks are unit cubes with assigned colors, which are symbols that include but are not limited to 'red', 'green', 'blue', and 'yellow'. Two or more blocks may share the same color. No two blocks may share the same (X,Y,Z) coordinates. Blocks can sit on each other but cannot float in the air. Unique block positions are therefore legal if (1) they are of the form (X, 0, Z), or (2) they are of the form (X, i, Z), i > 0, such that another block legally occupies the coordinates (X, i-1, Z).

Drones, like blocks, are unit cubes, but there is only one of them, and the drone can not share the coordinates of any block. Drones have two properties blocks do not have, however: drones can fly, so they do not need to be supported by the table top, and drones are controllable. Drones support the following actions:

- **Attach().** This command is only valid if the drone is at position (i,j,k) and there is a block at position (i, j-1, k). This causes the block to attach to the drone, and thereafter the block moves with the drone, always occupying the grid square immediately below it.
- **Move(dx, dy, dz)**, where dx, dy & dz are elements of the set {-1, 0, 1}. Obviously, this command changes the drone's coordinates by adding the vector (dx, dy, dz), so long as the destination square is unoccupied, and if the drone has a block attached then the square below the destination square must also be unoccupied.
- **Release().** This command is only valid if the drone has a block attached, in which case the block is released. Releasing a block does not change its X or Z coordinates, but may cause the Y coordinate to be decremented in order to create a legal block position (i.e. the block falls until it hits another block or the table top).
- **Speak(string).** This command will not be used in this assignment, but for future assignments your drone should be able to output text strings.

Drone world actions are discrete and sequential. For example, a drone can attach a block and move, but it cannot do both at the same time.

Drone world simulators should be a distinct object from the search algorithms you analyze in this assignment. The interface between the two should be limited to the four commands (methods) above, plus an **initialize(filename)** method and a **state()** method. The state method returns the current of the simulation for the search algorithm. It should return a container of N+1 elements, where N is the number of blocks in the current simulation (the extra element is for the drone). Each element contains an (x,y,z) coordinate and either a block color or the symbol 'drone'. The initialize method takes an initial state of the simulator in the form of a filename, where the file contains N+1 rows, each containing an (x,y,z) coordinate and a color or the word 'drone'. The initialize method should verify that the initial state is valid (i.e. it contains exactly one drone, and all blocks and the drone have legal coordinates). Similarly, the simulator should ensure that each action is legal.

Search algorithms should also be able to read goal states in the same format, except that any coordinate and/or color name in the file might be replaced by '?', which is a wildcard. For example, if a goal state included: (?, 3, ?, red) is would imply that a red block must be stacked on top of three other blocks.

It is highly recommended that drone world simulators be able to log the sequence of drone actions; otherwise, it is hard to evaluate the performance of the search algorithms, leaving your team with nothing to write about in the paper. Some form of graphical interface is also recommended, as it can help produce figures for the paper and/or presentation.

Hints

- Note that I am not telling you what algorithms to compare
- Nor am I telling you how to measure search algorithm performance
- Nor am I providing initial drone world configurations

Be creative. Teach me something…

… but this says start working on this assignment as quickly as possible (as soon as your team is set). Because the task isn't to build a simulator or implement search algorithms. That would eb easy, undergraduate stuff. The assignment is to think about search algorithms, how they differ and how they are similar. Then think of performance measures and configuration patterns that will demonstrate these properties and give you something to write about. You only implement the simulator and search algorithms to test your ideas; it's the ideas (supported by data) that get graded.