**Project**

You and your team must propose a final project that demonstrates one or more of the AI techniques we covered over the semester. You can choose to use the Lego agent if you wish. The proposals are due April 28th.

In your proposal you tell me what problem you are going to solve, why this is significant, how you will solve it, what AI theory and techniques you will be using. Your proposal will have an Introduction, Background, proposed Approach (how you intend on solving this problem), how you will evaluate your work and your hypothesized results in it.

We will discuss your proposals in class to critic ideas.

For your deliverable, you will present your work and demonstrate it during your final time. You will also turn in group code and a group write-up that follows the structure specified on blackboard.

Ideas:

* Play capture the flag with another team using your Lego agent.
* Implement Wumpus World with your Lego.

Multiagent system software:

# Jade(JAVA Agent DEvelopment Framework): http://jade.tilab.com

Jadex:

Jadex download: http://sourceforge.net/projects/jadex/

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.7243&rep=rep1&type=pdf

* Play Conway’s Game of Life

Develop a simple distributed application to play the game of Life. You will assume a 5x5 world. Each cell in the world will be represented by a different agent. Each agent will have the same set of very simple rules:

1. if the cell is alive and it has 2 or 3 living neighbors, then the cell stays alive
2. if the cell is alive and it has less than 2 or more than 3 living neighbors, then it dies
3. if the cell is dead and it has 2 or 3 living neighbors, then it becomes alive

In addition to the 25 cell agents, there will be some bookkeeping agents, too. An initializing agent will use *random* to populate the initial world. A displaying agent will show the world after each generation. A control agent will make sure that the focus loops through all cell agents. You may build other agents if you need them.

Communication between agents will be done explicitly using messages. You will define the format of the message, but, in general, the message should have information about the sender, the receiver, and its contents. The messages should be used to inform cell agents about the state of their neighbors, their initial value, etc.

* Implement a simple Contract Net system. There is a single contractor agent that will receive the tasks, decompose them, accept and evaluate bids, and then assign the tasks. There will be five bidding agents.

For simplicity, we will call the contractor C and the bidding agents B1, B2, B3, B4 and B5.

Through a user interface (outside the agents), the user will select the task that is to be executed. The tasks will be mathematical operations, and they will be expressed in prefix notation. For example:

(sin (+ 10 (/ 10 3))

B1 will be able to perform: +, -, exp, \*

B2 will be able to perform: +, -, \*, /

B3 will be able to perform: exp, sin, cos

B4 will be able to perform: +, /, exp

B5 will be able to perform: +

Here is how your program will work:

1. The user inputs the task and the contractor reads it. E.g.:

(sin (+ 10 (/ 10 3)))

1. The contractor decomposes the task. You are not expected to perform complete decomposition, but only to figure out which operations are to be performed. You may assume that only executable operations will be given to the system. E.g.:

(sin (+ 10 (/ 10 3)) -> (sin + /)

1. The contractor posts the tasks to be performed and the bidders bid on them. You will need to establish a way to bid on tasks and a way to evaluate the bids.
2. If a bidder cannot perform the complete task, the contractor needs to decide on a set of bidders who can complete the task and assign the appropriate subtask to each one. For example, for the preceding example, C could assign (sin) to B3, and (+ /) to B2.
3. You will need to provide sufficient output to inform the user of the process of the operations (e.g. task decomposition, bidding, task assignment).

Or

Set up a blackboard, matchmaking or contract net of your own design.

* Develop a case-based system for design. Select a domain, store some objects as cases, implement similarity techniques and some simple adaptation rules. Examples: design of homes (parameters: number and types of rooms, windows, doors, size of backyard, etc.);
* Play soccer with SoccerBots: <http://www.cs.cmu.edu/~trb/TeamBots/Domains/SoccerBots/#YOUROWN>
* Analyze some datasets using neural nets: http://neuroph.sourceforge.net
* Analyze some datasets using data mining: http://www.cs.waikato.ac.nz/ml/weka/