Video Game AI, Spring 2013
HW #3
Due May 9 at 12 midnight

This assignment is to be done individually. You can talk to other students to understand the provided code, and to get ideas, but all new code should be your own.

In this assignment you will implement behaviors for a behavior tree in provided code on the course web page. Some behaviors have already been implemented, while some still need to be implemented. The assignment has three parts:

1. Understand how the provided code works. Being able to continue a behavior tree where it left off is not a trivial task. Thus, it is important that you understand how the provided code works before you start writing new code. (Most programmers will have to work with others' code, so this is a useful skill in general.)
   a. You should only modify BTUnit.cpp/.h. At the top of the .h file you can change the number of agents to assist in debugging.
   b. Each agent has their own copy of the behavior tree. It is updated automatically when the behavior tree is edited.
   c. Every node in the behavior tree has a variable named `sharedData` which can be used to store any relevant information for maintaining history in the tree. The convention is that a value of -1 means that there is no history. This is generally used to mark the current executing path in the behavior tree.
2. Implement actions, decorators and queries. Some of these you will have to make up yourself. These do not have to be complex, but you might want to design them to be useful in part 3.
   a. Three queries have been implemented. Make up and define three more. (Feel free to contact the instructor if you are unsure how a query can be implemented in the provided environment.)
   b. Three actions have been implemented. Make up and define three more. (Feel free to make some actions perform differently on each unit; calling `simInfo->GetCurrentUnit()` will provide the current unit id. [1-3 for each of the dogs])
   c. Two decorators have been implemented. Define the last decorator and make up and define three more.
   d. One composite has been implemented. Implement the other two. [Hint: you may want to treat the sharedData as a bit array in order to implement ~?.] [Tip: If you click/tap on a composite tree node, a –/+ sign will appear to allow you to add or remove children from the composite.]
3. Build the strongest dog behavior possible given your behavior tree primitives. Because dogs are slower, you can make the behavior as strong as possible. Behavior trees are not saved between runs, so you can either write out or take a screen shot of the behavior tree. (Hit command-shift-4 and then space bar and then clicking on a window will capture that window to a file on the desktop. Hitting command-shift-control-4 and then space plus clicking on a window will put the picture on the clipboard. Opening the Preview application and creating a new document will allow you to save the file elsewhere on the computer.)

Turn your final code into your SVN repository.