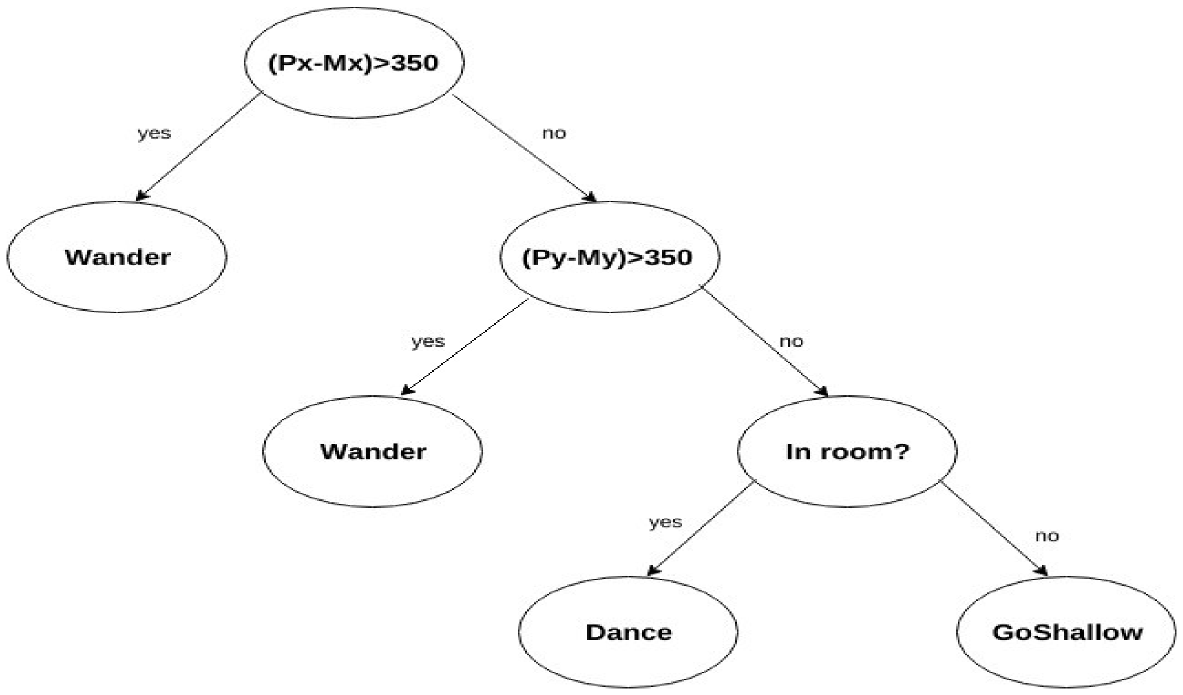
CSC 584
HW3

Rishabh Sinha
rsinha2@ncsu.edu

Game Description

The game that has been implemented by me has a theme of Sea Monsters. The Monster in my game is a Shark and the protagonist is a gold fish. In the game, the Aim of the shark is to eat the goldfish, while that of the goldfish is survival.
The Game Environment consists of a large lake represented in blue, Since both the protagonist (Gold Fish) and the antagonist (Shark) are fishes, they can swim anywhere in the blue region. The lake also consists of rocky boundaries on the sides as well as certain other rocky formations which divide the region into four some what separate sub lakes which is connected together in the middle. These rocky formations as well as the boundary is represented in gray color in the game. Both the characters cannot enter the rocky regions of the game environment and they represent the obstacles in the environment for both the character. The Game Environment also contains marsh land that have represented in green, these are basically shallow waters which can be accessed by the game protagonist because of its small size but is an obstacle for the shark as it cannot swim in such shallow regions of the environment.
Thus the game simply is a pursuit evasion game, in which the the monster pursues the goldfish which is the evader. The game continues until the monster catches the goldfish, after which the below mentioned behavior tree resets and the game restarts once again.

Decision Tree



A decision tree is a decision making algorithm that is used for Game AI. Decision Tree as the name suggests is a

Tree whose traversal determines what action will be carried out by the AI character.

The Architecture of a decision tree is very similar to a flowchart-like structure in which each internal node represents a condition where in a value of an attribute is tested. Each branch of the decision tree describes the outcomes of the conditional test on the variable. And the leaf nodes represent the final classification which in case of Game AI is the action that will be taken by the character.

To determine the value of Action (class) that will be carried out by a character for a given data record is determined by the traversal down from the root to any one of the leaf node of the tree, Where in at each node a condition is tested and the branch is chosen according to the result of the condition at each node.

For the Purpose of this project I have implemented a decision tree that controls the behavior of my in game character. The concept of my game is very simple and has been described in brief above. The Decision tree implemented for this part of the project specifically concentrates on determining the behavior of the goldfish. The goldfish has 3 possible action choices, these action choices are wander, seek marsh as well as dance.
The Wander Algorithm is basically a path following form of wander where in the character selects positions on the map in neighboring areas at random and the follows path to that node and than continues on further,
The Seek Marsh Algorithm is a safety first algorithm for the goldfish, which if it senses danger seeks the marshy land and sits in the marsh until the Shark has moved a certain distance away from the character.
The third action is the dance action, in which the goldfish, when it sits in the marshy land does a kind of break-dance, where in the character rotates, while it sits in its position waiting for the shark to move away.

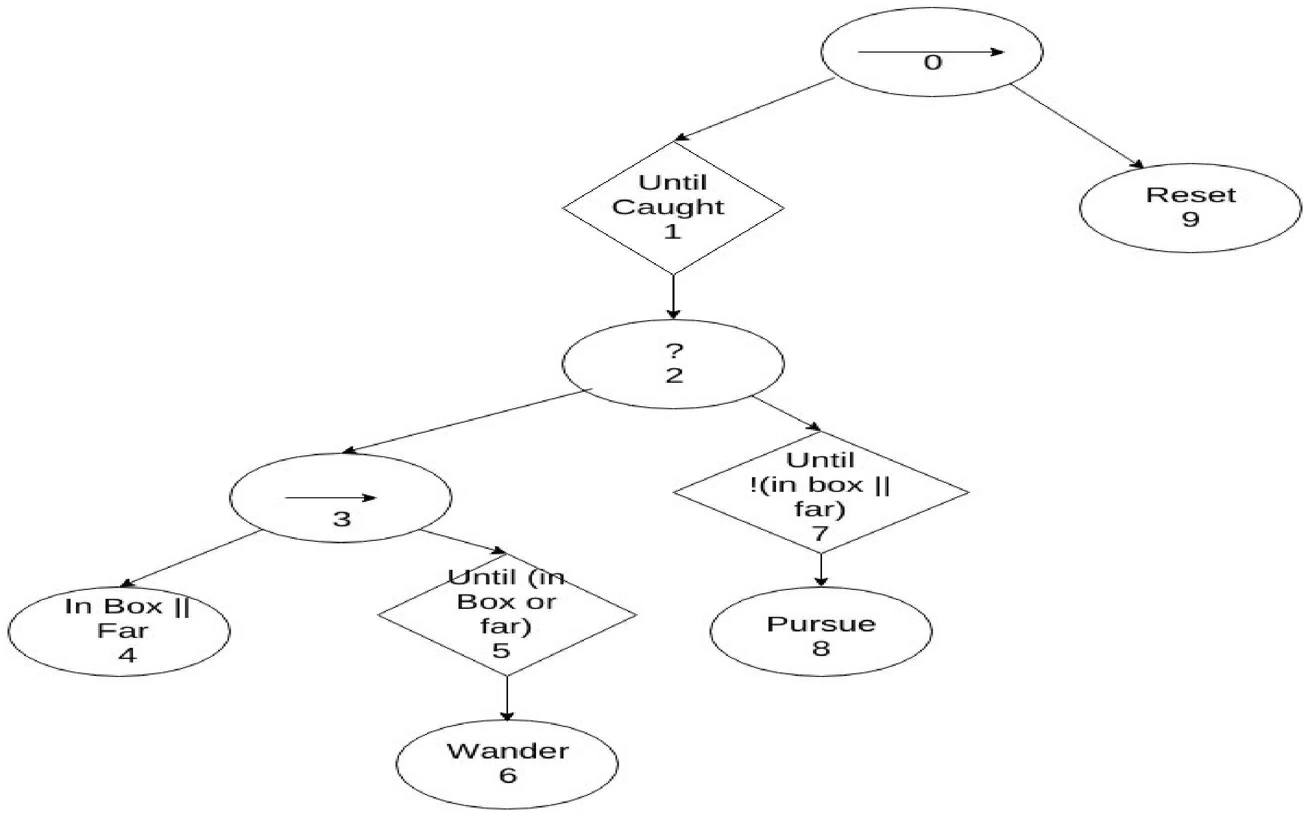
The Flow of Control in Decision Tree starts from the root node. The root node of the decision tree for the goldfish character checks the distance of the shark from the goldfish (x coordinate wise), if the distance of the shark from the goldfish is more than a 350 pixels in x, then the goldfish does its default behavior which is just wander. The wander, behavior defined above is not the standard wander behavior, but as mentioned above takes care of path following problem, by choosing random nodes in its neighborhood.

The next condition if the above condition happens to be false is a check of the same case for difference in y coordinate for the characters, if that is again more than 350, the goldfish does wander, else we check whether the goldfish is already in the green rectangles. If the goldfish is in the green rectangles then in that case the goldfish can be considered safe. As the shark cannot enter that region. Thus the goldfish whenever in these specific regions does the dance action. The dance action in the above example can be seen as a side to side movement of the character in the particular region.

If the result of the above 3 conditional check is no, then in that case, the goldfish does a kind of flee operation, it is not exactly flee, But basically the goldfish tries to run to one of the green regions, which is the swampy land, which is an obstacle for the shark. So this basically occurs if the distance between x and y coordinate between the two is both less than 350 pixels as well as the goldfish is not present in the one of the four green regions, The conditions for the green regions is once again given in terms of start and end x,y coordinate of each of the individual green regions. Then in that case the goldfish tries to hide.

Thus at each iteration of the game loop, the game happens to, traverse the entire extent of the decision tree to know the current state of the environment that is the x, y values of the character itself, the x,y values of the shark. To determine which action would it undertake, that is whether it would do the dance behavior, the flee behavior or the wander behavior. A visual representation of the above explained decision tree is given above.

Behavior Tree



The Behavior tree in the game implementation, has been used to control the monster character which in the game is a shark. Behavior tree is an encoded graph structure that produces behavior. Behavior trees are a state-full structure where in the state of the traversal is maintained.

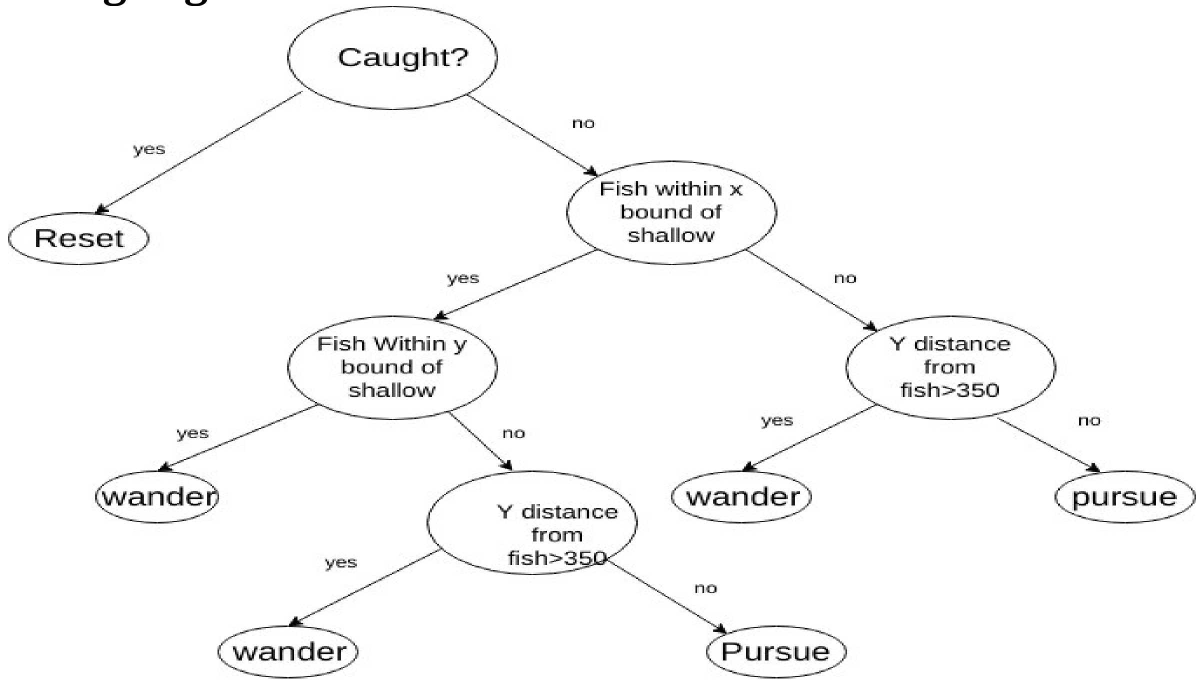
The Behavior tree in my implementation consists of 3 different types of composites and 9 different states. The root node (state 0) is a sequence of a decorator (state 1) and and a reset (state 9) action which is a leaf.
The decorator runs until caught becomes true, caught is a variable that becomes true when the goldfish has been caught by the shark and false otherwise. So if the value of caught is not true, then the control moves to the next node in the true, that the decorator is running multiple number of times, that is the selector (state 2). This selector consists of 2 child nodes, these child nodes are a sequence (state 3) and a decorator (state 7). The purpose of the selector to basically choose between the actions wander and pursue based on certain condition. The sequence once again has 2 child nodes. These child nodes consist of a condition (state 4), that checks whether the goldfish is inside the green swamps or not and also a decorator (state 5) that runs wander until this condition is true. If the goldfish is in green swamp, then sequence progresses and the shark does wander (state 6) which is run multiple times by the help of a decorator.

If false is returned by state 3 of the selector (state 2) that is wander action has not been carried out, then the control moves to state 7 which is a decorator. This decorator is responsible for running pursue action (state 8) multiple times. The pursue action is run until the goldfish is not caught, or the goldfish does not enter the swamp land, after which the behavior tree moves back to state 2 and then 1 (initial decorator) after which the entire sub tree is run once again.
If the goldfish gets caught, then in that case, the control backtracks to node 0 and then node 9 which is the state for reset operation is executed. The reset operation is an operation that is responsible for resetting the entire game once again and restarting it once again from the initial state.

I have implemented the behavior tree, using a set of nested if else statements. On the outer level, the if statements are involved in checking the current state, that is the current place where it is present in the behavior tree. I have also maintained a previous state, variable which is checked at the inner level of if else, the previous basically is used to determine how we have arrived at a particular state, that is if we have arrived to that state from the top, and all the below branches have to be executed or from one of the child nodes, and accordingly the next state of the tree is selected.

There are also boolean values associated with states which tells the parent if the child has returned true or false and accordingly next decision is taken, at the current state of the behavior tree. The Behavior tree was, not very easy to visualize while designing, but once, it was drawn and implemented, It behaved exactly like I expected.

Learning Algorithm



The Learning Algorithm implemented is one which reads data and prints the Decision tree that is learned from the data. Then, the decision tree as printed by the learning algorithm has been hand implemented to control the Shark which is the monster.
The Decision tree learning algorithm implemented by me, Selects nodes at each level based on information Gain, Node with highest information gain is selected at each level. Thus the parameter which has the highest information gain overall is made the root node, And the tree grows from there.
For the Decision tree learning algorithm, I have used parameters for the data as,
1. The difference between the x coordinate of the shark and goldfish
2. The difference between the y coordinate of the shark and goldfish
3. the current x coordinate of goldfish (to know if its in the green region or not.
4. the current y coordinate of goldfish (to know if its in the green region or not.
5. caught boolean value. (needed for reset action)

All the above values have been converted to binary for the purpose of data collection. For example, the difference between x is 0, if the 2 characters are less than 300 pixels away from each other on x coordinate and

and 1 if their distance on x coordinate is greater than 300 coordinate. Same has been done for difference in y coordinates. This value has been selected, because for my behavior tree, I have checked if the 2 characters are less than 300 in either direction only then will the shark pursue the goldfish else he will just do wander.

Similarly the coordinate x of the goldfish is 1, if it is inside the x boundaries of the green shallow regions which is a barrier for the shark, The same is true for the coordinate y of goldfish. Similarly, Caught is a boolean value that represents, that distance between the 2 character in either direction is less than 25 coordinates, which considering for the size of characters (approx 50). is correct.

I collected the data for the learning part of the assignment by running the previously implemented decision tree and the behavior tree, and recorded the values into a .txt file. Following which I implemented the learning algorithm discussed in class to implement the decision tree. The Algorithm, basically greedily selects the parameter with max information gain (Difference between entropy of parent tree and current node(parameter)). Once we have the root node selected, We split the data according to the possible splits of the parameter and accordingly split the data to rerun the algorithm on each of the split. This goes on to grow the tree and terminates whenever we have a pure value at a node in which case, we replace it with the associated action and make it a leaf node. So ultimately we have the associated decision tree that is constructed.

For the purpose of this assignment, the decision tree that was learned from the data, was used to implement another decision tree to control the monster that is the shark.
The Actions that is the classes for the decision tree are of 3 different types. These tasks are pursue the goldfish, wander and reset.

The Decision tree thus constructed from the data is one which has its root node as the caught parameter, if the value of the caught parameter is 1, we reset the game, (so that it can restart), otherwise we, check the value of x coordinate of character (goldfish), if the value of it is 1,that is, the goldfish is in the x boundary for green swamps, we check the value of y coordinate of the character (goldfish), if this value is also 1, this means the goldfish is also in the y boundary for, swamps and thus it is surely present in the swamp, thus the algorithm does wander, if the value for y of character is 0, then goldfish is not in the swamp, then the algorithm uses distance between y coordinate of the 2 characters to check which action is performed. If the distance between the y coordinates is 0, then the monster does pursue else it does wander. Same is true of value of x Coordinate of goldfish is 0, in the second node from the top, where the character then checks distance between the y coordinates and if it is 0, then the monster does pursue else it does wander.

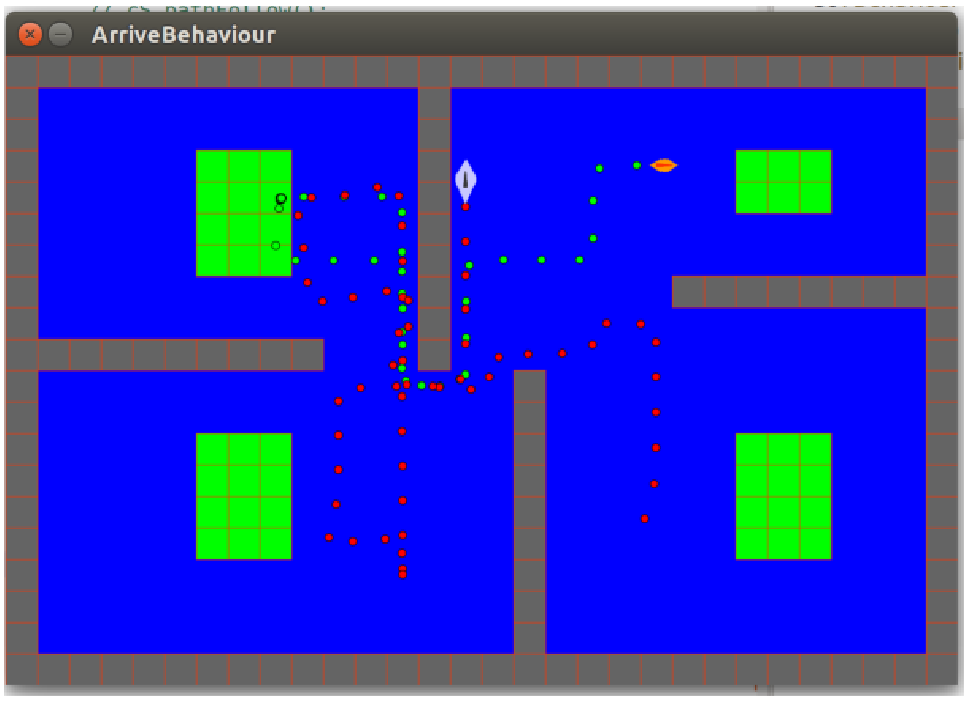
One thing to note from the behavior tree given out by the, learning algorithm, it does not use the information of distance of x coordinates between the two characters, which is unexpected. This may be because, of the shape of the environment, and the behavior of the 2 characters, the difference in x is almost never greater than 300 if the difference between y is less than 0. This is a very rare case.
When implemented this Decision tree, it performs almost identically to the behavior tree and it is very difficult to find out the difference between the 2.
The Learning Algorithm generated tree has one performance difference as compared to the Behavior tree implemented in the last part of the assignment that is, the behavior tree since it has 9 states, works on a higher latency, as path for both wander as well as pursue is not updated every draw cycle, but it is updated once every 3 to 4 cycles, due to change of states. While for the decision tree learned in the final part, the functions for wander and pursue (implemented as path follow) are updated every draw cycle.

Algorithm	Average time to Catch
Behaviour Tree	20.2 seconds
Learnt Decision Tree	25.3 seconds

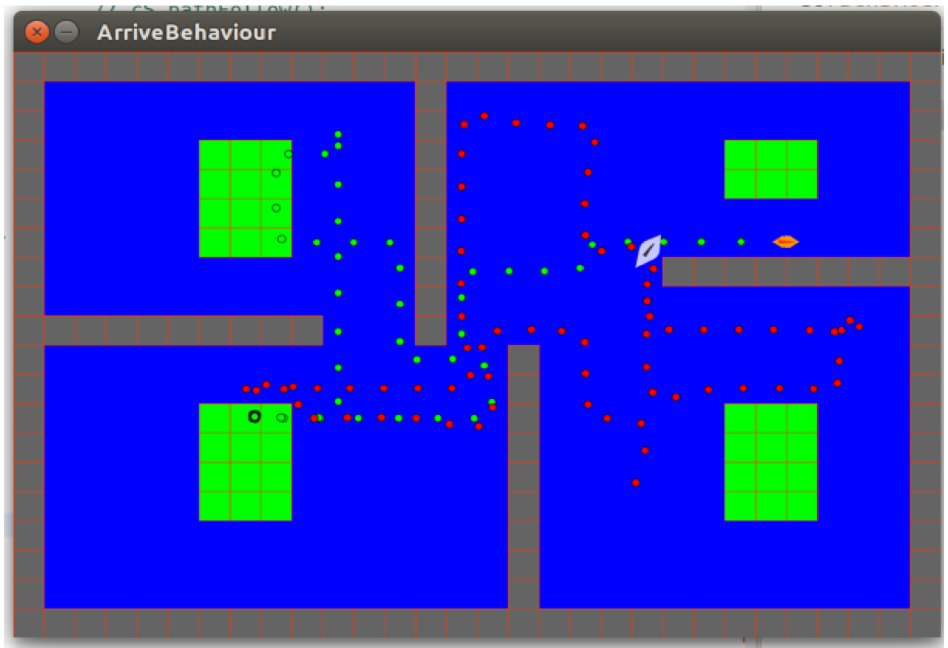
Thus we can see that the Learned Algorithm has a performance slightly inferior to the Behavior tree, This is unexpected as it should be the other way round, as the monster in the decision tree version, is less constrained for pursue operation. Thus it is very difficult to explain, but this could be affected by a few outliers.

For the purpose of collecting data to implement the learning algorithm, I used file write operation while running the Behavior tree. This printed values on every iteration of the decision tree. The I further ran the Algorithm multiple times to just get the values for the caught case, where the algorithm has to reset. This was to make sure, that reset action was equally represented.

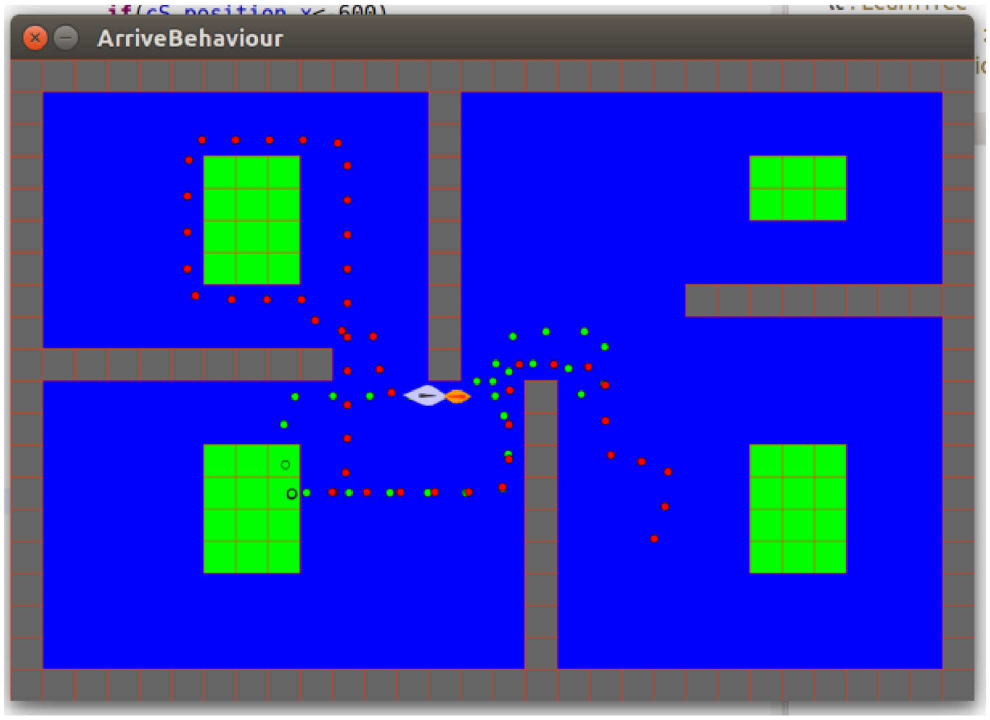
Appendix



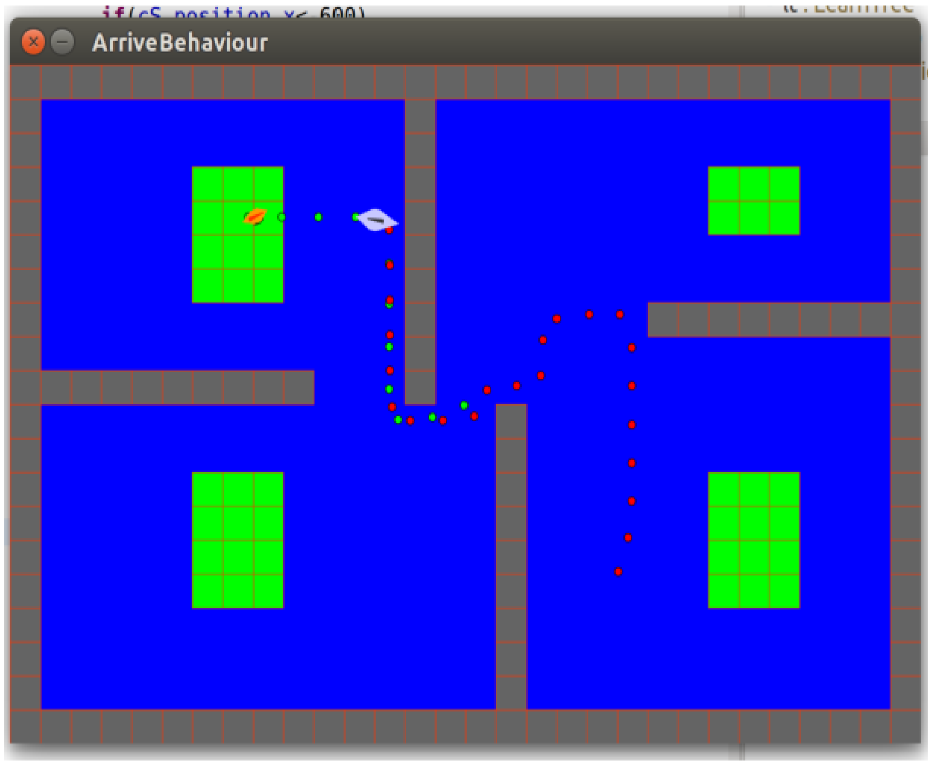
Running Main Class for Behavior Tree, Depicting pursue for shark and arrive at shallow region for the goldfish.



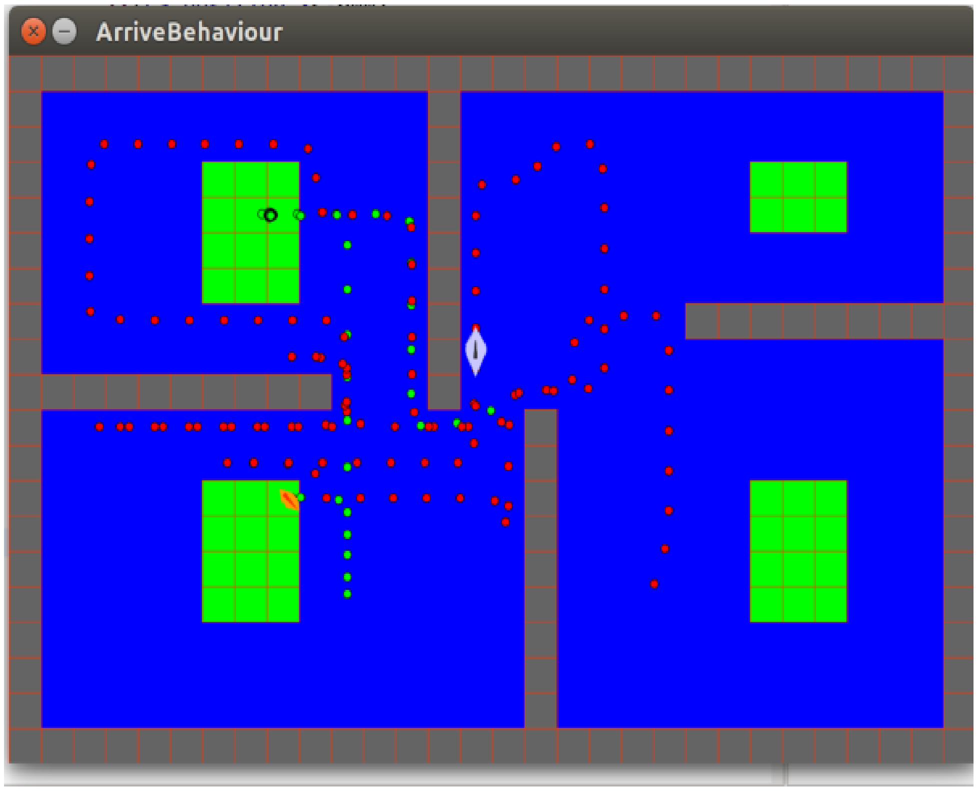
Running Main Class for (Learned) Decision Tree, Depicting pursue for shark and arrive at shallow region for the goldfish.



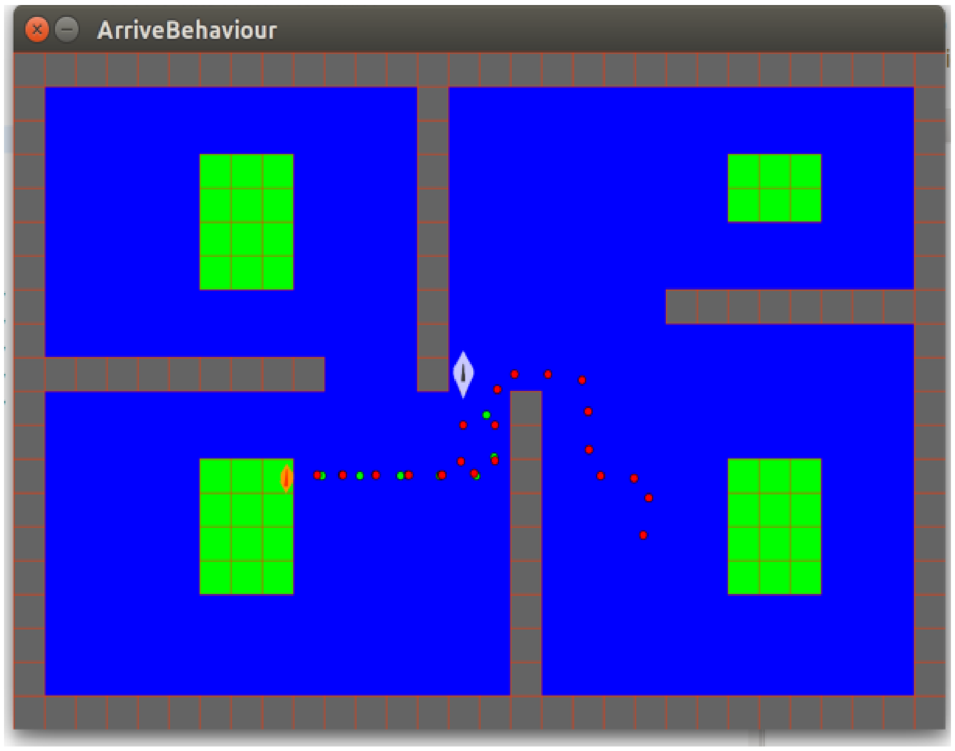
Depiction of catching condition for Behavior Tree



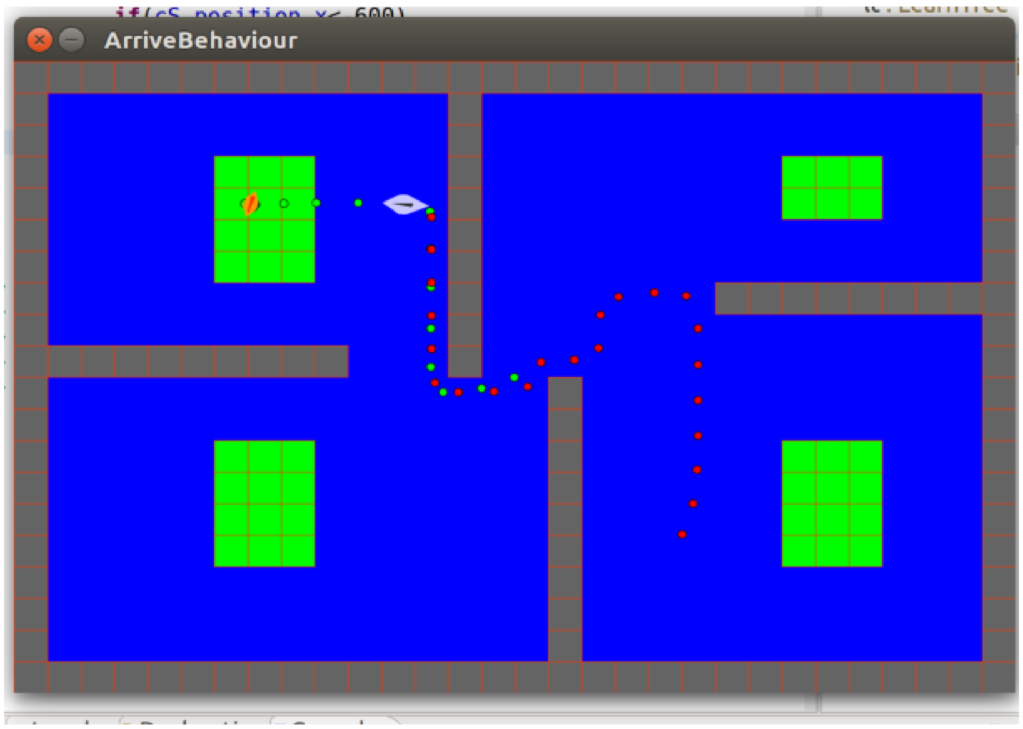
Goldfish takes safe retreat and dances while shark pursues for Behavior tree



Shark Wanders away, while goldfish is dancing in shallow region for Behavior tree.



Shark Wanders away, while goldfish is dancing in shallow region for Decision Tree (Learning) tree.



Shark pursuing, while goldfish is dancing in shallow region for Decision Tree (Learning) tree.