

Assignment 3 Write up

Student ID: 200110436

Unity ID: ychen74

Name: Yi-Chun Chen

Outline

Brief Description about Program Logic

Analysis and Implementation Description

Part 0: Game World Setting

Part 1: Decision Tree

Part 2: Behavior Tree

Part 3: Decision Tree Learning

Appendix

Brief Description about Program Logic

This program was developed according to following components:

- Basic class for kinematic data and behaviors (Only related are listed)
 - KinematicData and SteeringData: The two classes are the data structures which contains kinematic and steering variables—position, velocity, accelerations and so on.
 - KinematicOperations: This class provides basic operations of kinematic variables, such as computing velocity from acceleration, compute next position, etc.
 - Seek: This class implements steering seek with arriving.
 - TimeController: This class is used to compute elapsed time to control decision rate.
- Graph and Pathfinding (Only related are listed)
 - GraphData and GraphGenerator will automatically generate the link edge between input nodes. The map is a tile graph.
- Decision Tree and Behavior Tree (for each part of assignment)
 - Behavior Tree package contains the data structures for implementing behavior tree, including Sequence, Selector, Random Selector, and Action nodes.
 - Decision Tree package contains the data structures for implementing decision tree, including Decision nodes, and Action nodes.

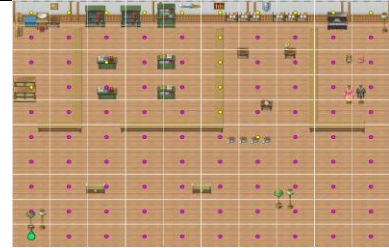


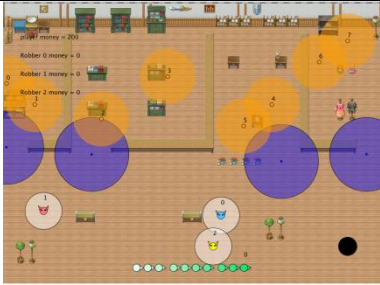
The program logic is simple. In every display loop, when character and monsters have to make decisions (decision rate is controlled by timer with 0.01 second), they will call decision tree method and behavior tree method respectively. Then, the decision/ behavior trees will help they to make decisions in each round.

Analysis and Implementation Description

Part 0: Game World Setting

For this assignment, to make character and monster able to make interesting decisions using decision/ behavior tree, there are some changes in game world setting (comparing to last assignment). The settings are listed below:

- **Scenario:** The world is a shopping mall with 4 stores, and character has some needed things in stores (2/ each store). The monsters are robbers; instead of eat the character, they will rob some money from the character when they get him.
- **Map:** The world was represented as a tile map, with rectangle obstacles. The edge between tiles will automatically avoid the nodes which are inside the obstacles. And there are some special nodes on the map, which are the store enter points, and the location of needed items. The figures below show how the world looks like.
- **Character and Monsters:** The green shape is the character with decision tree. The Blue one is the monster with first version of behavior tree (figure 6). The Red one is the monster with alternative behavior tree (figure 8 in appendix). The yellow one is the monster with learned decision tree.

Figure 1. Map with Dirichlet domains. The nodes are the center of each tile, and the white lines show the Dirichlet domains.	Figure 2. Map with Nodes and edges, each node links to eight directions. The edge will avoid the obstacle when created.
	
Figure 3. The real test environment, 40 x 40 tiles.	Figure 4. Special nodes on the map. The blue ones are the store enter points, and the orange ones are randomly generated needed item. And both the player's money and monster's money are showed on left-up corner.
	

Part 1: Decision Tree

This section shows the decisions that the character has to make, in order to achieve its goal, and the decision branches of the decision tree.

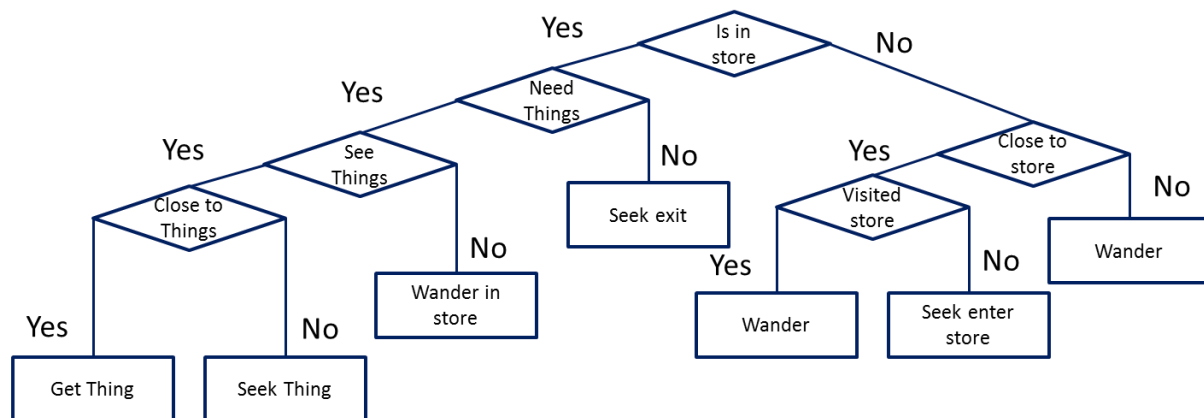
Strategy:

The environment states are represented by some Boolean parameters, which help the character to construct the decision tree:

- **Store area:** the store areas are recorded by two vectors, one is the coordinate of left-up corner, the other is the coordinate of the area's right-down corner.
- **Player position, store enter points, item locations:** the variables are coordinates.
- **Is character in store:** the character will check whether its position is in a store area.
- **Is character close to any store:** the character will check whether it close to a store area by distance.
- **The store was visited:** If the store didn't have needed items or was visited, this will return true.
- **Character need items in the store:** In the store, there are some needed items.
- **Character see needed item:** When character wander in store, the needed items come into its vision.
- **Character close to needed item:** When character is close enough to needed item.

The character will wander in the shopping mall. When it close to some stores and find that there are some needed thing in the store, it will seek to enter the store and wander in the store (look like shopping behavior). Once the character sees the needed item, it will seek the item's position and get it. If all needed items are collected, the player will leave the store and keep wandering in the shopping mall. The decision tree structure of character is shown below:

Figure 5. Character's decision tree.



Part 2: Behavior Tree.

This section shows the behavior tree of monsters. And the needed parameters and data structures are described below.

Strategy:

The behavior tree nodes reference the pseudo code in section 5.4 in textbook. All of the nodes: Sequence, Selector, Random selector and Action node implement the Task interface, so they can be combined together as a tree structure.

The basic idea of monsters' behavior is try to find the character and rob it. So the monster will wander in the shopping mall, but not in store (stores are safe area), in order to find character. If they see the character, they need to decide that whether the character is in store or not. If the character is in store, monsters can choose to wander again or wait in store door. If the character is not in store, the monsters will try to seek the character and rob it. After they rob some money, they will be regenerated in other position of the map.

The parameters used for deciding environment state for monster are listed in below:

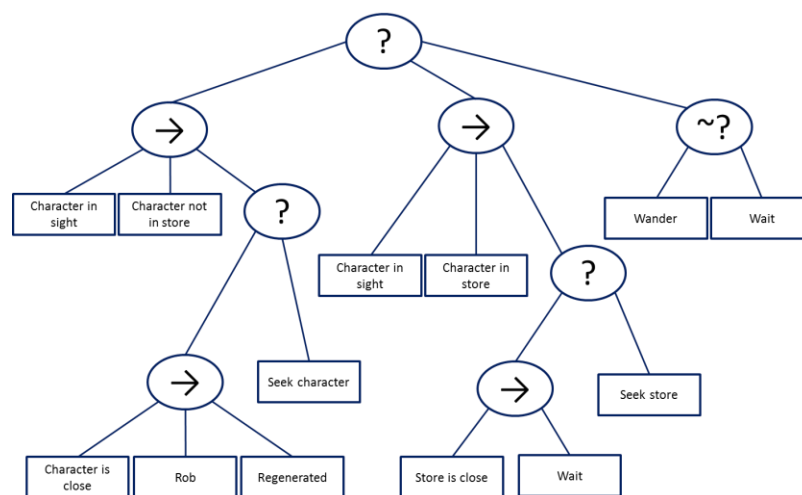
- **Is character in sight:** the monster will check its position to see whether character appear in sight area.
- **Is character in store:** the monster will check whether the character is inside the stores?
- **Is character close enough:** only when the monster is close enough to character, then the monster can rob money from character.
- **Close to store:** when monster chase the character and character goes into store. The monster will wait outside the store, until character fully out of its vision.

Here are the actions that monster can take:

- **Rob character:** if the character is close enough, the monster will rob 10 dollar from the character.
- **Seek character:** if monsters see character, and the character is not in store, they will seek the character.
- **Seek store:** if character goes into store, the monster will seek the store enter point, then decide it should wait or wander again.
- **Wait:** the monster will wait in same position.
- **Wander:** the monster will wander in the map.

The below is the behavior tree for monsters:

Figure 6. The behavior tree for monsters.



According to observing the execution results of this tree. In appendix, Figure 8 shows an alternative behavior tree for monsters. It removes a check action (character in sight) from the

second Sequence node, and removes the Random Selector from this tree to prevent useless Wait action to delay the monster movement. In the alternative behavior tree, when the player go into stores, the monster will wait in the store's door, in order to rob the character.

Part 3: Decision Tree Learning

This section discusses the decision tree learning result, the learned decision is shown below.

Strategy:

The learning method used in this assignment reference the ID3 algorithm in textbook section 7.6. The recorded data use following Boolean variables as parameters to model the states of environment (similar to part 2):

- **Is character in sight:** the monster will check its position to see whether character appear in sight area.
- **Is character in store:** the monster will check whether the character is inside the stores?
- **Is character close enough:** only when the monster is close enough to character, then the monster can rob money from character.
- **Close to store:** when monster chase the character and character goes into store. The monster will wait outside the store, until character fully out of its vision.

Here are the possible actions for monsters:

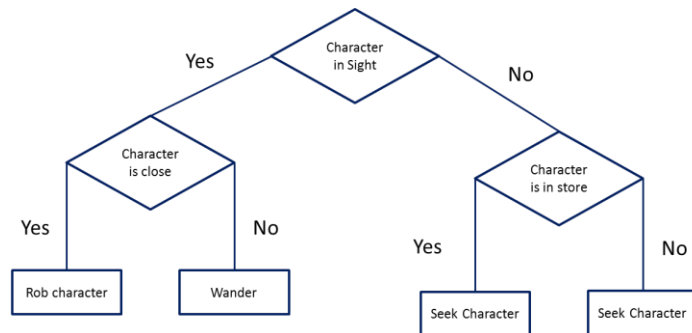
- **Rob character:** if the character is close enough, the monster will rob 10 dollar from the character.
- **Seek character:** if monsters see character, and the character is not in store, they will seek the character.
- **Seek store:** if character goes into store, the monster will seek the store enter point, then decide it should wait or wander again.
- **Wait:** the monster will wait in same position.
- **Wander:** the monster will wander in the map.

After learning, the generated decision become simpler, some of not so frequently actions are removed by the learning process. Also, originally, the monsters have a constraint that they cannot wander into store, but after learning, they become able to follow the character and then go into the stores. That is because, in the recorded data, if the character runs into store, it will easily run out of monster's sight, which makes one of the branches in original behavior tree rarely happen. (The branch that when character in monster's sight, and the player runs into stores).

In this experiment, about 100 records are collected. When the program is executed once, about 25 records will be collect, because character only has limited amount of money. Once the character owns 0, the monster cannot rob it anymore. And, because if monsters become unable to rob the player, they will keep wander. To prevent some bias of data being repeat recorded, once player have no money, no more data will be recorded. Therefore, in each round, monsters usually make about 25 decisions. Then 100 records are the data from 4 rounds of executions.

The figure below show the learned result of decision tree:

Figure 7. the learned decision tree.



The monsters decision become simple, when they see the character, they will check whether it is closes enough or not. In the character is close to monster, it will be robbed. Otherwise, monsters will wander. Then, if the character is in stores, monster will seek the character.

Because of removing some infrequently actions, the monster with learned tree usually can rob more money, which will be consider as having higher performance , than other monsters. The experiment and result pictures are shown in appendix.

Appendix

Figure 8. The alternative behavior tree for monster

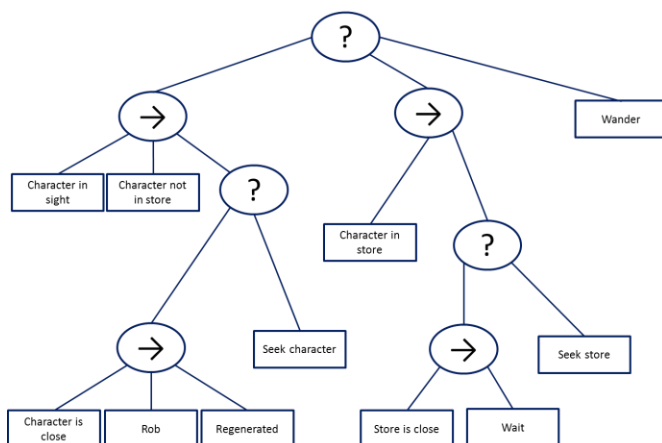


Table1: the experiment results (several rounds executions), the result is how much money they rob from the character.

Round	Behavior Tree v1 (Blue)	Behavior Tree v2 (Red)	Learned tree (Yellow)
1 (get money)	10	50	140
2 (get money)	50	60	90
3 (get money)	70	60	70
4 (get money)	30	40	130
5 (get money)	0	60	140
6 (get money)	0	90	110