HW4 HTN Tree

Tuesday, December 8, 2020   2:22 AM

How I Designed My Tree

Be A Monster

Growl

Cooldown

M0

React Angrily

M0

M1

Shake

Turn Red

Stay Red

Cooldown

Go to Spawn Location

M1

Turn Purple

Idle Behavior

Projectile Attack

Go to Nearest Crate

M0

Pick It Up

Throw Crate

Go to Nearest Rock

M1

Pick It Up

Throw Rock

M0

Move Left

Move Right

Move Right

Move Left

M1

Move in Circle
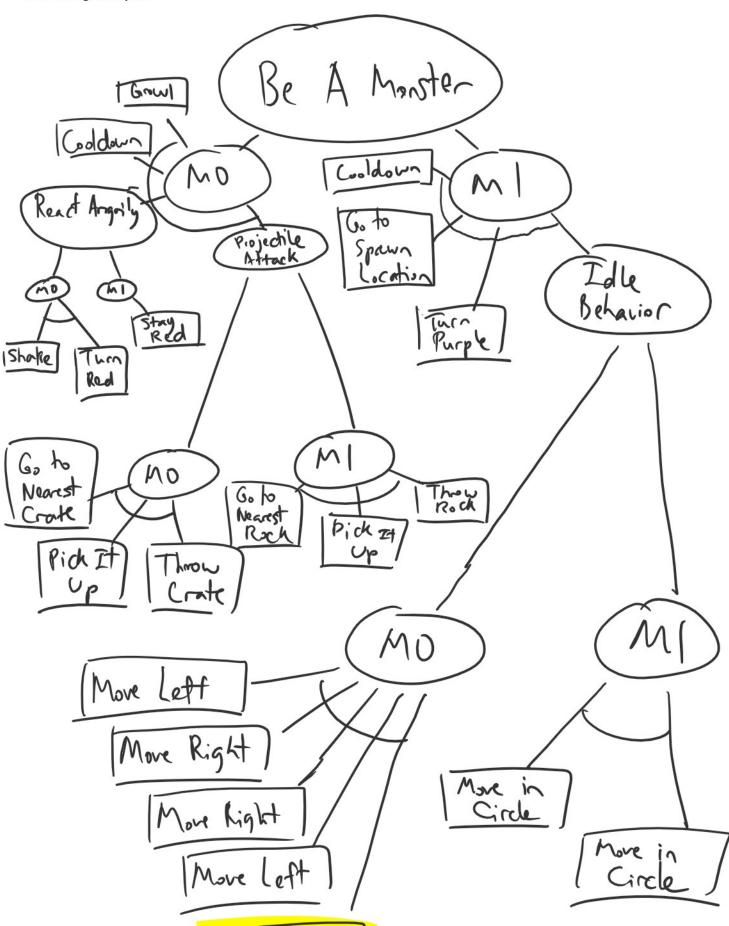
Move in Circle

World State Vector
<isPlayerInRange, isRed, isHoldingObstacle, numCrates>
The first three are booleans.
The last one is an integer, but should be >= 0 (negative crates doesn't make sense).

isPlayerInRange: is the player in range
isRed: is the monster red
isHoldingObstacle: is the monster holding an obstacle
numCrates: how many undestroyed crates are there in the map

Preconditions
To growl, player has to be in range.
To go to span location, player has to not be in range.
To shake, monster must not be red.
To stay red, monster must be red.
To throw crate or rock, monster must be holding an obstacle.
To go to the nearest crate, there must be at least one crate left.

(I didn't add preconditions for Go To Spawn Location and Turn Purple, because if you already are
at the spawn location and are purple, these actions won't do anything anyways, and they are
cheap.)

Postconditions
After turning red, the monster is red.
After turning purple, the monster is not red.
After picking up an obstacle, the monster is holding an obstacle.
After throwing a rock or crate, the monster is not holding an obstacle.
After throwing a crate, there is one less crate.

More Info
Pre and postconditions in code:

```csharp
private bool Precondition(string taskName, WorldState state) {
    switch (taskName) {
        case "Growl":
            return state.isPlayerInRange;
        case "Go to Spawn Location":
            return !state.isPlayerInRange;

        case "Shake":
            return !state.isRed;
        case "Stay Red":
            return state.isRed;

        case "Throw Crate":
        case "Throw Rock":
            return state.isHoldingObstacle;
        default:
            return true;

        case "Go to Nearest Crate":
            return state.numCrates > 0;
    }
}

private void Postcondition(string taskName, WorldState state) {
    switch (taskName) {
        case "Turn Red":
            state.isRed = true;
            break;
        case "Turn Purple":
            state.isRed = false;
            break;

        case "Pick It Up":
            state.isHoldingObstacle = true;
            break;
        case "Throw Rock":
            state.isHoldingObstacle = false;
            break;
        case "Throw Crate":
            state.isHoldingObstacle = false;
            state.numCrates--;
            break;
    }
}
```

Part of my world state class:
```csharp
public class WorldState {
    public bool isPlayerInRange;
    public bool isRed;
    public bool isHoldingObstacle;
    public int numCrates;
```

This is what the console printed about my tree:
Be a Monster
  Attack
    Growl
    Cooldown
    React Angrily
      Get Angry
        Shake

- Turn Red
  - Stay Angry
    - Stay Red
- Projectile Attack
  - Crate Attack
    - Go to Nearest Crate
    - Pick It Up
    - Throw Crate
  - Rock Attack
    - Go to Nearest Rock
    - Pick It Up
    - Throw Rock
- Idle
  - Cooldown
  - Go to Spawn Location
  - Turn Purple
  - Idle Behavior
    - Pace Back and Forth
      - Move Left
      - Move Right
      - Move Right
      - Move Left
      - Double Barrel Roll
    - Move in Circles
      - Move in Circle
      - Move in Circle