Inteligência Artificial para Jogos

2020-2021

Laboratory Guide

Lab 7 – Stochastic environments, multiple players and MCTS Optimizations

In this Laboratory, you will work on an extended version of the previous scenario, performing optimizations in the MCTS algorithm. However, you will also have to deal with a stochastic environment and other characters making decisions in the game. Life just got harder for Sir Uthgard.

But Sir Uthgard now has 3 new actions available:

- **Level Up:** Increases the level of the character when enough XP has been reached (10*current level XP required to level up). Leveling up will increase the maximum HP by 10, however it will not heal Uthgard. A character's xp is reset when he levels up. Skeletons give 3 XP each, Orcs 10 XP and the Dragon gives 20 XP.
- **Rest:** Utghard can now rest for 5 seconds, and recover 2 hp at the end of the rest.
- **Teleport:** Only available after Sir Uthgard attains level 2. Sir Uthgard calls upon the Deity of Helm to help him get out of trouble by teleporting him at the cost of 5 Mana to the relative safety of the starting position.

In addition to these actions, the **Sword Attack** action has been substantially changed for the stochastic version of the world and now has non-deterministic effects:

- When performing a sword attack, Uthgard now needs to perform an attack roll. Only when the attack roll surpasses the enemy's AC (armour class) the attack is successful in destroying the target. In short, Uthgard has a 90% chance to kill a skeleton with a sword attack, 70% to kill an Orc, and 50% to kill the dragon. If the enemy is not killed, Uthgard will have to attack it again meaning that in some cases Uthgard will have to perform several Sword attacks to kill an enemy.
- While performing the sword attack, Uthgard will receive damage from its enemies. The damage received is now stochastic. Skeletons cause 1D6 damage, Orcs cause 1D10 +2 damage, and the dragon causes 2D12[1] damage.

In the new version of the game (released in Laboratory 6), **enemies will now attack Sir Uthgard when he approaches them, and do more damage**. To deal with the enemies' initiative to attack in MCTS simulations, when simulating future states of the world, a check will be made to detect if the main character is too close to an enemy. When this happens, the next player to play in that state will be player 1 and the only action considered in that state is

[1] 1D6 corresponds to the roll of a six-sided die, 2D10 to the roll of two ten-sided dice.

attacking the main character with an enemy attack action (equivalent to a sword attack)[2]. If this does not occur, then the next player in that state will be player 0 (the main character) and the normal actions will be considered. This mechanism is already implemented, but you must be sure to always call the CalculateNextPlayer method after a new state is created and action effects are applied. See the example below.

```
WorldModel newState = parent.State.GenerateChildWorldModel();
action.ApplyActionEffects(newState);
newState.CalculateNextPlayer();
```

Also now Sir Uthgard must complete his objectives in 150 seconds. Time is of the essence.

1) Explore the the source code

a) Integrate the new source code provided into your Unity project. You only need to update the files under the Utils, GameManager and DecisionMakingActions directories.

b) Add the following properties to AutonomousCharacter.cs:

```
i)    public const float RESTING_INTERVAL = 5.0f;
ii)   public const int REST_HP_RECOVERY = 2;
iii)  public bool Resting = false;
iv)   public float StopRestTime;
```

c) Start by analyzing the new classes and changes to older classes. A new LevelUp action was added to the ForwardModelActions, as well as the EnemyAttack action for the enemies.

2) Implement Rest and Teleport

a) Create the new actions Rest and Teleport under the DecisionMakingActions folder. Add those actions to the action list in the AutonomousCharacter class. Implement the actions correctly (you can take a look at the GameManager to have a better idea of what you need to do.

3) Extend MCTS to handle the new stochastic environment

a) The MCTS algorithm can easily work with stochastic environments. One simple approach is to perform multiple playouts (instead of a single playout) when a new node is selected for expansion. Implement this mechanism and see the differences in behaviour.

4) Implement MCTS Biased Playout

a) Implement an optimization to the playout process that consists in biasing the random selection of actions. There are several options here. You can use heuristic information about the actions themselves or extract heuristic information from features of the child states to determine the best actions. Decide what approach and what heuristics will be used. Implement this in the MCTSBiasedPlayout class.

b) Try out the resulting behavior. You will quickly realize that you need to adjust some parameters. What are those?

[2] This is a trick to simplify the decision-making process of the enemy when running simulations.