Adam Gressen
CS 5150

Mario AI

Introduction

Making Mario AI appear human is not a simple task. In order to make him believably human I had to first decide what that means. I settled on the following goals:

Imperfection

Despite what my mother tells me, I am not perfect. Mario should not be either. He must strive for imperfection. This means he should leave bricks and coins behind, get hurt by enemies, and mess up jumps.

Whimsical

During a single playthrough a player may be juggling multiple goals simultaneously. I try to give Mario this same dynamic personality by giving him different targets throughout the game.

Win

Everyone loves to win. Mario should be able to eventually complete the level.

These goals are reasonable because they rely mainly on real time observations of the level. Essentially, Mario is always checking whether he should do all of his actions. This seems to be similar to how humans play the game, and allows the AI to choose the best decision at any given time.

AI Architecture

The scale of this project calls for a simple implementation. My first thought was to use a finite state machine, but I quickly realized that most states would have similar behaviors since there are not many options in terms of input. Ultimately, I decided to implement something that works something like a behavior tree. It is not complete anarchy, but I am hesitant to call it a behavior tree because it is not structured like a true behavior tree.

Structure

My AI is a series of functions that handle each of Mario's behaviors. They are TryLeft, TryRight, TryJump, TryFire, and TryQBrick. Some are pretty self explanatory. TryLeft and TryRight decide whether Mario should go left or right respectively. TryJump tells Mario whether or not to jump. TryFire

controls when Mario shoots fireballs and when he runs. TryQBrick allows Mario to target and hit question bricks. I have ordered them in the code so that the most important behaviors are lower in getAction. This allows each behavior to run, but emphasizes the final behavior.

Decisions, decisions

As Mario goes through the level he is always trying to break bricks. If he misses a brick, he does not worry about it…...unless it is a question brick. He wants to break every question brick. Mario jumps if there is something in his way, but there is also a chance that he will randomly jump as he progresses through the level. Mario is usually moving right, but will head left if he missed a question brick or there are several more enemies to his right than his left. If an enemy gets past him then Mario will usually hunt them down and jump on them. This presents a good example of how my implementation accomplishes believability. When Mario heads back to hunt down an enemy or hit a question brick he continues to jump randomly and also targets regular bricks. His actions are consistently believable because all of his behaviors are always running.

Room for Improvement

Obviously my implementation leaves room for improvement. I could have written a true behavior tree, which would have been more extensible and easier to modify. I shied away from this for two reasons:

1. Code overload – It would have required a lot more code than I ended up writing. I wanted to keep this short and sweet.
2. Overcomplicated – My behavior tree would have had maybe three behaviors. This number of behaviors does not necessitate construction of a tree in my opinion.

That being said, here are areas in which I could have improved my implementation:

1. Organization – If I had created a separate class to hold helper functions it would have been much cleaner in TuringAgent.java.
2. Complexity – Sometimes simplicity is great. However, if I had implemented a true behavior tree than I would have been able to make my AI more complex. I cannot claim that this would have made the end result better, but it would have potentially given me more opportunities to explore other aspects of Mario's personality. For example, maybe Mario gets frightened when he is in tiny mode, and if he is frightened then he would not chase down enemies that get past him.

Performance

My AI seems to perform well and has fulfilled all of the goals I outlined above.

Successes

If the difficulty is 0, my agent can definitely get to the end of the level. My AI is a Level 9 Goomba-Slayer. They really do not offer him a challenge at all. Mario is a greedy guy. He is able to get

a good amount of the bricks, including all of the question bricks. However, he also leaves some behind because he's not perfect.

Failures

I spent many hours trying to target fire flowers. However, I could not get Mario to successfully retrieve them every time. He would often get stuck attempting to jump for one. I removed this from the final version of my AI because I didn't want to submit anything that would be such an obvious giveaway that the player isn't human. Additionally, sometimes Mario fires too quickly. I threw in a dash of randomness to try to slow him down, but this does not always fix the problem. Mario is also not very good at dealing with gaps, which causes him to lose some levels that a human would probably be able to beat.

Conclusion

I fulfilled my goal of developing a human-like Mario AI. He can't beat the harder levels, but Mario looks believable under most circumstances. Mario makes mistakes, performs some random actions, and has different targets throughout the level. I used a very simple AI architecture based on the functionality of a behavior tree.