Members

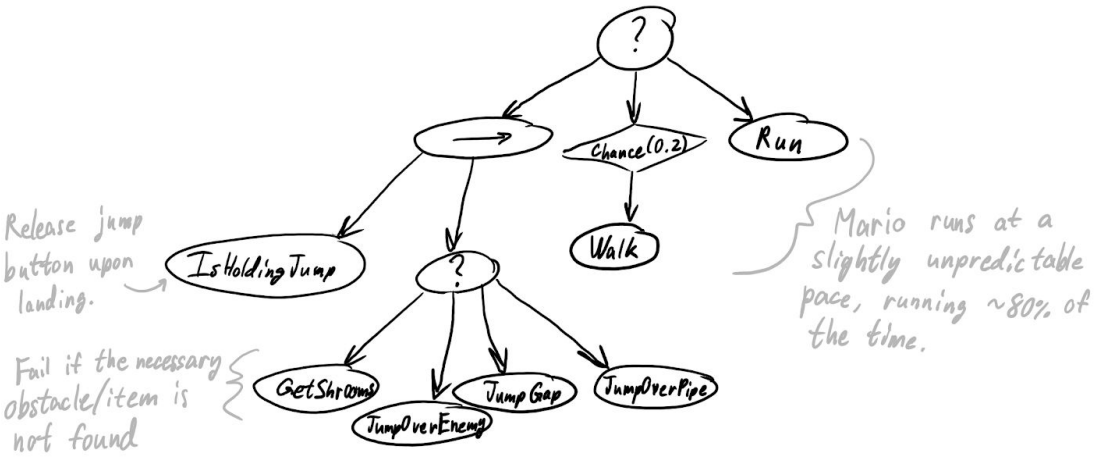
Neer Jariwala
William Lucca

Goals

The main behavior goals for our agent were to be able to jump over enemies, obstacles, and gaps. These behaviors were implemented using a behavior tree with a selector. We chose to focus on these behaviors as they were the most common situations found in a Mario level. Also, with our own experiences playing Mario, we felt that avoiding enemies and traversing the level were the two most important parts a player must keep in mind when playing a level. Overall, we felt we were able to achieve these behaviors. However, the behaviors do not perform exactly as we desired.

Technique

To implement our agent, we used a behavior tree. The tree used a selector to cycle through a list of “tasks” until it successfully completed one. Our tasks included a variety of actions: GetShrooms, JumpOverEnemy, JumpGap, JumpOverPipe, Run, and Walk. The former four were implemented in their own selector node in that specific order. In this way, we were able to control the priority of each task should multiple tasks be successful. For instance, we made getting mushrooms the highest priority, so if a mushroom is found, the agent will go after it before leaping over the subsequent enemies and pits. If there are no obstacles or items, the agent will either walk or run to continue moving forward.



Testing Process

We developed and tested our agent in parallel. Using some of the predefined levels from the framework, we tested each of our different tasks. By manipulating the priority of each type of task and how far the agent looks around it to detect obstacles and items, we were able to produce a behavior that looked more human. After we ensured that these actions were behaving correctly and able to dodge obstacles and pick up items, we focused our attention to a specific level (1-1 from the original set). Here we tried a handful of random seeds for the randomized portions of our tasks, until we settled on a seed that produced a runthrough that looked most believably like a human player.

Pros and Cons

One pro of our behavior tree implementation was how easy it was to build the tree itself once the possible actions had been created. The structure of sequence nodes and selector nodes made the agent's priorities very transparent, and simple tweaks in the ordering of actions could produce noticeably different behaviors. In addition to this, we were able to use a decorator node to add a random probability for an action to fail to ensure the agent is not too predictable.

One of our major cons is the AI's ability to not function perfectly with gaps. There are times where the agent jumps a gap right into an enemy, something a player can usually avoid. Another major fault of our agent is its inability to detect platforms that change in elevation greatly. If a level requires a player to act with extreme precision, there is a good chance our agent will not be able to successfully complete the level. Also, since we used a delay between each time the tree ran, there is a chance that the agent will not be able to make a decision quickly enough and make an error that a human would not make.

Something we would do differently is change how we detected enemies and gaps. Our current system is able to succeed, but fails in certain situations. As such, we would want to create a Task capable of predicting further ahead to test if an enemy or gap is safe to jump over.