Interactive Automated Sword Fighting Simulations with Game Tree Search

R. Christine Uyemura
Faculty Advisor: Dr. Stephen Lane

MOTIVATION

Scenes with many characters battling can be time intensive and costly if all of the animation is done manually, and completely automating the animation process runs the risk of creating animations that are noticeably similar. This project proposes a method for efficiently producing automated animations that are capable of interactivity.We implement an algorithm that has been successfully used to simulate other zero-sum games such as tic-tac-toe and checkers.

OBJECTIVE

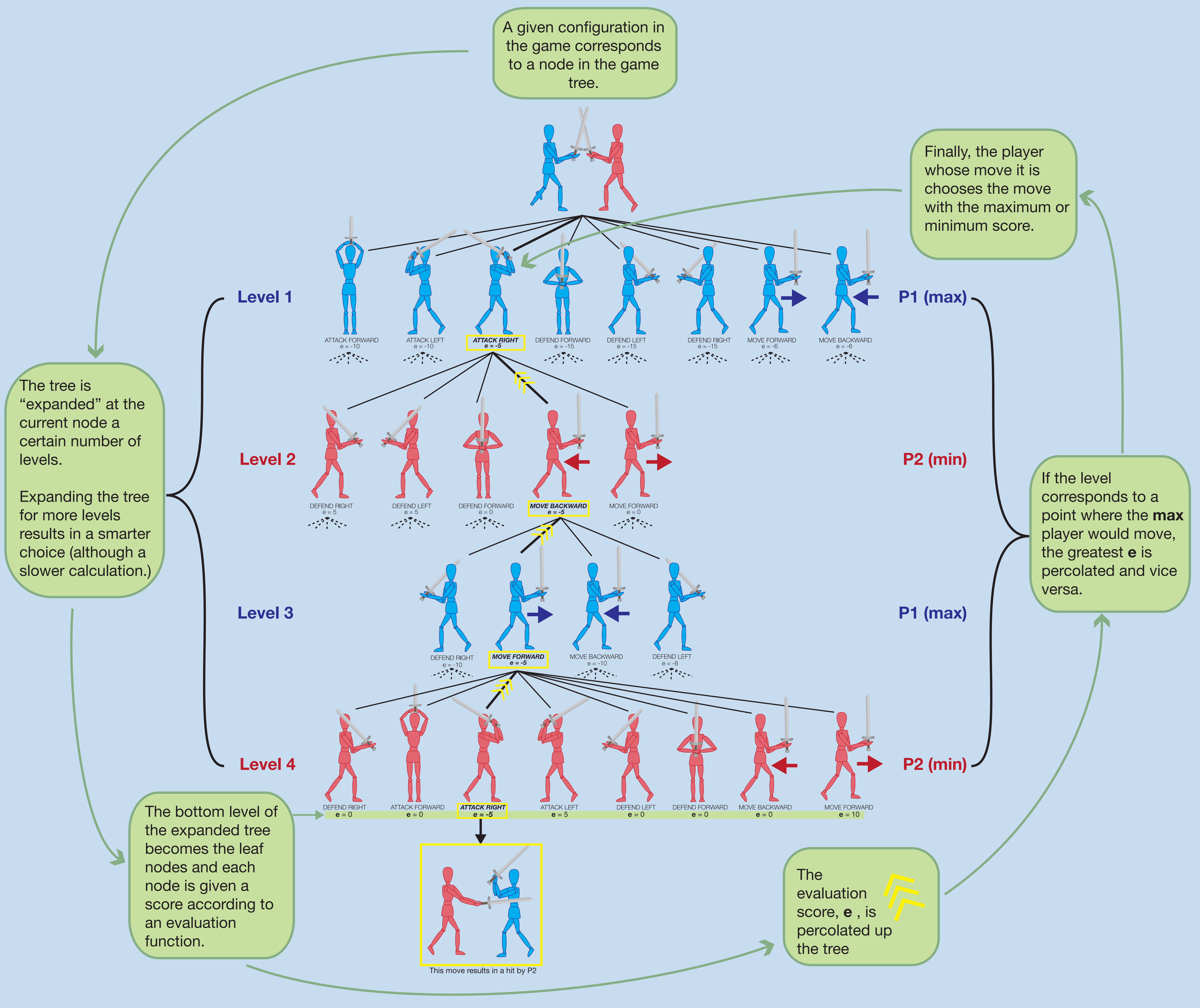
Apply the Minmax Game Tree search algorithm, which has been used to efficiently and intelligently simulate play of other zero-sum games, to a game of swordplay. Create sword fighting animations that can be automated or interactive.

METHODOLOGY

A character is built in the Unity Game Engine that is able to perform the desired motions. There are three attack motions (left, forward, right) and three defend motions (left, forward, right) in addition to step forward and step backward.

The game tree algorithm is implemented as shown. A game configuration consists of the current position and motion state (what move the player is performing, if any, and for how many seconds) of each player. The evaluation function is determined heuristically. It takes into account these factors and also adds in randomness for the sake of creating imperfect players. Motions can also be pruned from the tree if a player is not within the correct range of the other player to perform the motion.

IDEA



A given configuration in the game corresponds to a node in the game tree.

Finally, the player whose move it is chooses the move with the maximum or minimum score.

The tree is "expanded" at the current node a certain number of levels.

Expanding the tree for more levels results in a smarter choice (although a slower calculation.)

If the level corresponds to a point where the **max** player would move, the greatest **e** is percolated and vice versa.

The bottom level of the expanded tree becomes the leaf nodes and each node is given a score according to an evaluation function.

The evaluation score, **e** , is percolated up the tree

This move results in a hit by P2

Senior Project Poster Day 2013