Before:

if ((i - states.Count) % 2 == 0){

values.Add(-((states.Count - (states.Count - i) % 2)) / (float)states.Count);

}else{

values.Add(((states.Count - (states.Count - i) % 2)) / (float)states.Count);

}

After:

float weight = ((states.Count - (states.Count - i) % 2)) / (float)states.Count;

if ((i - states.Count) % 2 == 0){

values.Add(-weight);

}else{

values.Add(weight);

}

Normalized the calculation, since it’s practically the same calculation, just with one being negative.

Before:

writer("(0) for First Row, (1) for Second Row, (2) for Third Row");

int row = int.Parse(readLine());

writer("How many pieces do you want to take away?");

int num = int.Parse(readLine());

After:

int row = promptForInt("(0) for First Row, (1) for Second Row, (2) for Third Row");

int num = promptForInt("How many pieces do you want to take away?");

private int promptForInt(String prompt){

bool success = false;

int userInput = 0;

do{

writer(prompt);

int.TryParse(readLine(), out userInput);

if (!success){

writer("That's not a number, try again.");

}

}while (!success);

return userInput;

}

Extracted a method and made it much more resistant to user stupidity.

Before:

if (bestMove == -1)

{

bestMove = numState;

}

else if (game.stateList[numState].getWeight() <= game.stateList[bestMove].getWeight())

{

bestMove = numState;

}

After:

if (isBestMove( game, numState, bestMove))

{

private bool isBestMove(GameManager game, int numState, int bestMove)

{

return (game.stateList[numState].getWeight() <= game.stateList[bestMove].getWeight() || bestMove == -1);

}

Consolidate conditional expression

Before:

int numrowsToMove = 0;

if (game.stateList[numState].TopRow < game.rows[0])

{

numrowsToMove++;

}

if (game.stateList[numState].MidRow < game.rows[1])

{

numrowsToMove++;

}

if (game.stateList[numState].BotRow < game.rows[2])

{

numrowsToMove++;

}

After:

if (numRowsToMove(game, numState) == 1)

{

Change: Consolidate conditional expression