Timothy Chu

George Ferguson

WRT 105

15 February 2016

Project 1 Write-up

Note: I wasn't quite sure how to do a write-up for a computer science project, so I drew a lot of inspiration from this website

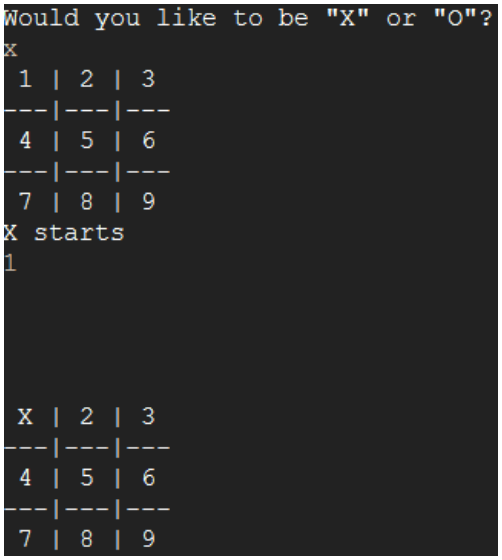
http://www.cs.ucl.ac.uk/teaching_learning/msc_cgvi/projects/project_report_structure/

Introduction/Background

Well, this is a program that plays Tic Tac Toe. A user plays against an AI coded in the project. We needed to implement the Mini-Max algorithm in order to give the AI some intelligence in choosing the next move. I have worked with the Mini-Max algorithm in the past. I used it to make a program that plays Connect 4. This was back in high school, and if I recall correctly, it did not function properly. Looking back at the project, the most difficult part of the project was making the ACTUAL Tic Tac Toe game. I saw making a program that allows two human players to play Tic Tac Toe as the first hurtle in completing the project. My problems with coding the Tic Tac Toe game became nine times harder with "super" Tic Tac Toe. I had to figure out how to have my code take in numbers as well as display the actual nine boards. Coding the AI was an entirely different story.

With my AI, I began by pulling up my old code from my failed Connect 4 mini-max driven program. I went ahead and copy-pasted it into my Tic Tac Toe program. Of course, it did

not work immediately. Of course, I changed all of the Connect 4 related functions to point to Tic Tac Toe things. I needed to change my heuristic function as well as somehow find a way to have it work with the nine board variant of Tic Tac Toe.



Analysis and Design/ Implementation

Again, I began with my mediocre Mini-Max code taken from my Connect 4 program. Before that, I needed to code a program that allows two human players to play Tic Tac Toe. Professor Ferguson mentioned in class that he would recommend using Enums to represent the tokens/players on the Tic Tac Toe board. At first, I went with that design, but I soon found out that it would give me headaches when it came time to write the user I/O. I wanted some way to display both the current tokens(marks, X/O) on the board as well as the numbers that corresponded to each position. I ended up creating an array of Strings. I also wrote a small piece of code that would translate a number to its corresponding position in the array. From there, it was relatively smooth sailing. I wrote a method that would estimate the heuristic of any given board in terms of the AI. I assigned a score based on the number of blanks and marks on the board. In addition, I returned some really high number whenever either player wins in any given board. With that, I finished the “basic” Tic Tac Toe. Yes, I could have used a “responsive” AI for the “basic” Tic Tac Toe for the sake of simplicity, but I went with utilizing Mini-Max for the basic Tic Tac Toe program in order to reduce the amount of code I needed to write for the Nine Board Tic Tac Toe. Yes, I did struggle at first with that implementation, but I did eventually succeed after playing around with my Mini-Max algorithm.

When I started my Nine Board Tic Tac Toe program, I literally began by copying and pasting my code form Basic Tic Tac Toe into a new project folder in Eclipse. I again started from the ground up. I first made the Nine Board Tic Tac Toe game. At that point, it was human vs human. After changing my "mark" function, Eclipse had errors showing up all over my Mini-Max method as well as my heuristic evaluation method. This gave me a starting point to change my basic Tic Tac Toe program into the Nine Board Tic Tac Toe variant. In the end, I only had to add more parameters to methods and method outputs, as well as some more for loops to accommodate for the nine boards.

```
public void mark(Player p, int b, int j){
    currBoard=j;
    int i=0;
    for(; j>3; j-=3){
        i++;
    }
    j--;
    if(board[b][i][j]==Player.Q){
        switch (p) {
        case X:
            board[b][i][j] = Player.X;
            v[b][i][j]="X";
```

Testing

In order to figure out if my program was actually working or not, I played Tic Tac Toe. I played it a lot. Actually testing the program aided me a lot with fixing my Mini-Max algorithm. This applies for both programs. For the nine board tic tac toe, I wasn't sure of the strategies involved, so I ended up downloading an app for it on my phone and having my program play against that. Yes, it was time consuming and quite annoying to do this, but it was probably the easiest way to make sure that the mini-max algorithm was functioning properly. It was able to win against the nine board tic tac toe app on my phone, so I assumed that it was working!

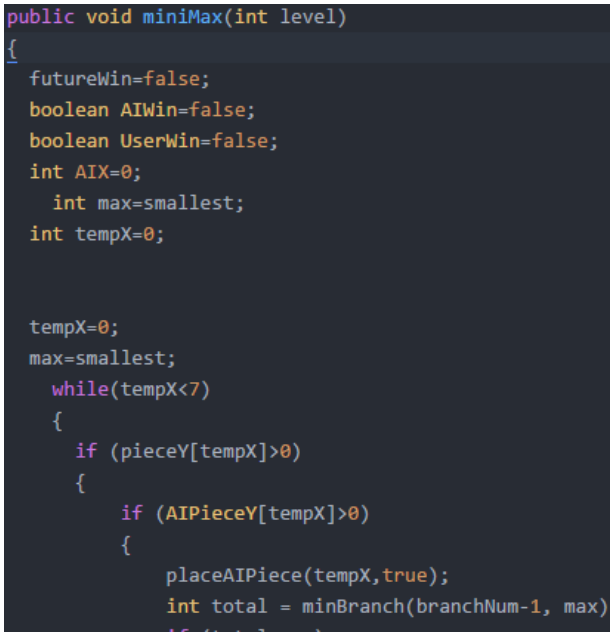
```
Utility of AI move is 9950



  (1)   |   (2)   |   (3)
 1 X 3 | 1 2 O | 1 O X
 4 5 6 | X 5 6 | 4 5 6
 7 8 9 | 7 8 9 | 7 8 9
-----------------------
  (4)   |   (5)   |   (6)
 1 2 3 | 1 2 O | 1 2 3
 X O 6 | 4 X O | 4 5 X
 O 8 9 | 7 8 O | 7 O 9
-----------------------
  (7)   |   (8)   |   (9)
 1 2 3 | 1 2 3 | 1 2 3
 O X 6 | 4 5 6 | 4 X 6
 7 8 9 | X 8 9 | 7 8 9
O won
```

Conclusion

As usual, there is always room for improvement. Looking back, I wish I hadn't copied my code from my old Connect 4 program. I felt as if I spent more time actually reworking it to make it function with my Tic Tac Toe program than it would have taken for me to write the minimax code from scratch. There is almost always room for optimization. I feel as though I could have improved upon my heuristic function in my nine board tic tac toe program if I were more familiar with that version of the game. Just like every other student, *I wish I had started sooner.*



```
public void miniMax(int level)
{
  futureWin=false;
  boolean AIWin=false;
  boolean UserWin=false;
  int AIX=0;
    int max=smallest;
  int tempX=0;


  tempX=0;
  max=smallest;
    while(tempX<7)
    {
      if (pieceY[tempX]>0)
      {
          if (AIPieceY[tempX]>0)
          {
              placeAIPiece(tempX,true);
              int total = minBranch(branchNum-1, max)
```