CS4200/5200 Artificial Intelligence Method

Program #2 Computer Science Department
40 points Bowling Green State University
Fall 2017

Due date: November 09, 2017, Thursday
Time: Electronic submission of program files due on Canvas by **6:00PM** on the due date

This program assignment is designed to help you learn about alpha-beta pruning using a zero-sum two player game, Othello. **Othello begins with four disks in the center, two black (i.e., player's) and two white (i.e., computer's), on an 8-by-8 board. The black goes first. A player can put a disk of their color anywhere on the board that meets the following two conditions: (1) there must be a disk of the opponent's color directly adjacent to the square in which the disk is place and (2) on the other side of the opponent's disk, there may be more disks of the opponent's as well, but they must end in another disk of the player who is making the move. That player then flips all of the opponent's disks from the point where the move was made until the next disk of the player's color. This flipping is done in all eight directions (i.e., up, down, left, right, and diagonals). The player with the most disks at the end of the game wins.** The program using graphical interface must be developed using the OpenGL graphics library with GLUT on our departmental Windows machines using Visual Studio (i.e., test your program on the machines before submitting it).

Assignment: In this assignment, you will develop a player-versus-computer Othello game. This is an individual assignment.

Details:

- The computer's move should be done **using the alpha-beta pruning algorithm**.
- You are free to choose your own evaluation function. However, it cannot be just the difference of the disk counts in different colors. You must consider the fact the edges and the corners of the board give a player huge advantage over the opponent. You must include the details of evaluation function you chose in the banner.
- Passing option is not allowed unless no move is possible.
- You are free to choose any other parameters (e.g., maximum search depth) for your program. However, those must be stated clearly in the banner.
- When the "Q/q" key is pressed, exit the game.

- Students have two options for the game interface: (1) text board game and (2) graphical board game. (The graduate students must use Option 2.)

- Option (1) The text board game is to show the state of the game on the standard output screen. Use "O" for black piece player and "X" for white piece player (i.e., computer). Use "-" for empty space. Use these symbols in 8 x 8 matrix/array form to show the state of the game. The player makes his/her move by specifying the row and column index using standard keyboard input. Illegal moves should not be allowed.

- Option (2) The graphical interface is to use OpenGL interface. The display is to be pixels 512 in height by 712 in width. The display is to be divided into two areas, separated by a vertical dividing line. The dividing line should be at 200 pixels from the right of the display. The left area of the display (512x512) is the main game board and the right area is used to display the current number of disk for each player. The player makes his/her move using left-mouse click on the board. Illegal moves should not allowed. The flipping of the disk may be animated (bonus); each disk will be flipped (i.e., gradually change the color of the disk one to another) sequentially in a reasonable time.

Extra points: If you want to add extra features to the program, a small number of extra credit points may be awarded. Some examples of features are: right-mouse button click will void the player's last move or implementation of a "RESET" button in the right side of the display. Undergraduate students implementing the graphical interface will receive bonus points.

Structure and Documentation Note: The program should have a modular design and a reasonable amount of documentation. This means that major/important features, functions/methods, and data structures should be very clearly documented. Remember, you can reduce some documentation work that might otherwise be needed by careful choice of variable and function/method names. Object-oriented approaches must be very clearly described; careful choice of object names is not sufficient for other readers of your program to understand its structure. Programs are also expected to be reasonably efficient.

The initial comment section of the program should include a concise description of the architecture of your program. Your goal in the initial comment section should include providing a concise description of the program's methodology (including any critical functions and data structures) to the grader. Anyone reading the initial comment section should be able to very quickly understand the structure of the program.

Building OpenGL programs: If you want to use an object-based approach, you are free to minimally modify the template program to work with your object-based framework. Please, no use of STL for C++ program unless you get the instructor's approval. You are restricted to NO MORE THAN **TWO** header files and only **ONE** C/C++ file.

Submitting the Program: You will need to submit all program source files (i.e., .c, .cpp, .h files), but NOT Visual Studio (VS) files (e.g., don't include entire VS project files). In addition to the program files, submit a separate MS Word document which includes copy of the source codes and screen captures of your outputs. (Don't zip files into one file.) No email submission will be accepted. **Please note that submissions not following all the instructions will get penalized (e.g., may result in 0 credit).**

Grading the Program:

Functionality	29 pts.	
Documentation	5 pts.	
Structure	3 pts.	
Efficiency	3 pts.	
---		
	40 pts.	(+ Extra)