Lab 8: 3D Tic-Tac-Toe

Overview:
Khan Academy has a great video that shows how to create a memory game. This is followed by getting you started in creating a tic-tac-toe game. Both games use a 2D grid or array to represent the game state.

In this lab, you will be creating a 3D game of Tic-Tac-Toe. However, you will NOT be doing 3D graphics. Think of 3D Tic-Tac-Toe as just 3 different 2D tic-tac-toe games. You will draw three 2D boards on the canvas and the player will make a move each turn in one of the three boards. A player wins when getting three-in-a-row in the usual way on any one of the three levels. A player can also win by getting three-in-a-row vertically, either in a single column or on a diagonal. Additionally, the middle space on the middle level is banned. For example, each board below shows a win case for player 0:

You are free to choose the style in which you draw the three layers of the 3x3x3 3D tic-tac-toe board.

The two on the left are examples I coded in JavaScript. The first is very basic, but meets all the requirements. The second has more style, and is drawn using just the 2D JavaScript/Processing primitives we have been working with this semester. The X and O sprites are images I created in Photoshop. I have posted the image files on the website. If you want to use images, you may use those or use some third party asset or draw your own.

The second example highlights the winning cells with a darker background and brighter, thicker boarder. When the player or AI wins, your program is required to recognize and report the win, but it is not required that there is a graphic element to the reporting.
Grading Rubric [20 points total]:

[Web Site: 3 point]: In Blackboard Learn, submit a link to a web page you create that runs your working game.

[Drawing the Board: 2 points]: Draw a set of 3 tac-tac-toe boards with the middle board missing the middle spot.

[Getting Input: 2 points]: When it is the user’s turn, and the user clicks on an empty spot, your board draws an X in that spot. Be sure to use the Khan Academy video on the memory game as it will help greatly with this.

[No Cheating: 3 points]: Each AI turn, your AI must make a legal move and the game must always prevent the player from making an illegal move (For example, an illegal move is to move in a location already taken or in the center location of the middle board).

[Recognizing a Win: 4 points]: Your game must recognize when either the player or the AI has attained three-in-a-row, must report the winner and must query for a new game.

[Taking a Win: 3 points]: Your AI does not need to be super smart; however, if the AI can win on its current turn, then the AI must make a winning move.

[Blocking a Win: 3 points]: If your AI cannot win in a single move, and the player can win on his or her next move, then the AI must block at least one possible three-in-a-row that the player could make.
Extra Credit:

[Getting Smarter: +5]: Make your AI significantly smarter than the base requirement by adding a key additional strategy.

[Wizard: +5]: After earning the +5 for Getting Smarter, make your AI even smarter by adding yet another key strategy.

[Polish: up to +10]: A normal credit program needs to draw a board with lines, and shapes that get drawn in response to user actions. A well-polished program has the perfect lines and the perfect shapes that are summoned into existence not just when required, but with style to really make the user experience pop.

[4×4×4×4: +20]: Replace the 3×3×3 game requirements with a program that plays 4×4×4×4 Tac-Tac-Toe AND teach it to play at the Wizard level as defined above for the 3×3×3 game. Note, 4×4×4×4 Tac-Tac-Toe does not have any banded spots on any levels.

The 4×4×4×4 layout shown above is not required for this extra credit option. You are free to come up with your own layout. This example layout is nice in that it gives the 3D appearance without using any 3D graphics: The vertical are drawn at an angle on top of a solid blue rhombus with simple, slightly slanted images for the Xs and Os.