Connect 4 AI Game

Daniel Shandley

Candidate Number: 0064

Centre Number 64395

Godalming College

Analysis

Background

In today's world more and more people have less and less free time, this means that people want to make the most out of that time and might not play a game of Connect 4 when it is offered by another person, not only that but if they did agree to play they might not be up to the standard of the challenger. The solution to both problems is to investigate if making an AI Connect 4 player is possible.

(Connect 4 is sometimes known as: Captain's Mistress, Four Up, Plot Four, Find Four, Four in a Row and Four in a Line. I will always call the game Connect 4)

Here is some basic information on Connect 4.

Connect 4 is a game involving 42 discs, 21 yellow and 21 red.
The game is played in a grid of 7 by 6 holes placed vertically with a sliding gate at the bottom.
The game is played by two players.
Each player is given 21 discs of either yellow or red.
When a disc is dropped over a column it will fall until it hits the gate at the bottom or another disc.

Rules of Connect 4:

A player is picked at random to start.
The players will alternate dropping one disc of their assigned colour.
A when a disc is dropped it must be in a grid space and not wedged in between two rows or columns.
If a column is full then a dis cannot be dropped in that column.
Once a disc is in a valid location it cannot be moved until the end of the game.
The winner is the person who can create a straight line of their own colour, that line can be horizontal, vertical or diagonal.
The game is a draw if all 42 spaces on the grid are filled and there are no lines of four of the same colour.
Once the game has been won all the discs should be removed by sliding the small tag connected to the gate at the bottom to the left or right depending on which side of the grid you are on.

Can and cannots of Connect 4:

Two discs cannot occupy the same space on the grid.
Discs can only be put in from the top slots.
Discs can only be dropped into a grid space when flat against the same plain that the grid is on.

This is what the grid and discs look like:

Strategies:

A player should place a piece with the intention to either block the opponents four or build your own four.
If there is no way to win then try to draw.
The if you are starting then always choose the centre as it is the strategically most sound position due to it having the most possibilities for a connect 4.
Don't start with piece in the corner as you can only make a connect 4 diagonally away from the side and vertically.
When playing against a human they will not be able to see all the future possible positions of the discs so you can plan ahead and create what are essentially traps, they will force your opponent to either let you win quickly or help you win by using one of their discs to make a connect 4.
If you are playing against a player who is not looking into future moves, then as long as you get down two of your discs next to each other and have two spaces on both sides then you have a guaranteed win.
If you are ever able to create a 7, that doesn't touch the top or side, safely then do so because it will mean a garneted win.
Equally so you should look out for those tactics in your opponent's move and avoid falling for a trap.
There is a way to win every time if you start, but it requires you to see all possible outcomes, this is because Connect 4 is a solved game.

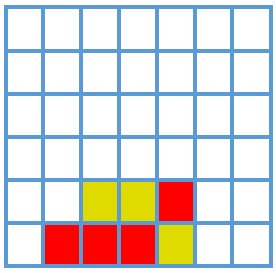
Connect 4 is a solved game as of 1988 (not called Connect 4 at the time) by James Dow Allen, this means that if you play the first move you can always win no matter what, as long as you start, place your disk in the centre and see all future possible outcomes (probably

only possible by a computer). When both players follow the perfect moves, the player would have just played what is called a perfect game.

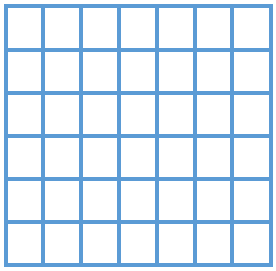
I have chosen this investigation because artificial intelligence and machine learning are extremely helpful and will become more important in coming years. Connect 4 is a simple game that has 1905333170621 different possible winning states, to make a program to play Connect for would be quite interesting to do and see whether it can play a perfect game.

Board States

(Red starts)



In this situation, it is obvious that yellow would place a disc in the far left otherwise, they would lose.



Other game rules:

The game of Connect 4 is usually played in the way talked about above, but there are a few different rules.

Pop Out: Has the same rules as standard Connect 4 but allows a player, on their turn, to remove, or pop out, one of their discs from the bottom row, hence the name. By removing a piece from the bottom, all of the pieces above the aforementioned piece move down a row, this changes the dynamic of the grid and adds more possibilities in gameplay.

Pop 10: This game is vastly different form Connect 4. Each player takes a turn at placing a disc in the grid, a disc must be placed in the lowest spot on the grid so that a row is filled and the discs can then be placed on the next one, if a four in a row is made it does not mean the game has been won. Continue to take turns for placing discs until the grid is full. Once it is full then the person who placed the first disc will take a disc of their own type out of the bottom, if the disc was part of a four in a row then the disc is kept and you get to draw

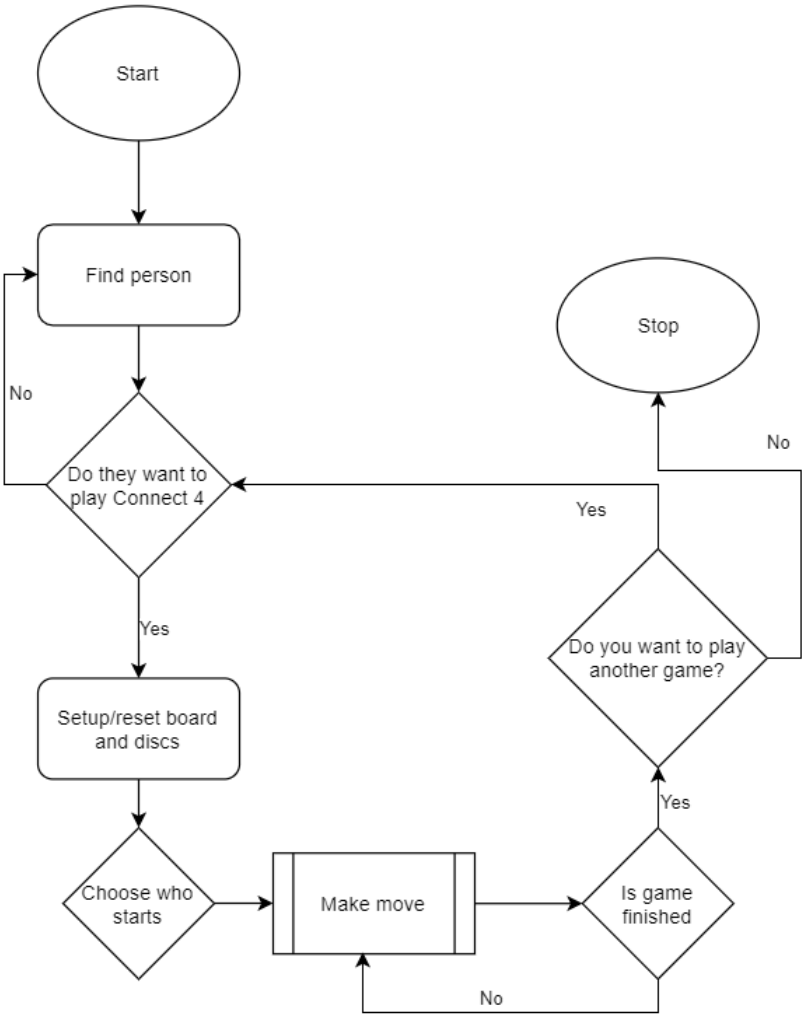
another disc, if not then it is dropped down any open column. Each player will repeat this process taking turns until one of the players have collected 10 of their discs, at that point the person with those 10 discs wins.

5-in-a-Row: The same rules as Connect 4 but five discs must be joined and it is done in a grid of nine columns and six rows. This game mode would expand the possible game play options by a huge amount.

Power Up: This game mode has the same rules as Connect 4 but has a special disc that has an anvil icon on it, when the anvil token is placed then all of the discs below it will be removed leaving the anvil piece at the bottom, this piece would be a perfect counter move against an otherwise certain loss situation.

Current system

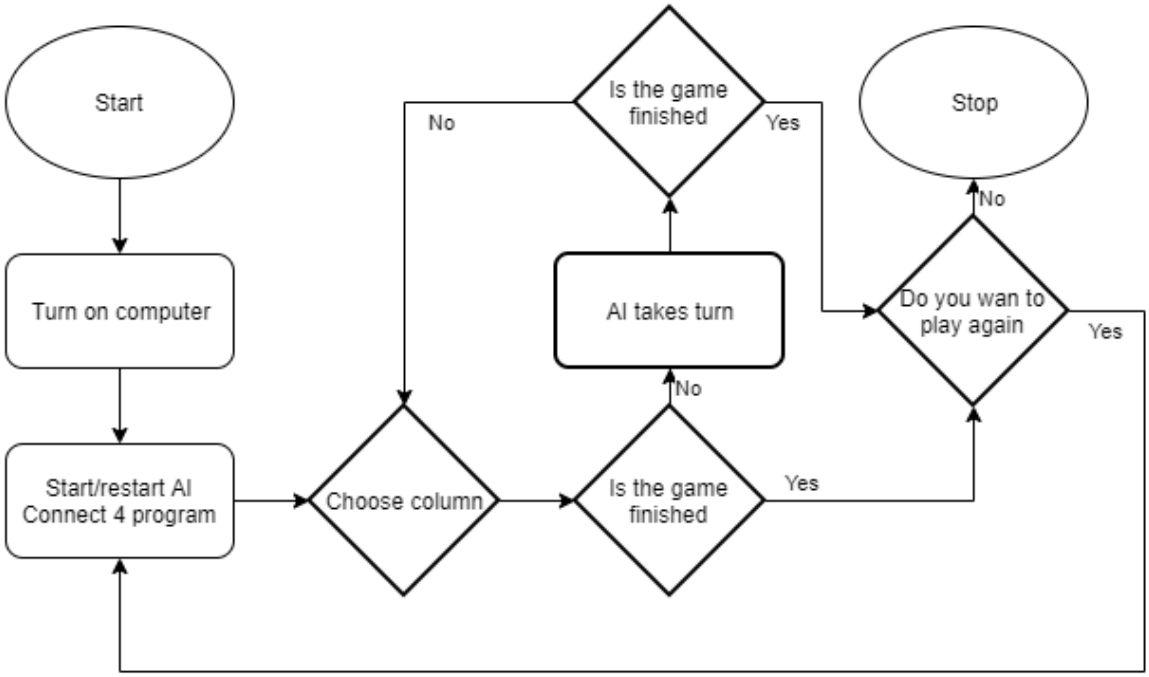
As things are right now the way that a parson plays Connect 4 goes like this:
1) Finding another person who is available
2) Ask that person if they want to play a game of Connect 4
3a) If they say no then restart from step 1
3b) If they say yes then set up board and distribute discs
4) Play game
5)Once the game has ended then if you want to play another game then repeat from step 2, if not then end.
Flowchart:



The flow chart shows the process of getting a player and then playing with that player, the person must go through multiple different processes that might not come up in the same thing but done with an AI.

New AI included system

Flowchart:



This new system shows that there is no uncertainty when it comes to whether you will be able to play a game of Connect 4 with a challenging opponent. Investigating if an AI Connect 4 player is possible is not just an interesting question but will also help someone who want to play Connect 4 but might not have the opponent to play it with.

End User

This program that I will write is to show the examiner that a game of Connect 4 can be played with an AI, that AI can be better or just as good at playing Connect 4 as the user. The End user is a person who wants to play connect four against an opponent who has a range of difficulties, or someone who wants to play connect four and has nobody to play with.

Acceptable Limitations

System limitations

I will need to learn some knowledge to make the final project, the best way to do so is to break up the program into separate sections, Connect 4 game and Minimax code.

1st Program:

I will create a Connect 4 game in Visual Basic in OOP design. It will allow two human players to play Connect 4.

2nd Program:

Create a Tic Tac Toe game played between two people in Visual Basic in OOP design.

3rd Program:

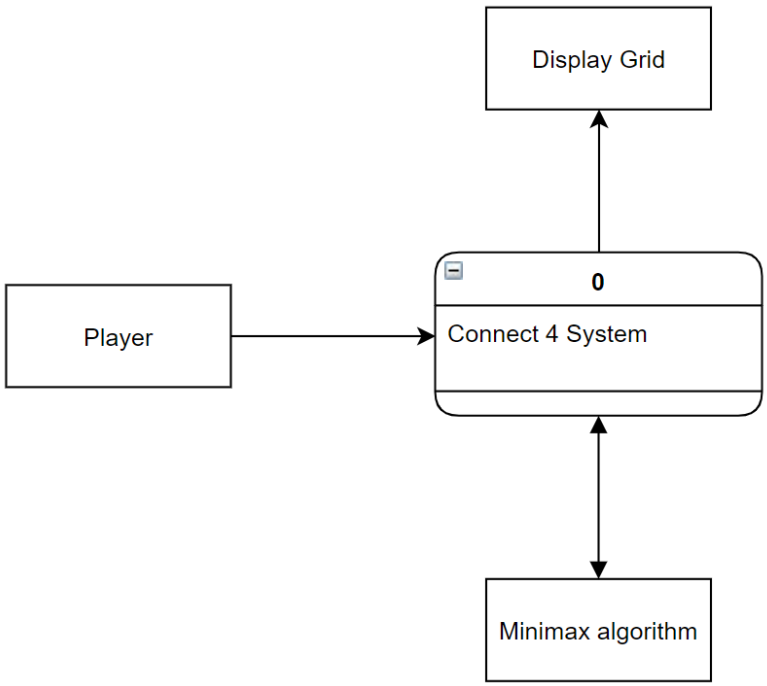
The Tic Tac Toe game will have the second player be replaced by a Minimax algorithm.

4th Program:

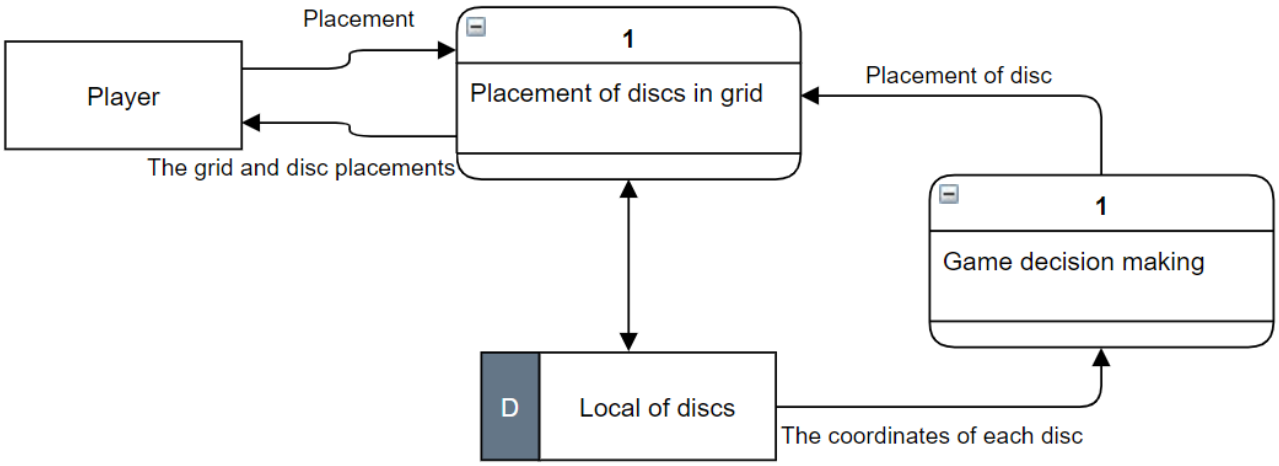
This program will be the combination of a Minimax program and my Connect 4 game. This will be the result.

DFD:

Level 0:



Level 1:



Requirements

1. Provide the user with an AI with changeable difficulty to play Connect 4 with
   1.1. The user must be able to play connect four with the AI at any time.
   1.2. The AI will have different difficulties that the user will choose from.
   1.3. The AI will operate by using a Minimax algorithm
      1.3.1. The algorithm will place a disc depending on the current disc positions.
      1.3.2. The algorithm will place a disc depending on the predictions of the future moves.
         1.3.2.1. The algorithm's decision that is dependent on the predicted moves, it will aim to get a win at the earliest point assuming that the opponent plays perfectly within its own perception.
         1.3.2.2. If the algorithm finds that it cannot win then it will take the longest route to a loss or will try to draw.
   1.4. The game of Connect 4 will be a fresh game with nothing in the grid whenever a new game is started.
2. The user interface should be simple enough that a child can figure out how to play.
   2.1. The menus will be clear and simple
      2.1.1. Every menu will have text that is written in large clear text.
      2.1.2. The main menu will give you the option to either play Connect 4 with an AI or load a saved game.
      2.1.3. The AI difficulty menu will allow the user to choose the difficulty of the AI or go back.
         2.1.3.1. The choices of difficulties are: Easy, Moderate, Medium, Hard, and Unyielding.
   2.2. Playing the game must be simple to understand.
      2.2.1. To the side of the grid will be an information box telling the user how to place a disc.
      2.2.2. The grid and discs will be blue, yellow and red to make it clear that they are different entities.

3. The program must not have long processing times.
   3.1. The AI decision-making process should not be long, less than 10 seconds.
   3.2. The placements of the discs should not take long to process, less than 3 seconds.
4. Should be able to run on a standard desktop computer.
5. The user should be able to place a disc only during their turn in allowed columns.
   5.1. The column will have an indicator, possibly an arrow that when clicked will place a disc at the lowest point on the grid.
      5.1.1. The disc can only be placed in a grid space that is not occupied.
      5.1.2. A disc can only be placed in a space above a disc or grid bottom.
   5.2. If a disc cannot be placed in a certain column then there will be a cross that will indicate that that column on the grid cannot be played.
      5.2.1. When the cross is clicked an error message will come up to say that the player cannot place their piece in that column.
   5.3. When it is not the users turn the user cannot place a disc and must wait for the AI to end their turn to then place their disc.
6. The user should be given the ability to see the rules without having to search for them.
   6.1. During the game there will be a toggle on the side that when clicked will show the rules of Connect 4.
   6.2. When clicked again the tab will disappear.

Requirement explanation

1. The purpose of this is so that the investigation is what it has been set out to be.
   1.1. This is half of the investigation
   1.2. The difficulty will be determined by how many moves it will look ahead to.
   1.3. I could use other algorithms, but I have chosen a minimax algorithm because it seems to be easier to code.
2. If the user interface is complicated, then the effort to play a game of Connect 4 real life might be less therefor making the feature of being able to play by yourself useless. The program will be written in Visual basic using forms.
      2.1.4. By having the option of different difficulties any user can enjoy playing since they can have a chance at winning while being challenged.

      2.1.3.1 The difficulty is determined by how many steps ahead the algorithm looks ahead at.

3. If the processing time is long, whether that be in the initial start-up or the Minimax processing, might make the program less desirable over the game in real life.
   3.1. The AI should be able to emulate a real player of connect four this means that the processing time for the move should not be a significant time; it would make the user feel frustrated and board.
4. The normal person does not have a highly powerful desktop computer, which is why this program should be designed so that it works well on standard computers.
5. The game will follow the rules of connect 4, if it didn't then it would cease being connect four.

6. If during the match the user forgets the rules, they can simply click on a button to be shown the rules, this is much easier than the alternative, that being opening a browser and searching up for the rules.

Design

Overall system design

The system is composed of five classes: `Main_Menu, Player_Name_and_Colour, AI_difficulty, Connect_4_Game, move.`

The first three classes are their own forms with the last two being in one form, the last of which is only there to make the undo feature, in the testing mode, work, therefore the code while technically is OOP, resembles structured much more.

Description of modular design

In the program all the specific processes that are done are modular and are utilized throughout different sections of the code, for example, both the Game sub and AIMove sub will call upon IsMoveValid, AssignGridSpace and IsGameFinished which are all held in the class Board.

Definition of data requirements

Random picker pics who starts

Validation required

The only inputs that matter are the

Stored information

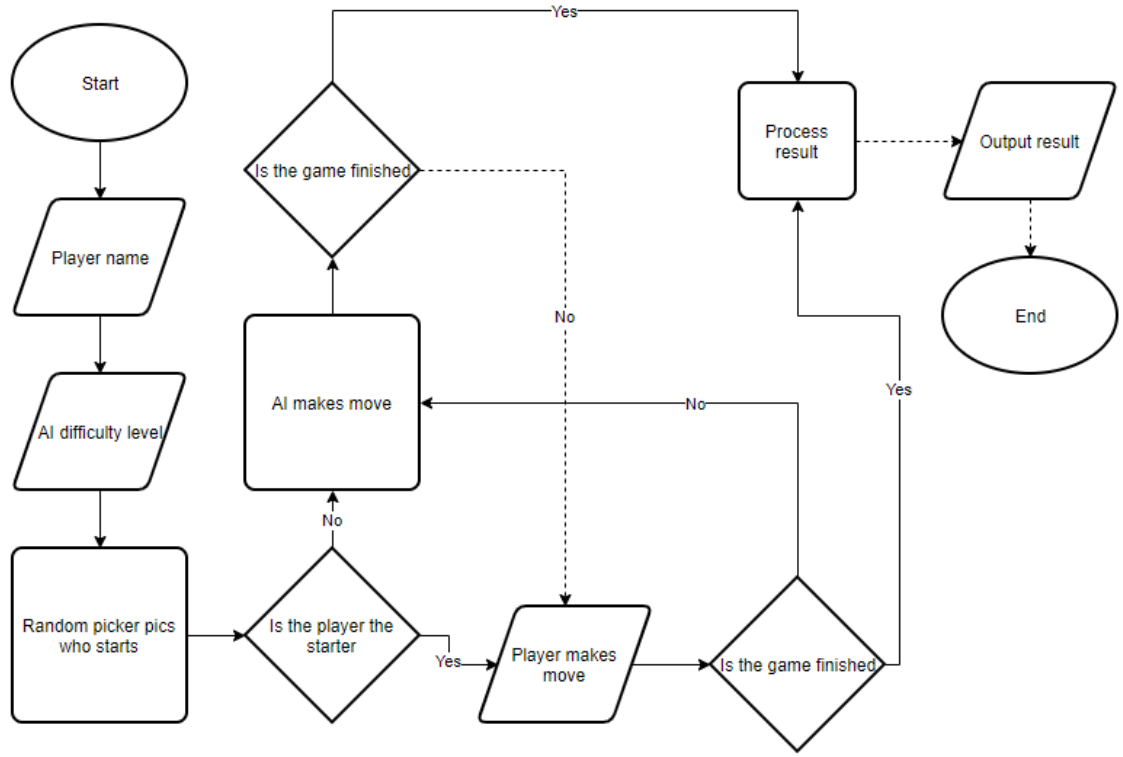
The program will give the user the ability to save and load a game; this will be done through stream writer and reader. On the game screen will be a button with the words “Save game” on it, when clicked will take the text of each image box and using stream writer write those states in a text file named “Save” in the debug folder, the file will also have the players name, if it is the players turn, the players colour, the AI’s colour and AI’s difficulty. When the LOAD SAVED GAME button is selected from the main menu using the stream reader the text file will be read with the first 42 lines being the board, every six lines being a column, the grid will then be drawn.

Class:

Board
grid(6, 7)

Game
TurnNum
Won
Draw
Winner
Turn
TurnSymbol
Column
Board

Flow:



Pseudo-code:

Here is the pseudo-code for the win state checking code:

```
FUNCTION IsFourInARow AS BOOLEAN
       FOR column <- 1 to 7 STEP + 1
              InARow <- 0
              FOR row <- 1 to 6 STEP + 1
              IF grid(row, column) <- CurrentColour
                     InARow <- InARow + 1
                     IF InARow <- 4 Return True
              ELSE
```

```
                InARow = 0
            END IF
            END FOR
        END FOR
        FOR row <- 1 to 6 STEP + 1
            InARow <- 0
            FOR column <- 1 to 7 STEP + 1
            IF grid(row, column) <- CurrentColour
                InARow <- InARow + 1
                IF InARow <- 4 Return True
            ELSE
                InARow = 0
            END IF
            END FOR
        END FOR
        FOR row <- 1 to 3 STEP + 1
            FOR column <- 1 TO 4 STEP + 1
                DiagonalNum <- 0
                InARow <- 0
                IF gird(row + DiagonalNum, column) <- CurrentColour
                    HoldColumn <- Column
                    FOR HoldColumn <- HoldColumn TO HoldColumn + 3 STEP + 1
                        IF grid(row + DiagonalNum, HoldColumn) <- CurrentColour
                            DiagonalNum <- DiagonalNum + 1
                            InARow <- InARow + 1
                            IF InARow <- 4 Return True
                        ELSE
                            DiagonalNum <- 0
                            InARow <- 0
                        END IF
                    END FOR
                END IF
            END FOR
        END FOR
FOR row <- 1 to 3 STEP + 1
            FOR column <- 1 TO 4 STEP - 1
                DiagonalNum <- 0
                InARow <- 0
                IF gird(row + DiagonalNum, column) <- CurrentColour
                    HoldColumn <- Column
                    FOR HoldColumn <- HoldColumn TO HoldColumn – 3 STEP - 1
                        IF grid(row + DiagonalNum, HoldColumn) <- CurrentColour
                            DiagonalNum <- DiagonalNum + 1
                            InARow <- InARow + 1
                            IF InARow <- 4 Return True
                        ELSE
                            DiagonalNum <- 0
                            InARow <- 0
                        END IF
                    END FOR
                END IF
            END FOR
        END FOR
END FUNCTION
```

Here is the pseudo-code for the AI heuristic/Minimax:

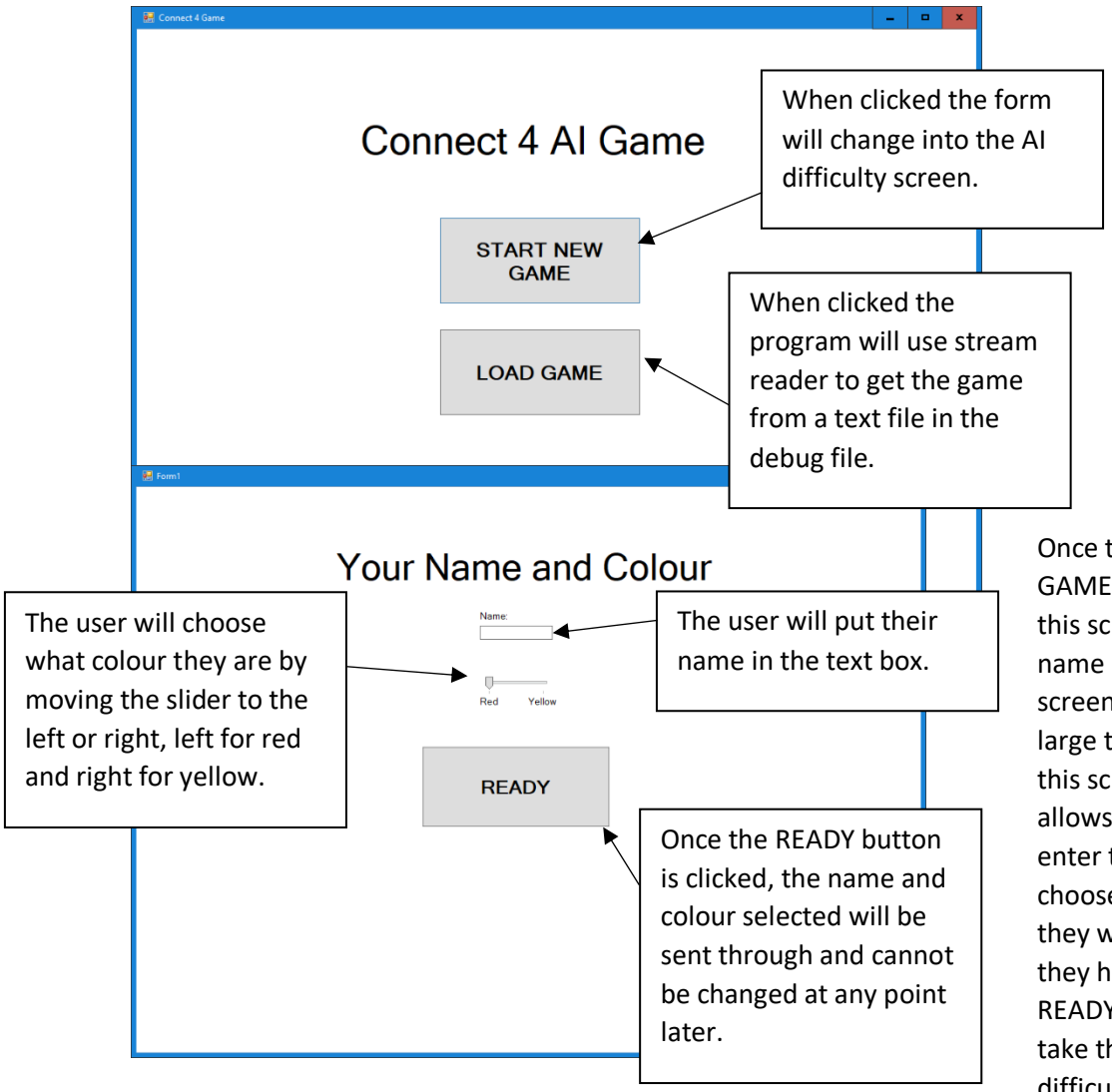
```
FUNCTION minimax(node, depth, maximizingPlayer) AS INTEGER
    IF depth = 0 OR Draw() OR WinningMove() then
```

```
        RETURN NodeValue
    END IF
    IF maximizingPlayer then
        value := −∞
        FOR Node = 1 to 7
            NodeValue := max(NodeValue, minimax(Node, depth − 1, FALSE))
        END FOR
        RETURN NodeValue
    ELSE
        value := +∞
        FOR Node = 1 to 7
            value := min(NodeValue, minimax(Node, depth − 1, TRUE))
        END FOR
        RETURN NodeValue
    END IF
END FUNCTION
```

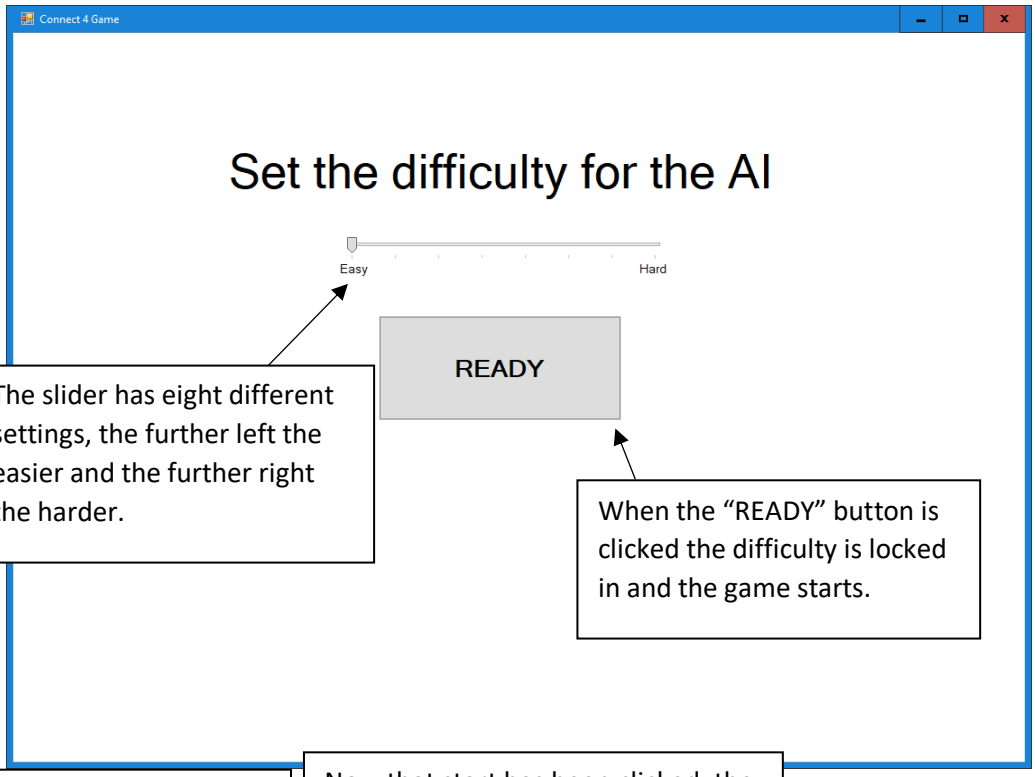
GUI

Here is a basic design for the GUI that I have made:

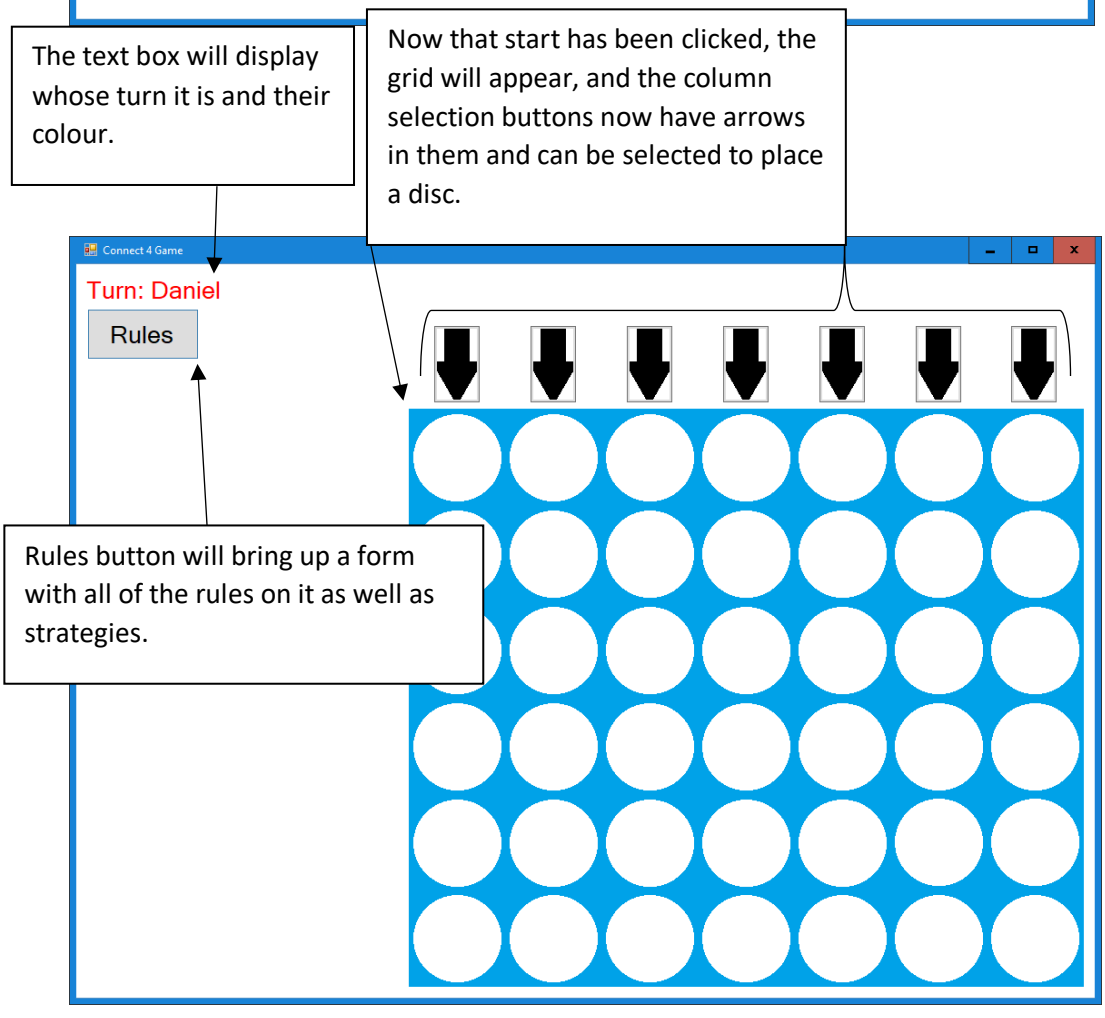
The form will start with just two buttons: START NEW GAME and LOAD SAVED GAME and will have the text “Connect 4 AI Game” written in large. START NEW GAME will show a new page named Player name and colour. When LOAD SAVED GAME is clicked, the most recent saved game will be loaded, and the user is taken straight to the game screen where they resume the game they saved.



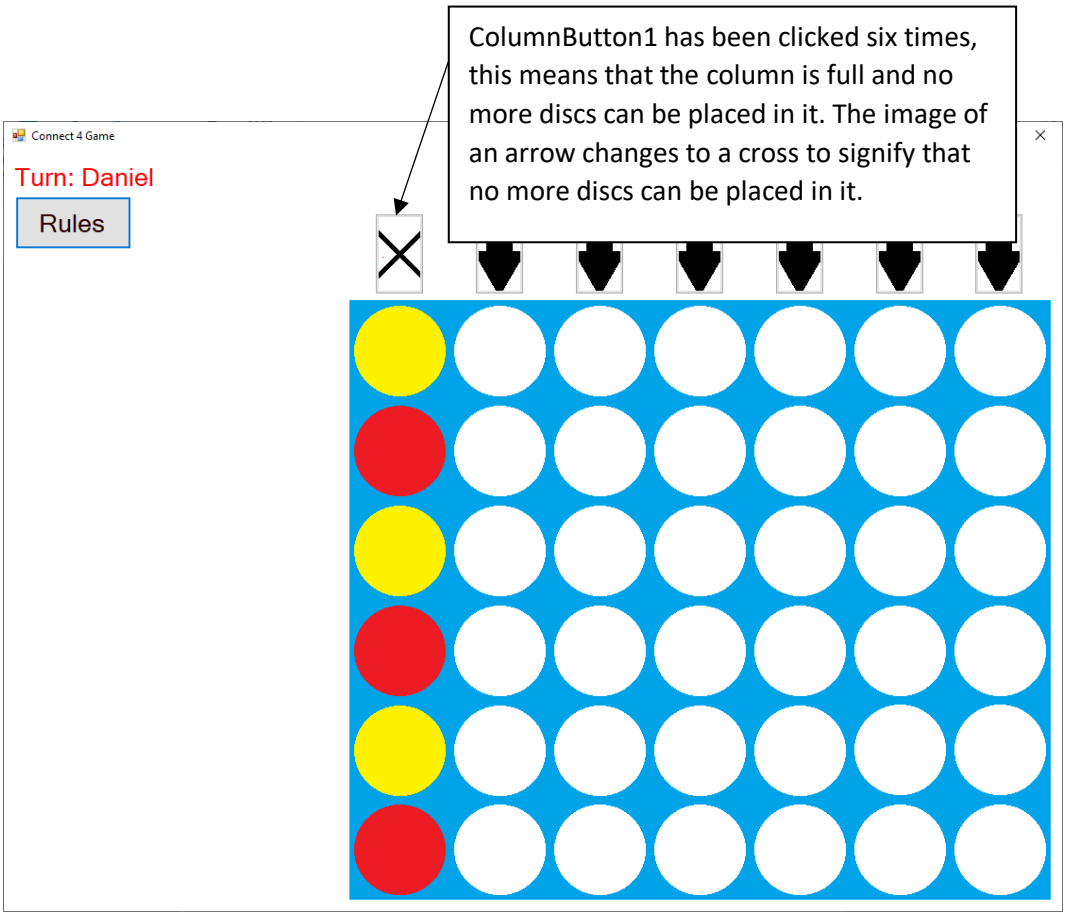
Once the START NEW GAME button is clicked this screen, the Player name and colour screen, appears, it has large text stating what this screen does, and it allows the user to enter their name and choose what colour they want to use. Once they have clicked the READY button, it will take them to the AI difficulty screen.



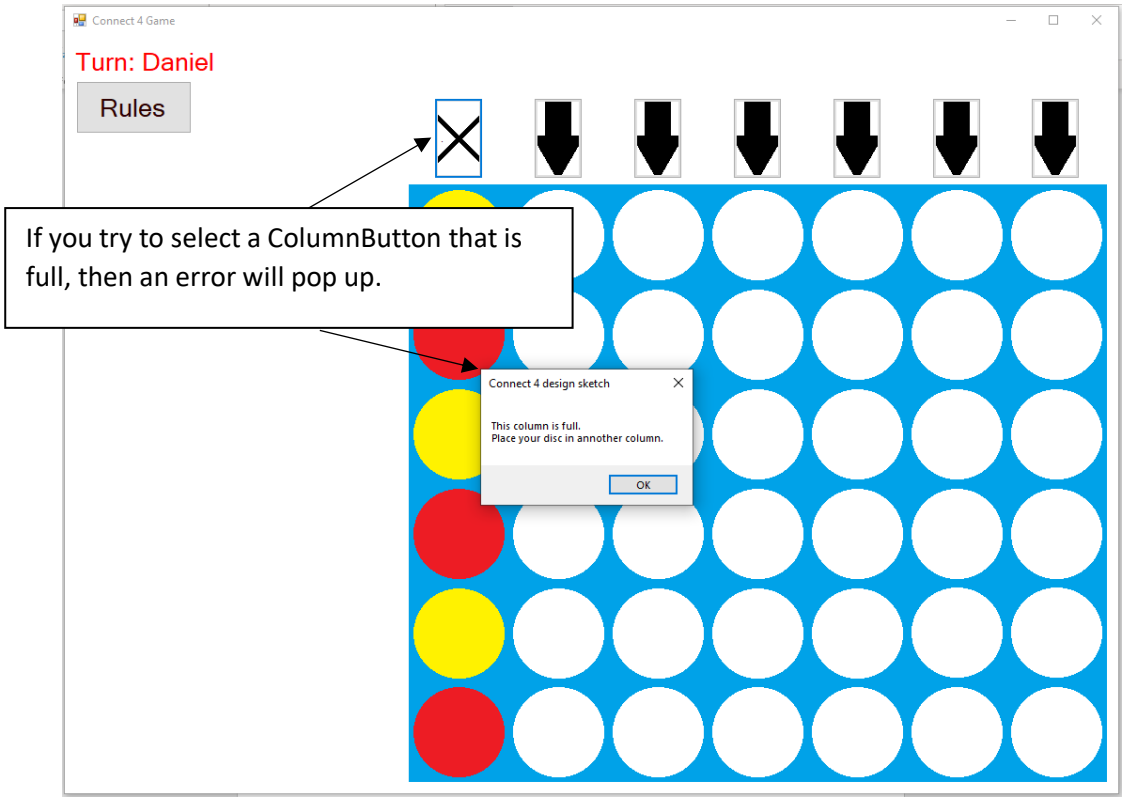
Once the READY button is clicked this screen, the AI difficulty screen, appears, it has large text stating what this screen does, and it allows the user to choose what difficulty AI they want to play up against. Once they have chosen the difficulty they then have to click ready, which will take them to the game screen.



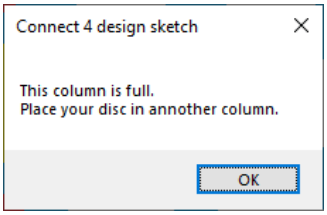
If the game was loaded then what ever was on the grid would be there otherwise the grid will be clear and whomever the random picker picks to start, will start.



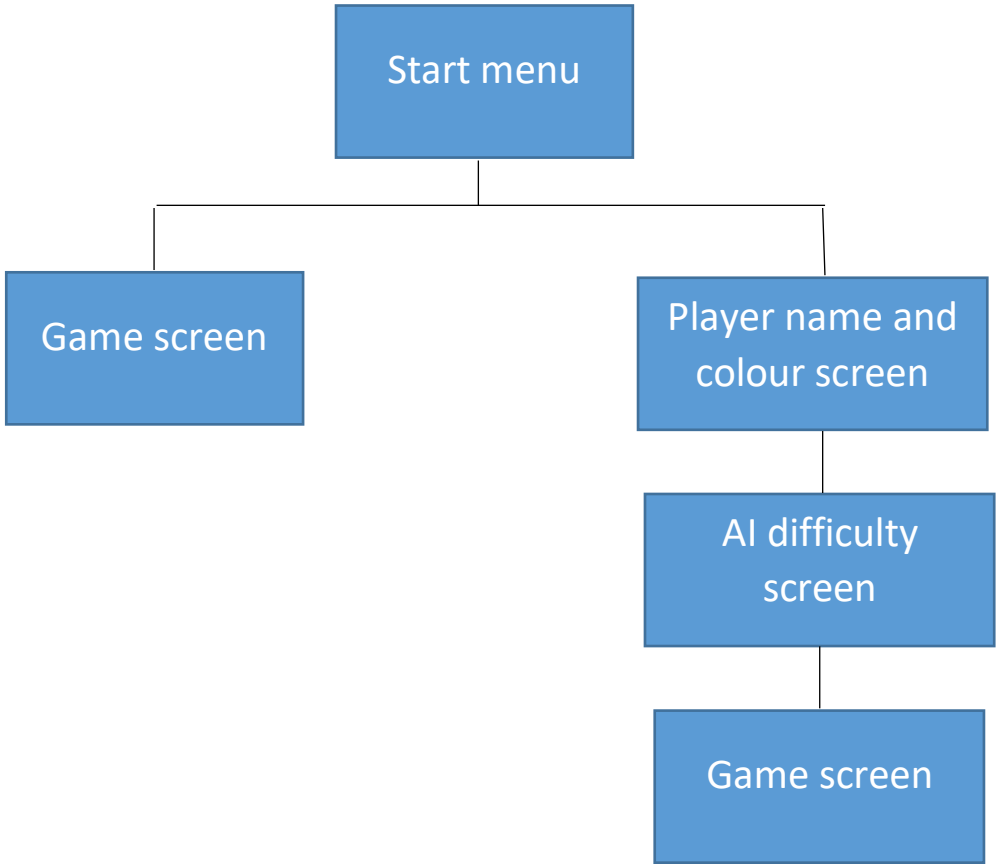
Whatever ColumnButton is selected the lowest empty spot in that column will be filled with the appropriate colour disc, if a column if full you will not be able to place a disc in it.



When a column is selected that is full an error message will popup in a Message Box saying:



Navigation diagram:



Technical Solution

The requirements:

7. Provide the user with an AI with changeable difficulty to play Connect 4 with
    7.1. The user must be able to play connect four with the AI at any time.
    7.2. The AI will have different difficulties that the user will choose from.
    7.3. The AI will operate by using a Minimax algorithm
        7.3.1. The algorithm will place a disc depending on the current disc positions.
        7.3.2. The algorithm will place a disc depending on the predictions of the future moves.
            7.3.2.1. The algorithm's decision that is dependent on the predicted moves, it will aim to get a win at the earliest point assuming that the opponent plays perfectly within its own perception.
            7.3.2.2. If the algorithm finds that it cannot win then it will take the longest route to a loss or will try to draw.
    7.4. The game of Connect 4 will be a fresh game with nothing in the grid whenever a new game is started.
8. The user interface should be simple enough that a child can figure out how to play.
    8.1. The menus will be clear and simple
        8.1.1. Every menu will have text that is written in large clear text.
        8.1.2. The main menu will give you the option to either play Connect 4 with an AI or load a saved game.
        8.1.3. The AI difficulty menu will allow the user to choose the difficulty of the AI or go back.
            8.1.3.1. The choices of difficulties are: Easy, Moderate, Medium, Hard, and Unyielding.
    8.2. Playing the game must be simple to understand.
        8.2.1. To the side of the grid will be an information box telling the user how to place a disc.
        8.2.2. The grid and discs will be blue, yellow and red to make it clear that they are different entities.
9. The program must not have long processing times.
    9.1. The AI decision-making process should not be long, less than 10 seconds.
    9.2. The placements of the discs should not take long to process, less than 3 seconds.
10. Should be able to run on a standard desktop computer.
11. The user should be able to place a disc only during their turn in allowed columns.
    11.1. The column will have an indicator, possibly an arrow that when clicked will place a disc at the lowest point on the grid.
        11.1.1. The disc can only be placed in a grid space that is not occupied.
        11.1.2. A disc can only be placed in a space above a disc or grid bottom.
    11.2. If a disc cannot be placed in a certain column then there will be a cross that will indicate that that column on the grid cannot be played.

11.2.1. When the cross is clicked an error message will come up to say that the player cannot place their piece in that column.

11.3. When it is not the users turn the user cannot place a disc and must wait for the AI to end their turn to then place their disc.

12. The user should be given the ability to see the rules without having to search for them.

12.1. During the game there will be a toggle on the side that when clicked will show the rules of Connect 4.

12.2. When clicked again the tab will disappear.

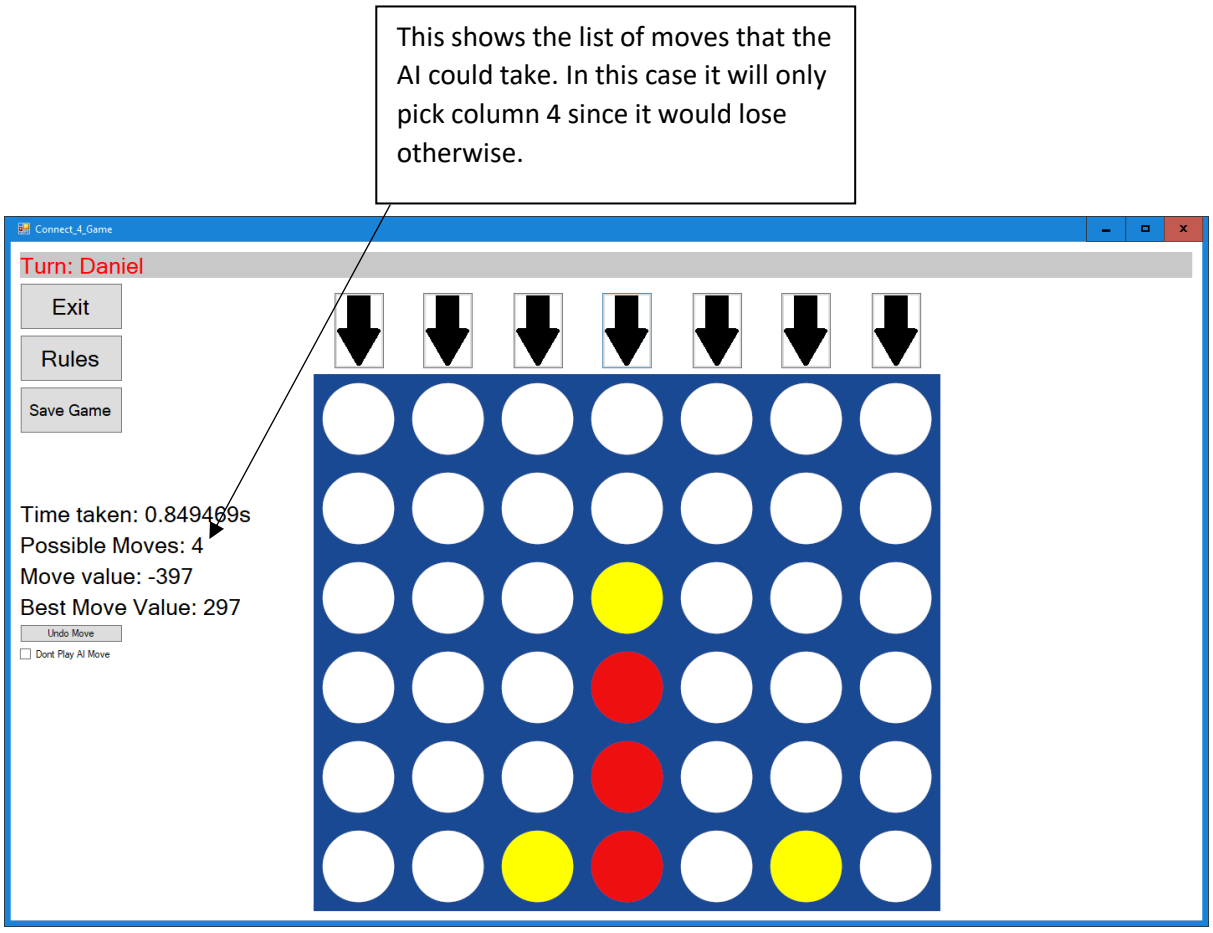
Proof of requirements being met:

(All screen shots of the program are with the testing mode turned on since it gives data on each move the AI makes as well as some extra functionality that help with testing and demonstration. When an image is referenced it is talking about the images with the code at the end of the document)

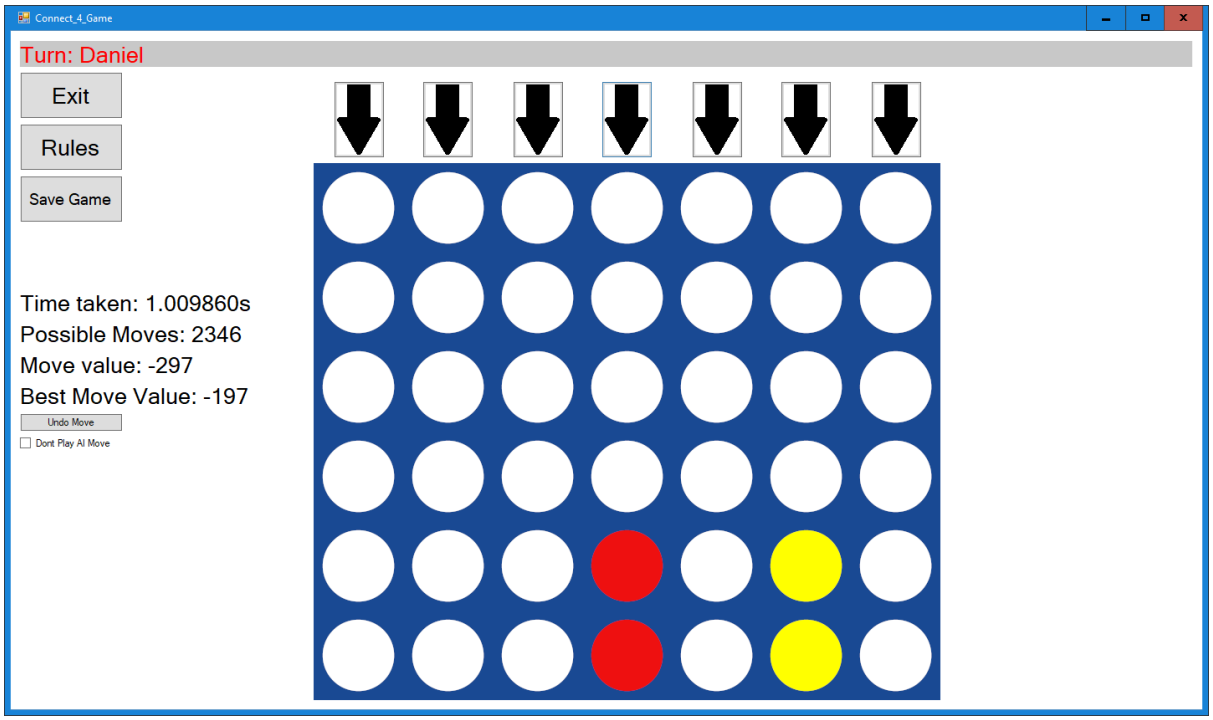
1.1 As long as the user has a computer with VB installed as well as the program this requirement is met.

1.2 As seen in image 4 the user is given 5 difficulties to choose from, the difficulty works by setting the number of moves that the AI looks forward to, the harder difficulties have a higher number of moves that it will check.

1.3.1 This is achieved since the code will find that when there is a three in a row of the opponent's discs and an empty space along that line it will place a disc to stop it.



1.3.2.1



In this screenshot, it shows that the best move is not just 4 since by looking ahead it assumes that I would not place my disc in 4 since it would be blocked the next round making it a pointless move.

1.3.2.2

1.4 When the player selects “START NEW GAME” from the main menu and then goes through all of the settings and clicks “READY” as shown in image 4 a clear grid will appear as seen in image 6.

2.1.1 As seen in Image 1 through 6 this requirement is met.

2.1.2 Image 1 shows this

2.1.3.1 Image 4 shows this

2.2.1 On the left of image 6 is a button labelled rules, when clicked the rules of connect 4 appear in an image as well as other information including how to play, this can be seen in image 5

2.2.2 The image for 1.3.2.1 shows this

3.1

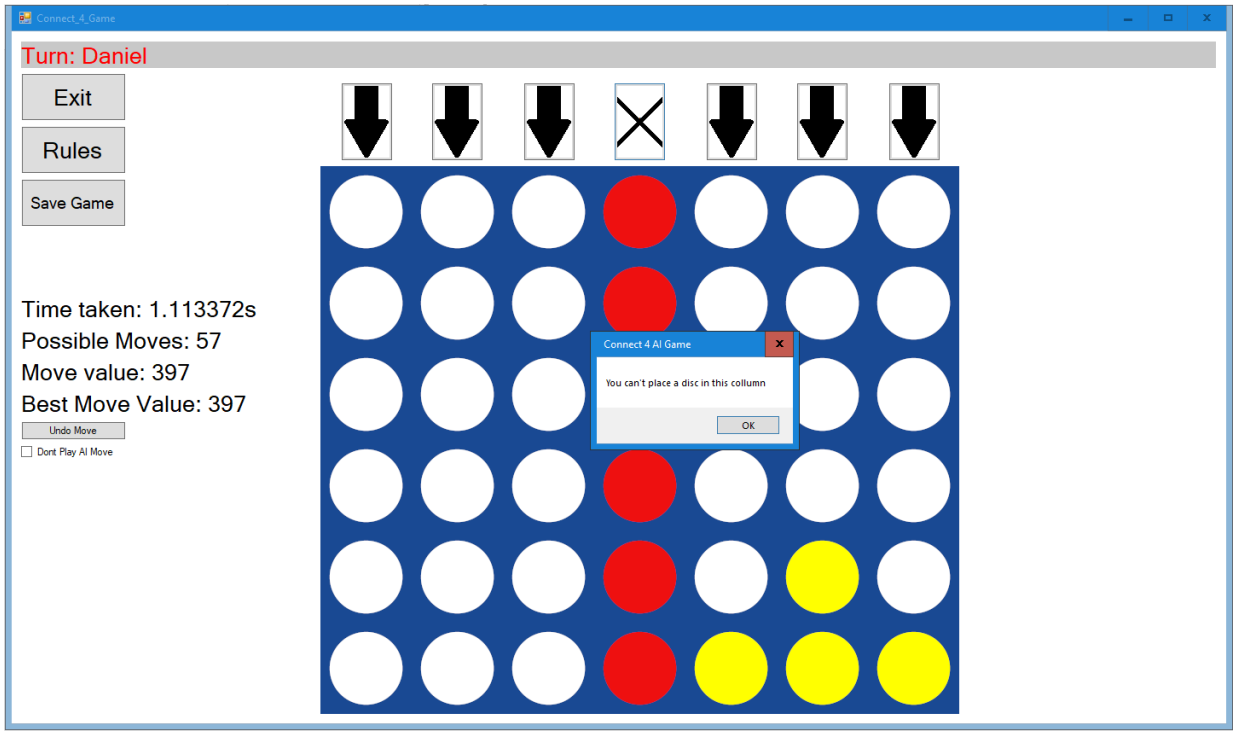
3.2 Once the minimax has decided where it will place its disc or when the player selects a column the disc is placed in less than half a second.

4 It can

5.1.1 The image for 1.3.2.1 shows that two discs are stacked meaning that they do not occupy the same space

5.1.2 The image for 1.3.2.1 shows this

5.2.1

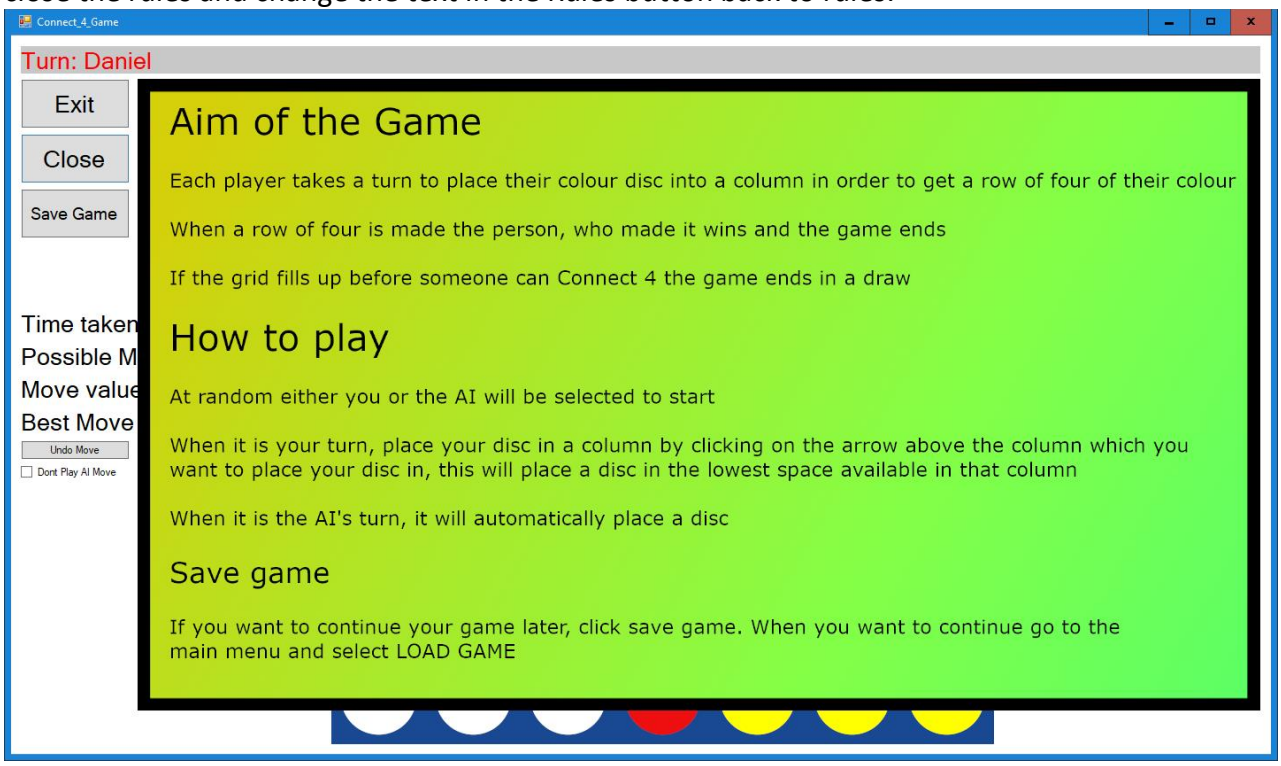


The message box reads “You can't place a disc in this column”.

5.3 When the AI is processing it’s go all buttons are made inactive other than the “Rules” button.

6.1 The button labelled “Rules” will show a image with various info on the game including rules, this can be seen in image 5.

6.2 When the image is shown the label for the rules button changes to close and when clicked will cause the image to go away. If the Exit button is clicked at this point, it will also close the rules and change the text in the Rules button back to rules.

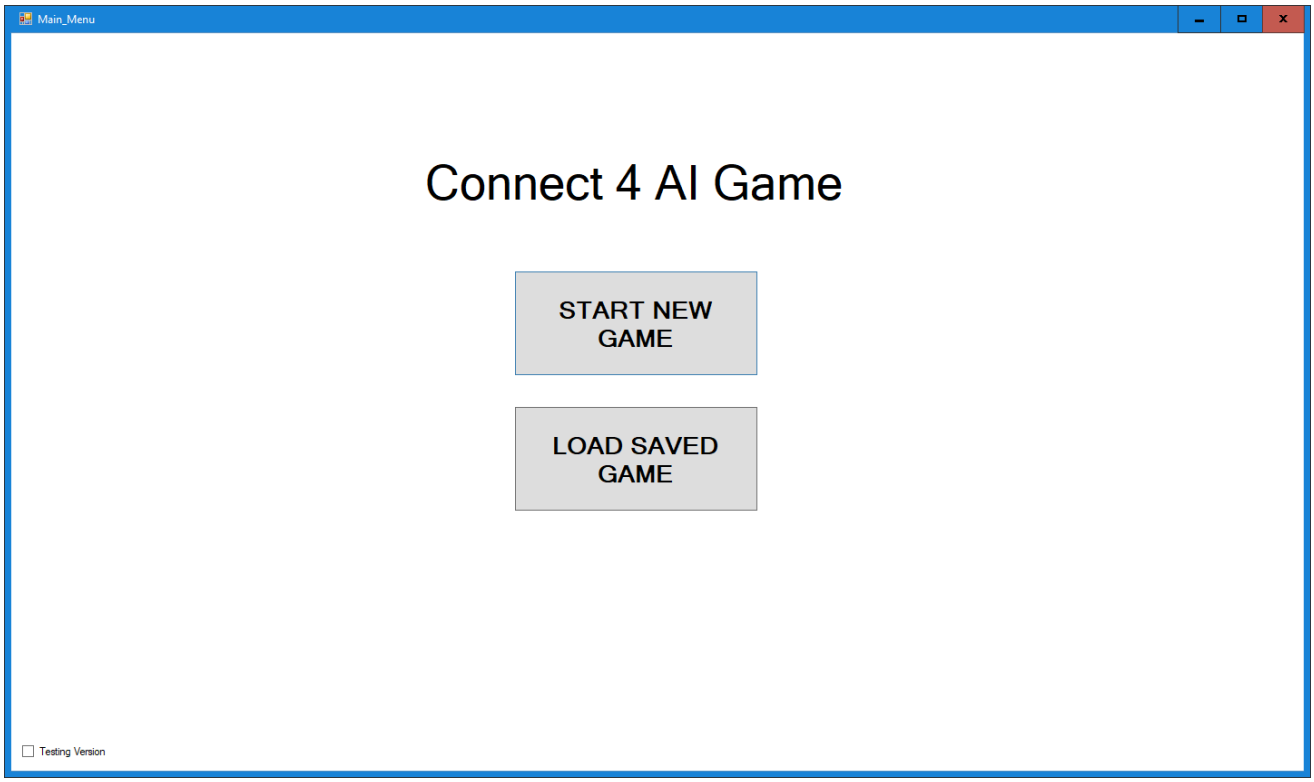


Overview of code

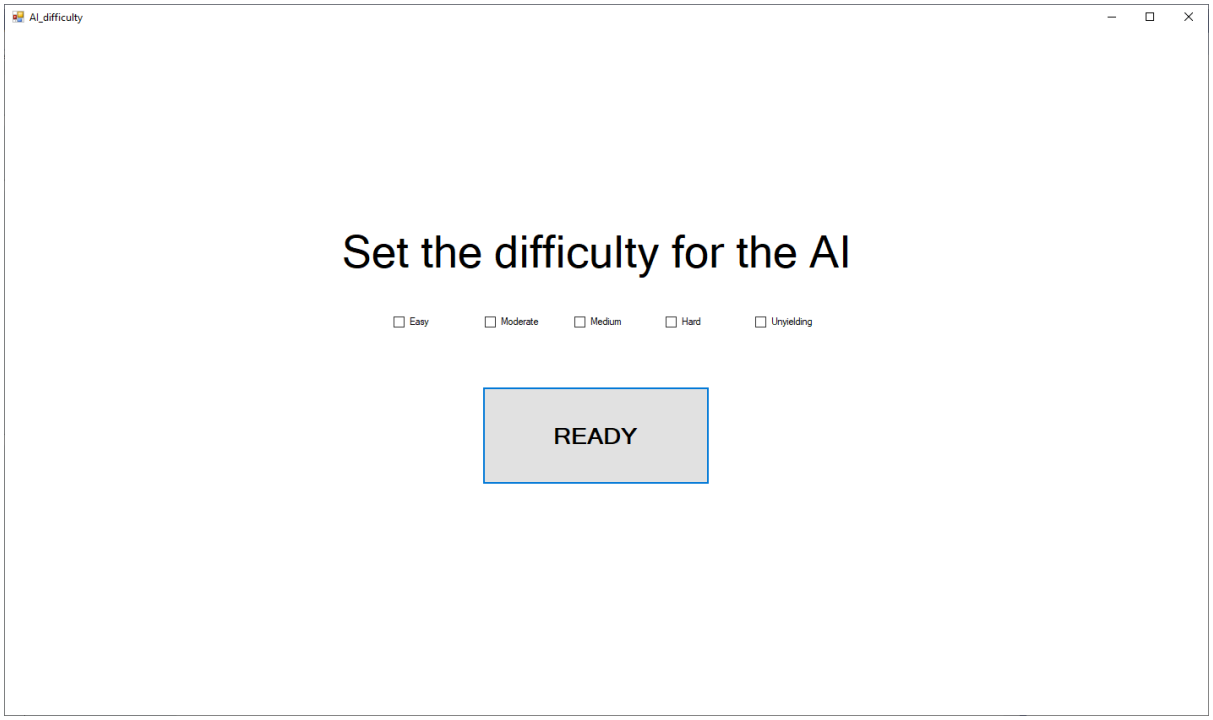
The code is written in a structured style.
The minimax algorithm is used to get the move that the AI will make.
The minimax algorithm is sped up by use of alpha beta pruning.
The code works as intended.

Changes from original design

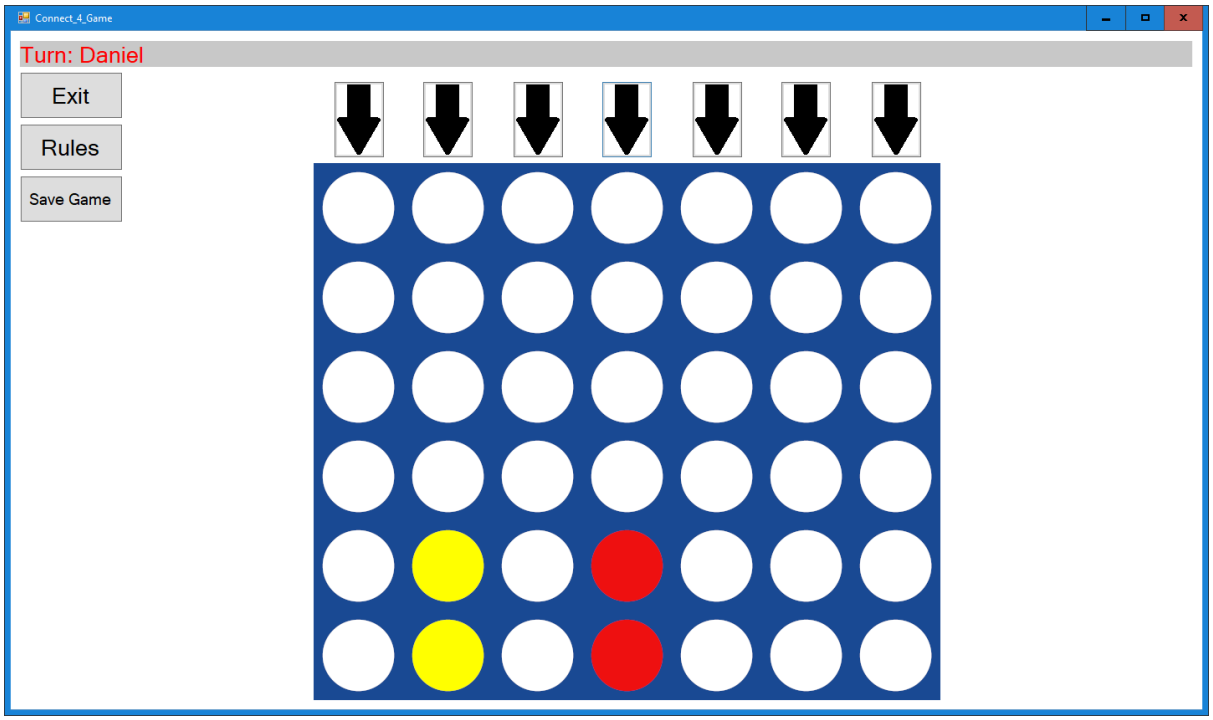
The main menu design has a check box that toggles testing mode on and off, this can be found in the bottom left. The LOAD GAME button now says LOAD SAVED GAME



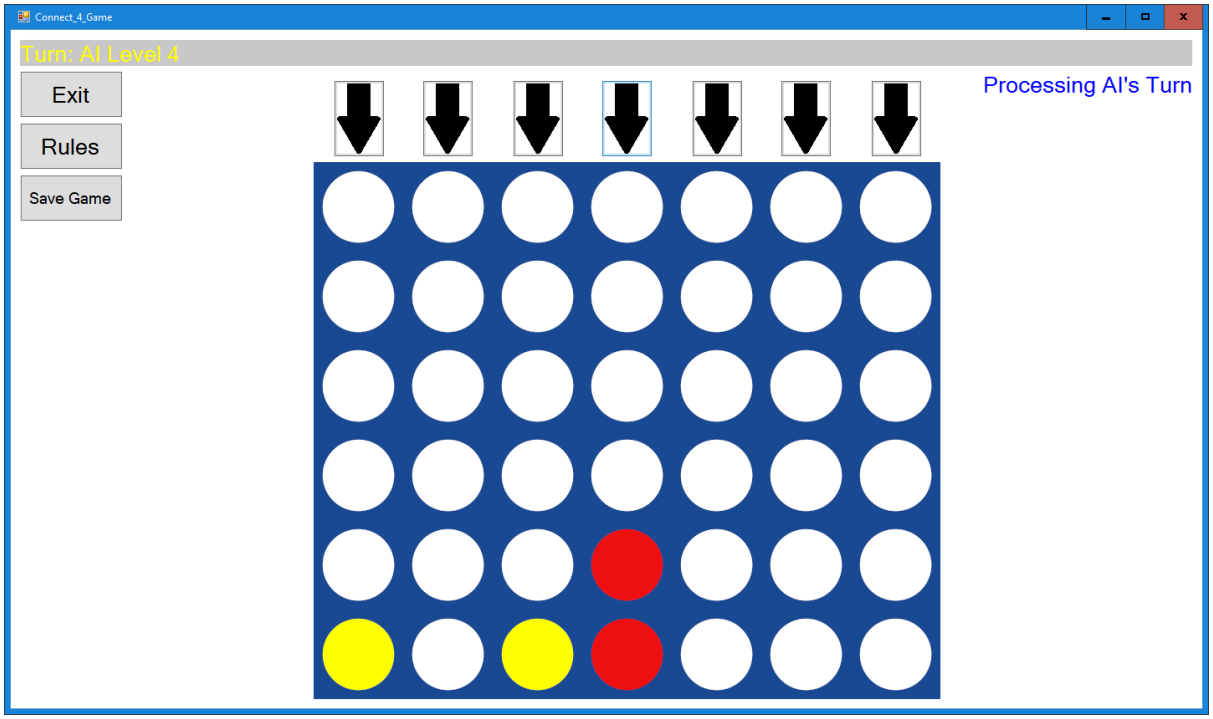
The selection for difficulty also changed from being a slider to being five separate check boxes that uncheck when another is selected.



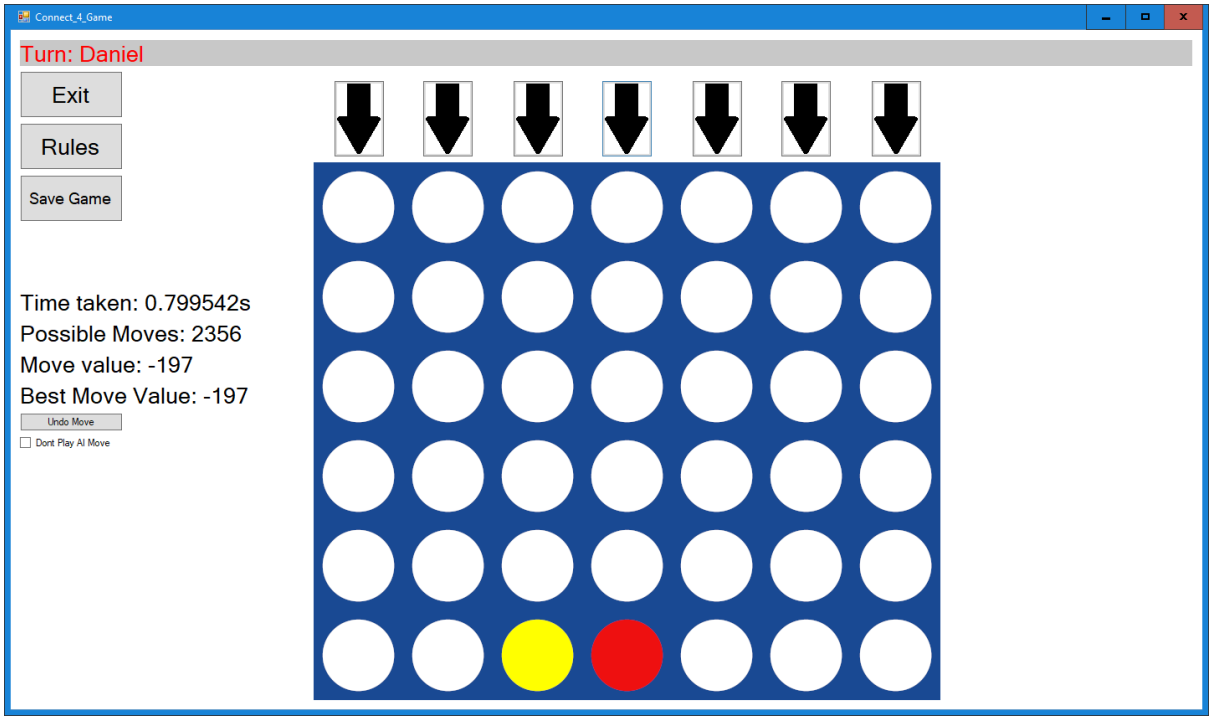
The game screen has now got a cleaner look with better looking images for the board and the discs. The game screen now has an exit button and the text box with the turn name and colour is now grey to better define it from the rest of the screen. The new version also does not have an AI turn button so it will run the AI code as soon as it is the AI's turn. The save game button is now larger and has larger font size.



When the AI is processing its turn, it now says in the top right "Processing AI's turn"



With the addition of testing mode, some more features are present. Once the AI has taken its turn, a text box will display the time taken by the AI as well as the possible moves that it could have chosen. While the AI runs the numbers for move value and best move value will change, this gives me the ability to step through the code and easily get the value and best value at that moment. The testing mode gives the user the ability to remove a disc that has been placed by clicking the remove disc button, when clicked the most recently placed disc it removed and the turn switched to the colour that was removed. There is a checkbox that allows the toggling off and on of the auto run of the AI, when checked the AI will not run the moment that it is its turn.



Testing

Test Number	**Description** (You will describe what the test is looking for and how you will perform it)	**Data type** (This will be Typical, Erroneous or Extreme. If you don't know what this means read below)	**Expected result** (How are you expecting the system to respond to your input data? Should it give an error message? Will the response be any different to a normal one?)	**Pass/Fail** (Did it succeed or fail? You are aiming for the vast majority to be passes)	**Cross Reference** (link this test to your screen shot that proves it works)
1	Clicking a column button that is not full to see how the grid will respond	Typical	When a column button that is over a column that is not full is clicked, the lowest empty space in that column should be taken by the players disc colour.	Pass	See video in link [1]
2	Clicking a column button that is full to see how the grid will respond	Typical	When a column button that is over a column that is full is clicked, an error message will come up saying you cannot place a disc in this column, the grid will not change.	Pass	See video in link [2]
3	Selection of difficulty by clicking on the check boxes to see what effect this will have on the AI's behaviour	Typical	On the AI Difficulty select screen the user is provided with five check boxes that have different difficulties written next to them, when one is clicked the box should be crossed, if another is clicked while one still has a cross in it the previous one will be de-selected and the new one selected. When the player clicks the ready button, the game will retrieve the difficulty for the AI where the difficulty will be appropriate to the labels.	Pass	See video in link [3]
4	Name input to see what	Typical	A name will be input into the text box on the player	Pass	See video in link [4]

	will be displayed in the game on the players turn and when the player wins		name and colour screen, when it is the players turn the players inputted name should be present in the top left of the game screen. When the player wins, a message should come up with the players name stating that they have one and once acknowledged will have the player turn box say that the player has won. If the name that is input ins longer than 60 characters, the name will be set to those first 60 characters with an ellipsis at the end.		
5	Choosing the players colour by moving a slider to either red or yellow to see what happens in a game	Typical	On the player name and colour screen a slider will be automatically selected on red, it can stay there or be moved to the right to hover over yellow. When the game starts if it is the players turn the colour of the text in the top left showing whose turn it is will be the colour selected earlier, if it is the AI's turn then the colour will be the one not selected, the colour of the discs should also follow that.	Pass	See video in link [5]
6	Saving and loading the game by running a game, clicking the save button, exiting to main menu and clicking the LOAD SAVED GAME button to see what the grid	Typical	After playing a few pieces clicking the save game button, exiting to go to the main menu and then click the LOAD SAVED GAME button, these things should be the same as before: position of the discs in the board, player name, player and AI colour and the turn (i.e. the player's turn since the AI will do its move as soon as the player has done theirs meaning that you	Pass	See video in link [6]

	will look like, who's turn it is, what colour the player is, the difficulty of the AI, who's turn it is and the name of the player.		could not save it on the AI's turn.		
7	Search for four in a row in every direction this can be done by step by stepping through the code to see it working	Typical	The code must be able to find four consecutive colours in all directions (horizontal, vertical, diagonally / and diagonally \), when it has found one then it should display the message (player name) has won; the buttons for the disc placement and save game should be made inactive and will do nothing if clicked. Bellow the save game button the user will be prompted to click the exit button to return to the main menu.	Pass	See video in link [7]
8	Check for draw by loading a file with the game one move away from a draw and then clicking the only working placement button.	Typical	When the last disc has been placed and there are no connect 4s then a message should pop up saying that the game is a draw, once acknowledged the text in the turn display box (the text box in the top left) will change to text that is black and says the game is a draw. The buttons for the disc placement and save game should be made inactive and will do nothing if clicked.	Pass	See video in link [8]
9	What happens when there is more than 4 in a row, this will be done	Typical	When the player's disc is placed in the empty space, there should be no difference with just getting four in a row producing the	Pass	See video in link [9]

	by loading a save that has the bottom row completely full of the player's colour of disc other than the centre one that will be empty, it will be the players turn and I will click on the centre button.		same outcome shown in test 7.		
10	Does the minimax pick the best option each round when it starts? By stepping through the code as it processes the move we are able to see how it will choose its move.	Typical	The different difficulties should give different results. With difficulty 1, 2 and 3 it should pick any of the columns due to its limitation when looking ahead. For 4 and 5 it should find that column 4 is the best choice since it gives the most possibilities for connect fours.	Pass	See video in link [10]
11	Does the minimax pick the best move in a game? By loading a series of saves for each difficulty, I can step through the code to see if the minimax is finding the best move.	Extreme	The move that the minimax will make should either stop an opponent's four in a row, delay an opponent's inevitable four in a row, create its own four in a row or get the longest string of discs possible appropriate for the difficulty.	Pass	See video in link [11]

12	Will the game run with all of its features being fully utilized? This can be done by using every button and input device in a session.	Typical	All the beta tests must be passed in the same session, the game must fully work.	Pass	See video in link [12]
13	White box test of the minimax algorithm with a tree diagram on paint to trace out what decision the minimax should make. I will use a saved game.	Typical	Before I allow the AI to make a move I will trace out a tree diagram in paint and show what the algorithm will go through and what result is given, I will then use f11 to step through the code to show the program following what I drew out at the beginning. It will be a success if the match.		See video in link [13]

Testing references

[1] https://youtu.be/OeWCK67klLs

[2] https://youtu.be/JDRzoBb7vpc

[3] https://youtu.be/yz8ACWKuips

[4] https://youtu.be/ri0yhSM-Jtk

[5] https://youtu.be/wHHGvr4_A0A

[6] https://youtu.be/EhiRozQsYv0

[7] https://youtu.be/AtDmAN8lkdY

[8] https://youtu.be/tgHZUaTBoGE

[9] https://youtu.be/w72xw1QQk9I

[10] https://youtu.be/JhjVeIjD8K8

[11] https://youtu.be/ad7jxTu_S8c

[12] https://youtu.be/GfoPEzUb8Vc

[13] https://youtu.be/VsmU_JilUS4

Evaluation

Requirements

See "Proof of requirements being met:" above page 20

I met almost all of my requirements; the only one that I did not achieve is the processing time being less than 10 seconds (3.1). The hardest difficulty on a college computer will take an average max time of about 52 seconds and on my computer at home, which is a bit more powerful, will take 20 seconds, all other difficulties meet this requirement even on college computers. Two of the requirements were achieved but were combined in the process, 2.2.1 and 6.1 are part of the same function.

How could it be improved if revisited?

The product could be improved by making the AI's turn faster; this might be achieved by optimizing the code in some way.

I would make it so that the forms for the main menu, player name and colour select and AI difficulty select are all in the same space instead of moving around each time a button is clicked that takes you to the next form, this can be annoying especially is you are going back and forth between the different forms.

I would allow the user to select the column that they want to place their disc in by clicking any of the images in that column; I would do this because sometimes the user tries to click on the image instead of the button.

If I were to revisit the program, I would like to add some more game modes like, two-player mode (maybe more) and different board sizes with different length of win lines.

Independent feedback

Third Party:

Questions asked:

1) What requirements are not met in this program?
2) What did you not like about the program?
3) What did you like about the program?
4) Final notes?

Third party response:

1) 3. The AI processing time is too long; it leaves the player waiting for their turn.
2) When the AI is randomly selected to place the first disc, it should be very quick to do so; as it is there is a long wait for what should be a simple choice of placement
3) I was surprised at how the AI was able to predict my moves, even on Medium difficulty, and could not beat it on the top two levels. The game rules are clear and easy to understand.

4) I feel that the rest of the requirements were met and I really enjoyed playing the game.

Summary of third party feedback:

The program is too long meaning that requirement 3 is not met, other than that, the program is good.

Response to feed back:

The speed of the AI is slow at the highest difficulty but is something that I am not sure how to fix.

By hard coding the first placement of the disc to be 1-7 for difficulties 1-3 and 4 for difficulties 4 and 5, I am no longer using the minimax algorithm, which is one of the focuses of the project and is in the requirements.

Code

```
Public Class Main_Menu
    Dim LoadGame As Boolean
    Dim TestingVersion As Boolean
    Public Sub Main_Menu_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        LoadGame = False
    End Sub
    Private Sub StartButton_Click(sender As Object, e As EventArgs) Handles StartButton.Click
        LoadGame = False
        Player_Name_and_Colour.Show()
        Me.Hide()
    End Sub

    Private Sub LoadButton_Click(sender As Object, e As EventArgs) Handles LoadButton.Click
        LoadGame = True
        Me.Hide()
        Connect_4_Game.Show()
    End Sub
    Function LoadGameFunc() As Boolean
        'Called from Connect_4_Game
        If LoadGame = True Then
            Return True
        Else
            Return False
        End If
    End Function

    Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged
        If TestingVersion Then
            TestingVersion = False
        Else
            TestingVersion = True
        End If
    End Sub
    Function TestingGet() As Boolean
        Return TestingVersion
    End Function
End Class



Public Class Player_Name_and_Colour
    Dim IsTheChosenColourYellow As Boolean
    Dim NameOfPlayer As String
    Private Sub StartButton_Click(sender As Object, e As EventArgs) Handles StartButton.Click
```

```
        'If the name is more than 60 characters long then the name is set to the first 60 characters and an ellipsis
        If Len(NameInput.Text) > 60 Then
            NameOfPlayer = NameInput.Text.Substring(0, 59) & "..."
        Else
            NameOfPlayer = NameInput.Text
        End If
        AI_difficulty.Show()
        Me.Hide()
    End Sub

    Private Sub ColourChoiceBar_Scroll(sender As Object, e As EventArgs) Handles ColourChoiceBar.Scroll
        If IsTheChosenColourYellow = False Then
            IsTheChosenColourYellow = True
        Else
            IsTheChosenColourYellow = False
        End If
    End Sub
    Function ColourFunction() As Boolean
        'Called from Connect_4_Game to get the players colour
        If IsTheChosenColourYellow = True Then
            Return True
        Else
            Return False
        End If
    End Function
    Function NameFunction() As String
        'Called from Connect_4_Game to get the players name
        Return NameOfPlayer
    End Function
    Private Sub Player_Name_and_Colour_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        NameInput.Text = ""
    End Sub
    Private Sub Player_Name_and_Colour_Close(sender As Object, e As EventArgs) Handles MyBase.Closed
        Main_Menu.Show()
        Me.Hide()
    End Sub
End Class



Public Class AI_difficulty
    Dim AIDifficulty As Integer
    Dim Run As Boolean
    Dim CheckBox(5) As CheckBox
    Public Sub AI_difficulty_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CreateDifficultyCheckBoxes()
        Run = True
    End Sub
```

```
Sub CreateDifficultyCheckBoxes()
    '5 check boxes are created with different values and names
    Dim ListOfDifficultys() As String = {"Easy", "Moderate", "Medium", "Hard", "Unyielding"}
    For i = 1 To 5
        CheckBox(i) = New CheckBox
        With CheckBox(i)
            .Name = "CheckBox" & i
            .Left = (300 + 107 / 2) + 107 * i
            .Top = 300
            .Width = 80
            .Height = 92
            .Text = ListOfDifficultys(i - 1)
            .Tag = i
            AddHandler .CheckedChanged, AddressOf CheckBoxChange
        End With
        Me.Controls.Add(CheckBox(i))
    Next
End Sub
Private Sub ReadyButton_Click(sender As Object, e As EventArgs) Handles ReadyButton.Click
    Dim ZeroAIDifficulty As Boolean = True
    Dim NumOfNotChecked As Integer = 0
    'Checks for a check box to be checked
    For a = 1 To 5
        If ZeroAIDifficulty = True Then
            If CheckBox(a).Checked = True Then
                ZeroAIDifficulty = False
                NumOfNotChecked = 0
            Else
                NumOfNotChecked += 1
                If NumOfNotChecked = 5 Then
                    ZeroAIDifficulty = True
                    AIDifficulty = 0
                End If
            End If
        End If
    Next
    If AIDifficulty <> 0 Then
        Connect_4_Game.Show()
        Me.Hide()
    Else
        MsgBox("Select a difficulty before continuing")
    End If
End Sub
Function GetAIDifficulty() As Integer
    'Returns the chosen difficulty to Connect_4_Game
    Return AIDifficulty
End Function
```

```
    Sub ChangeOtherCheckBoxes(ByRef NewAIDifficulty As Integer)
        'If one check box is cheked any one that was previously cheked is un-checked
        For i = 1 To 5
            If CheckBox(i).Checked = True And CheckBox(i).Text <> CheckBox(NewAIDifficulty).Text Then
                Run = False
                CheckBox(i).Checked = False
                Run = True
            End If
        Next
    End Sub
    Sub CheckBoxChange(sender As Object, e As EventArgs)
        If Run = True Then
            Dim ChangedTextBox As CheckBox = CType(sender, CheckBox)
            Dim NewAIDifficulty As Integer = ChangedTextBox.Tag
            ChangeOtherCheckBoxes(NewAIDifficulty)
            AIDifficulty = CheckBox(NewAIDifficulty).Tag
        End If
    End Sub
    Private Sub AI_difficulty_Closed(sender As Object, e As EventArgs) Handles MyBase.Closed
        Main_Menu.Show()
        Me.Hide()
    End Sub
End Class



Public Class Connect_4_Game
    ReadOnly Board(7, 6) As PictureBox
    ReadOnly StoreBoard(7, 6) As String
    ReadOnly DropButton(7) As Button
    ReadOnly ImageFolder As String = "images/"
    Dim EndOfGame As Boolean
    Dim PlayerTurn As Boolean
    Dim TurnColour As String
    Dim PlayerColour As String
    Dim AIColour As String
    Dim AIName As String
    Dim NameOfPlayer As String
    Dim WinningMove As Boolean
    Dim OriginalDepth As Integer
    Dim AllowButtonClick As Boolean = True
    Dim SwapSwitch As Boolean
    Dim DontPlayAITurn As Boolean
    Dim movelist As New Stack(Of move)
```

```
    Public Sub Connect_4_Game_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        CreateButtons()
        TestingMode()
        If Main_Menu.LoadGameFunc = False Then
            OriginalDepth = AI_difficulty.GetAIDifficulty
            AIName = "AI Level " & OriginalDepth
            SetStartNameAndColour()
            CreateGrid()
            EndOfGame = False
            If PlayerTurn = False Then AIMove()
        Else
            RetrieveSaveData()
            CheckForFullColumns()
            If PlayerTurn = False Then AIMove()
        End If
    End Sub
    Sub TestingMode()
        If Main_Menu.TestingGet Then
            ProcessingTime.Visible = True
            ListOfBestMoves.Visible = True
            MoveValueTextBox.Visible = True
            BestMoveValue.Visible = True
            UndoMove.Visible = True
            AIMoveCheck.Visible = True
        Else
            ProcessingTime.Visible = False
            ListOfBestMoves.Visible = False
            MoveValueTextBox.Visible = False
            BestMoveValue.Visible = False
            UndoMove.Visible = False
            AIMoveCheck.Visible = False
        End If
    End Sub

    Sub CreateButtons()
        For i = 1 To 7
            DropButton(i) = New Button
            With DropButton(i)
                .Name = "CollumnDrop" & i
                .Left = (225 + 107 / 2) + 107 * i
```

```
                .Top = 60
                .Width = 61
                .Height = 92
                .Text = i
                .Image = Image.FromFile(ImageFolder & "Arrow.png")
                AddHandler .Click, AddressOf ButtonClicked
            End With
            Me.Controls.Add(DropButton(i))
        Next
    End Sub
    Sub SetStartNameAndColour()
        Dim Ran As New Random
        Dim RanNum As Integer
        RanNum = Ran.Next(1, 3)
        NameOfPlayer = Player_Name_and_Colour.NameFunction
        If Player_Name_and_Colour.ColourFunction = True Then
            PlayerColour = "Yellow"
            AIColour = "Red"
        Else
            PlayerColour = "Red"
            AIColour = "Yellow"
        End If
        If RanNum = 1 Then
            PlayerTurn = True
            TurnColour = PlayerColour
            DisplayTurn.Text = "Turn: " & NameOfPlayer
        Else
            PlayerTurn = False
            TurnColour = AIColour
            DisplayTurn.Text = "Turn: " & AIName
        End If
        If TurnColour = "Yellow" Then
            DisplayTurn.ForeColor = ForeColor.Yellow
        Else
            DisplayTurn.ForeColor = ForeColor.Red
        End If
    End Sub
    Sub CreateGrid()
        For x = 1 To 7
            For y = 1 To 6
```

```
                StoreBoard(x, y) = "empty"
                Board(x, y) = New PictureBox
                With Board(x, y)
                    .Name = "R" & y & "y" & x
                    .Left = 255 + 107 * x
                    .Top = 800 - 107 * y
                    .Width = 107
                    .Height = 107
                    .Image = Image.FromFile(ImageFolder & "Grid piece.png")
                    .Text = "empty"
                End With
                Me.Controls.Add(Board(x, y))
            Next
        Next
    End Sub
    Sub RetrieveSaveData()
        Dim i As Integer = 1
        For x = 1 To 7
            For y = 1 To 6
                Board(x, y) = New PictureBox
                With Board(x, y)
                    Dim sr As New StreamReader("Save.txt")
                    Dim Line As String = System.IO.File.ReadAllLines("Save.txt")(i - 1)
                    i += 1
                    .Name = "R" & y & "y" & x
                    .Left = 255 + 107 * x
                    .Top = 800 - 107 * y
                    .Width = 107
                    .Height = 107
                    If Line = "empty" Then
                        .Image = Image.FromFile(ImageFolder & "Grid piece.png")
                    ElseIf Line = "Red" Then
                        .Image = Image.FromFile(ImageFolder & "Grid piece red.png")
                    Else
                        .Image = Image.FromFile(ImageFolder & "Grid piece yellow.png")
                    End If
                    .Text = Line
                    StoreBoard(x, y) = Line
                    sr.Close()
                End With
```

```
                Me.Controls.Add(Board(x, y))
            Next
        Next
        NameOfPlayer = System.IO.File.ReadAllLines("Save.txt")(42)
        i += 1
        PlayerTurn = System.IO.File.ReadAllLines("Save.txt")(43)
        i += 1
        PlayerColour = System.IO.File.ReadAllLines("Save.txt")(44)
        i += 1
        AIColour = System.IO.File.ReadAllLines("Save.txt")(45)
        i += 1
        OriginalDepth = System.IO.File.ReadAllLines("Save.txt")(46)
        AIName = "AI Level " & OriginalDepth
        If PlayerTurn = True Then
            DisplayTurn.Text = "Turn: " & NameOfPlayer
            TurnColour = PlayerColour
            If PlayerColour = "Yellow" Then
                DisplayTurn.ForeColor = ForeColor.Yellow
            Else
                DisplayTurn.ForeColor = ForeColor.Red
            End If
        Else
            DisplayTurn.Text = "Turn: " & AIName
            TurnColour = AIColour
            If AIColour = "Yellow" Then
                DisplayTurn.ForeColor = ForeColor.Yellow
            Else
                DisplayTurn.ForeColor = ForeColor.Red
            End If
        End If
    End Sub

    Sub ButtonClicked(sender As Object, e As EventArgs)
        'The players entire move happens here
        If AllowButtonClick = True Then
            AllowButtonClick = False
            If RulesPictureBox.Visible = False And EndOfGame = False Then
                If PlayerTurn = True Then
                    Dim clickedButton As Button = CType(sender, Button)
                    Dim PlacementChoice As Integer = clickedButton.Text
```

```
                If PlaceDisc(PlacementChoice) Then
                    movelist.Push(New move(PlayerColour, PlacementChoice))
                    If CheckForRuns(4, PlayerColour) Then WinningMove = True
                    WinOrDraw()
                    PlayerTurn = False
                    CheckForFullColumns()
                    AIMove()
                End If
            Else
                MsgBox("The AI must have their go first")
            End If
        End If
        AllowButtonClick = True
    End If
End Sub
Function PlaceDisc(ByVal PlacementChoice As Integer) As Boolean
    'Finds the lowest empty space in a collumn as long as the top of that collumn is empty
    If StoreBoard(PlacementChoice, 6) = "empty" Then
        For i = 1 To 6
            If Board(PlacementChoice, i).Text = "empty" And PlaceDisc = False Then
                If TurnColour = "Yellow" Then
                    StoreBoard(PlacementChoice, i) = "Yellow"
                    Board(PlacementChoice, i).Text = "Yellow"
                    Board(PlacementChoice, i).Image = Image.FromFile(ImageFolder & "Grid piece yellow.png")
                    PlaceDisc = True
                Else
                    StoreBoard(PlacementChoice, i) = "Red"
                    Board(PlacementChoice, i).Text = "Red"
                    Board(PlacementChoice, i).Image = Image.FromFile(ImageFolder & "Grid piece red.png")
                    PlaceDisc = True
                End If
            End If
        Next
    Else
        If PlayerTurn = True Then
            MsgBox("You can't place a disc in this collumn")
        End If
        PlaceDisc = False
    End If
    Return PlaceDisc
```

```
    End Function
    Function CheckForRuns(ByVal DiscsInARow As Integer, ByVal SearchColour As String) As Boolean
        Dim DiagonalNum As Integer
        Dim DiscsPlacedWithSpace As Integer
        Dim InARow As Integer
        Dim Holdx As Integer
        Dim AdditionalSpaces As Integer
        Dim EmptySpaceCounter As Integer
        'DiscsInARow will determine how many discs and spaces will be searched for
        If DiscsInARow = 4 Then
            DiscsPlacedWithSpace = 4
            AdditionalSpaces = 0
        ElseIf DiscsInARow = 3 Then
            DiscsPlacedWithSpace = 3
            AdditionalSpaces = 1
        ElseIf DiscsInARow = 2 Then
            DiscsPlacedWithSpace = 2
            AdditionalSpaces = 2
        ElseIf DiscsInARow = 1 Then
            DiscsPlacedWithSpace = 1
            AdditionalSpaces = 3
        End If

        'If at any point InARow = 4 the function returns true

        'Checks verticaly
        For x = 1 To 7
            InARow = 0
            For y = 1 To 6
                If StoreBoard(x, y) = SearchColour Or InARow >= DiscsPlacedWithSpace And StoreBoard(x, y) = "empty" Then
                    InARow += 1
                    If InARow = 4 Then Return True
                Else
                    InARow = 0
                End If
            Next
        Next

        'Checks horizontaly
        For y = 1 To 6
```

```
            For x = 1 To 4
                InARow = 0
                EmptySpaceCounter = 0
                For Steps = 0 To 3
                    If StoreBoard(Steps + x, y) = SearchColour Then
                        InARow += 1
                    ElseIf StoreBoard(Steps + x, y) = "empty" And EmptySpaceCounter < AdditionalSpaces Then
                        If y > 1 Then
                            If StoreBoard(Steps + x, y - 1) <> "empty" Then
                                InARow += 1
                                EmptySpaceCounter += 1
                            Else
                                InARow = 0
                                EmptySpaceCounter = 0
                            End If
                        Else
                            InARow += 1
                            EmptySpaceCounter += 1
                        End If
                    Else
                        InARow = 0
                        EmptySpaceCounter = 0
                    End If
                    If InARow = 4 Then Return True
                Next
            Next
        Next

        'Checks diagonal going up and right
        For y = 1 To 3
            For x = 1 To 4
                EmptySpaceCounter = 0
                DiagonalNum = 0
                InARow = 0
                Holdx = x
                For Holdx = Holdx To Holdx + 3
                    If StoreBoard(Holdx, y + DiagonalNum) = SearchColour Then
                        InARow += 1
                        DiagonalNum += 1
                    ElseIf StoreBoard(Holdx, y + DiagonalNum) = "empty" And EmptySpaceCounter < AdditionalSpaces Then
```

```
                    If y + DiagonalNum > 1 Then
                        If StoreBoard(Holdx, y + DiagonalNum - 1) = "empty" Then
                            InARow = 0
                            DiagonalNum = 0
                        Else
                            InARow += 1
                            EmptySpaceCounter += 1
                            DiagonalNum += 1
                        End If
                    Else
                        InARow += 1
                        EmptySpaceCounter += 1
                        DiagonalNum += 1
                    End If
                End If
                If InARow = 4 Then Return True
            Next
        Next
    Next

    'Checks diagonal going up and left
    For y = 1 To 3
        For x = 7 To 4 Step -1
            EmptySpaceCounter = 0
            DiagonalNum = 0
            InARow = 0
            Holdx = x
            For Holdx = Holdx To Holdx - 3 Step -1
                If StoreBoard(Holdx, y + DiagonalNum) = SearchColour Then
                    InARow += 1
                    DiagonalNum += 1
                ElseIf StoreBoard(Holdx, y + DiagonalNum) = "empty" And EmptySpaceCounter < AdditionalSpaces Then
                    If y + DiagonalNum > 1 Then
                        If StoreBoard(Holdx, y + DiagonalNum - 1) = "empty" Then
                            InARow = 0
                            DiagonalNum = 0
                        Else
                            InARow += 1
                            EmptySpaceCounter += 1
                            DiagonalNum += 1
```

```
                        End If
                    Else
                        InARow += 1
                        EmptySpaceCounter += 1
                        DiagonalNum += 1
                    End If
                End If
                If InARow = 4 Then Return True
            Next
        Next
    Next
    Return False
End Function

Sub WinOrDraw()
    'After being DontPlayAITurn, if there is a four in a row then the game will announce the winner and then prompt the player to
exit
    Dim WinnerName As String
    If WinningMove = True Then
        If PlayerTurn = True Then
            If NameOfPlayer = Nothing Or NameOfPlayer = " " Then
                WinnerName = TurnColour
            Else
                WinnerName = NameOfPlayer
            End If
        Else
            WinnerName = AIName
        End If
        MsgBox("Congratulations " & WinnerName & " you have won!")
        DisplayTurn.Text = WinnerName & " is the winner!"
        EndOfGame = True
        EndNotice.Text = "To return to the main menu"
        EndNotice2.Text = "click the Exit button"
    ElseIf Draw() = True Then
        'Else the game checks for a draw
        MsgBox("The game Is a draw!")
        DisplayTurn.Text = "The game is a draw"
        DisplayTurn.ForeColor = ForeColor.Black
        EndOfGame = True
        EndNotice.Text = "To return to the main menu"
```

```
            EndNotice2.Text = "click the Exit button"
        End If
        'If none of those conditions are met, the turn is changed and game continues
        If Not EndOfGame Then ChangeTurn()
    End Sub
    Sub CheckForFullColumns()
        For x = 1 To 7
            If StoreBoard(x, 6) <> "empty" Then
                DropButton(x).Image = Image.FromFile(ImageFolder & "Cross.png")
            Else
                DropButton(x).Image = Image.FromFile(ImageFolder & "Arrow.png")
            End If
        Next
    End Sub
    Function Draw() As Boolean
        'Each grid space is checked if an empty space is found then the grid is not full and the game is not a draw
        Dim IsFull As Boolean = True
        For x = 1 To 7
            For y = 1 To 6
                If Board(x, y).Text = "empty" Then
                    Return False
                End If
            Next
        Next
        Draw = IsFull
    End Function
    Sub ChangeTurn()
        If PlayerTurn = True Then
            PlayerTurn = False
            DisplayTurn.Text = "Turn: " & AIName
            TurnColour = AIColour
        Else
            PlayerTurn = True
            DisplayTurn.Text = "Turn: " & NameOfPlayer
            TurnColour = PlayerColour
        End If
        If TurnColour = "Yellow" Then
            DisplayTurn.ForeColor = ForeColor.Yellow
        Else
            DisplayTurn.ForeColor = ForeColor.Red
```

```
        End If
    End Sub
    Sub AIMove()
        Dim Start As Boolean = True
        Dim BestMove As Integer
        AllowButtonClick = False
        If EndOfGame = False And DontPlayAITurn = False Then
            If PlayerTurn = False Then
                Timer(Start)
                ProcessingText.Visible = True
                System.Windows.Forms.Application.DoEvents()
                BestMove = Minimax(True, BestMove, -99999999, 99999999, OriginalDepth)
                PlaceDisc(BestMove)
                movelist.Push(New move(AIColour, BestMove))
                WinOrDraw()
                ProcessingText.Visible = False
                Start = False
                Timer(Start)
            Else
                MsgBox("You must play your turn before letting the AI play")
            End If
        End If
        AllowButtonClick = True
        CheckForFullColumns()
    End Sub
    Sub Timer(Start)
        Static start_time As DateTime
        Static stop_time As DateTime
        Dim elapsed_time As TimeSpan
        If Start Then
            start_time = Now
        Else
            stop_time = Now
            elapsed_time = stop_time.Subtract(start_time)
            If Not EndOfGame Then ProcessingTime.Text = "Time taken: " & elapsed_time.TotalSeconds.ToString("0.000000") & "s"
        End If
    End Sub

    Function Minimax(ByVal MaximizeAI As Boolean, ByVal BestMove As Integer, ByRef Alpha As Integer, ByRef Beta As Integer, ByVal
depth As Integer) As Integer
```

```
'This function uses the minimax algorithm to find the best move for the AI
Dim Value As Integer
Dim HoldValue As Integer
Dim BestValue As Integer
Dim ListOfMoves(7) As Integer
Dim ListOfAllMovesVal(7) As Integer
Dim ListOfOpMoves(7) As Integer
Dim SizeOfList As Integer
Dim PlaceInList As Boolean
Dim Move As Integer
If depth = 0 Or Draw() Or CheckForRuns(4, AIColour) = True Or CheckForRuns(4, PlayerColour) = True Then
    SwapSwitch = False
    If MaximizeAI = True Then
        MaximizeAI = False
    Else
        MaximizeAI = True
    End If
    If MaximizeAI Then
        Value = EvealuateBoardMin(Value) + (OriginalDepth - depth)
        FakeChangeTurn()
        If Value > -558 Then
            HoldValue = EvealuateBoardMax(Value, depth)
            If HoldValue <> Value Then Value = HoldValue - (OriginalDepth - depth)
        End If
    Else
        Value = EvealuateBoardMax(Value, depth) - (OriginalDepth - depth)
        FakeChangeTurn()
        If Value < 558 Then
            HoldValue = EvealuateBoardMin(Value)
            If HoldValue <> Value Then Value = HoldValue + (OriginalDepth - depth)
        End If
    End If
    MoveValueTextBox.Text = ("Move value: " & Value)
    Minimax = Value
    Return Minimax
End If
If MaximizeAI Then
    BestValue = -99999999
    For Move = 1 To 7
        If PlaceDisc(Move) Then
```

```
                FakeChangeTurn()
                SwapSwitch = False
                Value = Minimax(False, BestMove, Alpha, Beta, depth - 1)
                If SwapSwitch Then Beta = 99999999
                PlaceInList = True
                BestValue = Max(BestValue, Value, Move, ListOfMoves, SizeOfList, PlaceInList, depth)
                PlaceInList = False
                BestMoveValue.Text = "Best Move Value: " & BestValue
                'Alpha-Beta pruning is used in order to increase the efficiency of the code
                Alpha = Max(Alpha, Value, Move, ListOfMoves, SizeOfList, PlaceInList, depth)
                If Beta <= Alpha Then
                    System.Windows.Forms.Application.DoEvents()
                    RemoveDisc(Move)
                    Exit For
                End If
                System.Windows.Forms.Application.DoEvents()
                RemoveDisc(Move)
            Else
                Value = -601
            End If
            If depth = OriginalDepth Then ListOfAllMovesVal(Move) = Value
        Next
        SwapSwitch = True
        FakeChangeTurn()
        If depth <> OriginalDepth Then
            Return BestValue
        End If
        Dim Ran As New Random
        ListOfBestMoves.Text = "Possible Moves: "
        For i = 1 To 7
            If ListOfMoves(i) <> 0 Then ListOfBestMoves.Text += CStr(ListOfMoves(i))
        Next
        BestMove = Ran.Next(1, SizeOfList + 1)
        BestMove = ListOfMoves(BestMove)
        FakeChangeTurn()
        Return BestMove
    Else
        BestValue = 99999999
        For Move = 1 To 7
            If PlaceDisc(Move) Then
```

```
                    FakeChangeTurn()
                    SwapSwitch = False
                    Value = Minimax(True, BestMove, Alpha, Beta, depth - 1)
                    If SwapSwitch Then Alpha = -99999999
                    PlaceInList = True
                    BestValue = Min(BestValue, Value, Move, ListOfOpMoves, PlaceInList)
                    BestMoveValue.Text = "Best Move Value: " & BestValue
                    PlaceInList = False
                    Beta = Min(Beta, Value, Move, ListOfOpMoves, PlaceInList)
                    If Beta <= Alpha Then
                        System.Windows.Forms.Application.DoEvents()
                        RemoveDisc(Move)
                        Exit For
                    End If
                    System.Windows.Forms.Application.DoEvents()
                    RemoveDisc(Move)
                Else
                    Value = 601
                End If
                ListOfAllMovesVal(Move) = Value
            Next
            SwapSwitch = True
            FakeChangeTurn()
            Minimax = BestValue
            Return Minimax
        End If
    End Function
    Function Max(ByRef BestValue As Integer, ByVal Value As Integer, ByVal Move As Integer, ByRef ListOfMoves() As Integer, ByRef
SizeOfList As Integer, ByVal PlaceInList As Boolean, ByVal depth As Integer
                ) As Integer
        Dim MovePlacedInList As Boolean
        If Value > BestValue Then
            BestValue = Value
            If depth = OriginalDepth Then
                If PlaceInList Then
                    ListOfMoves(1) = Move
                    SizeOfList = 1
                    For i = 2 To 7
                        ListOfMoves(i) = Nothing
                    Next
```

```
                End If
            End If
        ElseIf Value = BestValue And PlaceInList And depth = OriginalDepth Then
            MovePlacedInList = False
            For i = 1 To 7
                If ListOfMoves(i) = Nothing And MovePlacedInList = False Then
                    SizeOfList += 1
                    ListOfMoves(i) = Move
                    MovePlacedInList = True
                End If
            Next
        End If
        Return BestValue
    End Function

    Function Min(ByVal BestValue As Integer, ByVal Value As Integer, ByVal Move As Integer, ByRef ListOfOpMoves() As Integer, ByVal
PlaceInList As Boolean) As Integer
        Dim MovePlacedInList As Boolean
        If Value < BestValue Then
            BestValue = Value
            If PlaceInList Then
                ListOfOpMoves(1) = Move
                For i = 2 To 7
                    ListOfOpMoves(i) = Nothing
                Next
            End If
        ElseIf Value = BestValue And PlaceInList Then
            MovePlacedInList = False
            For i = 1 To 7
                If ListOfOpMoves(i) = Nothing And MovePlacedInList = False Then
                    ListOfOpMoves(i) = Move
                    MovePlacedInList = True
                End If
            Next
        End If
        Return BestValue
    End Function
    Function EvealuateBoardMax(ByVal Value As Integer, ByVal depth As Integer) As Integer
        For DiscsInARow = 4 To 1 Step -1
            If CheckForRuns(DiscsInARow, AIColour) Then
```

```
                If DiscsInARow = 4 Then
                    Value = 600
                    If depth = OriginalDepth - 1 Then WinningMove = True
                    Return Value
                ElseIf DiscsInARow = 3 And Value < 400 And Value > -358 Then
                    'If AI can get 3 in a row with 1 space
                    Value = 400
                ElseIf DiscsInARow = 2 And Value < 300 And Value > -258 Then
                    'If AI can get 2 in a row with 2 spaces
                    Value = 300
                ElseIf DiscsInARow = 1 And Value < 200 And Value > -158 Then
                    'If AI can get 1 in a row with 3 spaces
                    Value = 200
                ElseIf Value < 100 And Value > -58 Then
                    'If the next move can make a draw
                    Value = 100
                End If
            End If
        Next
        Return Value
    End Function
    Function EvealuateBoardMin(ByVal Value As Integer) As Integer
        For DiscsInARow = 4 To 1 Step -1
            If CheckForRuns(DiscsInARow, PlayerColour) Then
                If DiscsInARow = 4 Then
                    Value = -600
                    Return Value
                ElseIf DiscsInARow = 3 And Value > -400 And Value < 358 Then
                    'If AI can get 3 in a row with 1 space
                    Value = -400
                ElseIf DiscsInARow = 2 And Value > -300 And Value < 258 Then
                    'If AI can get 2 in a row with 2 spaces
                    Value = -300
                ElseIf DiscsInARow = 1 And Value > -200 And Value < 158 Then
                    'If AI can get 1 in a row with 3 spaces
                    Value = -200
                ElseIf Value > -100 And Value < 58 Then
                    'If the next move can make a draw
                    Value = -100
                End If
```

```
            End If
        Next
        Return Value
    End Function
    Sub RemoveDisc(ByRef Move As Integer)
        Dim Removed As Boolean = False
        For i = 6 To 1 Step -1
            If Removed = False Then
                If StoreBoard(Move, i) <> "empty" Then
                    StoreBoard(Move, i) = "empty"
                    Board(Move, i).Image = Image.FromFile(ImageFolder & "Grid piece.png")
                    Board(Move, i).Text = "empty"
                    Removed = True
                End If
            End If
        Next
    End Sub
    Sub FakeChangeTurn()
        'Only changes TurnColour
        If TurnColour = PlayerColour Then
            TurnColour = AIColour
        Else
            TurnColour = PlayerColour
        End If
    End Sub
    Private Sub SaveGameButton_Click(sender As Object, e As EventArgs) Handles SaveGameButton.Click
        'Using StreamWriter each collumn is written in a text file in the debug ImageFolder
        If AllowButtonClick = True Then
            AllowButtonClick = False
            If RulesPictureBox.Visible = False Then
                If Not EndOfGame Then
                    Dim sw As New StreamWriter("Save.txt")
                    For x = 1 To 7
                        For y = 1 To 6
                            sw.WriteLine(Board(x, y).Text)
                        Next
                    Next
                    'The rest of the parameters are stored after the grid
                    sw.WriteLine(NameOfPlayer)
                    sw.WriteLine(PlayerTurn)
```

```
                sw.WriteLine(PlayerColour)
                sw.WriteLine(AIColour)
                sw.WriteLine(OriginalDepth)
                sw.Close()
            Else
                MsgBox("You cannot save the game once it has finished")
            End If
        End If
        AllowButtonClick = True
    End If
End Sub
Private Sub RulesButton_Click(sender As Object, e As EventArgs) Handles RulesButton.Click
    If RulesPictureBox.Visible = False Then
        'Have it load in an image that has the rules of connect 4
        RulesPictureBox.Visible = True
        RulesPictureBox.BringToFront()
        RulesButton.Text = "Close"
    Else
        RulesPictureBox.Visible = False
        RulesButton.Text = "Rules"
    End If
End Sub
Private Sub ExitButton_Click(sender As Object, e As EventArgs) Handles ExitButton.Click
    If RulesPictureBox.Visible Then
        RulesPictureBox.Visible = False
        RulesButton.Text = "Rules"
    Else
        Main_Menu.Show()
        Me.Close()
    End If
End Sub
Public Sub Connect_4_Game_Close(sender As Object, e As EventArgs) Handles MyBase.Closed
    Main_Menu.Show()
    Me.Hide()
End Sub
Private Sub UndoMove_Click(sender As Object, e As EventArgs) Handles UndoMove.Click
    Dim tempMove As move
    If AllowButtonClick And Not EndOfGame Then
        Try
            tempMove = movelist.Pop()
```

```
                RemoveDisc(tempMove.GetColumn)
                ChangeTurn()
                If Not PlayerTurn And DontPlayAITurn = False And AllowButtonClick Then AIMove()
                CheckForFullColumns()
            Catch
                MsgBox("You cannot do this action")
            End Try
        End If
    End Sub
    Private Sub AIMoveCheck_DontPlayAITurnChanged(sender As Object, e As EventArgs) Handles AIMoveCheck.CheckedChanged
        If DontPlayAITurn = True Then
            DontPlayAITurn = False
            If Not PlayerTurn And AllowButtonClick Then AIMove()
        Else
            DontPlayAITurn = True
        End If
    End Sub
End Class
Class move
    Private Property colour As String
    Private Property column As Integer
    Sub New(ByVal colour As String, column As Integer)
        Me.colour = colour
        Me.column = column
    End Sub
    Function GetColumn() As Integer
        Return column
    End Function
End Class
```

Code Appendix