Assignment 2 Report

1 Team members

Name	ID	Department
Ahmed Ashraf Mohamed	2022446758	Business
Hashem Ahmed Abd El Hafiz	20221445676	AI
Abdelrahman Mohamed Abdelhadi	2022513643	AI
Antonuose Gerges Nageh	20221903971	AI

2 Herustic used

$$V(cell, piece) = \begin{cases} 100000 & \text{if the cell has 4 pieces of the same specified kind} \\ 100 & \text{if the cell has 3 pieces of the specified kind and 1 empty tile} \\ 1 & \text{if the cell has 2 pieces of the specified kind and 2 empty tiles} \\ -80 & \text{if the cell has 3 pieces of the enemy piece and 1 empty cell} \\ -100000 & \text{if the cell has 4 piece of the enemy piece} \end{cases}$$

2.1 Breakdown of the how the function will be used

We will be itearting over the board, where each row,column, postively sloped diagonals, and negatively sloped diagonals are going to broken down into cells of length 4, and each cell will be evaluated individually. The Score returned from evaluated cells is going increatement the global board Score or decrement it.

3 Data Structures used

1. `ArrayList` to the store the valid location where the piece can be dropped
2. `TreeNode` A custom made data Structure that maintains a tree used for printing the MiniMax Tree.
3. `GameSettings` A data Structure from FXGL library that initialize the settings
4. `Input` used to get the input from the user e.g. the mouse position
5. `UserAction` defines the actions that can be made by the user.
6. `Color` controls the color of the piece in the GUI
7. `Entity` a FXGL data structure that is used to show Entities in the game GUI.
8. `EntityType` defines the type of the Entities that is going to be used in the game.

9. `Text` JavaFX data Structure that initialize a text that is going to displayed in the game UI.

10. `Map<String,Object> vars` initializes observalbe game variable.

11. `FXGLMenu` used to create a custom Main Menu.

12. `Rectangle` used to define the shape of the button in the main menu.

13. `Animation` used to build animation when the Rectangle is hovered over.

14. `Point2D` used to to track the mouse position.

3.1 Creating the Game GUI

1. `createGUIBoard():void` creates the board in the game GUI.

2. `changeBallColor(Color color):void` Changes the ball color when the user has already played. It is redundent since there is no human second player.

3.2 Game Variables

1. `PLAYER,AI,EMPTY` used to denote the piece in the terminal game board.

2. `TILE_SIZE` used to control the size of the Tile.

3. `SPEED` used to control the ball speed when the user uses the keyboard controls

4. `MAX_ROWS,MAX_COLS` controls the Rows and columns of the game board.

5. `MAX_WIDTH,MAX_HEIGHT` controls the size of window using the maximum rows and columns

6. `OFFSET` when creating the cells for evaluation there is an offset where if the iterator reaches the fiveth column it will case an out-of-bounds error.

7. `CELL_LENGTH` used to define the length of the cells. This variable was created to avoid Magic numbers

8. `terminalBoard:int[][]`
   The main board which is going to updated througout the game runtime.

9. `boardTree` the tree that is going to be maintained throught the game runtime.

10. `playerScore,aiScore` Keeps track of the score of the game during its runtime.

3.3 Game functions

1. `createBoard():int[][]`
   creates an empty board.

2. `dropPiece(int[][] board, int row, int col, int piece):void`
   takes in the board,row,column, and the piece that is going to be drooped.

3. `isValidLocation(int[][] board, int col):boolean`
   takes in the board and the column and checks if the column has an empty space where the piece is going to be dropped.

4. `getOpenRow(int[][] board, int col):int` takes in the board and the column to get the next open row where the piece can be dropped.

5. `printBoard(int[][] board):void` prints the board in the terminal.

6. `GameOver(int[][] board, int piece):boolean` Classical check of the game. Checks if the and piece has a connect four

7. `GameOver(int[][] board):boolean` Custom check for the game where the game ends when there are no more valid locatoins

8. `evaluateCell(int[] cell, int piece):int` takes in the cell and the piece that is going to evaluated and returns the score.

9. `connectFour(int[] cell, int piece):int` takes in the cell and checks if there is a connect four

10. `countPieces(int[] arr, int piece):int` counts the pieces of the specified type in the cell.

11. `scorePosition(int[][] board, int piece):int` socres the game position for the game aiScore

12. `numberFours(int[][] board, int piece):int` counts the number of connect fours and updates the score of the game.

13. `makeColArr(int[][] board, int c):int[]`
    takes in the game board and the column and returns the column as an array

14. `makeRowArr(int[][] board, int r):int[]`
    takes in the game board and the row and retusn the row as an array

15. `isTerminalNode(int[][] terminalBoard):boolean`
    classical check if the node is terminal

16. `getValidLocations(int[][] board):ArrayList<Integer>` takes in the board and returns the valid location where the piece can be dropped.

17. `minimax(int[][] board, int depth, boolean maxPlayer, TreeNode node):int[]` implementes the minimax alogrithm and returns the column and score as an array where the column is index 0 and score is index 1. It takes as well the tree that is going to maintained throughout its run.

18. `minimax(int[][] board, int depth, int alpha, int beta, boolean maxPlayer, TreeNode node):int[]` implementes the minimax alogrithm with pruning.

19. `createGUIBoard():void` creats the gui board where the `terminalBoard` is going to be used

3.4 Entity Factory

The entity factory data structure is used to create the spawns or Entities at the correct loctaion during the game runtime A tree data Structure was used to print the MiniMax Tree during its run. But the alogrithm can be implemented without a tree as it implementes depth-first search, where there is no need for tree to decrease memory usage.

4 Libraries used

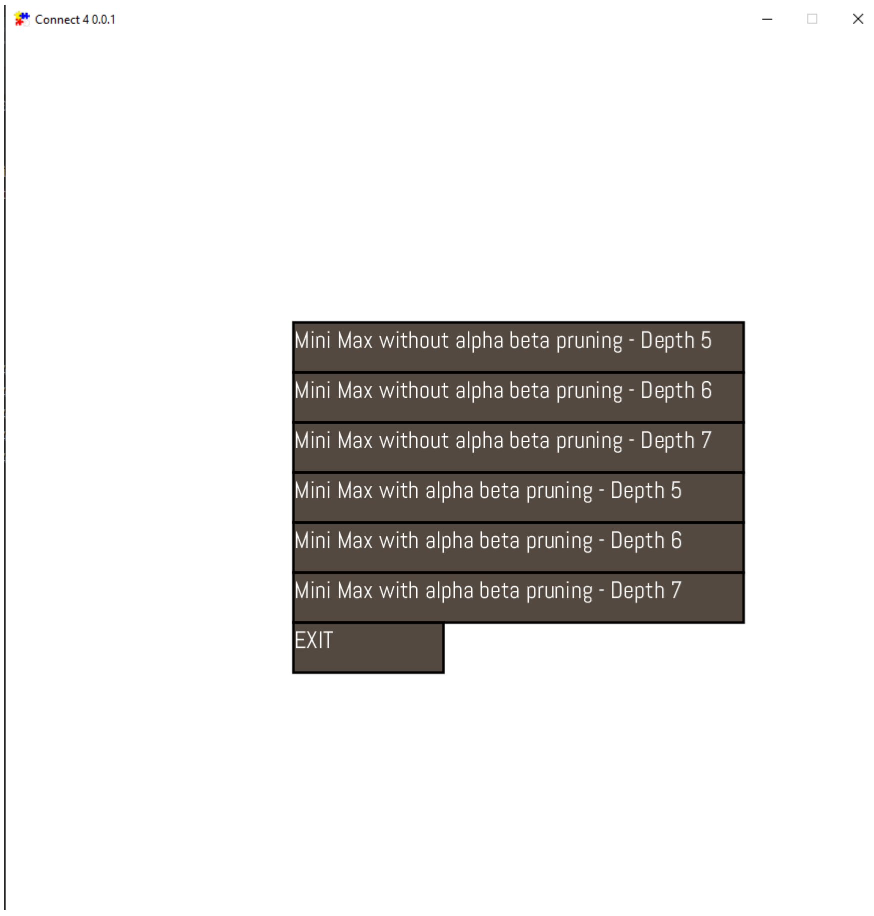
1. FXGL: Game engine library built upon JavaFX

2. JavaFX

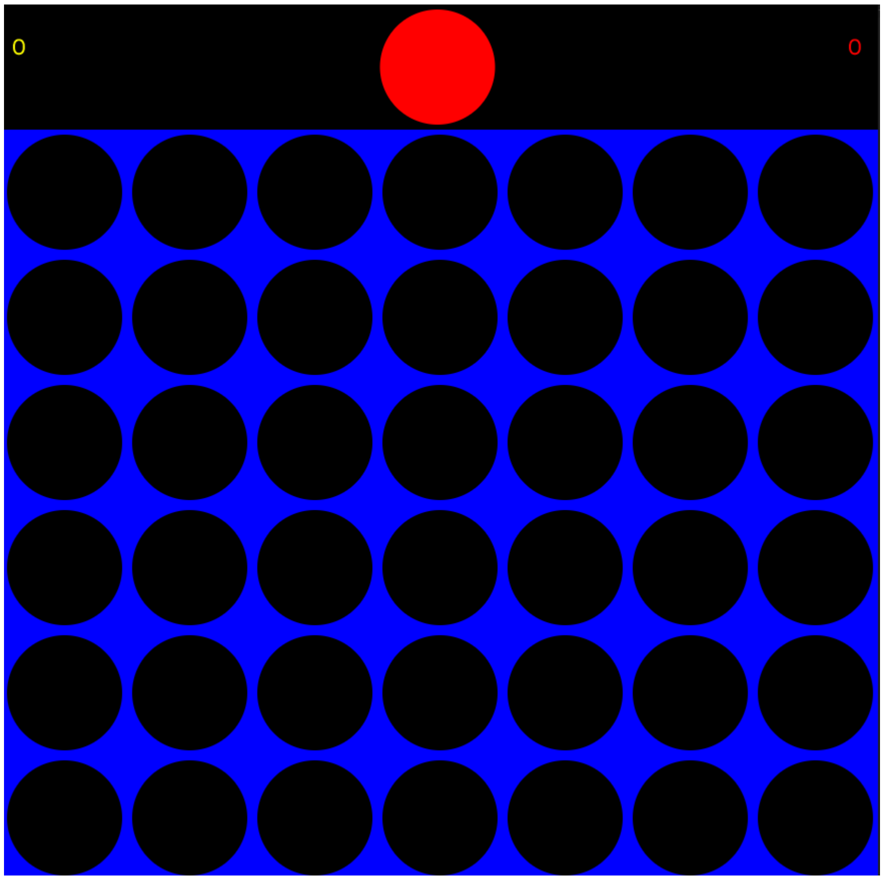
5 Sample runs

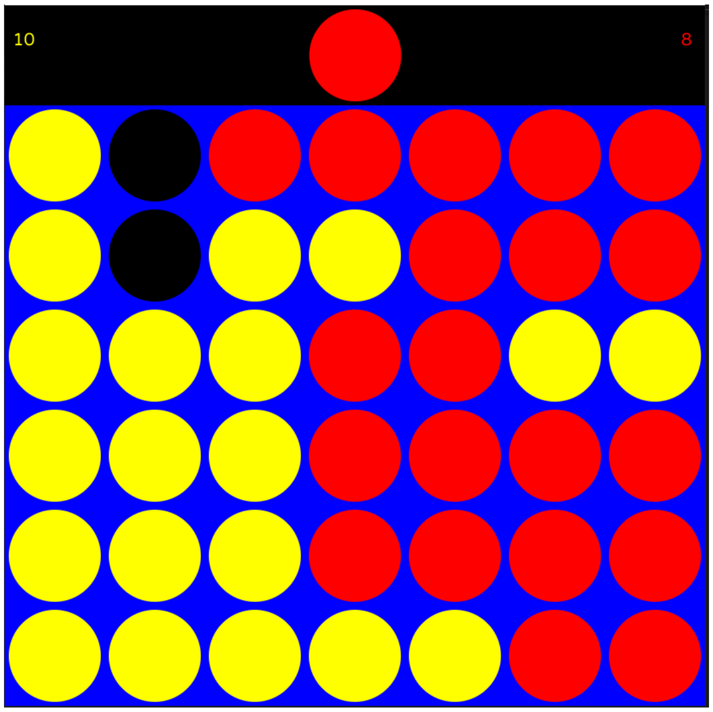
5.1 GUI

5.1.1 Main Menu


Connect 4 0.0.1
Mini Max without alpha beta pruning - Depth 5
Mini Max without alpha beta pruning - Depth 6
Mini Max without alpha beta pruning - Depth 7
Mini Max with alpha beta pruning - Depth 5
Mini Max with alpha beta pruning - Depth 6
Mini Max with alpha beta pruning - Depth 7
EXIT

5.1.2 Game





5.2 Mini Max Without $\alpha - \beta$ pruning Depth 5

We won't be able to show the whole tree since the tree is huge, so, we are showing parts of it not all of it. We also will be showing only two turns.

The first turn.

```
The new Board after the player played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
2
|---  1
|   |---  2
|   |   |---  1
|   |   |   |---  1
|   |   |   |   |---  0
|   |   |   |   |---  1
|   |   |   |   |---  1
|   |   |   |   |---  1
|   |   |   |   |---  0
|  |  |  |  |---  0
|  |  |  |  |---  0
|  |  |  |---  1
.
.
.
Time Taken : 267.0 Millie Seconds
Nodes Expanded : 19607
Depth : 5
alpha beta : false
The new Board after AI Played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 2 0 0 0 0 1
```

The second turn.

```
The new Board after the player played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
0 2 0 0 0 0 1
--------------------
5
|---  2
|  |---  23
|  |  |---  2
|  |  |  |---  101
|  |  |  |  |---  1
|  |  |  |  |---  3
|  |  |  |  |---  101
|  |  |  |  |---  101
|  |  |  |  |---  2
|  |  |  |  |---  1
|  |  |  |  |---  1
|  |  |  |---  101
.
.
.
Time Taken : 266.0 Millie Seconds
Nodes Expanded : 19606
Depth : 5
```

```
alpha beta : false
The new Board after AI Played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
0 2 2 0 0 0 1
--------------------
```

The final turn

```
2147483647
|---  2147483647

Time Taken : 0.0 Millie Seconds
Nodes Expanded : 1
Depth : 5
alpha beta : false
The new Board after AI Played :
2 1 1 1 1 1 1
2 1 1 1 2 1 2
2 2 1 2 1 2 1
2 2 2 2 1 1 1
2 2 2 2 1 1 1
2 2 2 2 2 1 1
--------------------
Player 2 Wins
Average time : 27.88095238095238
Average nodes expanded : 3357
```

5.3 Mini Max with $\alpha - \beta$ pruning-Depth 5

The first turn

```
    The new Board after the player played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
--------------------
2
|---  1:1:2147483647
|  |---  2:-2147483648:2
|  |  |---  1:1:2147483647
|  |  |  |---  1:-2147483648:1
|  |  |  |  |---  0:0:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |  |---  1:1:2147483647
|  |  |  |---  1:-2147483648:1
|  |  |  |---  1:-2147483648:1
|  |  |  |---  1:-2147483648:1
|  |  |  |---  1:-2147483648:1
|  |  |  |---  1:-2147483648:1
|  |  |  |---  1:-2147483648:1
|  |  |---  2:2:2147483647
.
.
.
Time Taken : 49.0 Millie Seconds
Nodes Expanded : 1501
Depth : 5
alpha beta : true
The new Board after AI Played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 2 0 0 0 0 1
--------------------
```

The second turn

```
    The new Board after the player played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
0 2 0 0 0 0 1
--------------------
5
|---  2:2:2147483647
|  |---  23:-2147483648:23
|  |  |---  2:2:2147483647
|  |  |  |---  101:-2147483648:101
|  |  |  |  |---  1:1:2147483647
|   |   |   |   |---  3:3:2147483647
|   |   |   |   |---  101:101:2147483647
|   |   |   |   |---  101:101:2147483647
|   |   |   |   |---  101:101:2147483647
|   |   |   |   |---  101:101:2147483647
|   |   |   |   |---  101:101:2147483647
|   |   |   |---  101:-2147483648:101
|   |   |   |   |---  2:2:101
|   |   |   |   |---  2:2:101
|   |   |   |---  2:-2147483648:2
.
.
.
Time Taken : 74.0 Millie Seconds
Nodes Expanded : 2359
Depth : 5
alpha beta : true
The new Board after AI Played :
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
0 0 0 0 0 0 1
0 2 2 0 0 0 1
--------------------
```

The final turn

```
Time Taken : 0.0 Millie Seconds
Nodes Expanded : 0
Depth : 5
alpha beta : true
The new Board after AI Played :
2 1 1 1 1 1 1
2 1 1 1 2 1 2
2 2 1 2 1 2 1
2 2 2 2 1 1 1
2 2 2 2 1 1 1
2 2 2 2 2 1 1
--------------------
Player 2 Wins
Average time : 10.261904761904763
Average nodes expanded : 787
```

6 Comparsion between MiniMax with pruning and Without pruning

6.1 At depth 5

Table 1: Depth 5

	With Pruning		Without Pruning	
Turn	Time	Nodes	Time	nodes
1	47.0	1501	363.0	19607
2	79.0	2359	209.0	19606
3	31.0	4028	97.0	19575
4	25.0	3390	85.0	19190
5	18.0	2244	42.0	9330
6	10.0	620	45.0	9330
7	33.0	2726	47.0	9329
8	31.0	4746	64.0	9302
9	11.0	2202	41.0	7615
10	9.0	1625	15.0	3903
11	27.0	2494	14.0	3901
12	9.0	1876	27.0	3859
13	13.0	1716	12.0	3505
14	15.0	753	5.0	1242
15	1.0	410	24.0	892
16	1.0	167	1.0	349
17	1.0	103	1.0	289
18	1.0	105	1.0	167
19	0.0	5	0.0	33
20	0.0	2	0.0	8
21	0.0	0	0.0	1
Average	17.24	1574.9	51.5	6715.9

Table 2: At Depth 6

	With Pruning		Without Pruning	
Turn	Time	Nodes	Time	nodes
1	120.0	4643	861.0	137255
2	110.0	8395	580.0	137217
3	273.0	53938	603.0	136609
4	27.0	3907	624.0	110069
5	19.0	4265	181.0	55953
6	28.0	5476	189.0	55951
7	18.0	2004	371.0	55519
8	89.0	5272	389.0	52347
9	8.0	1586	137.0	39070
10	5.0	1026	45.0	13195
11	16.0	1620	21.0	5439
12	32.0	1561	72.0	5287
13	16.0	1681	55.0	4643
14	6.0	1187	13.0	3075
15	6.0	596	4.0	1005
16	2.0	396	6.0	783
17	2.0	258	3.0	376
18	2.0	214	1.0	244
19	0.0	5	1.0	19
20	0.0	1	0.0	3
21	0.0	0	0.0	1
Average	37.1	4668.14	197.9	38764.76