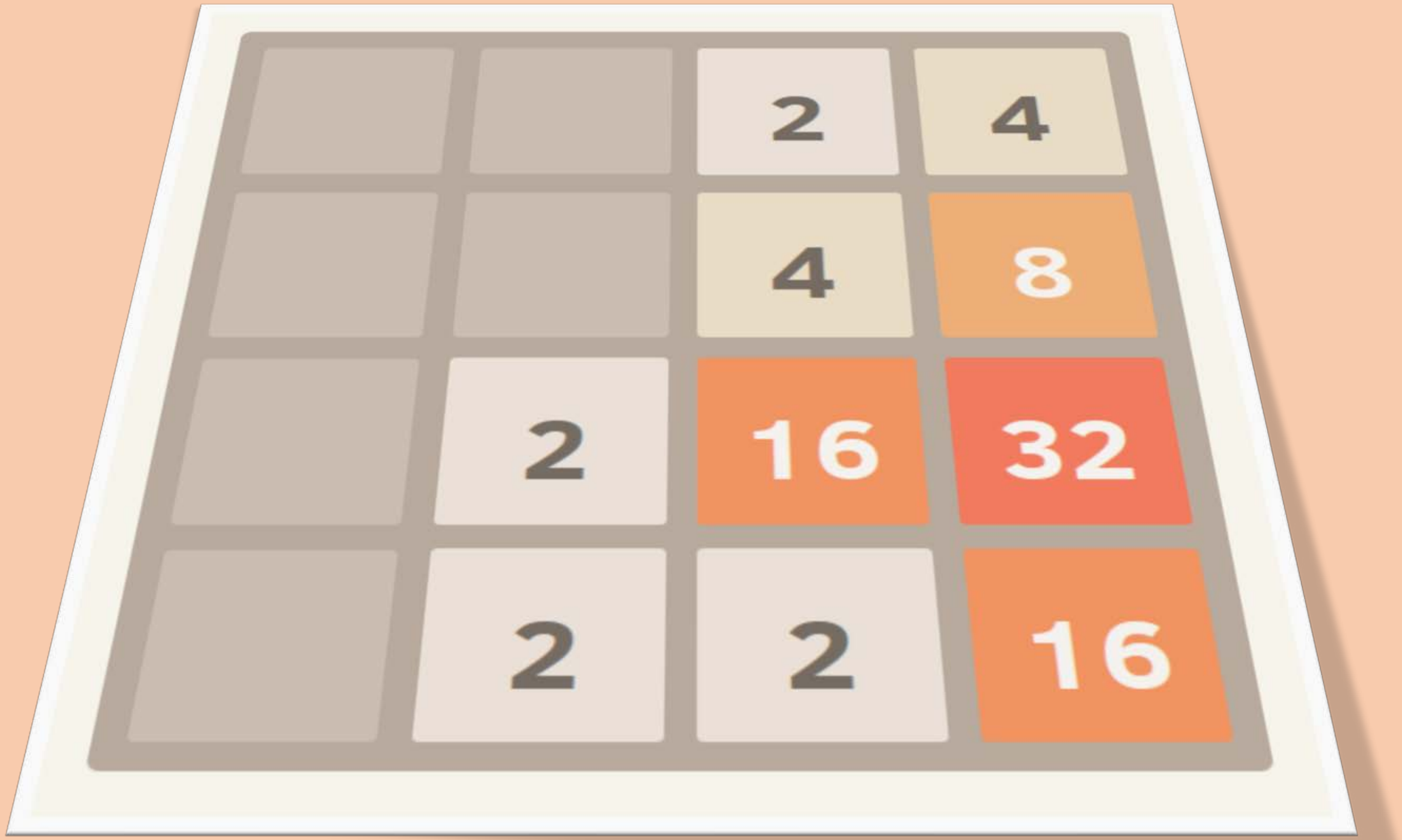
2048 GAME

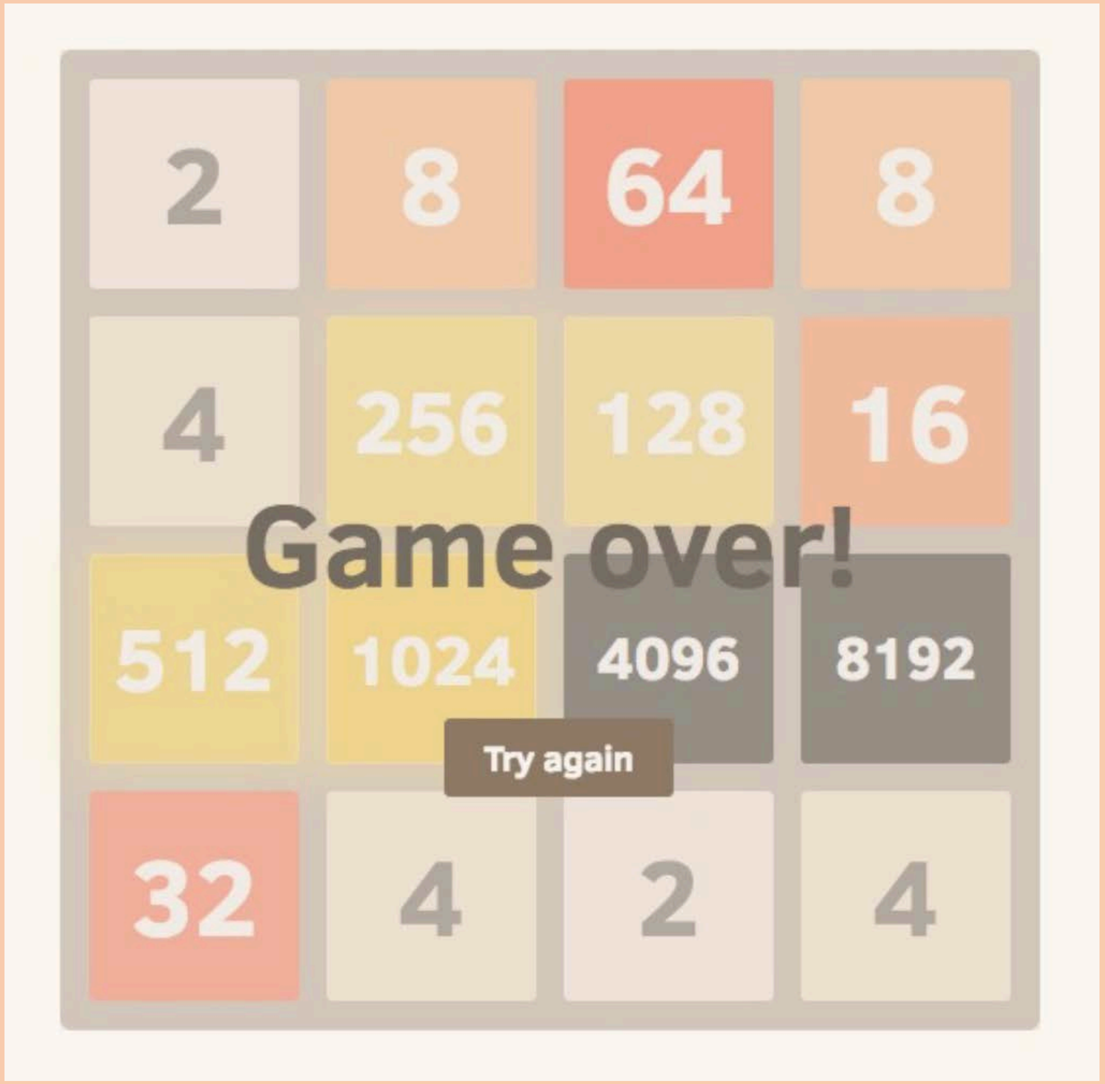


21632547- Mekiye **Helin DOĞAN**
21632662- **Firdevs KAMİŞLİ**

What is the 2048 Game?

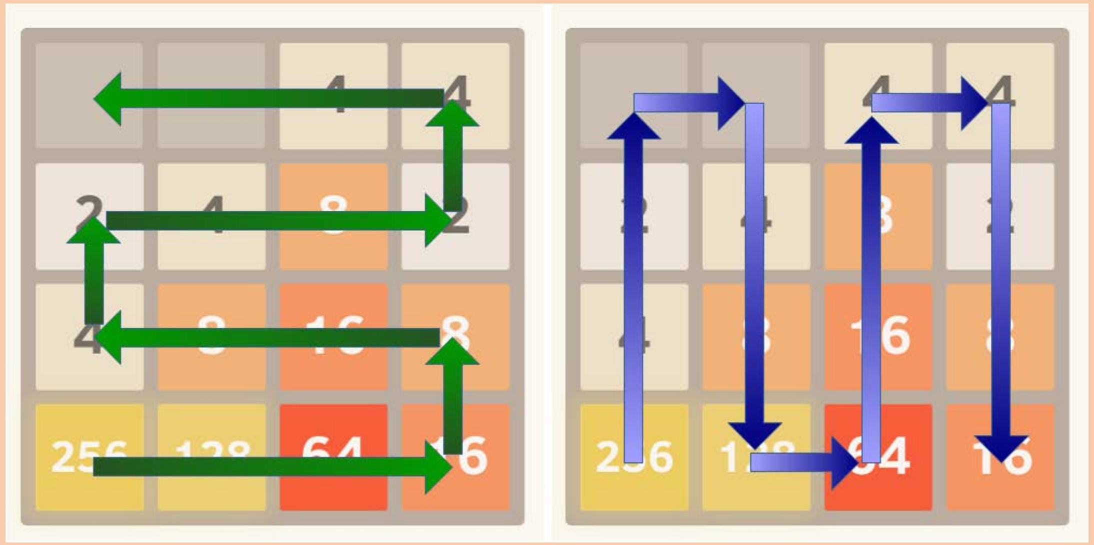
- 2048 is a single-player sliding block puzzle game designed by Italian web developer **Gabriele Cirulli**.
- The working principle is about collection.
- If you think you made the wrong move, you can undo the last 5 moves in the game. (Stack (LIFO))

What’s the goal of the game?

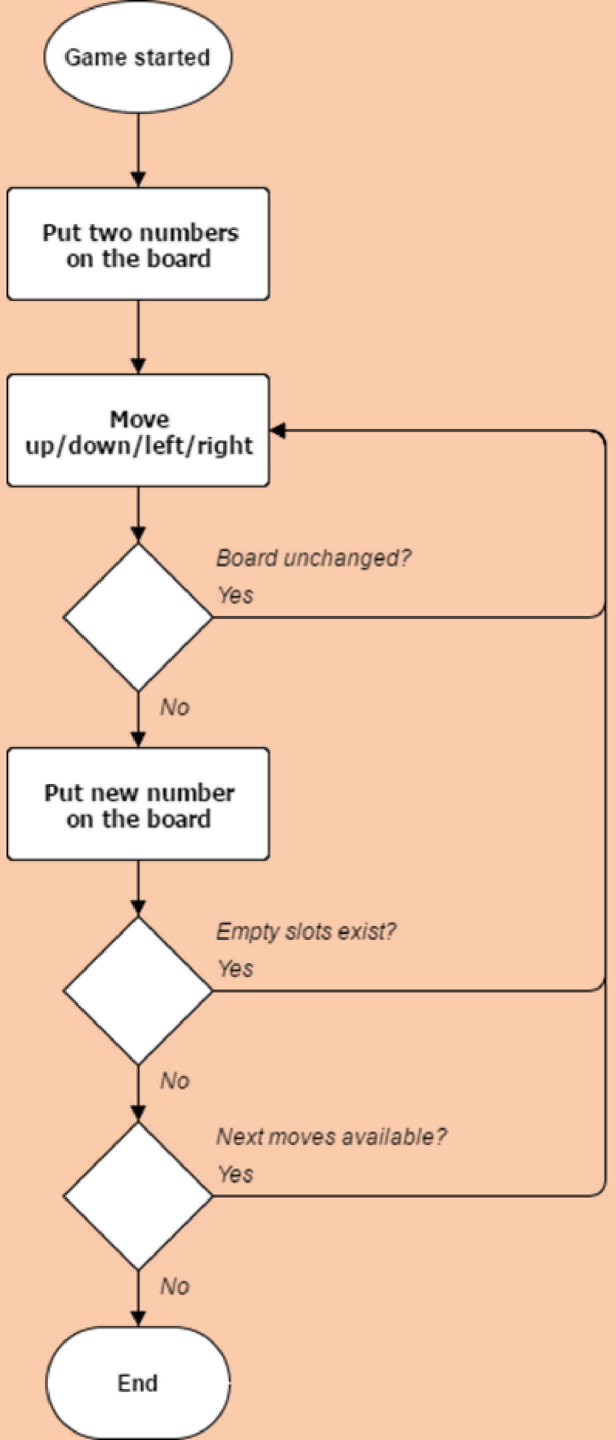


- ➢The goal is to reach 2048 but the game continues when you reach the goal.
- ➢To score as many points as possible.
- ➢If all blocks are full, game is over

How to Play?



- Swipe ↑, →, ↓, or ← to move the tiles. Every move generates a new tile at a random unoccupied position.
- Every move generates a new tile at a random blank position.
- When we swipe for one right, all blocks moves towards right. The same rule applies to all transactions

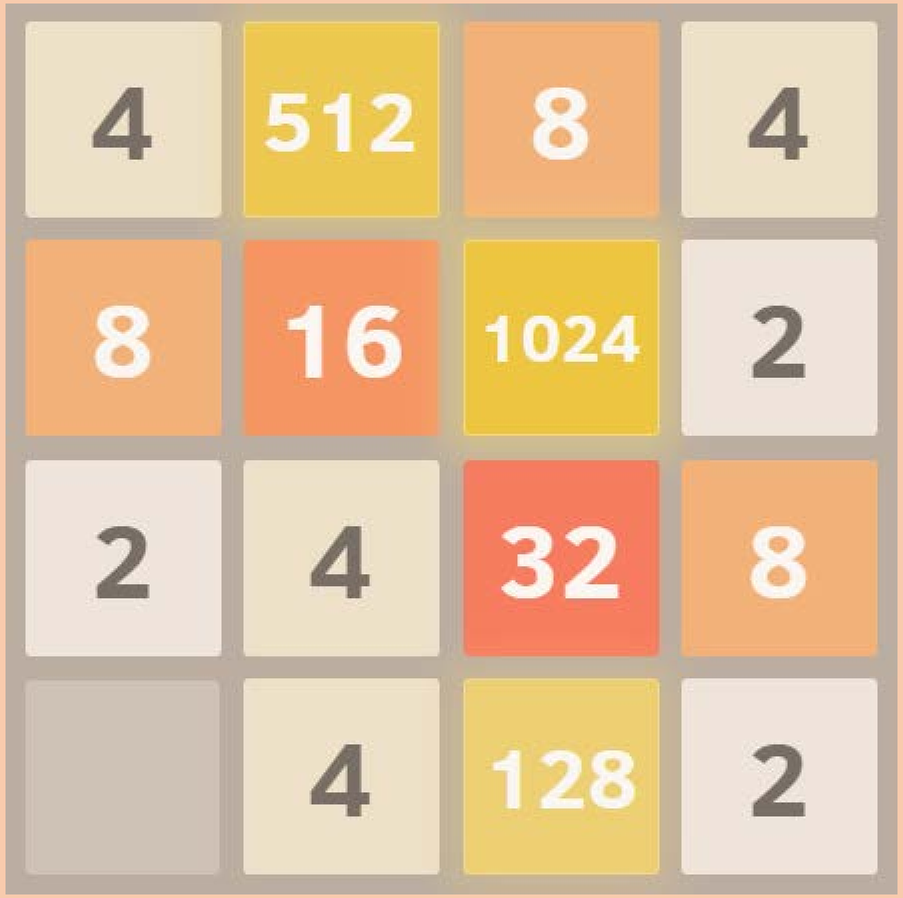


2048 Game Algorithm

- **Step1 :** Select board size(matrix size)
- **Step2 :** Press start button
- **Step3 :** Randomize the first two numbers.
- **Step4 :** Move up/down/left/right
- **Step5 :** Generate random number
- **Step6 :** Collect the same number. If array is empty, game is contuine. If array is not empty, game is over and go to step1
- **Step7:** The collection continues when the same numbers are added and the field is empty. The game is won when 2048 is achieved.

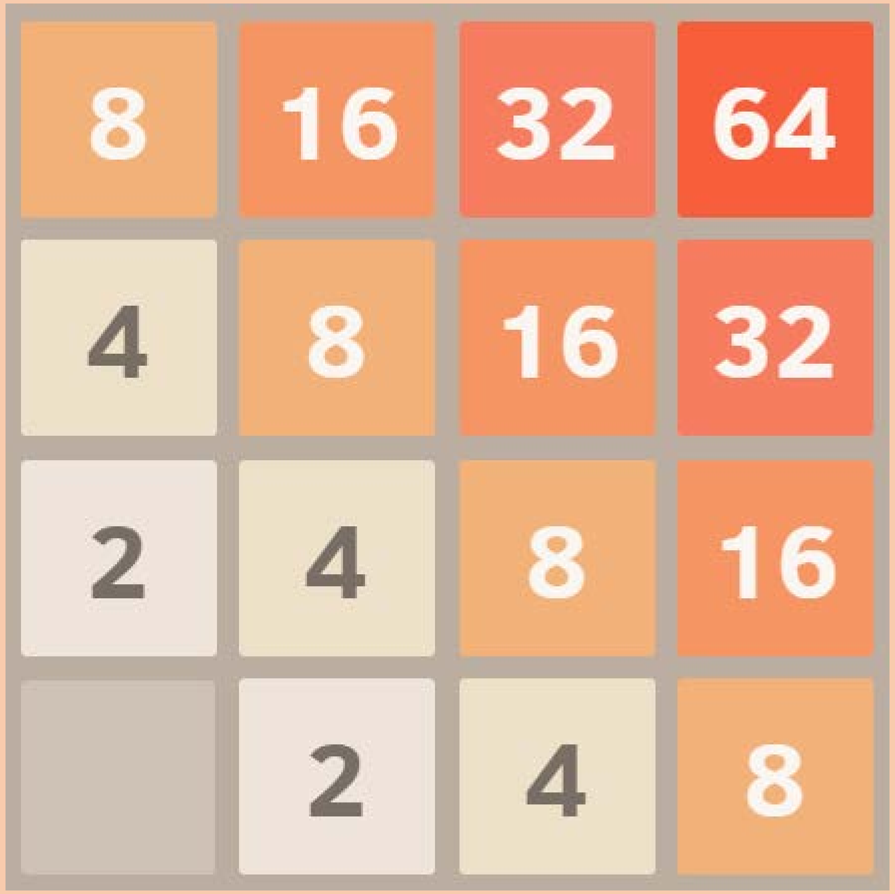
How to solve the 2048 game with Artificial Intelligence(AI)?

- It is optimal to keep the tile with the highest value in one of the corners
- Rows of tiles should be monotonic, so we can easily add them up
- The more empty tiles, the higher chance of not getting blocked
- The difference between adjacent tiles should be as small as possible
- ***The idea*** is just comparing resultant grids after every possible swipe and then picking the best move.



4	512	8	4
8	16	1024	2
2	4	32	8
	4	128	2

Not a really smooth grid



8	16	32	64
4	8	16	32
2	4	8	16
	2	4	8

Perfectly monotonic grid in both directions

Monte-Carlo (MC)

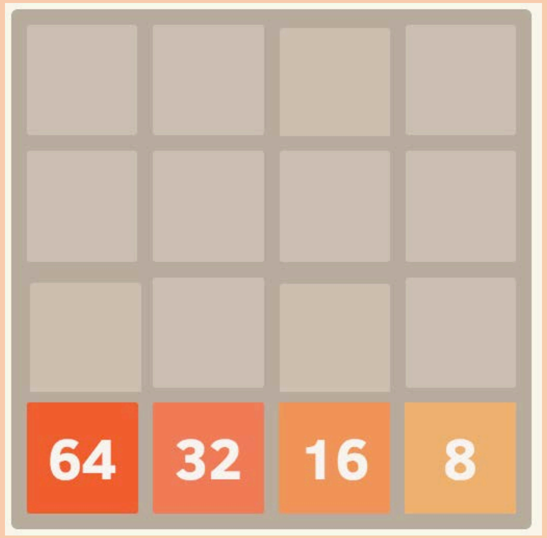
❑**One of the possible ways to solve the game of 2048 is to exploit the Monte-Carlo algorithm.**

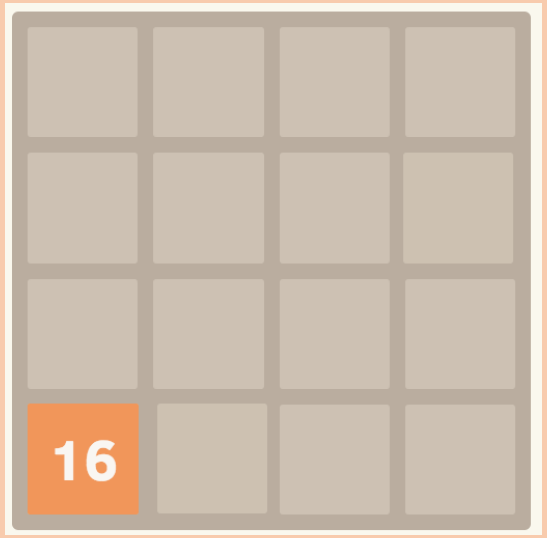
Monte-Carlo is a search algorithm that allows you to implement the most advantageous moves. This is a technique where the computer does a bunch of random simulations and tries to draw conclusions based on the results.

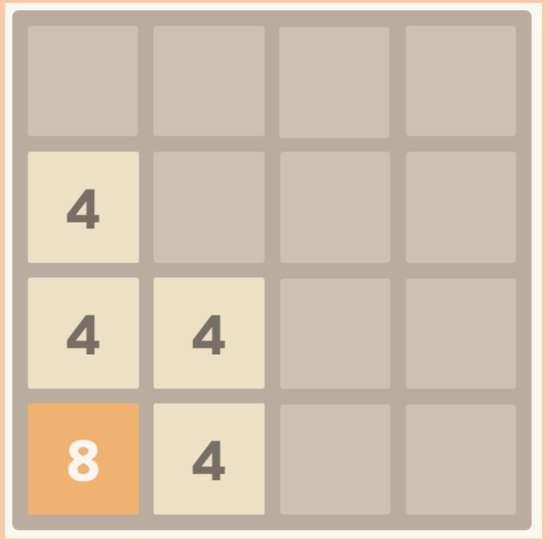
How to work?

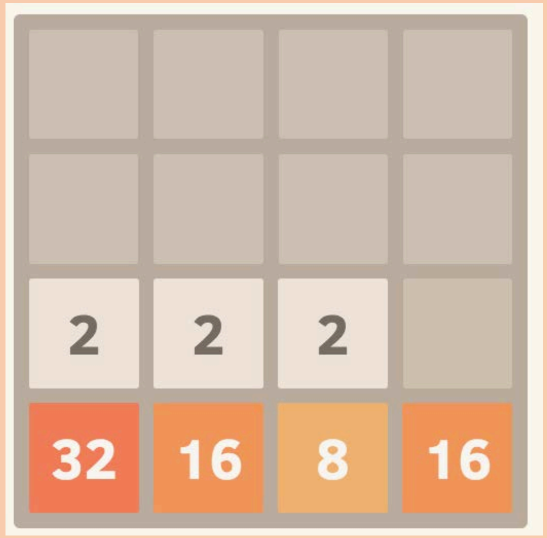
- Execute a series of background runs.
- Group them by the initial move.
- Count an average final score for each initial move.
- Pick the initial move with the highest average final score.

Implementation









1) Monotonically increasing or decreasing values on board edges

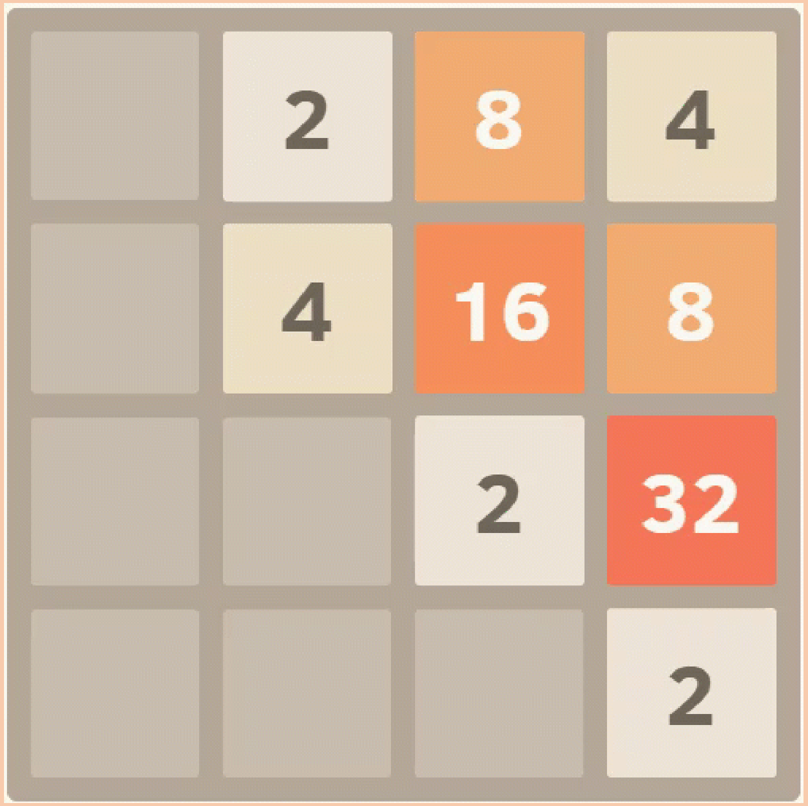
2) Emptiness of board

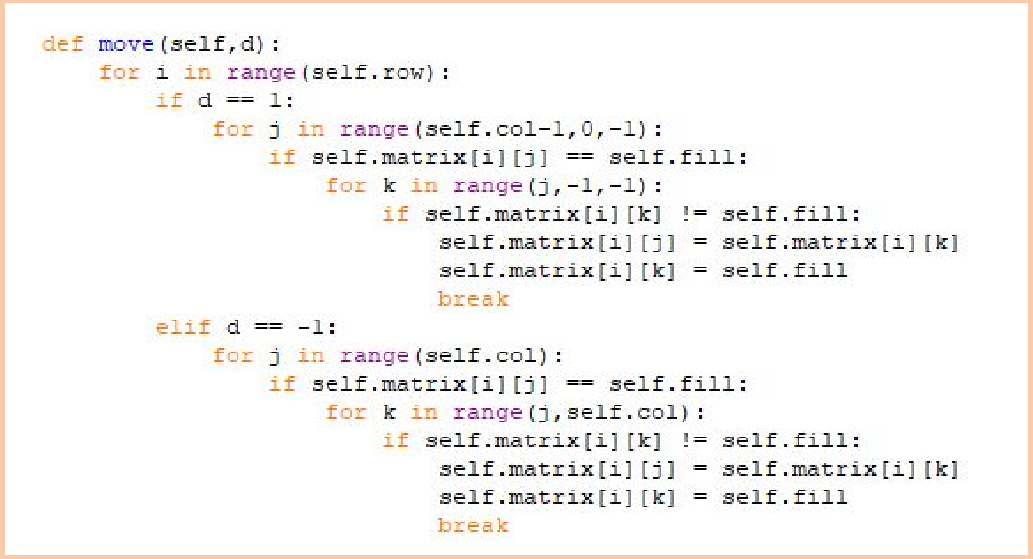
3) Ability to merge tiles

4) Keeping high values on edges

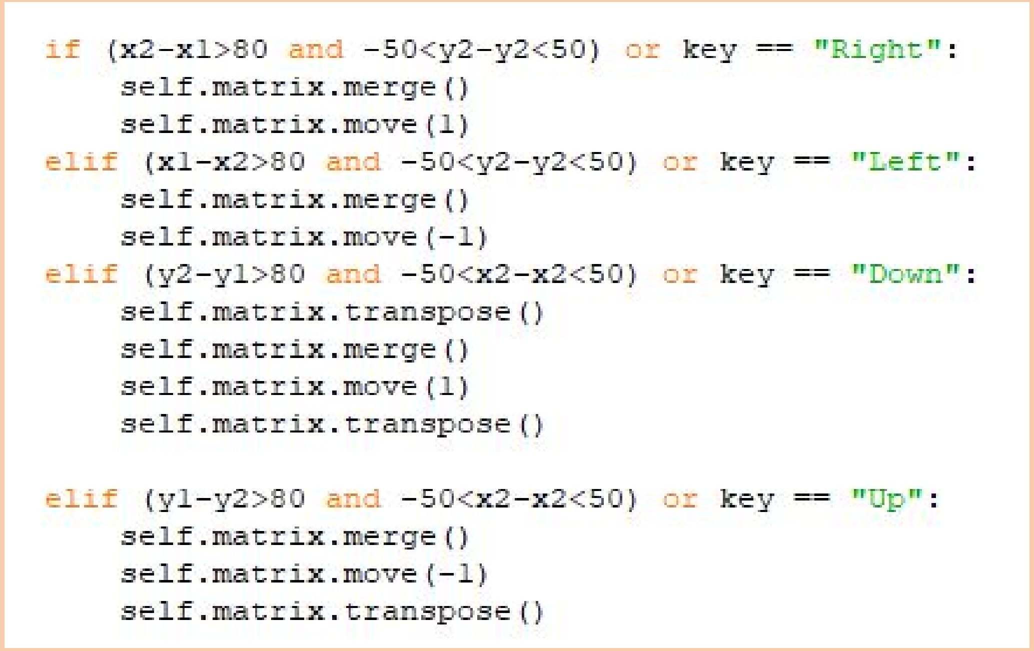
OUR 2048 GAME

Moves





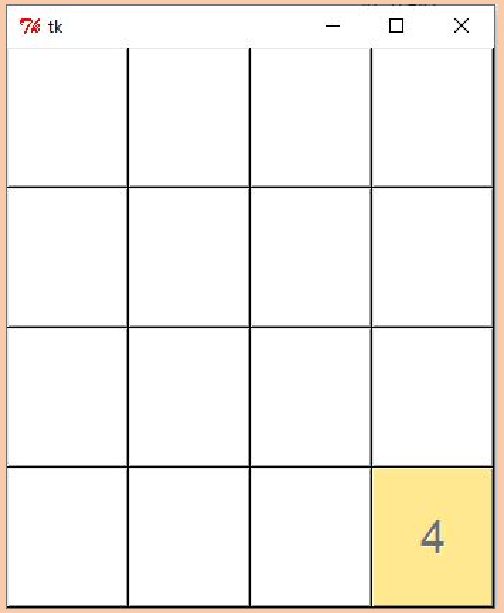
```
def move(self,d):
    for i in range(self.row):
        if d == 1:
            for j in range(self.col-1,0,-1):
                if self.matrix[i][j] == self.fill:
                    for k in range(j,-1,-1):
                        if self.matrix[i][k] != self.fill:
                            self.matrix[i][j] = self.matrix[i][k]
                            self.matrix[i][k] = self.fill
                            break
        elif d == -1:
            for j in range(self.col):
                if self.matrix[i][j] == self.fill:
                    for k in range(j,self.col):
                        if self.matrix[i][k] != self.fill:
                            self.matrix[i][j] = self.matrix[i][k]
                            self.matrix[i][k] = self.fill
                            break
```

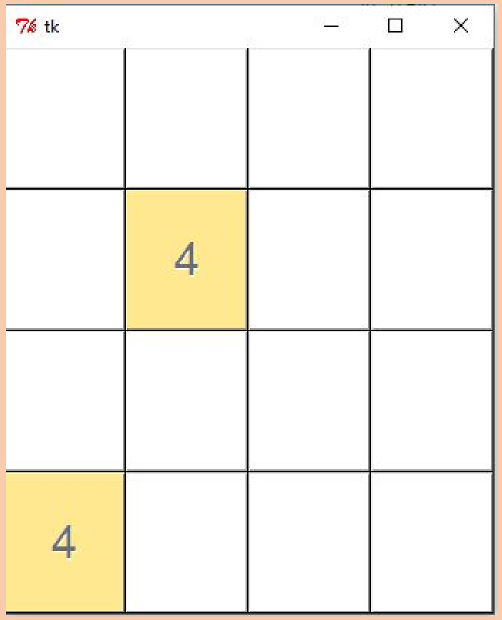


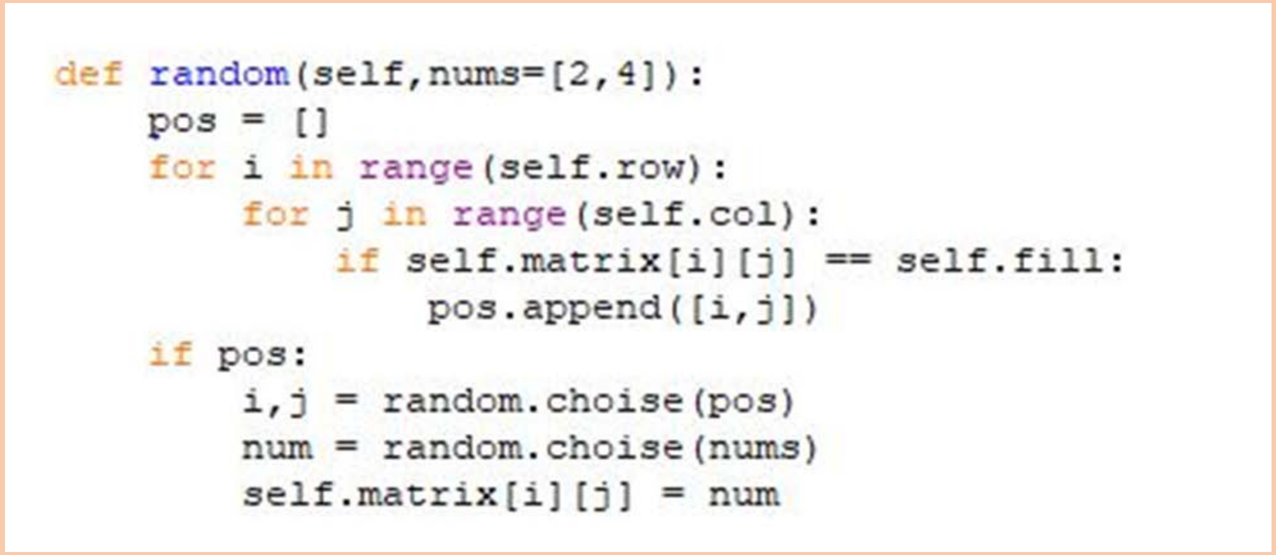
```
if (x2-x1>80 and -50<y2-y2<50) or key == "Right":
    self.matrix.merge()
    self.matrix.move(1)
elif (x1-x2>80 and -50<y2-y2<50) or key == "Left":
    self.matrix.merge()
    self.matrix.move(-1)
elif (y2-y1>80 and -50<x2-x2<50) or key == "Down":
    self.matrix.transpose()
    self.matrix.merge()
    self.matrix.move(1)
    self.matrix.transpose()

elif (y1-y2>80 and -50<x2-x2<50) or key == "Up":
    self.matrix.merge()
    self.matrix.move(-1)
    self.matrix.transpose()
```

Random number

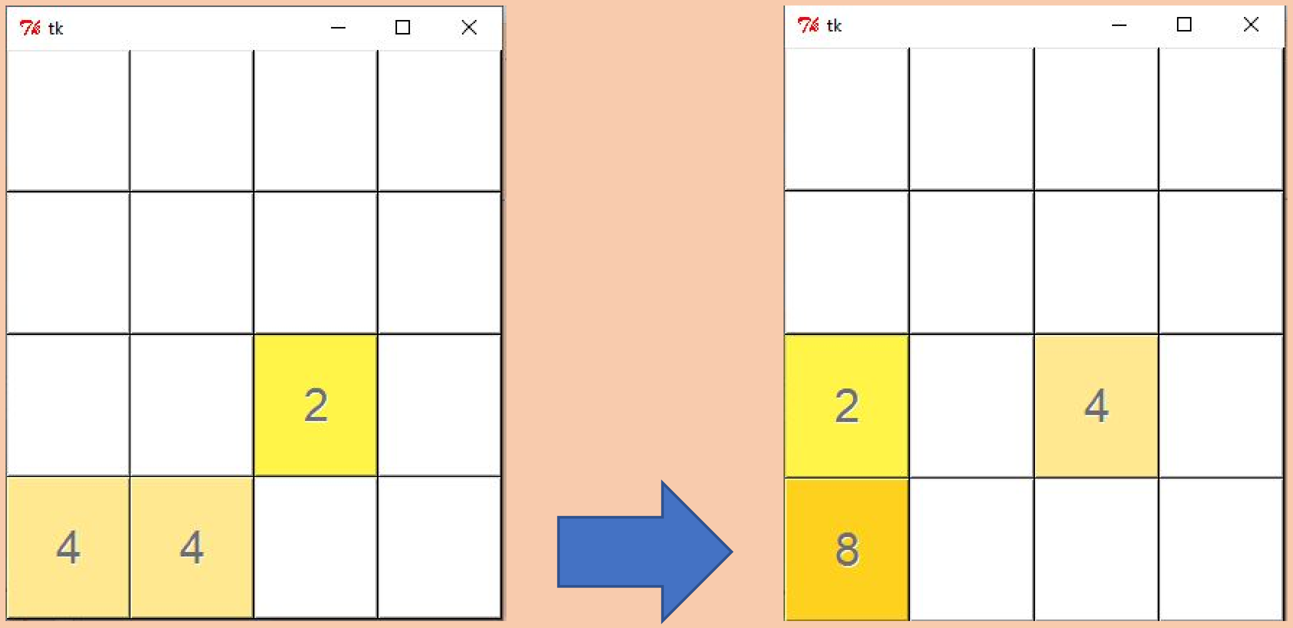






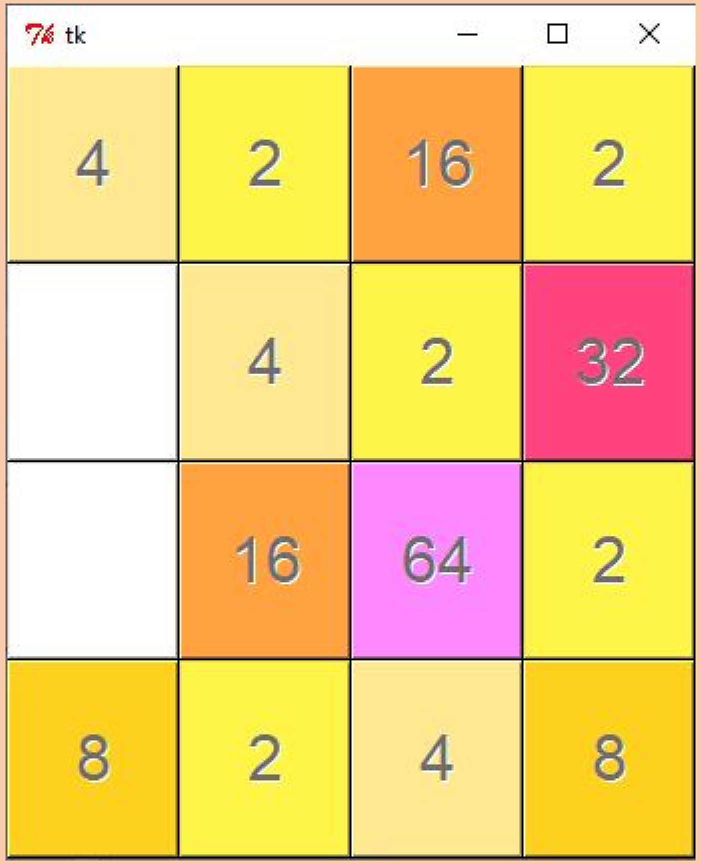
```
def random(self,nums=[2,4]):
    pos = []
    for i in range(self.row):
        for j in range(self.col):
            if self.matrix[i][j] == self.fill:
                pos.append([i,j])
    if pos:
        i,j = random.choise(pos)
        num = random.choise(nums)
        self.matrix[i][j] = num
```

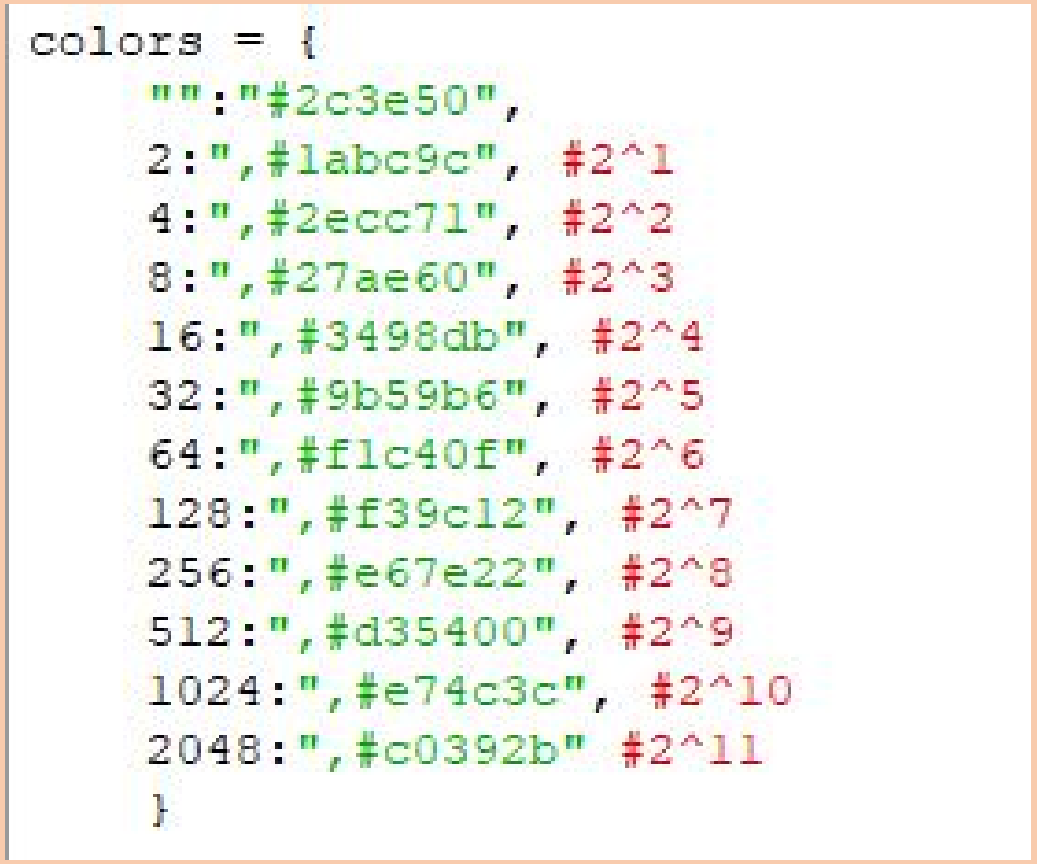
Merge



```
def merge(self):
    for i in range(self.row):
        for j in range(self.col-1,0,-1):
            if self.matrix[i][j] == self.matrix[i][j-1] and self.matrix[i][j] != self.fill:
                self.matrix[i][j] = 2*self.matrix[i][j]
                self.matrix[i][j-1] = self.fill
```

Colors





```
colors = {
     "":"#2c3e50",
     2:",#1abc9c", #2^1
     4:",#2ecc71", #2^2
     8:",#27ae60", #2^3
     16:",#3498db", #2^4
     32:",#9b59b6", #2^5
     64:",#f1c40f", #2^6
     128:",#f39c12", #2^7
     256:",#e67e22", #2^8
     512:",#d35400", #2^9
     1024:",#e74c3c", #2^10
     2048:",#c0392b" #2^11
     }
```

How we can improve our code?

- **Undo:** We can add undo button. Works with undo button stack logic.(for the last 5 moves)
- **Score List:** The users can log in online and play online. Scores can be saved. Score list can be created with online game scores.
- **Matrix size:** Create code of a button to choose matrix size of game before start playing.

References

- https://github.com/nayanraj210401/2048game/blob/master/2048.py
- https://github.com/silverstar194/2048-ai-monte-carlo
- https://towardsdatascience.com/2048-solving-2048-with-monte-carlo-tree-search-ai-2dbe76894bab
- https://towardsdatascience.com/2048-solving-2048-with-monte-carlo-tree-search-ai-2dbe76894bab?#86b0
- https://flatuicolors.com/palette/defo
- https://medium.com/@bartoszzadrony/beginners-guide-to-ai-and-writing-your-own-bot-for-the-2048-game-4b8083faaf53