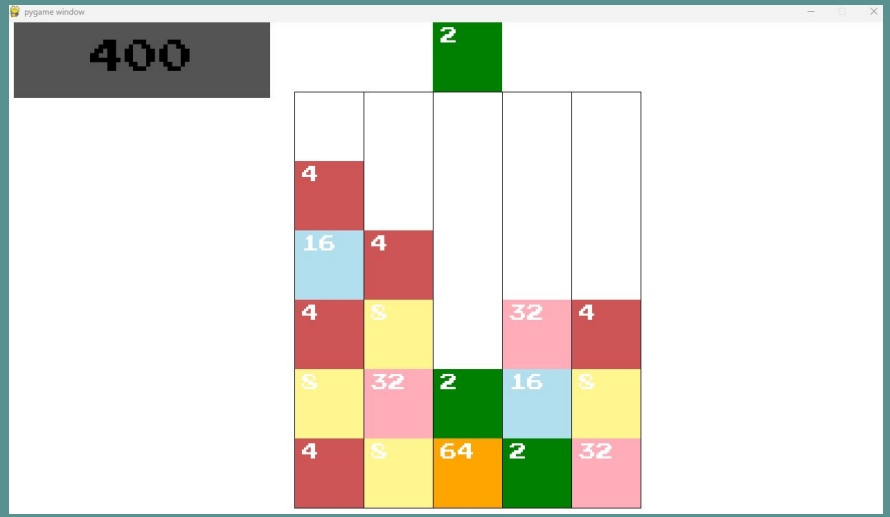
2048

Game by Daniil Ognev and Soumik Barua

Game Genre and Instructions



- 2048 is a single-player sliding tile puzzle video game
- The objective of the game is to match the tiles of the same number and color, that's how the player gains scores
- A player simply presses 1, 2, 3, 4, or 5 on their keyboard to make the tiles fall on the desired position
- The game ends when all the columns are filled and no more valid moves are possible

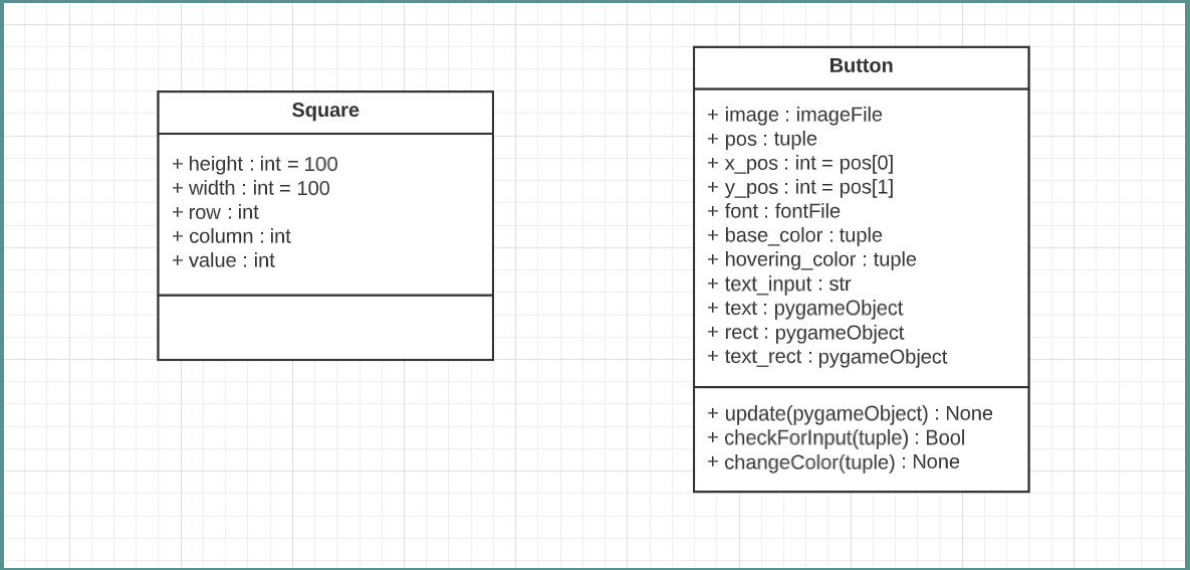
The AI Agent

```
def god_of_2048(board, current_square, score, intensity, depth = 1):
    possible_scores = []
    if intensity >= len(numbers):
        length = len(numbers)
    else:
        length = intensity + 5
    for column in range(len(board[0])):
        if not is_over(board, column):
            board_copy = [[i for i in row] for row in board]
            possible_square = current_square
            for row in range(len(board_copy)):
                if row == len(board_copy)-1 or board_copy[row+1][column] != 0:
                    board_copy[row][column] = current_square.value
                    possible_square = Square(row, column, current_square.value)
                    break
            possible_score = check_check_and_merge(board_copy, possible_square)
            height = 0
            for j in range(len(board_copy)):
                if board[j][column] != 0:
                    height += 1
            if depth > 1:
                depth -= 1
                for number in list(scores.keys())[:5+intensity]:
                    next_square = Square(None, None, number)
                    deeper_possible_scores = god_of_2048(board_copy, next_square, possible_score, intensity, depth)
                    best_score = deeper_possible_scores.index(max(deeper_possible_scores))
                    possible_score += best_score/ (5+intensity)

            possible_scores.append(possible_score - 4*height)
        else:
            possible_scores.append(-1 * float('inf'))
    return possible_scores
```

- The god_of_2048 function is our AI Agent which takes in different depths and calculates the possible scores
- Then it makes the move with the highest score among the 5 columns
- This way the AI achieves the highest score
- The higher the depth the higher the score

UML Class Diagram



- Square Class: It contains the size of a square tile instance, its row and column number which will be indexed into the board list and its value such as 2, 4, 8, ...
- Button Class: It contains an image, a tuple of a button's position, a font type, 2 tuples of base and hovering colors, a text for the button's name, a text, rectangle and text + rectangle pygame object

Game State

```
board = [[0 for i in range(5)] for j in range(6)]
```

- There are 5*6*5*6 = 900 possible game states due to the board consisting 6 rows and 5 columns
- There are 5 actions: 1, 2, 3, 4, and 5 which represent the 5 different columns

Acknowledgments

- Menu screen background image by The Verge: https://cdn.vox-cdn.com/thumbor/WR9hE8wvdM4hfHysXitls9_bCZI=/0x0:1192x795/1400x1400/filters:focal(596x398:597x399)/cdn.vox-cdn.com/uploads/chorus_asset/file/22312759/rickroll_4k.jpg
- Background Music - 2700s Sea Shanties by Lil Miner
- Click Sound Effect - Click Sound Effect (HD) by Joe TheSoundEffect Maker: https://www.youtube.com/watch?v=vzfqwCu2hi4&ab_channel=JoeTheSoundeffectMaker
- Tap Sound Effect - Tap Sound by Best Music BS: https://www.youtube.com/watch?v=ihLkK2pfeLI&ab_channel=BestMusicBS
- Square Merge Sound Effect - ka-ching sound effect by Alexnatoe: https://www.youtube.com/watch?v=vBYJdZlpBT4&ab_channel=Alexnatoe
- Font Style - West_england Font: https://www.fontspace.com/west-england-font-f9616