```
"""2048 Game Clone"""
import tkinter as tk
import random
import colours as c


class Game(tk.Frame):
    """Tkinter Game Class"""
    def __init__(self) -> None:
        tk.Frame.__init__(self)
        self.grid()
        self.master.title('2048')

        self.main_grid = tk.Frame(
            self, bg=c.GRID_COLOR, bd=3, width=600, height=600)
        self.main_grid.grid(pady=(100, 0))
        self.setup()
        self.start_game()

        self.master.bind("<Left>", self.left)
        self.master.bind("<Right>", self.right)
        self.master.bind("<Up>", self.up)
        self.master.bind("<Down>", self.down)

        self.matrix = None
        self.cells = None
        self.score_label = None
        self.score = None

        self.mainloop()

    def setup(self) -> None:
        """Set up the grid and score"""
        # make grid
        self.cells = []
        for i in range(4):
            row = []
            for j in range(4):
                cell_frame = tk.Frame(
                    self.main_grid,
                    bg=c.EMPTY_CELL_COLOR,
                    width=100,
                    height=100)
                cell_frame.grid(row=i, column=j, padx=5, pady=5)
                cell_number = tk.Label(self.main_grid, bg=c.EMPTY_CELL_COLOR)
                cell_number.grid(row=i, column=j)
                cell_data = {"frame": cell_frame, "number": cell_number}
                row.append(cell_data)
            self.cells.append(row)

        # make score header
        score_frame = tk.Frame(self)
        score_frame.place(relx=0.5, y=45, anchor="center")
        tk.Label(
            score_frame,
```

```
            text="Score",
            font=c.SCORE_LABEL_FONT).grid(
            row=0)
        self.score_label = tk.Label(score_frame, text="0", font=c.SCORE_FONT)
        self.score_label.grid(row=1)

    def start_game(self) -> None:
        """Start the game"""
        # create matrix of zeroes
        self.matrix = [[0] * 4 for _ in range(4)]

        # fill 2 random cells with 2s
        row = random.randint(0, 3)
        col = random.randint(0, 3)
        self.matrix[row][col] = 2
        self.cells[row][col]["frame"].configure(bg=c.CELL_COLORS[2])
        self.cells[row][col]["number"].configure(
            bg=c.CELL_COLORS[2],
            fg=c.CELL_NUMBER_COLORS[2],
            font=c.CELL_NUMBER_FONTS[2],
            text="2")
        while self.matrix[row][col] != 0:
            row = random.randint(0, 3)
            col = random.randint(0, 3)
        self.matrix[row][col] = 2
        self.cells[row][col]["frame"].configure(bg=c.CELL_COLORS[2])
        self.cells[row][col]["number"].configure(
            bg=c.CELL_COLORS[2],
            fg=c.CELL_NUMBER_COLORS[2],
            font=c.CELL_NUMBER_FONTS[2],
            text="2")

        self.score = 0

    # Matrix Manipulation Functions

    def stack(self) -> None:
        """Stack numbers"""
        new_matrix = [[0] * 4 for _ in range(4)]
        for i in range(4):
            fill_position = 0
            for j in range(4):
                if self.matrix[i][j] != 0:
                    new_matrix[i][fill_position] = self.matrix[i][j]
                    fill_position += 1
        self.matrix = new_matrix

    def combine(self) -> None:
        """Combine numbers"""
        for i in range(4):
            for j in range(3):
                if self.matrix[i][j] != 0 and self.matrix[i][j] ==
self.matrix[i][j + 1]:
                    self.matrix[i][j] *= 2
                    self.matrix[i][j + 1] = 0
```

```
                self.score += self.matrix[i][j]

    def reverse(self) -> None:
        """Reverse numbers"""
        new_matrix = []
        for i in range(4):
            new_matrix.append([])
            for j in range(4):
                new_matrix[i].append(self.matrix[i][3 - j])
        self.matrix = new_matrix

    def transpose(self) -> None:
        """Transpose numbers"""
        new_matrix = [[0] * 4 for _ in range(4)]
        for i in range(4):
            for j in range(4):
                new_matrix[i][j] = self.matrix[j][i]
        self.matrix = new_matrix

    # Add a new 2 or 4 tile randomly to an empty cell

    def add_new_tile(self) -> None:
        """Add new tile"""
        row = random.randint(0, 3)
        col = random.randint(0, 3)
        while self.matrix[row][col] != 0:
            row = random.randint(0, 3)
            col = random.randint(0, 3)
        self.matrix[row][col] = random.choice([2, 4])

    # Update the GUI to match the matrix

    def update(self) -> None:
        """Update Frame"""
        for i in range(4):
            for j in range(4):
                cell_value = self.matrix[i][j]
                if cell_value == 0:
                    self.cells[i][j]["frame"].configure(bg=c.EMPTY_CELL_COLOR)
                    self.cells[i][j]["number"].configure(
                        bg=c.EMPTY_CELL_COLOR, text="")
                else:
                    self.cells[i][j]["frame"].configure(
                        bg=c.CELL_COLORS[cell_value])
                    self.cells[i][j]["number"].configure(
                        bg=c.CELL_COLORS[cell_value],
                        fg=c.CELL_NUMBER_COLORS[cell_value],
                        font=c.CELL_NUMBER_FONTS[cell_value],
                        text=str(cell_value))
        self.score_label.configure(text=self.score)
        self.update_idletasks()

    # Arrow-Press Functions

    def left(self) -> None:
```

```
    """Move left"""
    self.stack()
    self.combine()
    self.stack()
    self.add_new_tile()
    self.update()
    self.game_over()

def right(self) -> None:
    """Move right"""
    self.reverse()
    self.stack()
    self.combine()
    self.stack()
    self.reverse()
    self.add_new_tile()
    self.update()
    self.game_over()

def up(self) -> None:
    """Move up"""
    self.transpose()
    self.stack()
    self.combine()
    self.stack()
    self.transpose()
    self.add_new_tile()
    self.update()
    self.game_over()

def down(self) -> None:
    """Move down"""
    self.transpose()
    self.reverse()
    self.stack()
    self.combine()
    self.stack()
    self.reverse()
    self.transpose()
    self.add_new_tile()
    self.update()
    self.game_over()

# Check if any moves are possible

def horizontal_move_exists(self) -> bool:
    """Check if horizontal move is possible"""
    for i in range(4):
        for j in range(3):
            if self.matrix[i][j] == self.matrix[i][j + 1]:
                return True
    return False

def vertical_move_exists(self) -> bool:
    """Check if vertical move is possible"""
```

```
        for i in range(3):
            for j in range(4):
                if self.matrix[i][j] == self.matrix[i + 1][j]:
                    return True
        return False

    # Check if the Game is Over (Win/Lose)

    def game_over(self) -> None:
        """Check if no more moves are possible"""
        if any(2048 in row for row in self.matrix):
            game_over_frame = tk.Frame(self.main_grid, borderwidth=2)
            game_over_frame.place(relx=0.5, rely=0.5, anchor="center")
            tk.Label(
                game_over_frame,
                text="You win!",
                bg=c.WINNER_BG,
                fg=c.GAME_OVER_FONT_COLOR,
                font=c.GAME_OVER_FONT).pack()
        elif (not any(0 in row for row in self.matrix)
              and not self.horizontal_move_exists() and
              not self.vertical_move_exists()):
            game_over_frame = tk.Frame(self.main_grid, borderwidth=2)
            game_over_frame.place(relx=0.5, rely=0.5, anchor="center")
            tk.Label(
                game_over_frame,
                text="Game over!",
                bg=c.LOSER_BG,
                fg=c.GAME_OVER_FONT_COLOR,
                font=c.GAME_OVER_FONT).pack()


if __name__ == "__main__":
    Game()
```