SGP TECHNICAL PAPER TETRIS GAME

Desai Sonu

Charusat university of science and technology

Cspit/IT department

20it025@charusat.edu.in

Yash Dalia

Charusat university of science and technology

Cspit/IT department

20it020@charusat.edu.in

• Abstract:

The purpose of the "TETRIS GAME" is for learn and play something new. Tetris can ease us through periods of anxiety by getting us to a blissfully engrossed mental state that psychologists call "flow."

Language- JavaScript with html

Operating system- Windows

• INTRODUCTION:

- Tetris is a puzzle game in which geometric shapes called "tetrominoes" fall down onto a playing field, and the player has to arrange them to form gapless lines. Tetris, video game created by Russian designer Alexey Pajitnov in 1985 that allows players to rotate falling blocks strategically to clear levels. ... The goal of the game is to prevent the blocks from stacking up to the top of the screen for as long as possible. combining the Greek numeral "tetra" (meaning four) and tennis, his favorite sport.

- It's not only enjoyable to play, but it also brings back good memories. For many, it was the first video game they played, and that's how their love for gaming started. Even after moving on to more advanced and challenging productions, Tetris will always be the one to make you feel nostalgic.

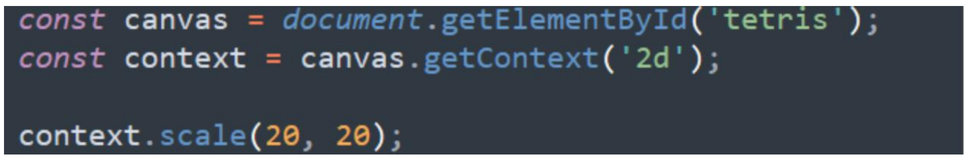
• Methodology:

- **Important part of project management is project planning.**
- **Initially, the scope of the project is defined and the appropriate methods for the project completing are determined.**
- **We have used JavaScript for our game development. Technology as the sub factors hardware, software and connectivity, the hardware needed should be identified before the introduction of the system.**
- **The game design is impeccable and clearly uses the technology of its time perfectly. So simple is its gameplay that it seems as if you could simulate it perfectly in a physical prototype, but this is much more difficult with a little experimentation. Without any context Tetris still makes sense.**

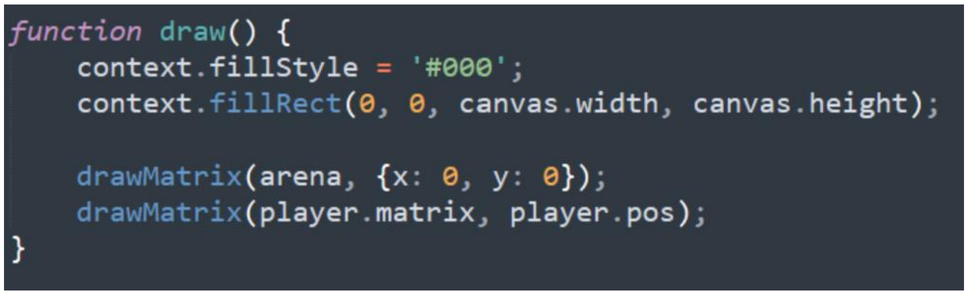
• <u>Explanation:</u>

1] I decided to make a Tetris game using JavaScript.

- **first of all, I faced a problem while deciding the playing window.**
- **to solve this, I performed a few lines of code which is shown below**

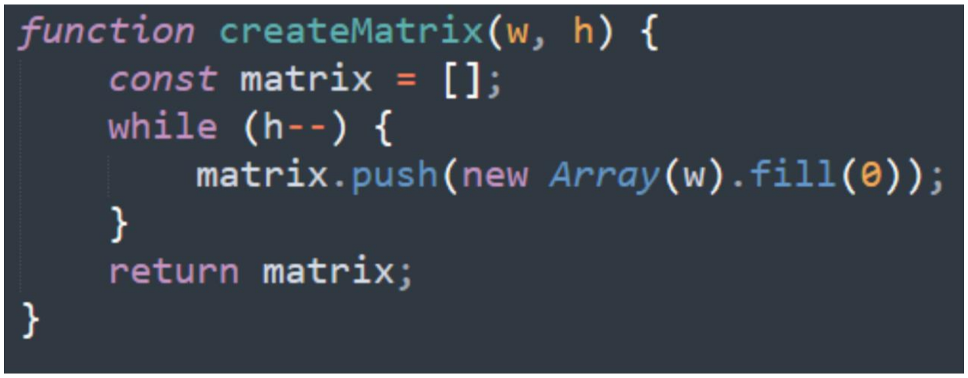


```
const canvas = document.getElementById('tetris');
const context = canvas.getContext('2d');

context.scale(20, 20);
```



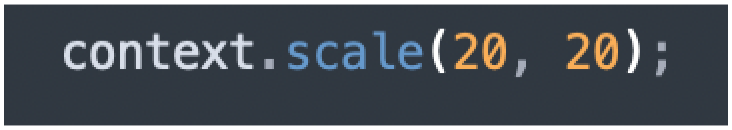
```
function draw() {
    context.fillStyle = '#000';
    context.fillRect(0, 0, canvas.width, canvas.height);

    drawMatrix(arena, {x: 0, y: 0});
    drawMatrix(player.matrix, player.pos);
}
```

2] in this game, the main elements are different shaped boxes. to make these boxes using JavaScript I used matrix instead of fetching pictures.



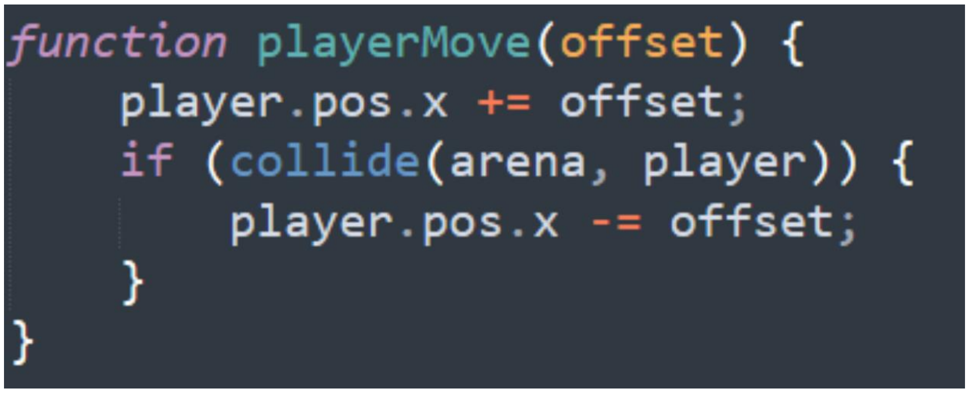
```
function createMatrix(w, h) {
    const matrix = [];
    while (h--) {
        matrix.push(new Array(w).fill(0));
    }
    return matrix;
}
```

3] to scale the Tetris box:

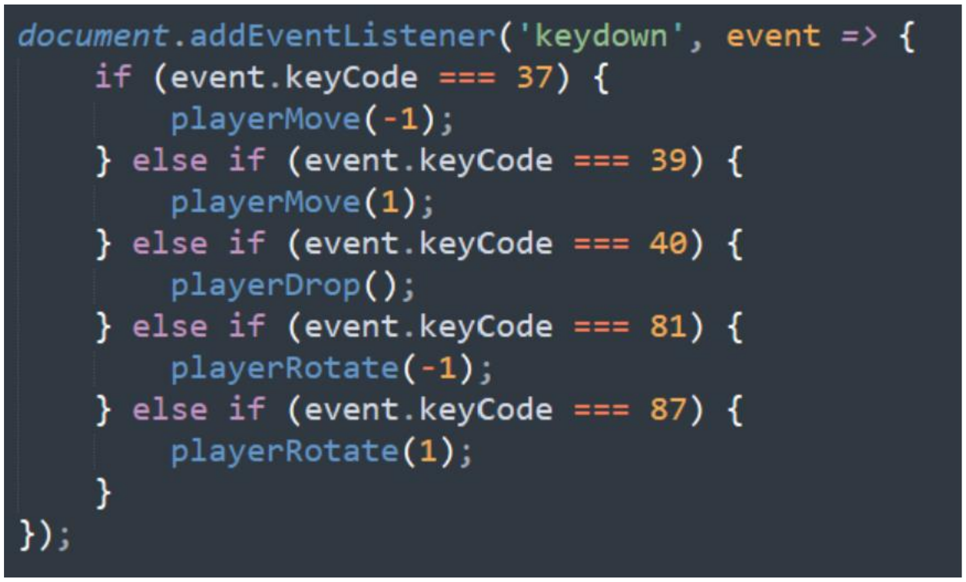


```
context.scale(20, 20);
```

4] to make the box move sideways:

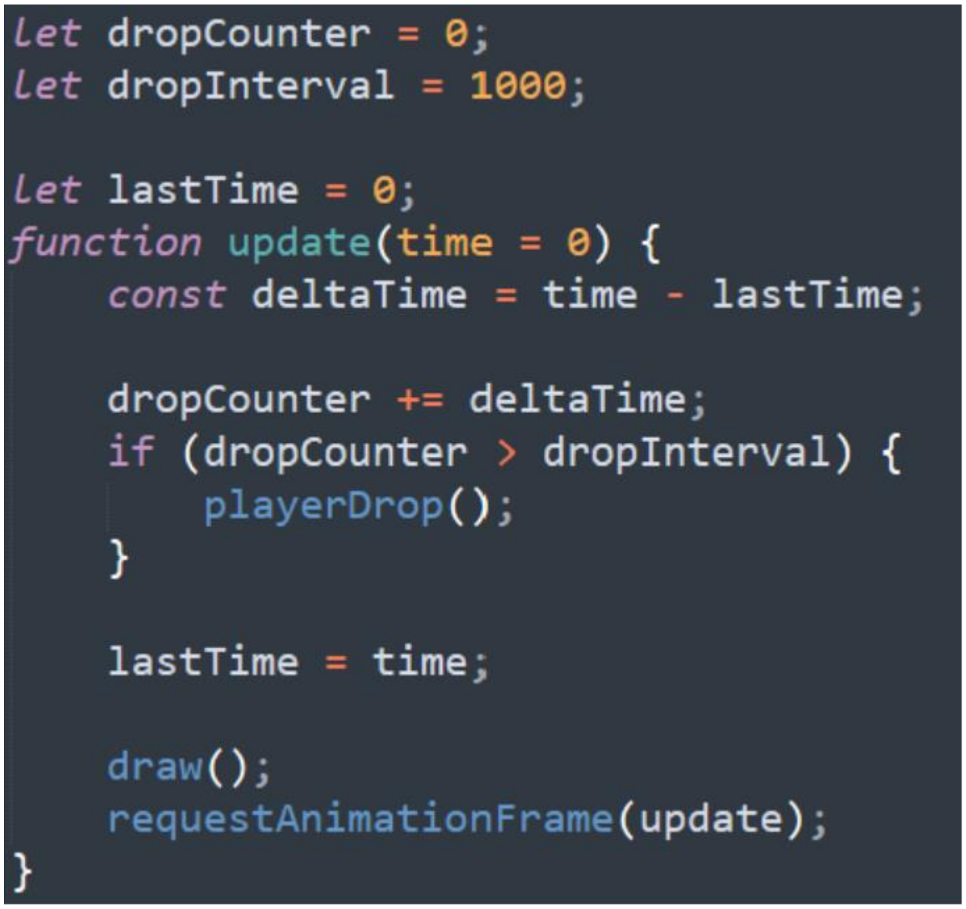


```
function playerMove(offset) {
    player.pos.x += offset;
    if (collide(arena, player)) {
        player.pos.x -= offset;
    }
}
```



```
document.addEventListener('keydown', event => {
    if (event.keyCode === 37) {
        playerMove(-1);
    } else if (event.keyCode === 39) {
        playerMove(1);
    } else if (event.keyCode === 40) {
        playerDrop();
    } else if (event.keyCode === 81) {
        playerRotate(-1);
    } else if (event.keyCode === 87) {
        playerRotate(1);
    }
});
```

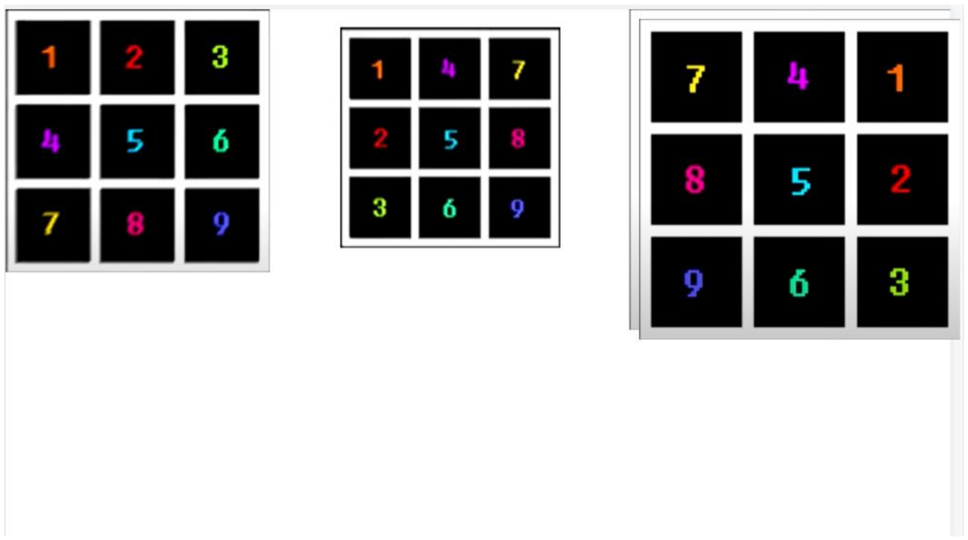
5] to make the box drop:

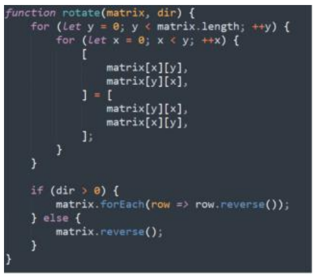


```
let dropCounter = 0;
let dropInterval = 1000;

let lastTime = 0;
function update(time = 0) {
    const deltaTime = time - lastTime;

    dropCounter += deltaTime;
    if (dropCounter > dropInterval) {
        playerDrop();
    }

    lastTime = time;

    draw();
    requestAnimationFrame(update);
}
```

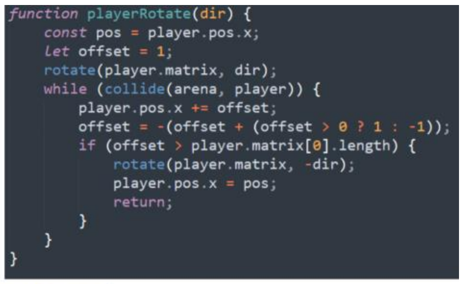
- **in this section, we had to reset the drop counter because we don't want any drops after pressing the down key.**
- **here are the keycodes I used for different inputs**
- **right arrow key: 39**
- **left arrow key: 37**
- **down arrow key: 40**

6] to rotate the Tetris we have to transpose all the matrix rows into columns. then reverse each row.

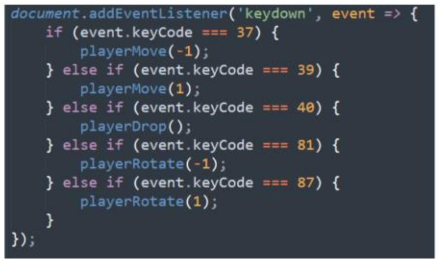




```
function rotate(matrix, dir) {
    for (let y = 0; y < matrix.length; ++y) {
        for (let x = 0; x < y; ++x) {
            [
                matrix[x][y],
                matrix[y][x],
            ] = [
                matrix[y][x],
                matrix[x][y],
            ];
        }
    }

    if (dir > 0) {
        matrix.forEach(row => row.reverse());
    } else {
        matrix.reverse();
    }
}
```



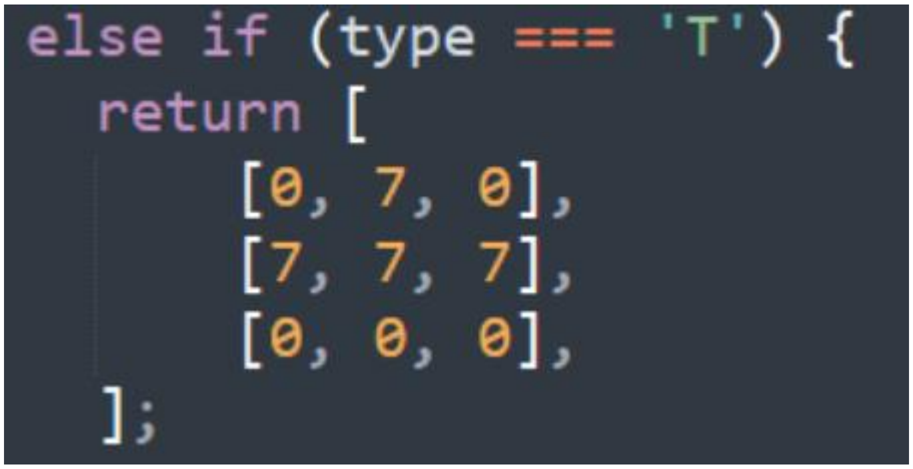
```
function playerRotate(dir) {
    const pos = player.pos.x;
    let offset = 1;
    rotate(player.matrix, dir);
    while (collide(arena, player)) {
        player.pos.x += offset;
        offset = -(offset + (offset > 0 ? 1 : -1));
        if (offset > player.matrix[0].length) {
            rotate(player.matrix, -dir);
            player.pos.x = pos;
            return;
        }
    }
}
```



```
document.addEventListener('keydown', event => {
    if (event.keyCode === 37) {
        playerMove(-1);
    } else if (event.keyCode === 39) {
        playerMove(1);
    } else if (event.keyCode === 40) {
        playerDrop();
    } else if (event.keyCode === 81) {
        playerRotate(-1);
    } else if (event.keyCode === 87) {
        playerRotate(1);
    }
});
```

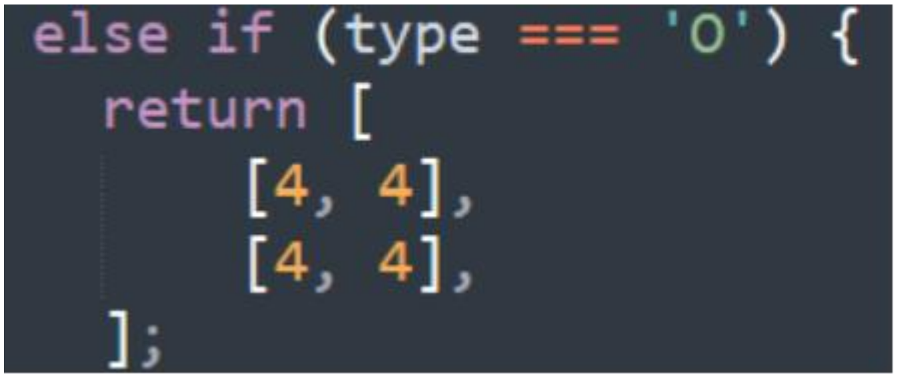
- **keycodes of inputs**
- **Q: 81**
- **W: 87**

7] for the "T" shaped Tetris:



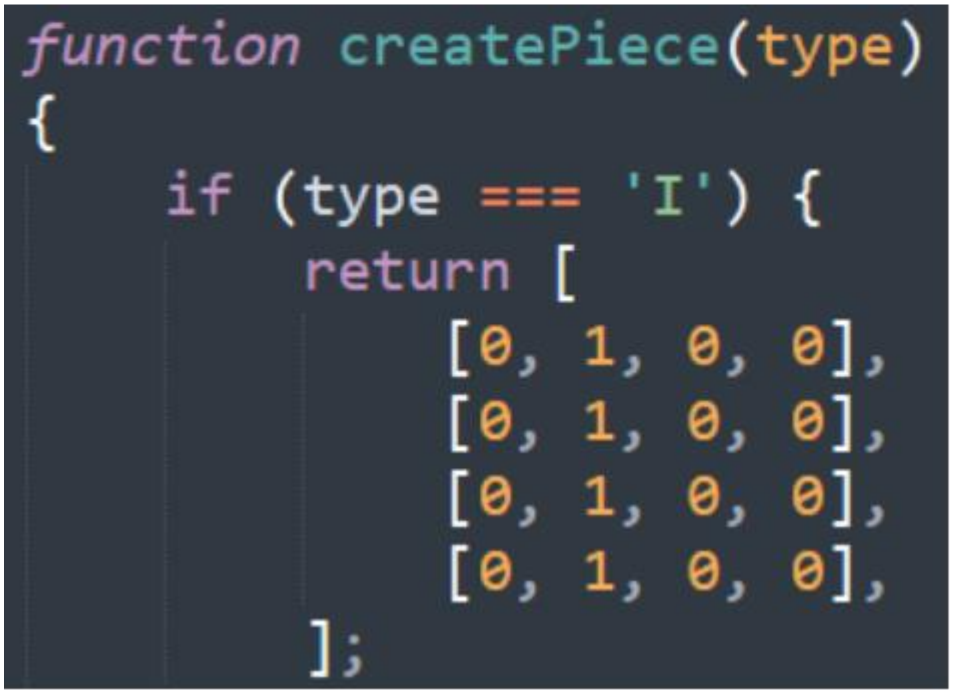
```
else if (type === 'T') {
  return [
      [0, 7, 0],
      [7, 7, 7],
      [0, 0, 0],
  ];
```

- **for the "o" shaped Tetris:**



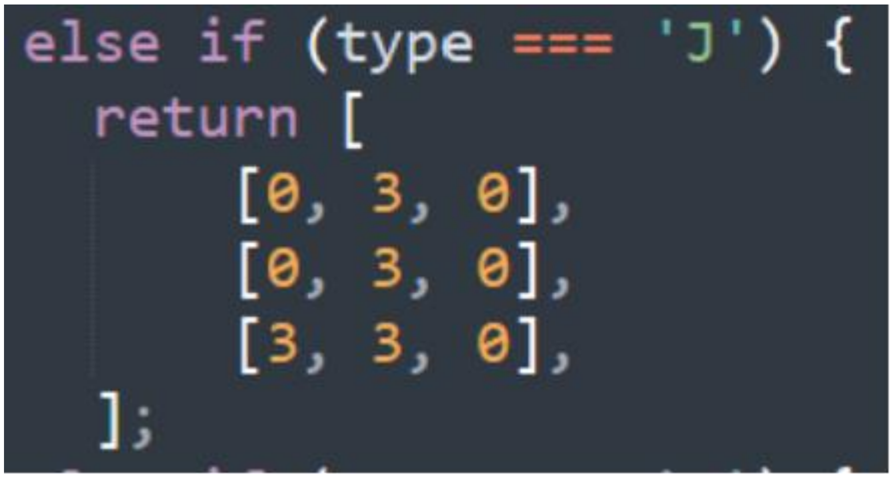
```
else if (type === 'O') {
  return [
      [4, 4],
      [4, 4],
  ];
```

- **for the "l" shaped Tetris:**



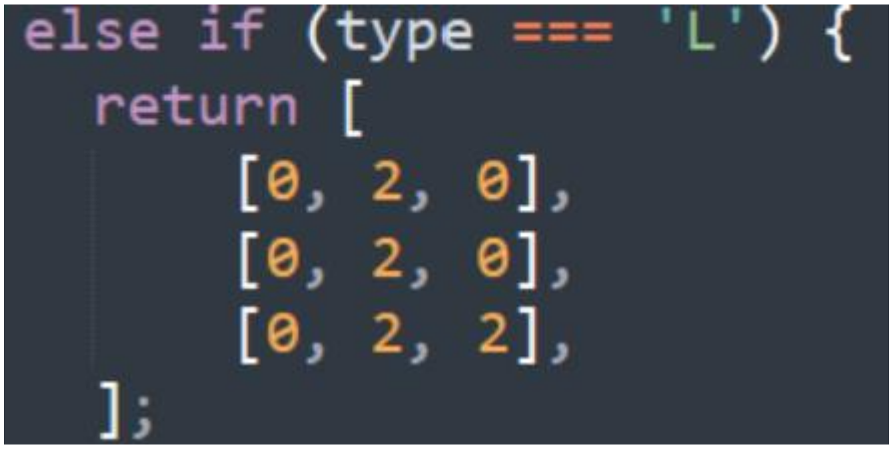
```
function createPiece(type)
{
    if (type === 'I') {
        return [
            [0, 1, 0, 0],
            [0, 1, 0, 0],
            [0, 1, 0, 0],
            [0, 1, 0, 0],
        ];
```

- **for the "j" shaped Tetris:**



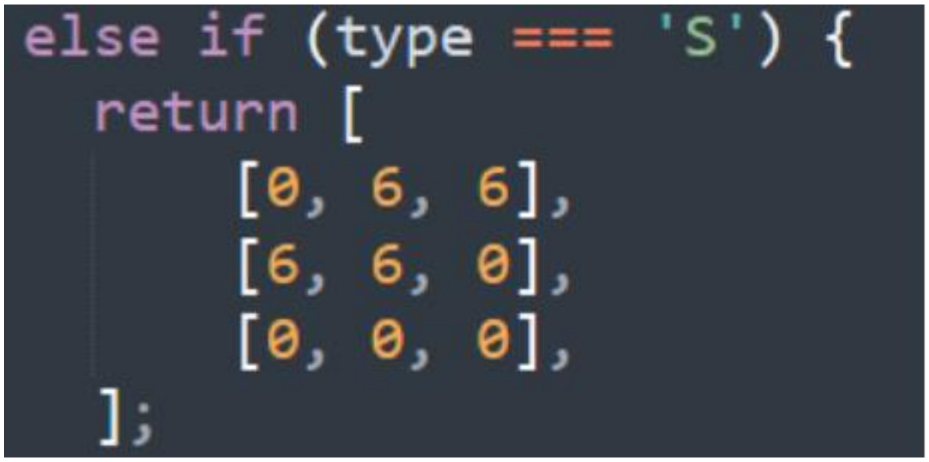
```
else if (type === 'J') {
  return [
      [0, 3, 0],
      [0, 3, 0],
      [3, 3, 0],
  ];
```

- **for the "|" shaped Tetris:**



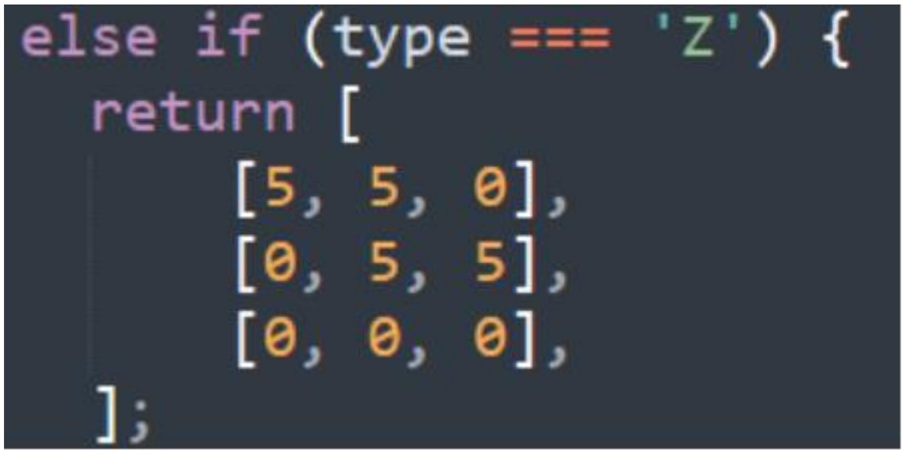
```
else if (type === 'L') {
  return [
      [0, 2, 0],
      [0, 2, 0],
      [0, 2, 2],
  ];
```

- **for the "s" shaped Tetris:**



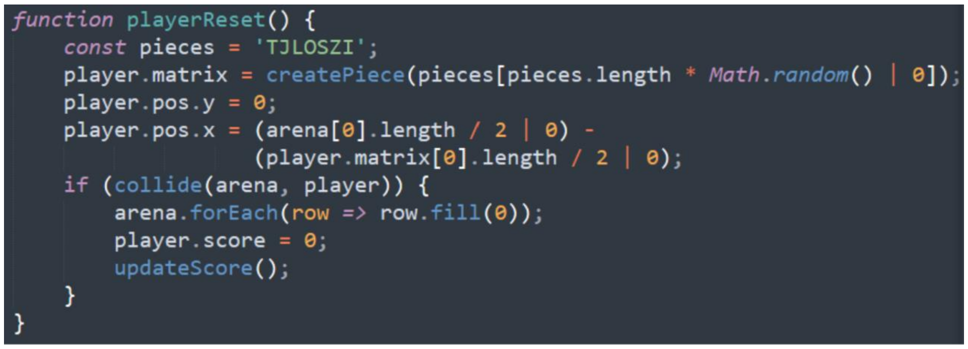
```
else if (type === 'S') {
  return [
      [0, 6, 6],
      [6, 6, 0],
      [0, 0, 0],
  ];
```

- **for the "z" shaped Tetris:**



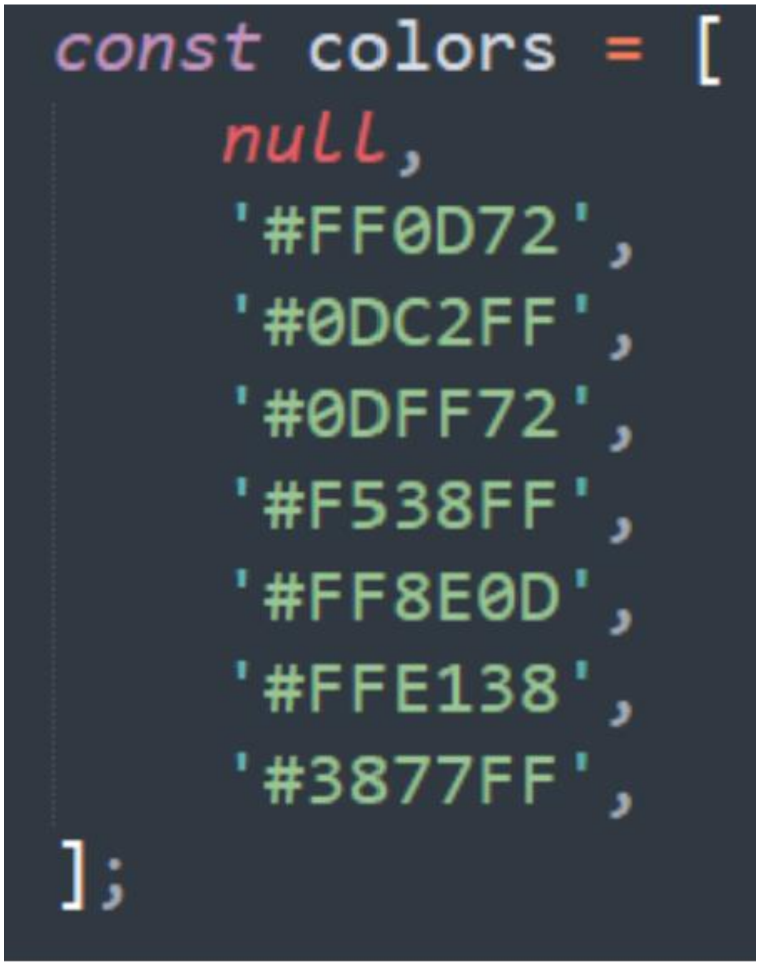
```
else if (type === 'Z') {
  return [
      [5, 5, 0],
      [0, 5, 5],
      [0, 0, 0],
  ];
```

8] to randomize the shapes I used the "math. random" function. I faced a problem when all the Tetris stack up and touched the top of the arena. to prevent that problem, we had to stop the game when the Tetris touch the top.



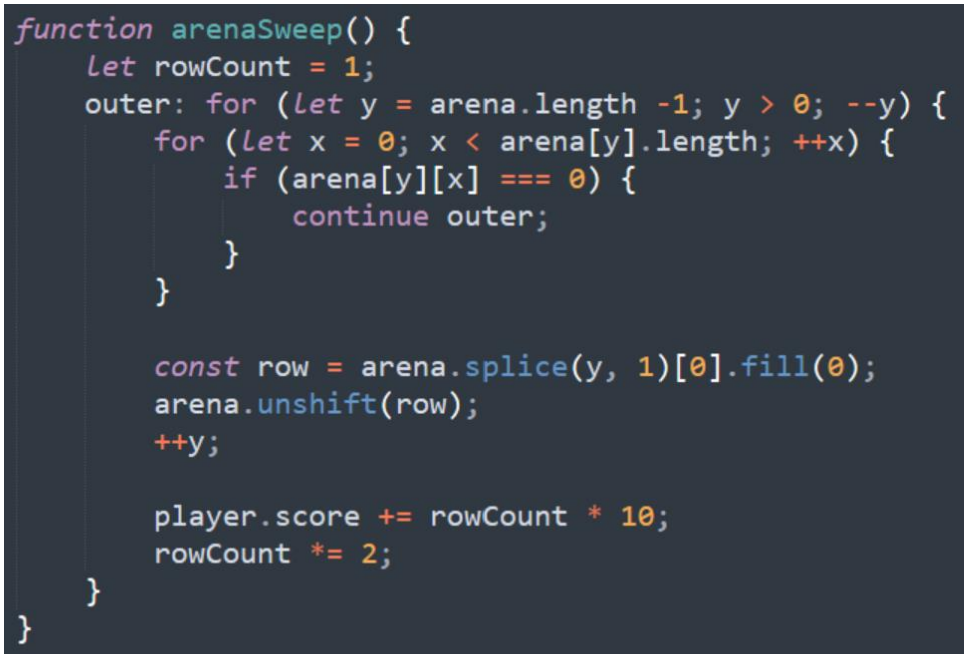
```
function playerReset() {
    const pieces = 'TJLOSZI';
    player.matrix = createPiece(pieces[pieces.length * Math.random() | 0]);
    player.pos.y = 0;
    player.pos.x = (arena[0].length / 2 | 0) -
                   (player.matrix[0].length / 2 | 0);
    if (collide(arena, player)) {
        arena.forEach(row => row.fill(0));
        player.score = 0;
        updateScore();
    }
}
```

9] to color the Tetris I created a new function:



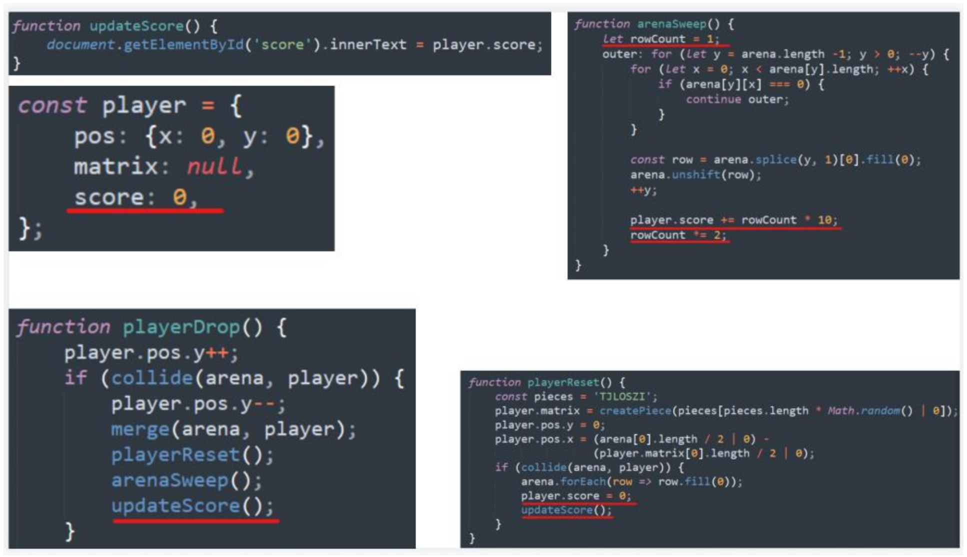
```
const colors = [
    null,
    '#FF0D72',
    '#0DC2FF',
    '#0DFF72',
    '#F538FF',
    '#FF8E0D',
    '#FFE138',
    '#3877FF',
];
```

10] when the whole column matches with the same color, we need to sweep the arena. for that, I created "arena sweep" function. now whenever the sweep function applies the score has to be added.



```
function arenaSweep() {
    let rowCount = 1;
    outer: for (let y = arena.length -1; y > 0; --y) {
        for (let x = 0; x < arena[y].length; ++x) {
            if (arena[y][x] === 0) {
                continue outer;
            }
        }

        const row = arena.splice(y, 1)[0].fill(0);
        arena.unshift(row);
        ++y;

        player.score += rowCount * 10;
        rowCount *= 2;
    }
}
```

11] now when the game resets we also need to reset the score. for that:



```
function updateScore() {
    document.getElementById('score').innerText = player.score;
}
```

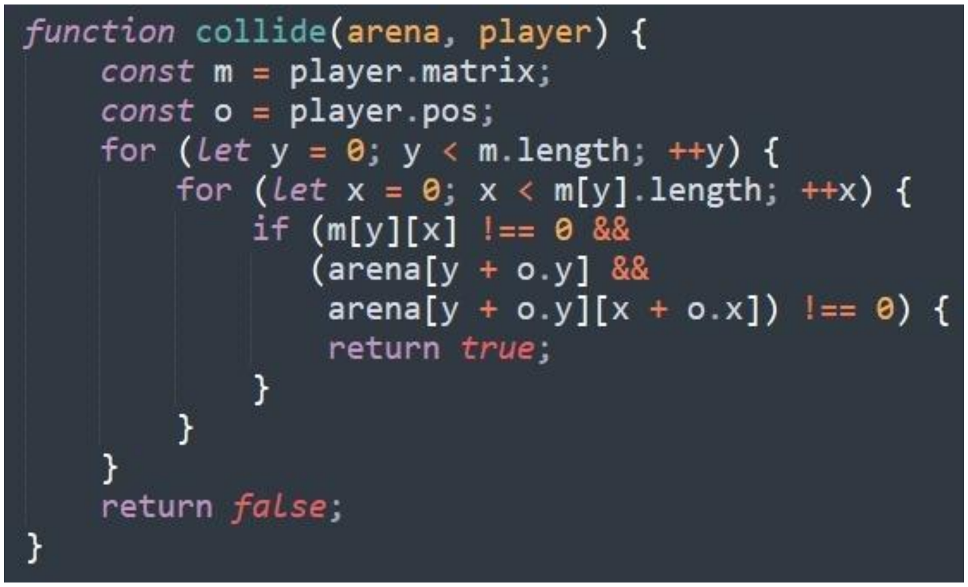
```
const player = {
    pos: {x: 0, y: 0},
    matrix: null,
    score: 0,
};
```

```
function arenaSweep() {
    let rowCount = 1;
    outer: for (let y = arena.length -1; y > 0; --y) {
        for (let x = 0; x < arena[y].length; ++x) {
            if (arena[y][x] === 0) {
                continue outer;
            }
        }

        const row = arena.splice(y, 1)[0].fill(0);
        arena.unshift(row);
        ++y;

        player.score += rowCount * 10;
        rowCount *= 2;
    }
}
```

```
function playerDrop() {
    player.pos.y++;
    if (collide(arena, player)) {
        player.pos.y--;
        merge(arena, player);
        playerReset();
        arenaSweep();
        updateScore();
    }
```

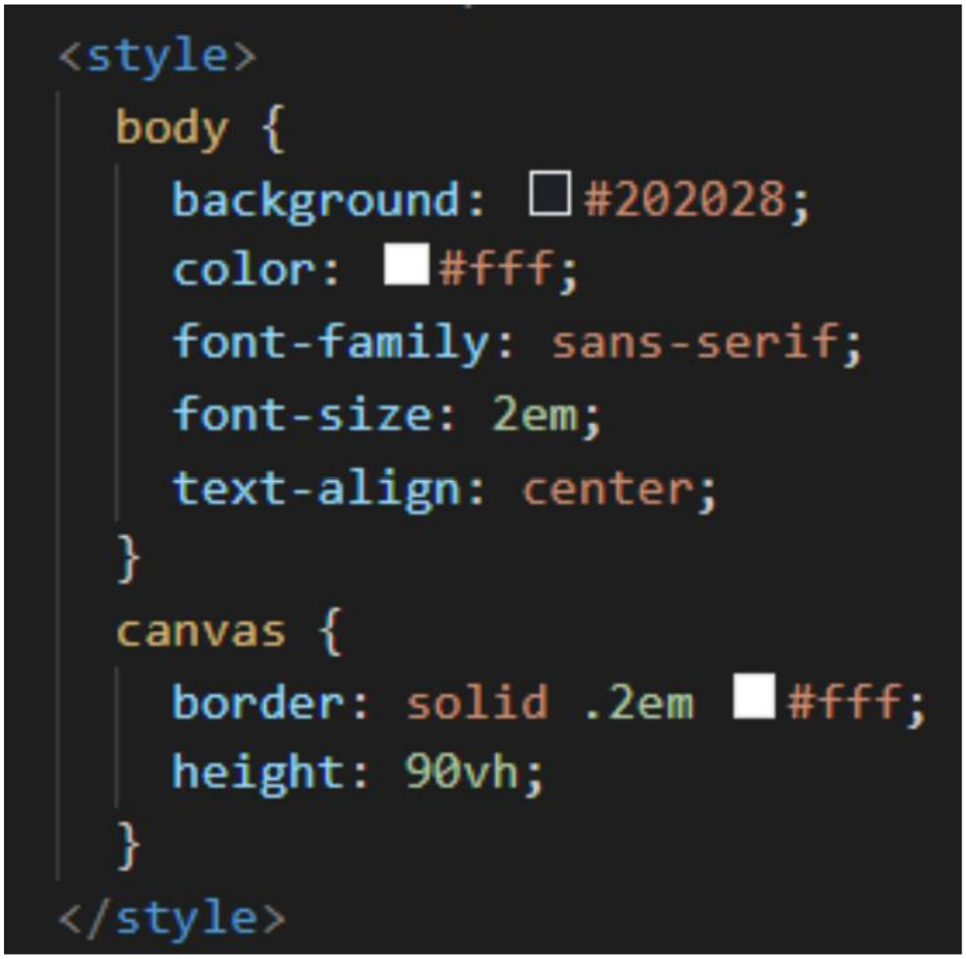
```
function playerReset() {
    const pieces = 'TJLOSZI';
    player.matrix = createPiece(pieces[pieces.length * Math.random() | 0]);
    player.pos.y = 0;
    player.pos.x = (arena[0].length / 2 | 0) -
                   (player.matrix[0].length / 2 | 0);
    if (collide(arena, player)) {
        arena.forEach(row => row.fill(0));
        player.score = 0;
        updateScore();
    }
}
```

12] to make sure the block won't go outside the canvas and stack up nicely I needed to create a "collide" function:



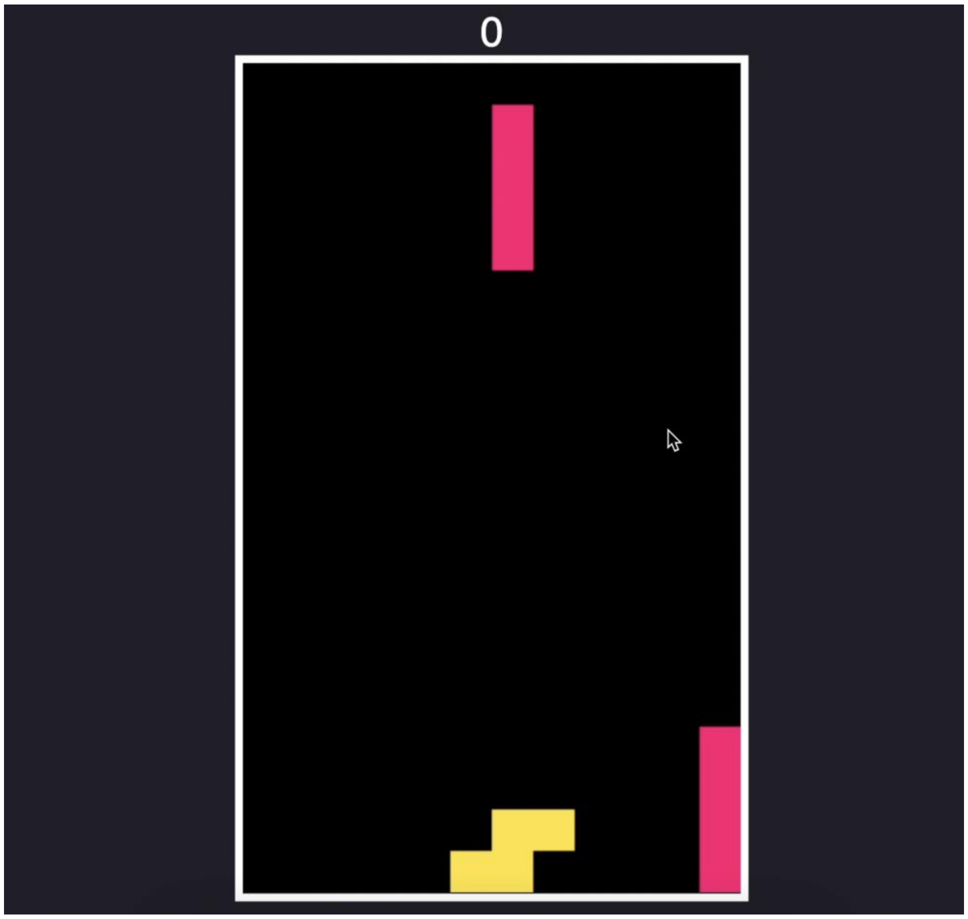
```
function collide(arena, player) {
    const m = player.matrix;
    const o = player.pos;
    for (let y = 0; y < m.length; ++y) {
        for (let x = 0; x < m[y].length; ++x) {
            if (m[y][x] !== 0 &&
               (arena[y + o.y] &&
                arena[y + o.y][x + o.x]) !== 0) {
                return true;
            }
        }
    }
    return false;
}
```

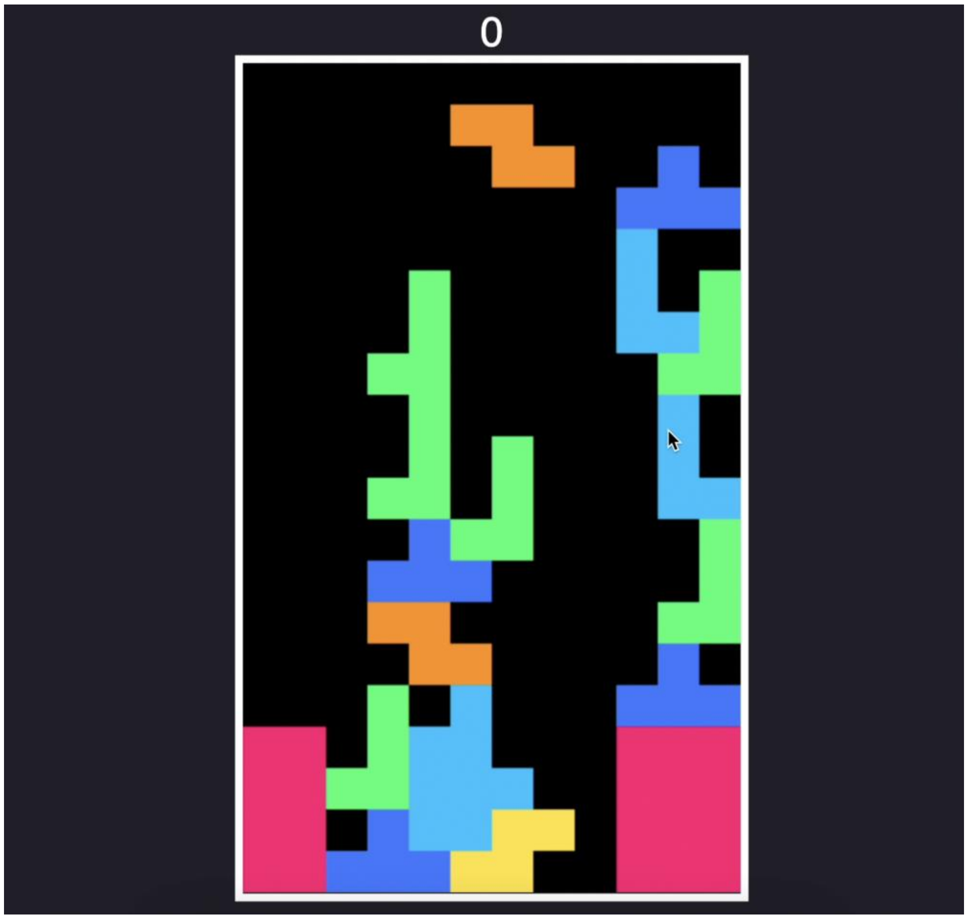
13] now for styling the background and borders i added the style tag to the html [<style>] (CSS):



```
<style>
  body {
    background: #202028;
    color: #fff;
    font-family: sans-serif;
    font-size: 2em;
    text-align: center;
  }
  canvas {
    border: solid .2em #fff;
    height: 90vh;
  }
</style>
```

- **Result:**


0



- **<u>Conclusion:</u>**

- **The Tetris effect (also known as Tetris syndrome) occurs when people devote so much time and attention to an activity that it begins to pattern their thoughts, mental images, and dreams. Those experiencing the effect may feel they are unable to prevent the thoughts, images, or dreams from happening.**

- **In 1994 Lynn Okazaki and Peter Frensch, in Journal of Applied Developmental Psychology, concluded playing Tetris had positive**

results on spatial skills, such as mental rotation, spatial perception, and spatial visualization.

- **Tetris can ease us through periods of anxiety by getting us to a blissfully engrossed mental state that psychologists call "flow."**

- **Reference:**

1. **https://youtu.be/K-pG2I0xRKY**
2. **https://freefrontend.com/javascript-tetris-games/**