Javascript

테트리스 게임

2022.11~ 권진이

목차

프로젝트 기획

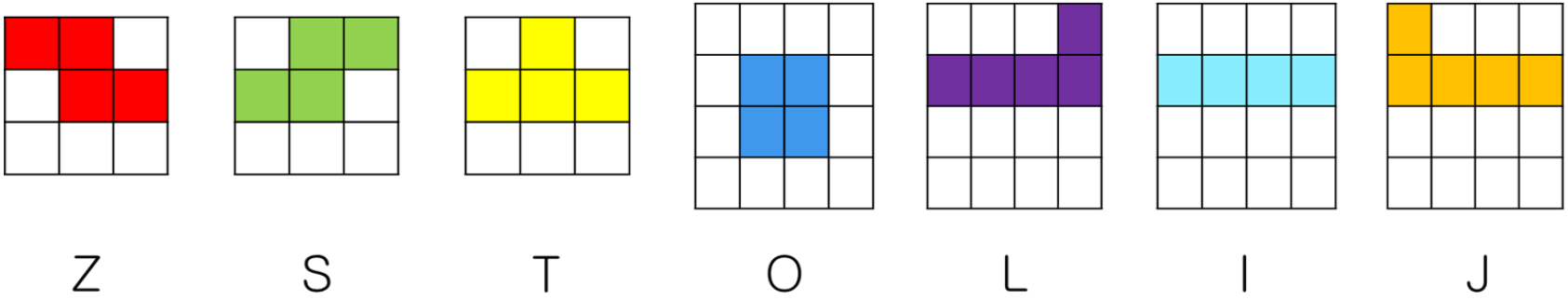
테트리스 게임 구현

- 오픈 소스 참조
- 코드 분석하기

- 7개의 테트리스 조각 만들기
- 테트리스 조각 회전
- 조각 빠르게 떨어뜨리기
- 마지막 한 줄 채우면 줄 삭제 및 점수 +10

구현

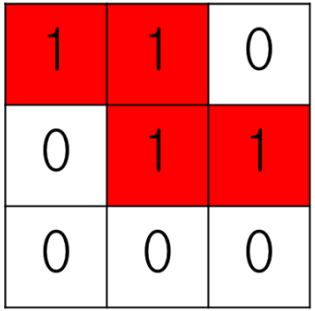
- 테트리스 조각들

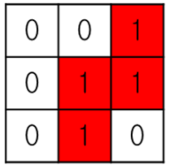


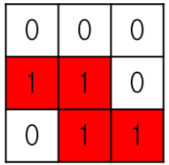
구현

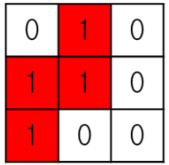
- 조각

Z





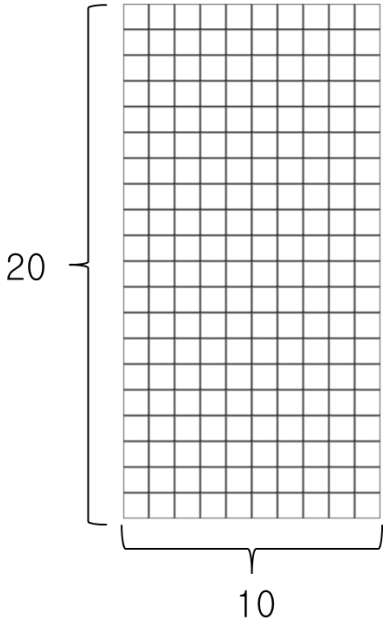





```
const Z = [
    [
        [1, 1, 0],
        [0, 1, 1],
        [0, 0, 0]
    ],
    [
        [0, 0, 1],
        [0, 1, 1],
        [0, 1, 0]
    ],
    [
        [0, 0, 0],
        [1, 1, 0],
        [0, 1, 1]
    ],
    [
        [0, 1, 0],
        [1, 1, 0],
        [1, 0, 0]
    ]
];
```

구현

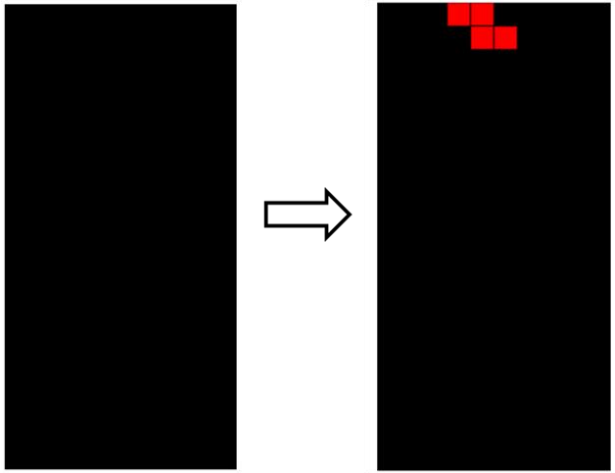
- 테트리스 보드판




```
const ROW = 20;
const COL = 10;
const SQ = 25;
const VACANT = "black";


// 한개의 칸 채우기
function drawSquare(x, y, color) {
    ctx.fillStyle = color;
    ctx.fillRect(x*SQ, y*SQ, SQ, SQ);

    ctx.strokeStyle = "black";
    ctx.strokeRect(x*SQ, y*SQ, SQ, SQ);
}

// 보드판 형성
let board = [];
for(r = 0; r < ROW; r++) {
    board[r] = [];
    for(c = 0; c < COL; c++) {
        board[r][c] = VACANT;
    }
}


// 보드판 그리기
function drawBoard() {
    for(r = 0; r < ROW; r++) {
        for(c = 0; c < COL; c++) {
            drawSquare(c, r, board[r][c]);
        }
    }
}
```

구현

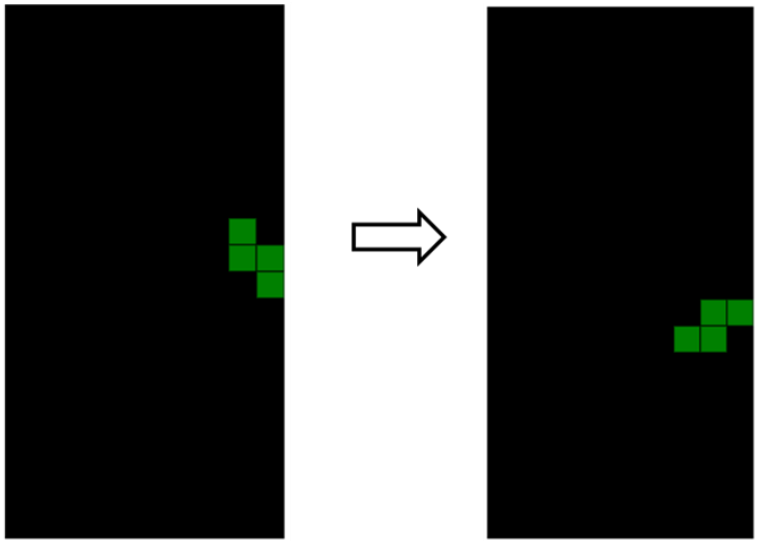
- 보드판에 조각 나타내기



```
// 조각
const PIECES = [
    [Z, "red"],
    [S, "green"],
    [T, "yellow"],
    [O, "blue"],
    [L, "purple"],
    [I, "cyan"],
    [J, "orange"]
];

// 랜덤 조각
function randomPiece(){
    let r = randomN = Math.floor(Math.random() * PIECES.length)
    return new Piece(PIECES[r][0], PIECES[r][1]);
}

let p = randomPiece();


// 조각 함수

function Piece(tetromino, color) {
    this.tetromino = tetromino;
    this.color = color;

    this.tetrominoN = 0;
    this.activeTetromino = this.tetromino[this.tetrominoN];

    this.x = 3;
    this.y = -2;
}

// 보드판에 조각 나타내기
Piece.prototype.fill = function(color){
    for(r = 0; r < this.activeTetromino.length; r++){
        for(c = 0; c < this.activeTetromino.length; c++){
            if(this.activeTetromino[r][c]){
                drawSquare(this.x + c, this.y + r, color);
            }
        }
    }
}
```

구현

- 조각 회전

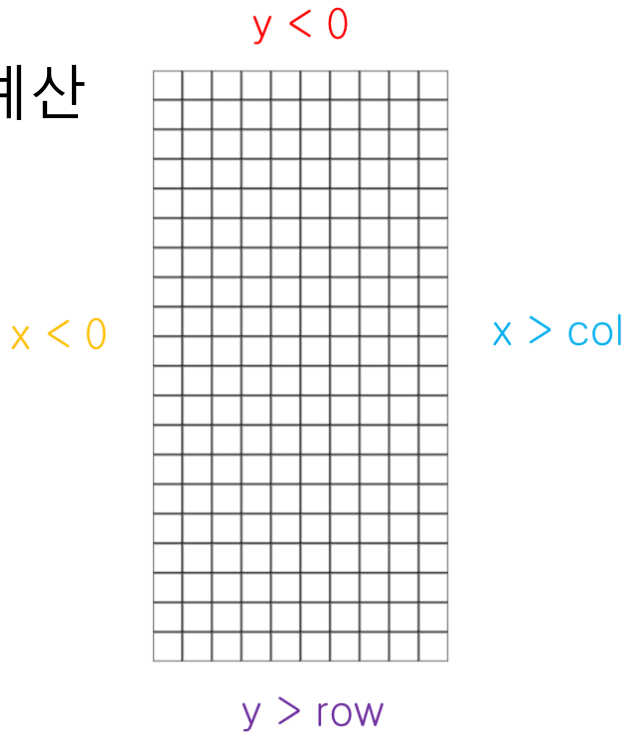


```
// 조각 회전
Piece.prototype.rotate = function() {
    let nextPattern = this.tetromino[(this.tetrominoN + 1)%this.tetromino.length];
    let kick = 0;

    if(this.collision(0, 0, nextPattern)){
        if(this.x > COL/2){
            kick = -1;
        }else{
            kick = 1;
        }
    }
    if(!this.collision(kick, 0, nextPattern)){
        this.unDraw();
        this.x += kick;
        this.tetrominoN = (this.tetrominoN + 1)%this.tetromino.length;
        this.activeTetromino = this.tetromino[this.tetrominoN];
        this.draw();
    }
}
```

회전했을 때 화면 밖으로 조각이 나가지 않도록 충돌 계산

구현

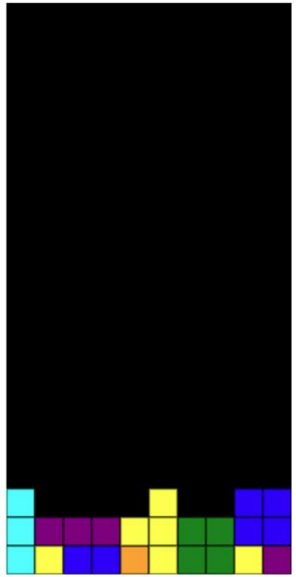
- 충돌 계산




```
Piece.prototype.collision = function(x, y, piece){
    for(r = 0; r < piece.length; r++){
        for(c = 0; c < piece.length; c++){
            // 비어있다면 충돌x
            if(!piece[r][c]){
                continue;
            }
            let newX = this.x + c + x;
            let newY = this.y + r + y;

            if(newX < 0 || newX >= COL || newY >= ROW){
                return true;
            }
            if(newY < 0){
                continue;
            }
            if(board[newY][newX] != VACANT){
                return true;
            }

        }
    }return false;
}
```

구현

- 한 줄 채우면 줄 삭제
- 점수 10 추가



점수 : 20

```
Piece.prototype.lock = function() {
    for(r = 0; r < this.activeTetromino.length; r++){
        for(c = 0; c < this.activeTetromino.length; c++){
            if(!this.activeTetromino[r][c]){
                continue;
            }
            if(this.y + r < 0){
                alert("Game Over");
                gameOver = true;
                break;
            }
            board[this.y+r][this.x+c] = this.color;
        }
    }
    for(r = 0; r < ROW; r++){
        let isRowFull = true;
        for(c = 0; c < COL; c++){
            isRowFull = isRowFull && (board[r][c] != VACANT);
        }
        if(isRowFull){
            for(y = r; y > 1; y--){
                for(c = 0; c < COL; c++){
                    board[y][c] = board[y-1][c];
                }
            }
            for(c = 0; c < COL; c++){
                board[0][c] = VACANT;
            }
            score += 10;
        }
    }
    drawBoard();
    scoreElement.innerHTML = score;
}
```

개선점

- 일시정지 기능 추가
- 재시작 기능 추가
- 일정 점수 이상 속도가 빨라지는 기능 추가

감사합니다