<!DOCTYPE html>

<html lang="ko">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>테트리스 게임</title>

<style>

body {

background: #111;

display: flex;

flex-direction: column;

align-items: center;

color: white;

font-family: sans-serif;

}

canvas {

background: #222;

border: 2px solid #fff;

margin-top: 20px;

}

.info {

margin-top: 10px;

}

</style>

</head>

<body>

<h1>🎮 테트리스</h1>

<canvas id="tetris" width="240" height="400"></canvas>

<div class="info">

<p>점수: <span id="score">0</span></p>

<p>레벨: <span id="level">1</span></p>

</div>

<audio id="bgm" src="https://www.bensound.com/bensound-music/bensound-littleidea.mp3" autoplay loop></audio>

<script>

const canvas = document.getElementById('tetris');

const context = canvas.getContext('2d');

context.scale(20, 20);

const scoreElem = document.getElementById('score');

const levelElem = document.getElementById('level');

const pieces = 'TJLOSZI';

function createPiece(type) {

if (type === 'T') {

return [

[0, 0, 0],

[1, 1, 1],

[0, 1, 0],

];

} else if (type === 'O') {

return [

[2, 2],

[2, 2],

];

} else if (type === 'L') {

return [

[0, 3, 0],

[0, 3, 0],

[0, 3, 3],

];

} else if (type === 'J') {

return [

[0, 4, 0],

[0, 4, 0],

[4, 4, 0],

];

} else if (type === 'I') {

return [

[0, 5, 0, 0],

[0, 5, 0, 0],

[0, 5, 0, 0],

[0, 5, 0, 0],

];

} else if (type === 'S') {

return [

[0, 6, 6],

[6, 6, 0],

[0, 0, 0],

];

} else if (type === 'Z') {

return [

[7, 7, 0],

[0, 7, 7],

[0, 0, 0],

];

}

}

function createMatrix(w, h) {

const matrix = [];

while (h--) {

matrix.push(new Array(w).fill(0));

}

return matrix;

}

function collide(arena, player) {

const [m, o] = [player.matrix, 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;

}

function merge(arena, player) {

player.matrix.forEach((row, y) => {

row.forEach((value, x) => {

if (value !== 0) {

arena[y + player.pos.y][x + player.pos.x] = value;

}

});

});

}

function rotate(matrix) {

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]];

}

}

matrix.forEach(row => row.reverse());

}

function playerDrop() {

player.pos.y++;

if (collide(arena, player)) {

player.pos.y--;

merge(arena, player);

playerReset();

arenaSweep();

updateScore();

}

dropCounter = 0;

}

function playerMove(dir) {

player.pos.x += dir;

if (collide(arena, player)) {

player.pos.x -= dir;

}

}

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;

player.level = 1;

dropInterval = 1000;

}

}

function playerRotate() {

const pos = player.pos.x;

let offset = 1;

rotate(player.matrix);

while (collide(arena, player)) {

player.pos.x += offset;

offset = -(offset + (offset > 0 ? 1 : -1));

if (offset > player.matrix[0].length) {

rotate(player.matrix);

rotate(player.matrix);

rotate(player.matrix);

player.pos.x = pos;

return;

}

}

}

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;

if (player.score >= player.level \* 100) {

player.level++;

dropInterval \*= 0.9;

}

}

}

function drawMatrix(matrix, offset) {

matrix.forEach((row, y) => {

row.forEach((value, x) => {

if (value !== 0) {

context.fillStyle = '#0f0';

context.fillRect(x + offset.x, y + offset.y, 1, 1);

}

});

});

}

function draw() {

context.fillStyle = '#000';

context.fillRect(0, 0, canvas.width, canvas.height);

drawMatrix(arena, {x: 0, y: 0});

drawMatrix(player.matrix, player.pos);

}

function updateScore() {

scoreElem.innerText = player.score;

levelElem.innerText = player.level;

}

let dropCounter = 0;

let dropInterval = 1000;

let lastTime = 0;

function update(time = 0) {

const deltaTime = time - lastTime;

lastTime = time;

dropCounter += deltaTime;

if (dropCounter > dropInterval) {

playerDrop();

}

draw();

requestAnimationFrame(update);

}

document.addEventListener('keydown', event => {

if (event.key === 'ArrowLeft') {

playerMove(-1);

} else if (event.key === 'ArrowRight') {

playerMove(1);

} else if (event.key === 'ArrowDown') {

playerDrop();

} else if (event.key === 'ArrowUp') {

playerRotate();

}

});

const arena = createMatrix(12, 20);

const player = {

pos: {x: 0, y: 0},

matrix: null,

score: 0,

level: 1

};

playerReset();

updateScore();

update();

</script>

</body>

</html>