```
<!DOCTYPE html>
<html lang="ky">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Жылаан × Математика</title>
</head>

<body>
    <!-- 🟦 Башкы бөлүк: оюн тууралуу абалды көрсөтөт -->
    <header>
        <div class="hud" aria-label="Status left">
            <!-- Режимди көрсөтөт (кошуу, алуу, ж.б.) -->
            <div class="chip">Mode: <b id="modeChip">—</b></div>
            <!-- Деңгээлди көрсөтөт -->
            <div class="chip">Level: <b id="levelChip">—</b></div>
        </div>

        <!-- Оюндун аталышы -->
        <h1>🐍 Snake × <span style="color:var(--accent)">Math</span></h1>

        <div class="hud" aria-label="Status right">
            <!-- Топтолгон упай -->
            <div class="chip">Score: <b id="scoreChip">0</b></div>
            <!-- Убакыт эсептегич (таймер) -->
            <div class="chip hidden" id="timeChipContainer">Time: <b id="timeChip">—</b>
</div>
            <!-- Жандар саны -->
            <div class="chip">Lives: <b id="livesChip">3</b></div>
        </div>

        <!-- Панель: маселе жана башкаруу баскычтары -->
        <div class="panel" role="group" aria-label="Expression & controls">
            <button class="btn" id="menuBtn" title="Main menu">≡ Menu</button>
            <div class="expr">Solve: <b id="exprText">—</b></div>
            <button class="btn" id="pauseBtn" title="Pause/Resume">⏯ Pause</button>
        </div>
    </header>

    <div class="wrap">
        <!-- Оюн талаасы -->
        <div class="board-wrap" id="boardWrap">
            <!-- Canvas — оюн талаасы (ячейкалар менен сетка) -->
            <canvas id="board" width="400" height="400" aria-label="Game board" tabindex="0">
</canvas>

            <!-- 📋 Башкы меню үстүнкү катмары -->
            <div class="overlay hidden" id="menu">
                <div class="card">
                    <h2>Start Game</h2>
                    <p>Режим жана деңгээлди танда. Бөлүүдө ар дайым бүтүн сан чыгат. Дубалга
жана өзүңө урунба. Туура
                        жоопту же, туура эмес жооп бир жанды алат.</p>
                    <div class="row">
                        <!-- Режим тандоо -->
                        <label>
                            <div style="margin-bottom:6px">Mode</div>
                            <select class="select" id="modeSelect">
                                <option value="add">Addition</option>
                                <option value="sub">Subtraction</option>
                                <option value="mul">Multiplication</option>
                                <option value="div">Division</option>
                                <option value="mix" selected>Mixed</option>
                            </select>
                        </label>
                        <!-- Деңгээл тандоо -->
```

```
                        <label>
                            <div style="margin-bottom:6px">Level</div>
                            <select class="select" id="levelSelect">
                                <option value="easy">Easy (1–9)</option>
                                <option value="med">Medium (10–99)</option>
                                <option value="mix">Mixed (1–99)</option>
                                <option value="adv">Advanced (to 3 digits)</option>
                                <option value="exp" selected>Expert (fast)</option>
                            </select>
                        </label>
                    </div>
                    <!-- Опциялар -->
                    <div style="margin-top:10px;">
                        <label style="display:flex; align-items:center; gap:8px; margin-
bottom:6px;">
                            <input type="checkbox" id="showHintsCheck" checked>
                            <span>Туура жоопко жашыл подсказка көрсөтүү</span>
                        </label>
                        <label style="display:flex; align-items:center; gap:8px; margin-
bottom:6px;">
                            <input type="checkbox" id="timerCheck" checked>
                            <span>Таймерди күйгүзүү</span>
                        </label>
                        <label style="display:flex; align-items:center; gap:8px;">
                            <input type="checkbox" id="sandboxCheck">
                            <span>Sandbox режим (чексиз оюн)</span>
                        </label>
                    </div>
                    <!-- Башкаруу баскычтары -->
                    <div style="display:flex; gap:8px; margin-top:10px;">
                        <button class="btn" id="startBtn">▶ Start</button>
                        <button class="btn" id="howBtn">❓ How to play</button>
                    </div>
                    <p class="notice">Башкаруу: жебе баскычтар / WASD • экрандагы D-pad •
сүрүү. <span
                            class="dot"></span></p>
                </div>
            </div>

            <!-- ℹ Кантип ойноо үстүнкү катмары -->
            <div class="overlay hidden" id="info">
                <div class="card">
                    <h2>How to play</h2>
                    <p>Классикалык жылаан эрежелери. Жогорку жакта мисал чыгат. Туура жоопко
жетсең упай кошулат, туура
                        эмес болсо жан кетет. 10 туура жооп бергенде деңгээл аяктайт.</p>
                    <div style="display:flex; gap:8px; margin-top:8px;">
                        <button class="btn" id="resumeFromInfo">OK</button>
                    </div>
                </div>
            </div>

            <!-- ✅ Деңгээл аяктады -->
            <div class="overlay hidden" id="levelDone">
                <div class="card">
                    <h2>Level Complete 🎉</h2>
                    <p>Сен <b id="solvedCount">0</b> мисалды туура чечтиң! Улантасыңбы же
жаңы деңгээл тандайсыңбы?</p>
                    <div style="display:flex; gap:8px; margin-top:8px; flex-wrap: wrap;">
                        <button class="btn" id="continueBtn">Continue</button>
                        <button class="btn" id="changeBtn">Change Mode/Level</button>
                    </div>
                </div>
            </div>

            <!-- ☠ Оюн бүттү -->
            <div class="overlay hidden" id="gameOver">
                <div class="card">
```

```
                <h2>Game Over 💀</h2>
                <p>Сенин упайың: <b id="finalScore">0</b>. Кайра аракет кыласыңбы?</p>
                <div style="display:flex; gap:8px; margin-top:8px; flex-wrap: wrap;">
                    <button class="btn" id="retryBtn">↻ Retry</button>
                    <button class="btn" id="goMenuBtn">≡ Menu</button>
                </div>
            </div>
        </div>
    </div>

    <!-- 📱 Мобилдик D-PAD -->
    <div class="controls" aria-label="Mobile controls">
        <button class="pad up" data-dir="up">▲</button>
        <button class="pad left" data-dir="left">◀</button>
        <button class="pad ok" id="okBtn">⏯</button>
        <button class="pad right" data-dir="right">▶</button>
        <button class="pad down" data-dir="down">▼</button>
    </div>

    <div class="notice">Tip: <b>Expert</b> деңгээлде жылаан ар бир жооптон кийин тезирээк болот.</div>
  </div>

  <script>
    // 🟦 Оюндун негизги классы
    class SnakeMathGame {
        constructor() {
            // 🟩 Оюн талаасынын өлчөмдөрү
            this.GRID_SIZE = 20;       // сетка (20×20)
            this.TILE_SIZE = 20;       // ар бир ячейканын өлчөмү
            this.CANVAS_SIZE = this.GRID_SIZE * this.TILE_SIZE; // талаанын көлөмү (400×400)
            this.TARGET_SOLVES = 10;   // бир деңгээлде чыгарыла турган мисалдардын саны
            this.START_LIVES = 3;      // баштапкы жандар саны

            // 🖼️ Canvas даярдоо
            this.canvas = document.getElementById('board');
            this.ctx = this.canvas.getContext('2d');
            this.canvas.width = this.CANVAS_SIZE;
            this.canvas.height = this.CANVAS_SIZE;
            this.ctx.imageSmoothingEnabled = false; // пикселди жумшартпайбыз

            // 📊 Оюн абалы — бардык маанилүү маалыматтар ушул объектте сакталат
            this.gameState = {
                running: false,      // оюн иштеп жатабы
                paused: false,       // токтотулуп турабы
                snake: [],           // жылаандын сегменттери
                direction: { x: 1, y: 0 },   // учурдагы багыт (оңго)
                nextDirection: { x: 1, y: 0 }, // кийинки багыт
                growQueue: 0,        // жылаан канчага чоңоёт
                answerTiles: new Map(), // жооп ячейкалары
                currentAnswer: 0,    // туура жооп
                currentExpression: '—', // чыгарылатурган мисал
                mode: 'mix',         // режим (кошуу, алуу ж.б.)
                level: 'exp',        // деңгээл
                score: 0,            // упай
                lives: this.START_LIVES, // жандар саны
                solved: 0,           // туура жооптор саны
                tickInterval: 140,   // жылаандын ылдамдыгы (миллисекунд)
                gameTimer: null,     // оюн таймери
                flashCounter: 0,
                showHints: true,     // туура жоопту көрсөтүү
                sandboxMode: false,  // чексиз режим
                timerEnabled: true,  // таймер күйүп турабы
                timeRemaining: 0,    // калган убакыт
                timerIntervalId: null, // таймердин ID
                tilePositions: []    // жооп ячейкаларынын жайгашуусу
```

```
        };

        // 🟨 Ар бир деңгээлге тиешелүү сандар (кошуу, алуу, бөлүү ж.б.)
        this.levels = {
            easy: { add: [1, 9], sub: [1, 9], mul: [1, 9], div: [1, 9], q: [1, 9],
time: 120, timeBonus: 10 },
            med: { add: [10, 99], sub: [10, 99], mul: [2, 12], div: [2, 20], q: [2,
12], time: 100, timeBonus: 8 },
            mix: { add: [1, 99], sub: [1, 99], mul: [2, 15], div: [2, 20], q: [2,
15], time: 90, timeBonus: 7 },
            adv: { add: [50, 999], sub: [50, 999], mul: [10, 99], div: [3, 60], q:
[3, 25], time: 75, timeBonus: 6 },
            exp: { add: [1, 999], sub: [1, 999], mul: [5, 99], div: [3, 80], q: [3,
30], time: 60, timeBonus: 5 }
        };

        this.init(); // оюнду баштапкы абалга келтирүү
    }

    // 🟦 Инициализация: окуя уккучтарды коюу, өзгөрмөлөрдү жүктөө
    init() {
        this.setupEventListeners();
        this.loadSettings();
        this.showMenu();
        this.draw();
    }

    // 🟦 Баскычтарды, клавиатураны жана сенсорду иштетүү
    setupEventListeners() {
        // 📌 Баскычтар
        document.getElementById('menuBtn').onclick = () => this.showMenu();
        document.getElementById('pauseBtn').onclick = () => this.togglePause();
        document.getElementById('howBtn').onclick = () => this.showInfo();
        document.getElementById('resumeFromInfo').onclick = () => this.hideInfo();
        document.getElementById('startBtn').onclick = () => this.startFromMenu();
        document.getElementById('continueBtn').onclick = () => {
this.hideLevelDone(); this.resume(); };
        document.getElementById('changeBtn').onclick = () => { this.hideLevelDone();
this.showMenu(); };
        document.getElementById('retryBtn').onclick = () => { this.hideGameOver();
this.startGame(); };
        document.getElementById('goMenuBtn').onclick = () => { this.hideGameOver();
this.showMenu(); };
        document.getElementById('okBtn').onclick = () => this.togglePause();

        // 📌 Экрандагы жебе баскычтар (мобилдик башкаруу)
        document.querySelectorAll('.pad[data-dir]').forEach(btn => {
            btn.onclick = () => this.setDirection(btn.getAttribute('data-dir'));
        });

        // 📌 Клавиатура менен башкаруу
        document.addEventListener('keydown', (e) => {
            if (["ArrowUp", "ArrowDown", "ArrowLeft", "ArrowRight", "
"].includes(e.key)) {
                e.preventDefault(); // баракты жылдырбоо
            }

            if (e.key === ' ') { // пробел — токтотуу/улантуу
                this.togglePause();
                return;
            }

            const keyMap = {
                ArrowUp: 'up', ArrowDown: 'down', ArrowLeft: 'left', ArrowRight:
'right',
                w: 'up', W: 'up', s: 'down', S: 'down',
                a: 'left', A: 'left', d: 'right', D: 'right'
```

```
            };

            if (keyMap[e.key]) {
                this.setDirection(keyMap[e.key]);
            }
        });

        // 📌 Сенсор менен башкаруу (сүрүү)
        let touchStart = null;
        this.canvas.addEventListener('touchstart', (e) => {
            if (e.touches[0]) {
                touchStart = { x: e.touches[0].clientX, y: e.touches[0].clientY };
                e.preventDefault(); // скроллду токтотобуз
            }
        }, { passive: false });

        this.canvas.addEventListener('touchmove', (e) => {
            e.preventDefault(); // скролл болбош үчүн
        }, { passive: false });

        this.canvas.addEventListener('touchend', (e) => {
            if (!touchStart) return;
            e.preventDefault();

            const touch = e.changedTouches[0];
            const dx = touch.clientX - touchStart.x;
            const dy = touch.clientY - touchStart.y;
            const absX = Math.abs(dx);
            const absY = Math.abs(dy);

            if (Math.max(absX, absY) > 24) {
                if (absX > absY) {
                    this.setDirection(dx > 0 ? 'right' : 'left');
                } else {
                    this.setDirection(dy > 0 ? 'down' : 'up');
                }
            }
            touchStart = null;
        }, { passive: false });
    }

    // ⚙ Интерфейс үстүнкү катмарларын көрсөтүү/жашыруу (меню, info, gameOver ж.б.)
    showMenu() { document.getElementById('menu').classList.remove('hidden');
this.pause(); }
    hideMenu() { document.getElementById('menu').classList.add('hidden'); }
    showInfo() { document.getElementById('info').classList.remove('hidden');
this.pause(); }
    hideInfo() { document.getElementById('info').classList.add('hidden');
this.resume(); }
    showLevelDone() {
document.getElementById('levelDone').classList.remove('hidden'); this.pause(); }
    hideLevelDone() { document.getElementById('levelDone').classList.add('hidden'); }
    showGameOver() { document.getElementById('gameOver').classList.remove('hidden');
this.pause(); }
    hideGameOver() { document.getElementById('gameOver').classList.add('hidden'); }

    // 🚀 Менюдан оюнду баштоо
    startFromMenu() {
        this.gameState.mode = document.getElementById('modeSelect').value;
        this.gameState.level = document.getElementById('levelSelect').value;
        this.gameState.showHints = document.getElementById('showHintsCheck').checked;
        this.gameState.sandboxMode = document.getElementById('sandboxCheck').checked;
        this.gameState.timerEnabled = document.getElementById('timerCheck').checked;
        this.saveSettings();
        this.hideMenu();
        this.startGame();
    }
```

```
// 🎮 Оюнду кайра баштоо (жаныраак)
startGame() {
    this.gameState.score = 0;
    this.gameState.lives = this.START_LIVES;
    this.gameState.solved = 0;
    this.gameState.flashCounter = 0;
    this.setupSpeed();
    this.resetSnake();
    this.spawnProblem();
    this.updateHUD();
    this.resume();

    // Таймер күйүп турган болсо
    if (this.gameState.timerEnabled) {
        const levelConfig = this.levels[this.gameState.level] || this.levels.mix;
        this.gameState.timeRemaining = levelConfig.time;
        this.startTimer();
    }
}

// 🟦 Жылаандын ылдамдыгын деңгээлге жараша коюу
setupSpeed() {
    const speedMap = { easy: 160, med: 150, mix: 145, adv: 140, exp: 130 };
    this.gameState.tickInterval = speedMap[this.gameState.level] || 140;
}

// 🟦 Жылаанды баштапкы абалга коюу (борбордон чыгат)
resetSnake() {
    const center = Math.floor(this.GRID_SIZE / 2);
    this.gameState.snake = [
        { x: center, y: center },
        { x: center - 1, y: center },
        { x: center - 2, y: center }
    ];
    this.gameState.direction = { x: 1, y: 0 };
    this.gameState.nextDirection = { x: 1, y: 0 };
    this.gameState.growQueue = 0;
}

// 🟦 Оюнду токтотуу/улантуу
togglePause() { this.gameState.paused ? this.resume() : this.pause(); }

pause() {
    this.gameState.paused = true;
    this.gameState.running = false;
    if (this.gameState.gameTimer) {
        clearInterval(this.gameState.gameTimer);
        this.gameState.gameTimer = null;
    }
    this.stopTimer();
}

resume() {
    const overlaysHidden = ['menu', 'info', 'gameOver', 'levelDone']
        .every(id => document.getElementById(id).classList.contains('hidden'));

    if (overlaysHidden) {
        this.gameState.paused = false;
        this.runGameLoop();
        this.startTimer();
    }
}

// 🟦 Негизги оюн цикли (ар тик сайын иштейт)
runGameLoop() {
    if (this.gameState.gameTimer) clearInterval(this.gameState.gameTimer);
    this.gameState.running = true;
    this.gameState.gameTimer = setInterval(() => this.tick(),
```

```
this.gameState.tickInterval);
        }

        // 🟦 Багытты өзгөртүү
        setDirection(dir) {
            const map = {
                up: { x: 0, y: -1 },
                down: { x: 0, y: 1 },
                left: { x: -1, y: 0 },
                right: { x: 1, y: 0 }
            };
            const newDir = map[dir];
            if (!newDir) return;

            // Жылаан артка бурулбайт
            if (this.gameState.snake.length > 1 &&
                newDir.x === -this.gameState.direction.x &&
                newDir.y === -this.gameState.direction.y) return;

            this.gameState.nextDirection = newDir;
        }

        // 🟦 Кокустук сан жана тандап алуу
        randomInt(min, max) { return Math.floor(Math.random() * (max - min + 1)) + min; }
        randomChoice(arr) { return arr[Math.floor(Math.random() * arr.length)]; }

        // 🟦 Математикалык мисал түзүү
        generateProblem() {
            const lvl = this.levels[this.gameState.level] || this.levels.mix;
            const mode = this.gameState.mode === 'mix'
                ? this.randomChoice(['add', 'sub', 'mul', 'div'])
                : this.gameState.mode;

            let a, b, ans, op;
            switch (mode) {
                case 'add': a = this.randomInt(...lvl.add); b =
this.randomInt(...lvl.add); ans = a + b; op = '+'; break;
                case 'sub': a = this.randomInt(...lvl.sub); b =
this.randomInt(...lvl.sub); if (b > a) [a, b] = [b, a]; ans = a - b; op = '−'; break;
                case 'mul': a = this.randomInt(...lvl.mul); b =
this.randomInt(...lvl.mul); ans = a * b; op = '×'; break;
                case 'div': b = this.randomInt(...lvl.div); const q =
this.randomInt(...lvl.q); a = b * q; ans = q; op = '÷'; break;
            }
            return { a, b, operator: op, answer: ans, expression: `${a} ${op} ${b}` };
        }

        // 🟦 Жооп ячейкаларын коюу
        spawnProblem() {
            const prob = this.generateProblem();
            this.gameState.currentAnswer = prob.answer;
            this.gameState.currentExpression = prob.expression;

            // эгер ячейка жок болсо жаңыларын түзөбүз
            if (this.gameState.tilePositions.length === 0 ||
this.gameState.answerTiles.size === 0)
                this.generateTilePositions();

            // жооп ячейкаларын толтуруу
            this.gameState.answerTiles.clear();
            const shuffled = [...this.gameState.tilePositions].sort(() => Math.random() -
0.5);

            // туура жооп
            this.gameState.answerTiles.set(shuffled[0], prob.answer);

            // адаштыруучу жооптор
            const distractors = [...this.generateDistractors(prob.answer,
```

```
prob.operator)];
                for (let i = 1; i < shuffled.length && i - 1 < distractors.length; i++) {
                    this.gameState.answerTiles.set(shuffled[i], distractors[i - 1]);
                }
                this.updateHUD();
            }

            // 🟦 Адаштыруучу жоопторду түзүү
            generateDistractors(ans, op) {
                const set = new Set();
                const variations = [1, 2, 3, 5, 10, this.randomInt(2, 12)];

                for (let i = 0; i < 100 && set.size < 3; i++) {
                    let d = Math.random() < 0.5 ? ans + this.randomChoice(variations) : ans -
this.randomChoice(variations);
                    if (op === '×' || op === '÷') if (Math.random() < 0.5) d = Math.round(ans
* (Math.random() < 0.5 ? 2 : 0.5));
                    d = Math.abs(Math.round(d));
                    if (d !== ans && d > 0) set.add(d);
                }
                return set;
            }

            // 🟦 Жооп ячейкаларынын жайгашуусу
            generateTilePositions() {
                this.gameState.tilePositions = [];
                const occupied = new Set(this.gameState.snake.map(s => `${s.x},${s.y}`));

                for (let n = 0; n < 4; n++) {
                    for (let i = 0; i < 200; i++) {
                        const x = this.randomInt(0, this.GRID_SIZE - 1), y =
this.randomInt(0, this.GRID_SIZE - 1);
                        const k = `${x},${y}`;
                        if (!occupied.has(k) && !this.gameState.tilePositions.includes(k)) {
                            this.gameState.tilePositions.push(k); break;
                        }
                    }
                }
                this.gameState.tilePositions.sort(() => Math.random() - 0.5);
            }

            // 🟦 Негизги тик (оюн бир кадам алдыга жылат)
            tick() {
                if (this.gameState.paused || !this.gameState.running) return;

                // жаңы баш координата
                this.gameState.direction = { ...this.gameState.nextDirection };
                const head = this.gameState.snake[0];
                const nx = head.x + this.gameState.direction.x, ny = head.y +
this.gameState.direction.y;

                // дубалга урунса
                if (nx < 0 || ny < 0 || nx >= this.GRID_SIZE || ny >= this.GRID_SIZE) {
this.loseLife('wall'); return; }

                // өзү менен урунса
                if (this.gameState.snake.some(s => s.x === nx && s.y === ny)) {
this.loseLife('self'); return; }

                this.gameState.snake.unshift({ x: nx, y: ny });
                const tile = `${nx},${ny}`;
                let ate = false;

                // жооп тайлына урунду
                if (this.gameState.answerTiles.has(tile)) {
                    const val = this.gameState.answerTiles.get(tile);
                    if (val === this.gameState.currentAnswer) {
                        ate = true; this.gameState.score += 10; this.gameState.solved++;
```

```
this.gameState.growQueue++;
                        if (this.gameState.timerEnabled) this.gameState.timeRemaining +=
(this.levels[this.gameState.level] || this.levels.mix).timeBonus;
                        if (this.gameState.level === 'exp' && this.gameState.tickInterval >
70) { this.gameState.tickInterval -= 3; this.runGameLoop(); }
                        if (!this.gameState.sandboxMode && this.gameState.solved >=
this.TARGET_SOLVES) { this.updateHUD(); this.showLevelDone(); return; }
                    } else { this.loseLife('wrong'); return; }

                    this.gameState.answerTiles.delete(tile);
                    if (this.gameState.answerTiles.size === 0) { this.gameState.tilePositions
= []; this.spawnProblem(); }
                    else if (ate) { this.spawnProblem(); }
                }

                // жылаандын узундугу
                if (ate || this.gameState.growQueue > 0) { if (!ate)
this.gameState.growQueue--; }
                else this.gameState.snake.pop();

                this.updateHUD(); this.draw();
            }

            // 🟦 Жанды жоготуу
            loseLife(reason) {
                this.pause(); this.gameState.lives--; this.gameState.flashCounter = 6;
                if (this.gameState.lives <= 0) { this.updateHUD(); this.draw();
this.showGameOver(); return; }
                this.updateHUD(); this.resetSnake();
                this.draw();
                setTimeout(() => { if (this.gameState.lives > 0) this.resume(); }, 1000);
            }

            // 🟦 Сүрөт тартуу (жылаан, тайлдар, подсказка)
            draw() {
                this.ctx.clearRect(0, 0, this.canvas.width, this.canvas.height);
                if (this.gameState.flashCounter > 0) { this.ctx.fillStyle =
`rgba(239,68,68,.2)`; this.ctx.fillRect(0, 0, this.canvas.width, this.canvas.height);
this.gameState.flashCounter--; }

                // жооп ячейкаларын тартуу
                this.ctx.textAlign = 'center'; this.ctx.textBaseline = 'middle';
this.ctx.font = '16px monospace';
                this.gameState.answerTiles.forEach((val, key) => {
                    const [x, y] = key.split(',').map(Number);
                    const px = x * this.TILE_SIZE + this.TILE_SIZE / 2, py = y *
this.TILE_SIZE + this.TILE_SIZE / 2;
                    const correct = val === this.gameState.currentAnswer;
                    const hint = this.gameState.showHints && correct;

                    this.ctx.fillStyle = hint ? 'rgba(34,197,94,.12)' :
'rgba(110,231,255,.1)';
                    this.ctx.fillRect(x * this.TILE_SIZE + 3, y * this.TILE_SIZE + 3,
this.TILE_SIZE - 6, this.TILE_SIZE - 6);
                    this.ctx.strokeStyle = hint ? 'rgba(34,197,94,.5)' :
'rgba(110,231,255,.25)';
                    this.ctx.strokeRect(x * this.TILE_SIZE + 3.5, y * this.TILE_SIZE + 3.5,
this.TILE_SIZE - 7, this.TILE_SIZE - 7);
                    this.ctx.fillStyle = hint ? '#22c55e' : '#9bd9ff';
                    this.ctx.fillText(String(val), px, py);
                });

                // жылаанды тартуу
                for (let i = this.gameState.snake.length - 1; i >= 0; i--) {
                    const seg = this.gameState.snake[i], head = i === 0;
                    this.ctx.fillStyle = head ? '#6ee7ff' : '#94a3b8';
                    const pad = head ? 2 : 3;
                    this.ctx.fillRect(seg.x * this.TILE_SIZE + pad, seg.y * this.TILE_SIZE +
pad, this.TILE_SIZE - 2 * pad, this.TILE_SIZE - 2 * pad);
```

```
            }
        }

        // 🟦 HUD жаңыртуу (интерфейстеги упай, жан, убакыт ж.б.)
        updateHUD() {
            document.getElementById('modeChip').textContent = { add: 'Addition', sub:
'Subtraction', mul: 'Multiplication', div: 'Division', mix: 'Mixed' }[this.gameState.mode];
            document.getElementById('levelChip').textContent = { easy: 'Easy', med:
'Medium', mix: 'Mixed', adv: 'Advanced', exp: 'Expert' }[this.gameState.level];
            document.getElementById('scoreChip').textContent = this.gameState.score;
            document.getElementById('livesChip').textContent = this.gameState.lives;
            document.getElementById('exprText').textContent =
this.gameState.currentExpression;
            document.getElementById('solvedCount').textContent = this.gameState.solved;
            document.getElementById('finalScore').textContent = this.gameState.score;

            if (this.gameState.timerEnabled) {
                document.getElementById('timeChipContainer').classList.remove('hidden');
                const m = Math.floor(this.gameState.timeRemaining / 60), s =
this.gameState.timeRemaining % 60;
                document.getElementById('timeChip').textContent =
`${m}:${s.toString().padStart(2, '0')}`;
            } else document.getElementById('timeChipContainer').classList.add('hidden');
        }

        // 🟦 Өзгөрмөлөрдү сактоо/жүктөө (локалдуу эс)
        loadSettings() {
            const m = localStorage.getItem('snakemath.mode'), l =
localStorage.getItem('snakemath.level'), t = localStorage.getItem('snakemath.timer');
            if (m) document.getElementById('modeSelect').value = m;
            if (l) document.getElementById('levelSelect').value = l;
            if (t !== null) document.getElementById('timerCheck').checked = t === 'true';
        }
        saveSettings() {
            localStorage.setItem('snakemath.mode', this.gameState.mode);
            localStorage.setItem('snakemath.level', this.gameState.level);
            localStorage.setItem('snakemath.timer', this.gameState.timerEnabled);
        }

        // 🟦 Таймер логикасы
        startTimer() {
            if (!this.gameState.timerEnabled || this.gameState.timerIntervalId ||
this.gameState.paused) return;
            this.gameState.timerIntervalId = setInterval(() => this.updateTimer(), 1000);
        }
        stopTimer() { if (this.gameState.timerIntervalId) {
clearInterval(this.gameState.timerIntervalId); this.gameState.timerIntervalId = null; } }
        updateTimer() {
            if (this.gameState.timeRemaining > 0) { this.gameState.timeRemaining--;
this.updateHUD(); }
            else { this.stopTimer(); this.loseLife('time'); }
        }
    }

    // 🟦 Оюнду ишке киргизүү
    const game = new SnakeMathGame();
    window.SnakeMath = game; // браузер консольдон текшерүү үчүн
</script>

</body>

</html>
```