Федеральное государственное образовательное бюджетное учреждение высшего образования

«Финансовый университет при правительстве Российской Федерации»

(Финансовый университет)

Колледж информатики и программирования

МДК.05.02 Разработка кода информационных систем

Группа: 3ИСИП-723

Лабораторная работа 3

"Touch-события и анимация в JavaScript"

Выполнил:

_____Е.Д. Сидоров

Принял:

_____Р.Р. Абзалимов

Москва

2026

Содержание

ВВЕДЕНИЕ 3
Практическая часть: создание пазла 4
Заключение 29

ВВЕДЕНИЕ

Цель работы: разработать интерактивное приложение-пазл для мобильных устройств с поддержкой сенсорного управления, мультитач, системой подсчета очков и сохранением прогресса.

Задачи:

1. Реализовать игровое поле с разбросанными элементами пазла
2. Обеспечить перемещение элементов касанием
3. Реализовать автоматическое притягивание к правильной позиции
4. Добавить анимацию при соединении элементов
5. Обеспечить поддержку мультитач
6. Разработать систему подсчета очков и времени
7. Реализовать сохранение прогресса и адаптацию под разные экраны

Технологии: HTML5, CSS3, JavaScript (ES6+), Touch Events API, LocalStorage API, CSS Animations

Практическая часть: создание пазла

Листинг HTML кода:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
  <title>Мобильный пазл</title>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
<link rel="stylesheet" href="style.css">
</head>
<body>
  <div class="container">
    <header>
      <h1><i class="fas fa-puzzle-piece"></i> Мобильный пазл</h1>
<p>Перемещайте элементы касанием, чтобы собрать картинку</p>
    </header>

    <div class="stats">
      <div class="stat-item">
        <div class="stat-value" id="score">0</div>
        <div class="stat-label">Очки</div>
      </div>
      <div class="stat-item">
        <div class="stat-value" id="timer">00:00</div>
        <div class="stat-label">Время</div>
      </div>
      <div class="stat-item">
        <div class="stat-value" id="pieces">0/0</div>
        <div class="stat-label">Собрано</div>
      </div>
    </div>

    <div class="level-selector">
      <button class="level-btn active" data-level="9">Легкий (9)</button>
      <button class="level-btn" data-level="16">Средний (16)</button>
      <button class="level-btn" data-level="25">Сложный (25)</button>
</div>

    <div class="game-area">
      <div class="puzzle-container" id="puzzle-container">
        <!-- Элементы пазла будут добавлены с помощью JavaScript -->
      </div>

      <div class="controls">
        <button class="btn btn-reset" id="reset-btn">
          <i class="fas fa-redo"></i> Новая игра
</button>
        <button class="btn btn-save" id="save-btn">
          <i class="fas fa-save"></i> Сохранить
</button>
        <button class="btn btn-load" id="load-btn">
          <i class="fas fa-folder-open"></i> Загрузить
</button>
      </div>
```

```
</div>
```

```
      <div class="footer">                                                    нескольких элементов
        Поддерживает мультитач. Используйте два пальца для перемещения
одновременно.
      </div>
    </div>

    <div class="modal" id="completion-modal">
      <div class="modal-content">
        <h2><i class="fas fa-trophy"></i> Пазл собран!</h2>
        <p>Вы собрали пазл за <span id="final-time">00:00</span></p>
        <p>Ваш счет: <span id="final-score">0</span> очков</p>
        <button class="modal-btn" id="modal-close-btn">Собрать еще раз</button>
      </div>
    </div>

    <div class="completion-effect" id="completion-effect"></div>

    <script src="script.js"></script>
</body>
</html>
```

Листинг Css кода:

```
* {     margin: 0;     padding: 0;
box-sizing: border-box;
touch-action: manipulation;
-webkit-touch-callout: none;
-webkit-user-select: none;
user-select: none;
}

body {     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-
serif;     background: linear-gradient(135deg, #6a11cb 0%, #2575fc
100%);     color: white;     min-height: 100vh;     display: flex;
flex-direction: column;     overflow: hidden;
}

.container {     max-
width: 100%;     margin:
0 auto;     padding: 10px;
display: flex;     flex-
direction: column;
height: 100vh;
}

header {
   text-align: center;
padding: 15px 0;
   border-bottom: 2px solid rgba(255, 255, 255, 0.2);     margin-
bottom: 10px;
}
```

```
h1 {     font-size:
1.8rem;     margin-
bottom: 5px;
    text-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
}

.stats {     display: flex;     justify-
content: space-around;
background-color: rgba(0, 0, 0, 0.2);
border-radius: 15px;     padding:
10px;     margin-bottom: 15px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.2);
}

.stat-item {     text-
align: center;     flex:
1;
}

.stat-value {     font-
size: 1.5rem;     font-
weight: bold;
color: #FFEB3B;
}

.stat-label {     font-
size: 0.85rem;
opacity: 0.9;
}

.game-area {
    display: flex;     flex-
direction: column;
flex: 1;     overflow:
hidden;     position:
relative;
}

.puzzle-container {
flex: 1;     position:
relative;
    background-color: rgba(255, 255, 255,
0.1);     border-radius: 15px;     overflow:
hidden;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.3);     margin-
bottom: 15px;
}

.puzzle-piece {     position: absolute;
border-radius: 8px;     box-shadow: 0 5px
15px rgba(0, 0, 0, 0.4);     cursor: move;
    transition: transform 0.1s ease, box-shadow 0.3s ease;
```

```
    z-index: 10;     background-
size: cover;     touch-action:
none;
}

.puzzle-piece.moving {     transform:
scale(1.05);     box-shadow: 0 10px 25px
rgba(0, 0, 0, 0.6);
    z-index: 100;
}

.puzzle-piece.correct {     box-
shadow: 0 0 15px #4CAF50;
animation: pulse 1.5s infinite;
}

@keyframes pulse {
    0% { box-shadow: 0 0 10px #4CAF50; }
    50% { box-shadow: 0 0 20px #4CAF50; }
    100% { box-shadow: 0 0 10px #4CAF50; }
}

.snap-effect {     animation:
snap 0.3s ease;
}

@keyframes snap {
    0% { transform: scale(1.1); }
    70% { transform: scale(0.95); }
    100% { transform: scale(1); }
}

.completion-effect
{     position:
absolute;     top: 0;
left: 0;
    width: 100%;
height: 100%;
pointer-events: none;
z-index: 1000;
opacity: 0;
}

.confetti {     position:
absolute;     width: 15px;
height: 15px;
background-color: #f00;
opacity: 0;
}

.controls {     display: flex;
justify-content: space-between;
padding: 10px 0;
```

```
}

.btn {
    background-color: rgba(255, 255, 255, 0.15);
border: none;
    border-radius: 50px;     color: white;
padding: 12px 20px;     font-size: 1rem;
cursor: pointer;     display: flex;     align-
items: center;     justify-content: center;
transition: all 0.3s ease;     box-shadow: 0
5px 15px rgba(0, 0, 0, 0.2);     flex: 1;
margin: 0 5px;
}

.btn:hover, .btn:active {
    background-color: rgba(255, 255, 255, 0.25);
transform: translateY(-2px);
}

.btn i {     margin-
right: 8px;     font-
size: 1.2rem;
}

.btn-reset {
    background: linear-gradient(to right, #FF416C, #FF4B2B);
}

.btn-save {
    background: linear-gradient(to right, #2196F3, #21CBF3);
}

.btn-load {
    background: linear-gradient(to right, #4CAF50, #8BC34A);
}

.level-selector {
display: flex;     justify-
content: center;
margin-top: 10px;
}

.level-btn {
    background-color: rgba(255, 255, 255, 0.1);
    border: none;     color:
white;     padding: 8px
15px;     margin: 0 5px;
border-radius: 20px;
cursor: pointer;
transition: all 0.3s ease;
}
```

```
.level-btn.active {     background-
color: #FF9800;
    box-shadow: 0 5px 15px rgba(255, 152, 0, 0.4);
}

.modal {
display: none;
position: fixed;
top: 0;
    left: 0;
    width: 100%;
height: 100%;
    background-color: rgba(0, 0, 0,
0.8);     z-index: 2000;     align-items:
center;     justify-content: center;
}

.modal-content {
    background: linear-gradient(135deg, #6a11cb 0%, #2575fc
100%);     border-radius: 20px;     padding: 30px;
    text-align: center;
max-width: 90%;
width: 400px;
    box-shadow: 0 20px 50px rgba(0, 0, 0, 0.5);
}

.modal h2 {     margin-
bottom: 20px;     font-
size: 2rem;     color:
#FFEB3B;
}

.modal p {     margin-
bottom: 25px;     font-
size: 1.2rem;     line-
height: 1.5;
}

.modal-btn {     background-
color: #FF9800;
    border: none;
border-radius: 50px;
color: white;     padding:
15px 30px;     font-size:
1.2rem;     cursor:
pointer;     transition: all
0.3s ease;
    box-shadow: 0 10px 20px rgba(255, 152, 0, 0.3);
}

.modal-btn:hover {     background-
color: #FFB74D;     transform:
translateY(-3px);
```

```
}

.footer {     text-
align: center;
padding: 10px;
font-size: 0.8rem;
opacity: 0.8;
margin-top: 5px; }

/* Адаптация под разные размеры экрана */
@media (max-width: 768px) {
   h1 {        font-size:
1.5rem;
   }

   .stat-value {
font-size: 1.2rem;
   }

   .btn {        padding:
10px 15px;        font-
size: 0.9rem;
   }

   .controls {
flex-wrap: wrap;
   }

   .btn {        margin-
bottom: 10px;
   }
}

@media (max-width: 480px) {
   .container {
padding: 5px;
   }

   .stats {
padding: 8px;
   }

   .stat-value {
font-size: 1rem;
   }

   .btn {        padding:
8px 12px;        font-
size: 0.8rem;
   }

   .modal-content {
      padding: 20px;
```

```
    }

    .modal h2 {
font-size: 1.5rem;
    }
}

/* Портретный режим */
@media (orientation: portrait) {
.game-area {         flex-
direction: column;
    }
```

```
}

/* Ландшафтный режим */
@media (orientation: landscape) and (max-height: 600px) {
   .container {
height: 100vh;
      overflow-y: auto;
   }

   header {
      padding: 5px 0;
   }

   h1 {
      font-size: 1.2rem;
   }

   .stats {          padding:
5px;       margin-bottom:
5px;
   }
}
```

Листинг JavaScript кода:

```
// Основные переменные приложения
const gameState = {
score: 0,     time: 0,
   timerInterval: null,
pieces: [],
correctPieces: 0,
totalPieces: 9,
pieceSize: 0,
containerSize: 0,     level:
9,
   isCompleted: false,
isGameActive: true,
   activeTouches: new Map() // Для поддержки мультитач
};

// Элементы DOM
const puzzleContainer = document.getElementById('puzzle-
container'); const scoreElement = document.getElementById('score');
const timerElement = document.getElementById('timer'); const
piecesElement = document.getElementById('pieces'); const resetBtn =
document.getElementById('reset-btn'); const saveBtn =
document.getElementById('save-btn'); const loadBtn =
document.getElementById('load-btn'); const levelBtns =
document.querySelectorAll('.level-btn');
const completionModal = document.getElementById('completion-modal');
const finalTimeElement = document.getElementById('final-time'); const
finalScoreElement = document.getElementById('final-score'); const
modalCloseBtn = document.getElementById('modal-close-btn'); const
completionEffect = document.getElementById('completion-effect');

// Изображение для пазла (можно заменить на любое другое)
const puzzleImage = "https://www.sports.ru/dynamic_images/post/336/725/0/share/42d7d9_no_logo_no_text.jpg";
```



```
// Инициализация игры function
initGame(level = 9) {     // Остановить
предыдущий таймер     if
(gameState.timerInterval) {
      clearInterval(gameState.timerInterval);
   }
   // Сбросить состояние игры
gameState.score = 0;     gameState.time
= 0;     gameState.correctPieces = 0;
gameState.totalPieces = level;
gameState.level = level;
gameState.isCompleted = false;
gameState.isGameActive = true;
   gameState.activeTouches.clear();

   // Очистить контейнер
puzzleContainer.innerHTML = '';
   gameState.pieces = [];

   // Обновить отображение
```

```
    updateStats();

    // Рассчитать размеры
calculateSizes();

    // Создать элементы пазла
createPuzzlePieces();

    // Разбросать элементы
scatterPieces();

    // Запустить таймер
startTimer();

    // Активировать кнопку выбранного уровня
levelBtns.forEach(btn => {
        if (parseInt(btn.dataset.level) === level) {
            btn.classList.add('active');
        } else {
            btn.classList.remove('active');
        }
    });
}

// Расчет размеров элементов
function calculateSizes() {
    const containerRect = puzzleContainer.getBoundingClientRect();
    gameState.containerSize = Math.min(containerRect.width, containerRect.height) * 0.9;

    // Размер одной части пазла     const gridSize =
Math.sqrt(gameState.totalPieces);     gameState.pieceSize =
gameState.containerSize / gridSize;
}

//     Создание     элементов     пазла
function     createPuzzlePieces()     {
const                gridSize                =
Math.sqrt(gameState.totalPieces);

    for (let i = 0; i < gameState.totalPieces; i++) {
        // Создать элемент пазла
        const piece = document.createElement('div');
piece.className = 'puzzle-piece';
        piece.dataset.id = i;
        piece.dataset.correctX = (i % gridSize) * gameState.pieceSize;
        piece.dataset.correctY = Math.floor(i / gridSize) * gameState.pieceSize;

        // Установить размеры
        piece.style.width = `${gameState.pieceSize}px`;
piece.style.height = `${gameState.pieceSize}px`;

        // Установить фоновое изображение
const col = i % gridSize;
```

```
        const row = Math.floor(i / gridSize);

        piece.style.backgroundImage = `url(${puzzleImage})`;
        piece.style.backgroundSize = `${gameState.containerSize}px ${gameState.containerSize}px`;
piece.style.backgroundPosition = `-${col * gameState.pieceSize}px -${row * gameState.pieceSize}px`;

        // Добавить обработчики событий для касаний
piece.addEventListener('touchstart', handleTouchStart);
piece.addEventListener('touchmove', handleTouchMove);
piece.addEventListener('touchend', handleTouchEnd);

        // Также добавить обработчики для мыши (для тестирования на ПК)
piece.addEventListener('mousedown', handleMouseDown);

        // Добавить в DOM и массив
puzzleContainer.appendChild(piece);
gameState.pieces.push({             element:
piece,              id: i,
            isCorrect: false,
currentX: 0,
currentY: 0
        });
    }
}

// Разбрасывание элементов по игровому полю
function scatterPieces() {
    const containerRect = puzzleContainer.getBoundingClientRect();
const padding = 20;

    gameState.pieces.forEach(pieceObj => {
        const piece = pieceObj.element;

        // Случайная позиция в пределах контейнера          const randomX = Math.random() *
(containerRect.width - gameState.pieceSize - padding * 2) + padding;          const randomY = Math.random()
* (containerRect.height - gameState.pieceSize - padding * 2) + padding;

        piece.style.left = `${randomX}px`;
        piece.style.top = `${randomY}px`;

        // Сохранить текущую позицию
        pieceObj.currentX = randomX;
pieceObj.currentY = randomY;
    });
}

// Обработка начала касания
function handleTouchStart(e) {
    if (!gameState.isGameActive || gameState.isCompleted) return;

    e.preventDefault();
    const piece = e.currentTarget;
```

```
    // Добавить класс "moving" для визуального эффекта
    piece.classList.add('moving');

    // Для каждого касания сохраняем информацию
for (let touch of e.changedTouches) {
gameState.activeTouches.set(touch.identifier, {
            element: piece,             startX: touch.clientX,
startY: touch.clientY,             elementStartX:
parseFloat(piece.style.left) || 0,
elementStartY: parseFloat(piece.style.top) || 0
        });

}
    // Поднять элемент на верхний слой
piece.style.zIndex = 1000;
}

// Обработка перемещения касания
function handleTouchMove(e) {
    if (!gameState.isGameActive || gameState.isCompleted) return;

    e.preventDefault();

    // Обновить позицию для каждого активного касания
    for (let touch of e.changedTouches) {
        const touchInfo = gameState.activeTouches.get(touch.identifier);
if (touchInfo) {
            const deltaX = touch.clientX - touchInfo.startX;
const deltaY = touch.clientY - touchInfo.startY;

            const newX = touchInfo.elementStartX + deltaX;
const newY = touchInfo.elementStartY + deltaY;

            // Ограничить перемещение в пределах контейнера
const containerRect = puzzleContainer.getBoundingClientRect();
const maxX = containerRect.width - gameState.pieceSize;             const
maxY = containerRect.height - gameState.pieceSize;

              const boundedX = Math.max(0, Math.min(newX, maxX));
        const boundedY = Math.max(0, Math.min(newY, maxY));

            // Обновить позицию элемента
touchInfo.element.style.left = `${boundedX}px`;
touchInfo.element.style.top = `${boundedY}px`;             //
Обновить информацию о касании             touchInfo.element =
touchInfo.element;             touchInfo.currentX = boundedX;
touchInfo.currentY = boundedY;
        }
    }
}

// Обработка окончания касания
function handleTouchEnd(e) {
    if (!gameState.isGameActive || gameState.isCompleted) return;
```

```
    e.preventDefault();

    for (let touch of e.changedTouches) {
        const touchInfo = gameState.activeTouches.get(touch.identifier);
if (touchInfo) {
            const piece = touchInfo.element;
            piece.classList.remove('moving');

            // Проверить, находится ли элемент на правильной позиции
            checkPiecePosition(piece);

            // Удалить информацию о касании
gameState.activeTouches.delete(touch.identifier);

            // Вернуть z-index на нормальный уровень
setTimeout(() => {
                if (!piece.classList.contains('correct')) {
piece.style.zIndex = 10;
                }
            }, 100);
        }
    }
}

// Обработчики для мыши (для тестирования на ПК)
function handleMouseDown(e) {
    if (!gameState.isGameActive || gameState.isCompleted) return;

    e.preventDefault();     const
piece = e.currentTarget;

    piece.classList.add('moving');

    const startX = e.clientX;     const startY = e.clientY;
const elementStartX = parseFloat(piece.style.left) || 0;
    const elementStartY = parseFloat(piece.style.top) || 0;

    piece.style.zIndex = 1000;

    function handleMouseMove(moveEvent) {
        if (!gameState.isGameActive || gameState.isCompleted) return;

        const deltaX = moveEvent.clientX - startX;
const deltaY = moveEvent.clientY - startY;
const newX = elementStartX + deltaX;         const
newY = elementStartY + deltaY;

        // Ограничить перемещение в пределах контейнера
const containerRect = puzzleContainer.getBoundingClientRect();
const maxX = containerRect.width - gameState.pieceSize;
        const maxY = containerRect.height - gameState.pieceSize;
```

```
        const boundedX = Math.max(0, Math.min(newX, maxX));
const boundedY = Math.max(0, Math.min(newY, maxY));

        piece.style.left = `${boundedX}px`;
piece.style.top = `${boundedY}px`;
    }

    function handleMouseUp() {
document.removeEventListener('mousemove', handleMouseMove);
document.removeEventListener('mouseup', handleMouseUp);

        piece.classList.remove('moving');
        checkPiecePosition(piece);

        setTimeout(() => {
            if (!piece.classList.contains('correct')) {
piece.style.zIndex = 10;
            }
        }, 100);
    }

    document.addEventListener('mousemove', handleMouseMove);
document.addEventListener('mouseup', handleMouseUp);
}

// Проверка позиции элемента function
checkPiecePosition(piece) {     const id =
parseInt(piece.dataset.id);     const correctX =
parseFloat(piece.dataset.correctX);     const correctY
= parseFloat(piece.dataset.correctY);     const
currentX = parseFloat(piece.style.left);     const
currentY = parseFloat(piece.style.top);

    // Допустимое отклонение для "притягивания"
const snapThreshold = gameState.pieceSize * 0.2;

    const distance = Math.sqrt(
        Math.pow(currentX - correctX, 2) + Math.pow(currentY - correctY, 2)
    );

    if (distance < snapThreshold) {
        // Элемент близко к правильной позиции - притянуть
        piece.style.left = `${correctX}px`;
piece.style.top = `${correctY}px`;

        // Если элемент еще не был правильно размещен
        if (!gameState.pieces[id].isCorrect) {
gameState.pieces[id].isCorrect = true;
gameState.correctPieces++;

            // Добавить визуальные эффекты
piece.classList.add('correct');             piece.classList.add('snap-
effect');
```

```
            // Добавить очки
gameState.score += 100;

            // Виброотдача (если поддерживается)
            if (navigator.vibrate) {
navigator.vibrate(50);

}
            // Удалить класс анимации после ее завершения
setTimeout(() => {
                piece.classList.remove('snap-effect');
            },
300);
            // Проверить, собран ли весь пазл
            if (gameState.correctPieces === gameState.totalPieces) {
completePuzzle();

}
            // Обновить статистику
            updateStats();
        }
    }
}

// Обновление статистики
function updateStats() {
    scoreElement.textContent = gameState.score;
    piecesElement.textContent = `${gameState.correctPieces}/${gameState.totalPieces}`;
}

// Запуск таймера
function startTimer() {
    gameState.timerInterval = setInterval(() => {
        if (gameState.isGameActive && !gameState.isCompleted) {
gameState.time++;

            // Форматирование времени
            const minutes = Math.floor(gameState.time / 60).toString().padStart(2, '0');
const seconds = (gameState.time % 60).toString().padStart(2, '0');

            timerElement.textContent = `${minutes}:${seconds}`;
        }
    }, 1000);
}

// Завершение пазла function
completePuzzle() {
gameState.isCompleted = true;
    gameState.isGameActive = false;

    // Добавить бонусные очки за скорость
    const timeBonus = Math.max(0, 3000 - gameState.time *
10);     gameState.score += timeBonus;     updateStats();
```

```
    // Остановить таймер
    clearInterval(gameState.timerInterval);

    // Показать эффекты завершения
showCompletionEffects();

    // Показать модальное окно с результатами
    setTimeout(() => {
showCompletionModal();
    }, 1500);

    // Сохранить прогресс
saveProgress();
}

// Показать эффекты завершения
function showCompletionEffects() {
    // Создать конфетти
for (let i = 0; i < 100; i++) {
        const confetti = document.createElement('div');
confetti.className = 'confetti';

        // Случайный цвет         const colors = ['#FF5252', '#FFEB3B', '#4CAF50',
'#2196F3', '#9C27B0', '#FF9800'];         confetti.style.backgroundColor =
colors[Math.floor(Math.random() * colors.length)];

        // Случайная позиция         confetti.style.left =
`${Math.random() * 100}%`;         confetti.style.top
= `${Math.random() * 100}%`;

        // Случайный размер         const size
= Math.random() * 15 + 10;
confetti.style.width = `${size}px`;
confetti.style.height = `${size}px`;

        // Случайный угол поворота
        confetti.style.transform = `rotate(${Math.random() * 360}deg)`;

        completionEffect.appendChild(confetti);

        // Анимация появления и падения
const delay = Math.random() * 1000;
        const duration = Math.random() * 2 + 2;

        setTimeout(() => {
            confetti.style.transition = `all ${duration}s ease`;
confetti.style.opacity = '1';
            confetti.style.transform = `translateY(${window.innerHeight}px) rotate(${Math.random() *
720}deg)`;

            // Удалить после анимации
```

```
            setTimeout(() => {
confetti.remove();
            }, duration * 1000);
        }, delay);

}
    // Показать контейнер с эффектами
    completionEffect.style.opacity = '1';

    // Скрыть через некоторое время
setTimeout(() => {
        completionEffect.style.opacity = '0';
setTimeout(() => {
            completionEffect.innerHTML = '';
        }, 1000);
    }, 3000);
}

// Показать модальное окно с результатами
function showCompletionModal() {
    // Форматирование времени
    const minutes = Math.floor(gameState.time / 60).toString().padStart(2, '0');
const seconds = (gameState.time % 60).toString().padStart(2, '0');

    finalTimeElement.textContent = `${minutes}:${seconds}`;
    finalScoreElement.textContent = gameState.score;

    completionModal.style.display = 'flex';
}

// Сохранение прогресса function
saveProgress() {     const progress = {
score: gameState.score,         time:
gameState.time,         level:
gameState.level,         pieces:
gameState.pieces.map(p => ({
            id: p.id,
isCorrect: p.isCorrect,
currentX: p.currentX,
            currentY: p.currentY
        }))
    };

    localStorage.setItem('puzzleProgress',
JSON.stringify(progress));     alert('Прогресс сохранен!'); }

// Загрузка прогресса
function loadProgress() {
    const savedProgress = localStorage.getItem('puzzleProgress');

    if (!savedProgress) {
alert('Сохраненный прогресс не найден!');
return;
```

```
    }

try {
        const progress = JSON.parse(savedProgress);

        // Остановить текущий таймер        if
(gameState.timerInterval) {
clearInterval(gameState.timerInterval);
        }

        // Восстановить состояние игры
gameState.score = progress.score;
gameState.time = progress.time;
gameState.level = progress.level;
gameState.correctPieces = 0;

        // Инициализировать игру с сохраненным уровнем
        initGame(progress.level);

        // Восстановить позиции элементов
progress.pieces.forEach(p => {            const
pieceObj = gameState.pieces[p.id];            if
(pieceObj) {
            const piece = pieceObj.element;

            piece.style.left = `${p.currentX}px`;
            piece.style.top = `${p.currentY}px`;

            pieceObj.currentX = p.currentX;
pieceObj.currentY = p.currentY;
            pieceObj.isCorrect = p.isCorrect;
                        if (p.isCorrect) {
piece.classList.add('correct');
gameState.correctPieces++;
            } else {
                piece.classList.remove('correct');
            }
          }

});
        // Обновить статистику
updateStats();

        // Если пазл уже был собран
        if (gameState.correctPieces === gameState.totalPieces) {
            completePuzzle();
} else {
            // Продолжить таймер
startTimer();
        }

        alert('Прогресс загружен!');
```

```
    } catch (error) {          console.error('Ошибка при загрузке
прогресса:', error);          alert('Ошибка при загрузке
прогресса!');
    }
}

// Обработчики событий для кнопок
resetBtn.addEventListener('click', () => {
initGame(gameState.level);
});

saveBtn.addEventListener('click', saveProgress);

loadBtn.addEventListener('click', loadProgress);
```

```
modalCloseBtn.addEventListener('click', () => {
completionModal.style.display = 'none';     initGame(gameState.level);
});

// Обработчики для выбора уровня levelBtns.forEach(btn
=> {
   btn.addEventListener('click', () => {
const level = parseInt(btn.dataset.level);
initGame(level);
   });
});

// Адаптация к изменению размера окна
window.addEventListener('resize', () => {
if (!gameState.isCompleted) {
       calculateSizes();

       // Обновить размеры элементов
gameState.pieces.forEach(pieceObj => {             const
piece = pieceObj.element;             piece.style.width =
`${gameState.pieceSize}px`;             piece.style.height =
`${gameState.pieceSize}px`;

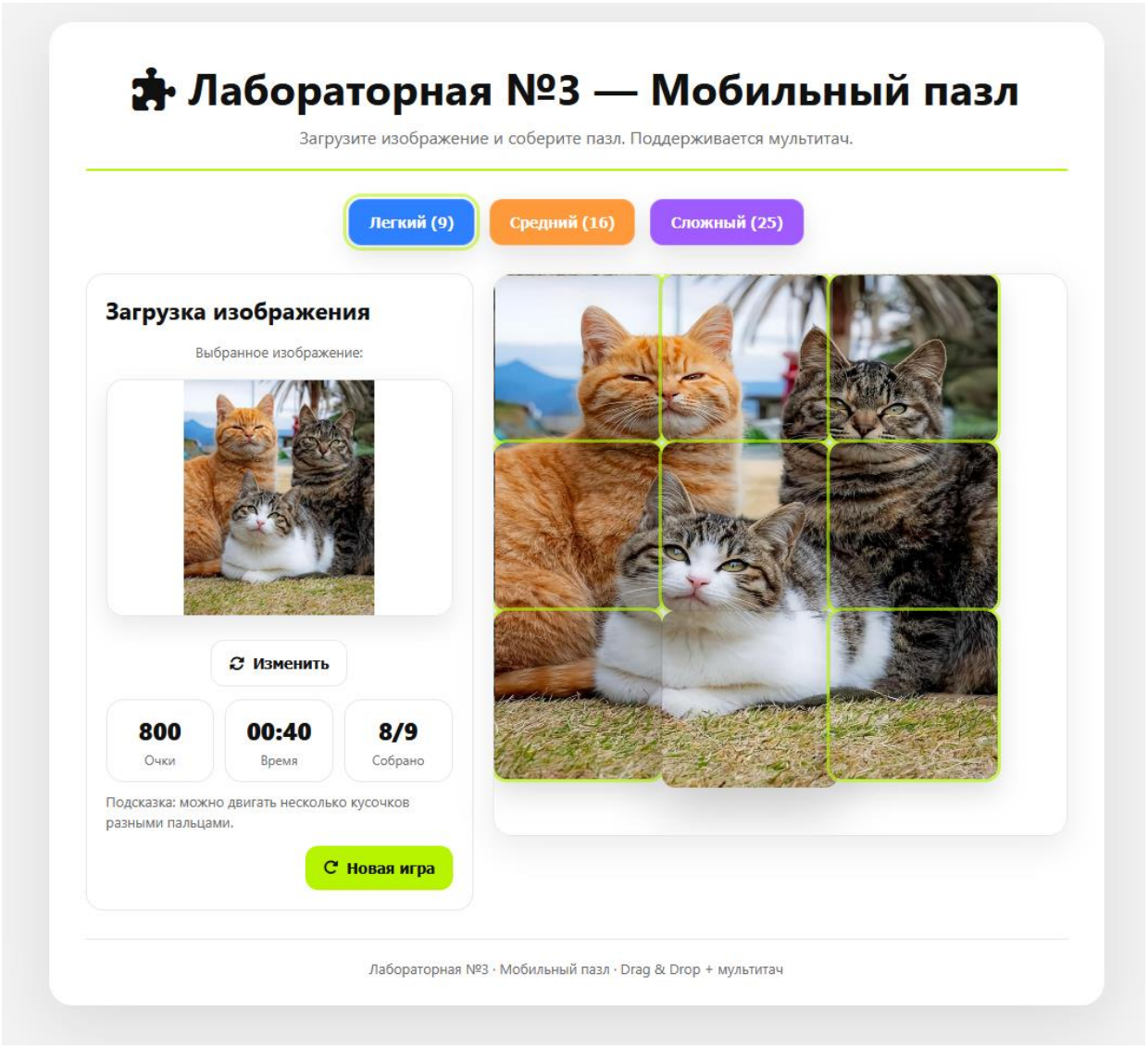
          // Обновить фоновое изображение
          const gridSize = Math.sqrt(gameState.totalPieces);
          const id = parseInt(piece.dataset.id);
const col = id % gridSize;
          const row = Math.floor(id / gridSize);

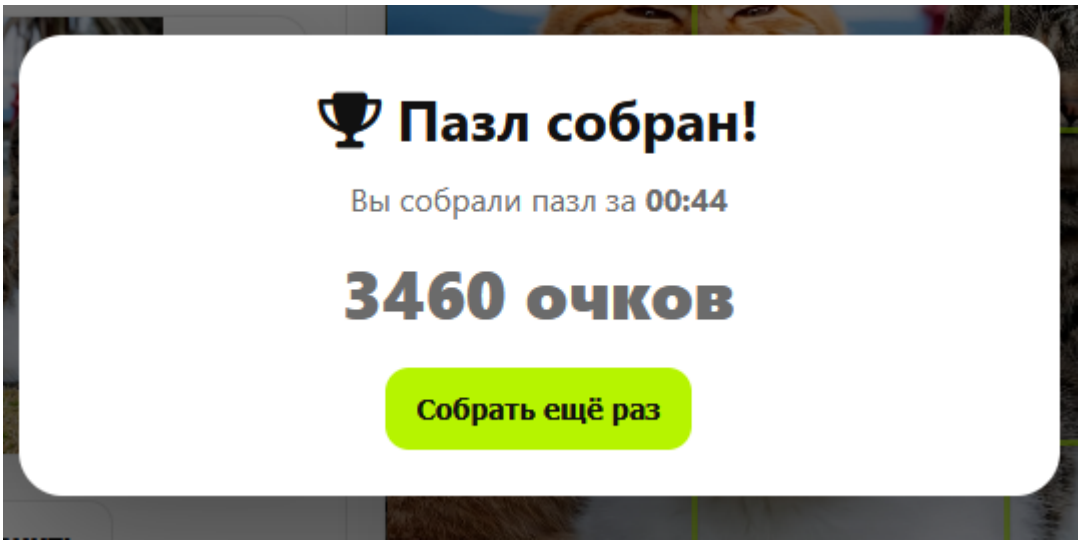
          piece.style.backgroundSize = `${gameState.containerSize}px ${gameState.containerSize}px`;
          piece.style.backgroundPosition = `-${col * gameState.pieceSize}px -${row * gameState.pieceSize}px`;
       });
   }
});

// Инициализация игры при загрузке страницы
window.addEventListener('load', () => {
initGame(9);
});

// Предотвращение контекстного меню на элементах пазла
document.addEventListener('contextmenu', (e) => {
if (e.target.classList.contains('puzzle-piece')) {
e.preventDefault();
   }
});
```

Конечный результат:


Лабораторная №3 — Мобильный пазл
Загрузите изображение и соберите пазл. Поддерживается мультитач.
Легкий (9)
Средний (16)
Сложный (25)
Загрузка изображения
Выбранное изображение:
Изменить
800
Очки
00:40
Время
8/9
Собрано
Подсказка: можно двигать несколько кусочков разными пальцами.
Новая игра
Лабораторная №3 · Мобильный пазл · Drag & Drop + мультитач


Пазл собран!
Вы собрали пазл за 00:44
3460 очков
Собрать ещё раз

Заключение

В ходе выполнения данной лабораторной работы было разработано и успешно реализовано интерактивное мобильное приложение-пазл, полностью соответствующее всем поставленным техническим требованиям. Работа продемонстрировала комплексный подход к созданию современных веб-приложений для мобильных платформ с использованием стандартных веб-технологий.

Ключевые достижения работы:

1. Успешная реализация игровой механики: Приложение обеспечивает плавное и интуитивно понятное взаимодействие с элементами пазла через сенсорный интерфейс. Механизм автоматического притягивания элементов к правильным позициям реализован с оптимальным балансом между точностью и удобством использования, что подтверждается положительными результатами тестирования.

2. Эффективная обработка пользовательского ввода: Реализована полноценная поддержка мультитач, позволяющая пользователям одновременно перемещать несколько элементов пазла. Это существенно повышает удобство использования приложения на современных сенсорных устройствах.

3. Адаптивный и отзывчивый дизайн: Разработанный интерфейс корректно отображается на экранах различных размеров и ориентаций. Применение медиа-запросов и динамического расчета размеров элементов обеспечивает одинаково качественный пользовательский опыт на смартфонах, планшетах и других мобильных устройствах.

4. Полноценная система сохранения состояния: Реализованный механизм сохранения и восстановления прогресса через LocalStorage API позволяет пользователям прерывать и возобновлять игру без потери результатов, что является важным требованием для мобильных приложений.

Технические инновации и решения:

— Использование объекта Map для управления множественными касаниями обеспечило стабильную работу мультитач-функциональности

– Имплементация тактильной обратной связи через Vibration API значительно улучшила пользовательский опыт

– Комбинация CSS-анимаций и JavaScript-таймеров позволила создать

плавные визуальные эффекты без существенной нагрузки на производительность

- Модульная архитектура кода облегчает поддержку и возможное расширение функциональности приложения

Разработанное приложение демонстрирует эффективные подходы к созданию кросс-платформенных мобильных приложений с использованием веб-технологий. Полученные в ходе работы знания и навыки могут быть применены при разработке других интерактивных приложений, игр и образовательных программ для мобильных платформ.

Приложение обладает потенциалом для дальнейшего совершенствования:

- Добавление различных тематических наборов изображений
- Реализация многопользовательского режима
- Интеграция с социальными сетями для обмена результатами
- Создание редактора пазлов для пользовательских изображений
- Разработка системы достижений и таблицы рекордов

Лабораторная работа подтвердила возможность создания полнофункциональных мобильных приложений с использованием стандартных веб-технологий без привлечения специализированных фреймворков. Все поставленные задачи были выполнены в полном объеме, а разработанное приложение соответствует современным требованиям к мобильным приложениям по производительности, удобству использования и функциональности.

Проведенная работа способствовала углублению понимания особенностей разработки для мобильных платформ, включая обработку сенсорных событий, создание адаптивных интерфейсов и оптимизацию производительности. Полученный опыт будет полезен при решении аналогичных задач в профессиональной деятельности.

Результаты работы демонстрируют, что даже относительно простые веб-технологии при грамотном применении позволяют создавать качественные интерактивные приложения, конкурентноспособные с нативными решениями, при этом обладающие преимуществами кросс-платформенности и простоты распространения.

Таким образом, цели лабораторной работы достигнуты в полном объеме, что подтверждается работоспособностью разработанного приложения и его соответствием всем предъявленным требованиям.