

Разработка интерактивного веб-приложения «Лопай шарики» с использованием HTML, CSS и JavaScript

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Статья посвящена разработке и практической реализации интерактивного веб-приложения «Лопай шарики» — аркадной головоломки, в которой пользователь собирает группы из двух и более одинаковых шариков для их удаления с игрового поля. В работе подробно описаны архитектура приложения, используемые технологии (HTML5, CSS3, JavaScript) и основные функциональные возможности: динамическое появление шариков сверху, плавная гравитационная анимация их падения, система уровней с автоматическим повышением сложности, визуальные эффекты при лопании шариков, светлая и тёмная цветовые схемы, функция подсказок, пауза с остановкой таймера, а также система рекордов с сохранением в localStorage. Приложение создаёт увлекательный игровой опыт и может быть использовано как образовательный инструмент для изучения основ клиентской веб-разработки или как развлекательный контент для развития реакции и стратегического мышления.

Ключевые слова: HTML, CSS, JavaScript, веб-приложение, аркада, интерактивность, анимация, рекорды, темы.

Implementation of an interactive web application «Pop the Bubbles» using HTML, CSS and JavaScript

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article is devoted to the development and practical implementation of an interactive web application «Pop the Bubbles» — an arcade puzzle where the user removes groups of two or more identical bubbles from the game board. The paper details the application architecture, technologies used (HTML5, CSS3, JavaScript), and key functional features: dynamic bubble spawning from the top, smooth gravity-based falling animation, an adaptive level system with increasing difficulty, visual particle effects upon bubble popping, support for dark and light themes, hint functionality, pause with timer suspension, and a leaderboard system using localStorage. The application delivers an engaging gameplay experience and can

serve as an educational tool for learning client-side web development fundamentals or as entertainment to enhance reaction time and strategic thinking.

Keywords: HTML, CSS, JavaScript, web application, arcade, interactivity, animation, leaderboard, themes.

1 Введение

1.1 Актуальность

В условиях цифровизации досуга всё большую популярность приобретают лёгкие, но увлекательные интерактивные приложения, сочетающие простоту управления с визуальной выразительностью. Аркадные головоломки с элементами физики и стратегии остаются востребованными благодаря своей доступности и способности развивать когнитивные навыки. Создание таких игр с использованием нативных веб-технологий (HTML, CSS, JavaScript) без сторонних библиотек подчёркивает возможности современного браузерного окружения и служит эффективным обучающим инструментом.

Актуальность данной работы обусловлена необходимостью в легковесных, кроссплатформенных играх с продуманной механикой и качественным пользовательским интерфейсом. Приложение «Лопай шарики» отвечает этим требованиям, предлагая динамичный геймплей, настраиваемую визуальную тему и систему мотивации через рекорды.

1.2 Обзор исследований

И. В. Смирнов рассмотрел применение HTML5 и CSS3 для создания анимированных веб-интерфейсов, подчеркнув важность использования ключевых кадров для реализации сложных анимаций [1]. А. С. Петрова исследовала методы перетаскивания и обработки указателей в JavaScript, что положено в основу интерактивности многих игр [2]. Д. А. Кузнецов представил программную реализацию системы управления мультимедийным контентом в браузере, включая воспроизведение аудио и управление громкостью с помощью элементов управления, что является важным аспектом для создания атмосферных приложений [3]. Е. М. Григорьева и М. П. Лебедев в своей работе описали подходы к созданию адаптивных веб-приложений с использованием Flexbox и Grid, включая панели управления и игровые поля [4]. П. Н. Тихонов подчеркнул важность оптимизации анимаций с помощью requestAnimationFrame для обеспечения плавности даже при большом количестве объектов [5].

1.3 Цель исследования

Целью исследования является разработка полнофункционального интерактивного веб-приложения «Лопай шарики» с использованием HTML, CSS и JavaScript.

2 Материалы и методы

Для разработки приложения использовалась интегрированная среда Visual Studio Code, поддерживающая подсветку синтаксиса и отладку для веб-

технологий. Архитектура приложения реализована в соответствии с принципом разделения ответственности:

- HTML5: определяет структуру игровых экранов: приветственное окно, игровое поле, экран завершения игры и таблицу рекордов.
- CSS3: отвечает за стилизацию, адаптивность, анимации (падение, взрыв частиц), а также переключение между светлой и тёмной темами с использованием CSS-переменных и классов.
- JavaScript: реализует всю игровую логику: спавн шариков, обработку кликов, поиск групп, гравитацию, систему уровней, таймер, паузу, подсказки и сохранение результатов в localStorage.

Визуальное оформление выполнено без внешних изображений — шарики представлены через цветные блоки с `border-radius: 50%`, что обеспечивает кроссплатформенную совместимость и высокую производительность.

3 Результаты

Разработанное интерактивное веб-приложение «Лопай шарики» представляет собой полноценную браузерную аркаду, реализованную исключительно на нативных веб-технологиях (HTML5, CSS3, JavaScript) без использования внешних библиотек. Игра функционирует автономно: после загрузки HTML-файла не требует подключения к интернету и корректно отображается на устройствах с различными размерами экранов.

3.1 Архитектура приложения

Приложение состоит из четырёх основных экранов, каждый из которых центрирован с помощью Flexbox и ограничен максимальной шириной 600 px для обеспечения удобства на мобильных устройствах.

Приветственное окно приложения (рис. 1) содержит название игры, краткое описание механики (сбор групп из ≥ 2 шариков, красная линия как условие проигрыша, бонус за полную очистку поля), а также кнопки «Играть», «Таблица рекордов» и переключатель темы.

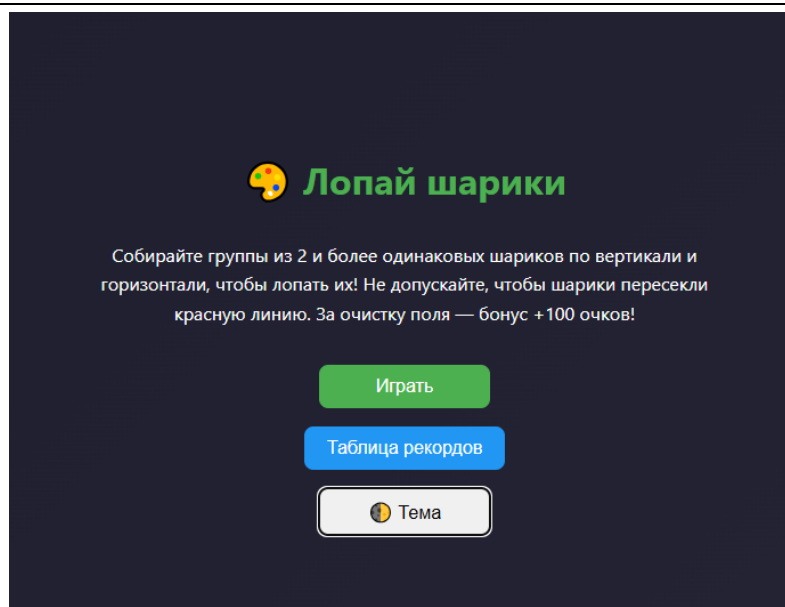


Рисунок 1 – Приветственное окно приложения

Игровой экран (рис. 2) включает информационную панель с уровнем, текущим счётом, рекордом и таймером, а также игровое поле размером 12×15 с динамически генерируемыми шариками. Красная «опасная линия» расположена на 4-й строке сверху ($\text{DANGER_ROW} = 3$), пересечение которой завершает игру.

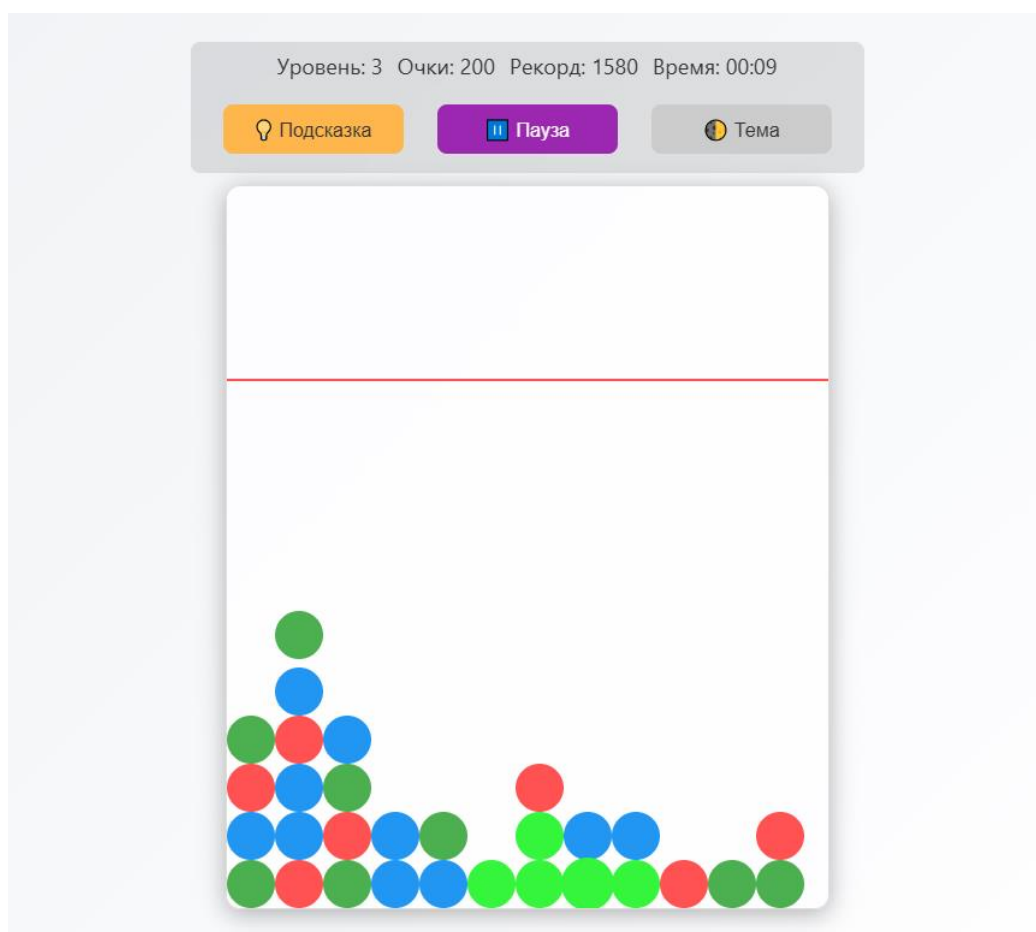


Рисунок 2 – Игровой экран приложения со светлой темой

При клике на шарик, входящий в группу из двух и более одинаковых, запускается анимация лопанья с разлётом цветных частиц (рис. 3). Эффект создаётся с помощью `requestAnimationFrame` и CSS-переходов, что придаёт динамику и визуальное удовлетворение от взаимодействия.

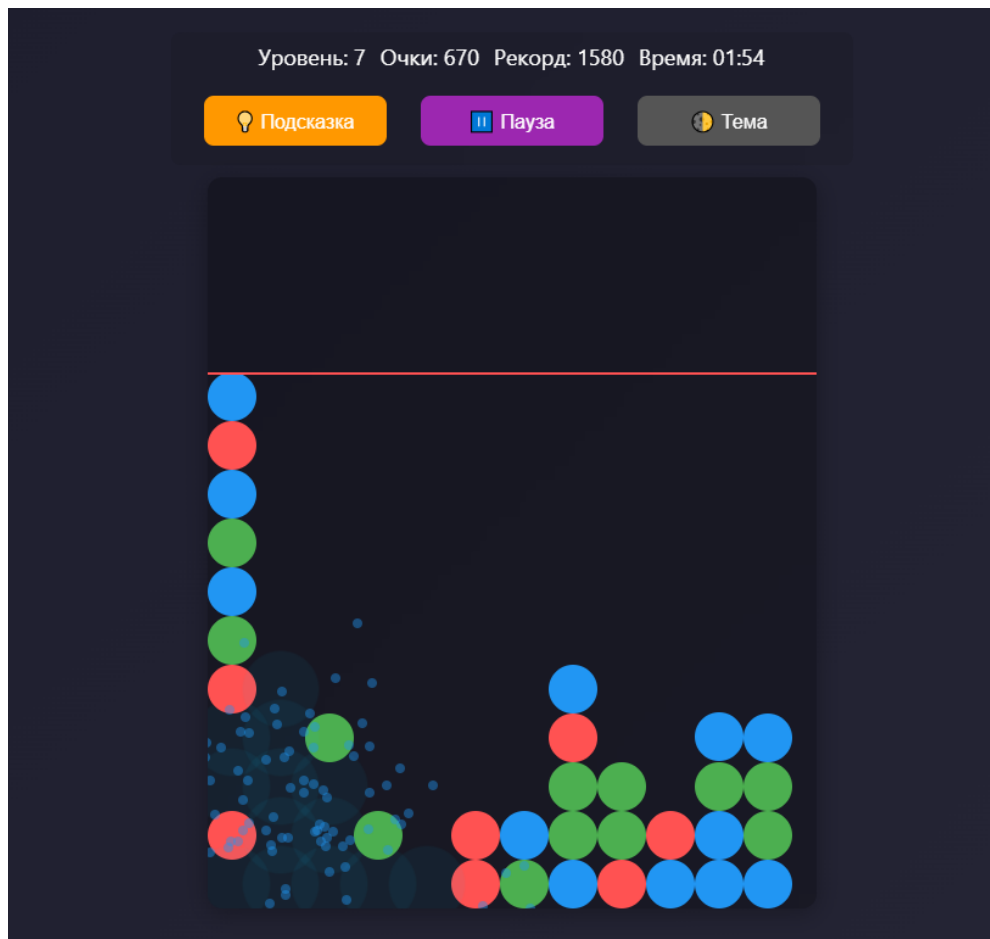


Рисунок 3 – Игровой экран приложения с темной темой и анимации лопанья

Экран завершения игры (рис. 4) отображает итоговые данные: уровень, очки и время. Если игрок установил новый рекорд (1 место) — появляются поле ввода имени и золотое уведомление 🏆.

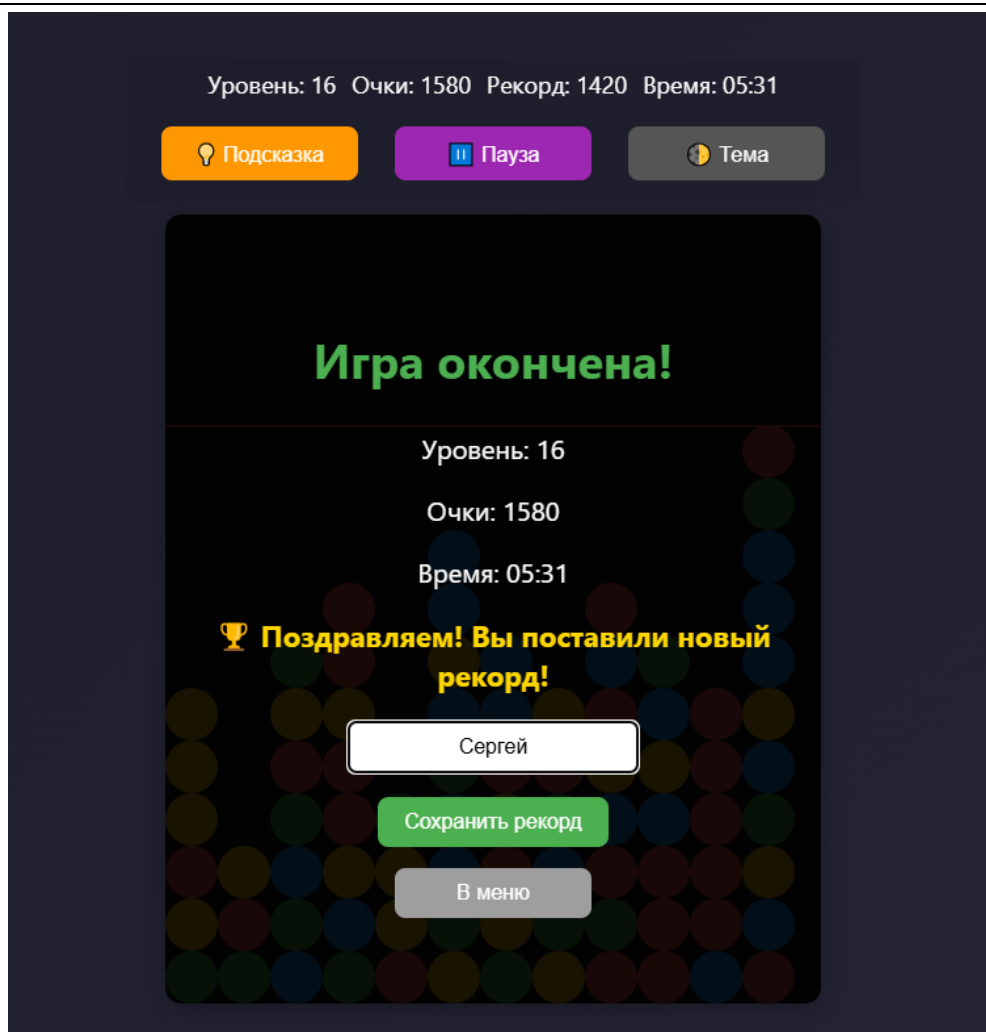


Рисунок 4 – Экран результата после прохождения уровня

«Таблица рекордов (Топ 10)» (рис. 5) поддерживает обе цветовые темы и отображает до 10 лучших результатов с указанием имени, количества очков и времени прохождения.

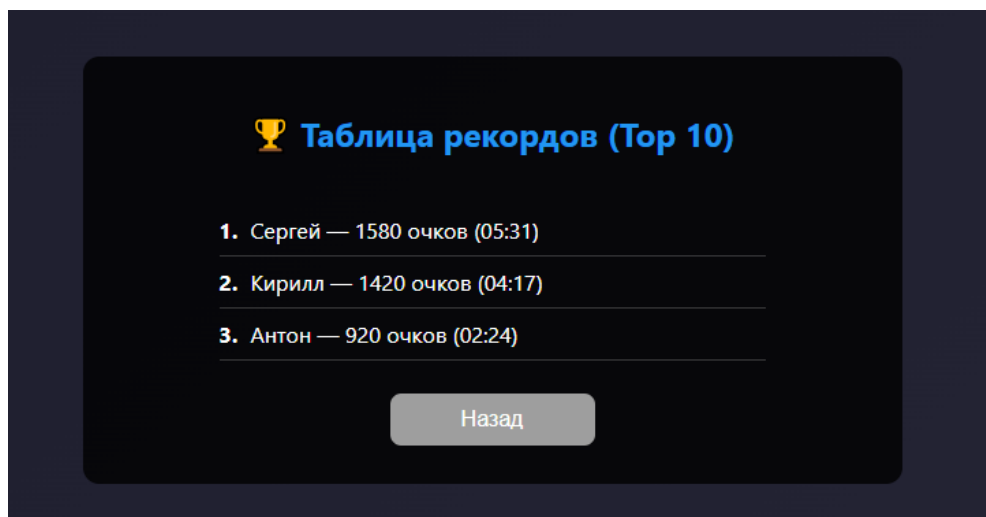


Рисунок 5 – Таблица рекордов

3.2 Правила игры и система начисления очков

Основная цель игры — собирать группы из двух и более одинаковых шариков, расположенных в соседних ячейках по вертикали или горизонтали. По клику на любой шарик, входящий в такую группу, все её элементы мгновенно «лопаются» с визуальной анимацией и начислением очков.

Критическое условие проигрыша — любой шарик пересекает красную линию, расположенную на 4-й строке сверху (рис. 2). Это обеспечивает чёткую и предсказуемую механику поражения: игра заканчивается уже при пересечении линии одним шариком.

Очки начисляются по 10 за каждый удалённый шарик, а при полной очистке игрового поля — бонус +100 очков:

```
score += group.length * POINTS_PER_BUBBLE; // 10 очков за шарик
// проверка и зачисление на бонус за полную очистку
setTimeout(() => {
  group.forEach(b => {
    if (b.cellEl.parentNode === gameEl) gameEl.removeChild(b.cellEl);
    grid[b.row][b.col] = null;});
  let isEmpty = true;
  for (let r = 0; r < ROWS; r++) {
    for (let c = 0; c < COLUMNS; c++) {
      if (grid[r][c] !== null) {
        isEmpty = false;
        break;}}
    if (!isEmpty) break;}
  if (isEmpty) {score += 100;}
```

Уровень повышается автоматически при наборе каждые 100 очков:

```
function updateUI() {
  scoreEl.textContent = score;
  const newLevel = Math.floor(score / POINTS_PER_LEVEL) + 1;
  if (newLevel > level) {
    level = newLevel;
    levelEl.textContent = level;}}
```

С ростом уровня увеличивается сложность:

- 1) Уровни 1–9: 3 цвета;
- 2) Уровни 10–24: 4 цвета;
- 3) С 25-го уровня: 5 цветов.

```
const COLORS = ['#FF5252', '#4CAF50', '#2196F3', '#FFC107', '#9C27B0'];
// Получить количество цветов по уровню
function getAvailableColors() {
  if (level <= 9) return COLORS.slice(0, 3);
  if (level <= 24) return COLORS.slice(0, 4);
  return COLORS;}
```

Одновременно ускоряется появление новых шариков:

```
clearInterval(spawnTimer);
spawnTimer = setInterval(spawnBubble, Math.max(300, SPAWN_INTERVAL -
(level - 1) * 50));
```

3.3 Основные функции и их реализация

Инициализация игры и сетки:

```
function startGame() {
    gameEl.innerHTML = '';
    const dangerLine = document.createElement('div');
    dangerLine.className = 'danger-line';
    dangerLine.style.top = (DANGER_ROW + 1) * CELL_SIZE + 'px';
    gameEl.appendChild(dangerLine);
    gameEl.appendChild(gameOverEl);
    initGrid();
    score = 0; level = 1; isGameOver = false;
    isAnimating = false; isPaused = false; pausedTime = 0;
    updateUI();
    clearInterval(spawnTimer);
    spawnTimer = setInterval(spawnBubble, Math.max(300, SPAWN_INTERVAL -
(level - 1) * 50));
    startTimer();}
```

Появление и падение шариков:

```
function animateFall(bubbleObj) {
    let row = 0;
    while (row < ROWS - 1 && grid[row + 1][bubbleObj.col] === null) {
        row++;
    }
    if (row <= DANGER_ROW) {
        endGame();
        return;
    }
    grid[bubbleObj.row][bubbleObj.col] = null;
    bubbleObj.row = row;
    grid[row][bubbleObj.col] = bubbleObj;
    const targetTop = row * CELL_SIZE;
    bubbleObj.cellEl.style.transition = 'top 1.0s cubic-bezier(0.2, 1.0,
0.4, 1.0)';
    bubbleObj.cellEl.style.top = targetTop + 'px';}
```

Поиск и лопание групп шариков:

```
function findGroup(start) {
    const visited = new Set();
    const group = [];
    const stack = [start];
    const targetColor = start.color;
    while (stack.length > 0) {
        const current = stack.pop();
        const key = `${current.row},${current.col}`;
        if (visited.has(key)) continue;
        visited.add(key);
        group.push(current);
        const dirs = [[-1,0], [1,0], [0,-1], [0,1]];
        for (const [dr, dc] of dirs) {
            const nr = current.row + dr;
            const nc = current.col + dc;
            if (nr >= 0 && nr < ROWS && nc >= 0 && nc < COLUMNS) {
                const nb = grid[nr][nc];
                if (nb && !visited.has(`${nr},${nc}`) && nb.color === targetColor)
            {
                stack.push(nb);
            }
        }
    }
    return group.length >= MIN_GROUP_SIZE ? group : [];
```

Создание частиц при лопанье шариков:

```
function createParticles(x, y, color) {
  for (let i = 0; i < 8; i++) {
    const particle = document.createElement('div');
    particle.className = 'particle';
    particle.style.backgroundColor = color;
    particle.style.left = x + 'px';
    particle.style.top = y + 'px';
    const angle = Math.random() * Math.PI * 2;
    const speed = 2 + Math.random() * 3;
    const dx = Math.cos(angle) * speed;
    const dy = Math.sin(angle) * speed;
    gameEl.appendChild(particle);
    let posX = x;
    let posY = y;
    let life = 30;
    const move = () => {
      life--;
      if (life <= 0) {
        particle.remove();
        return;
      }
      posX += dx;
      posY += dy;
      particle.style.left = posX + 'px';
      particle.style.top = posY + 'px';
      particle.style.opacity = life / 30;
      requestAnimationFrame(move);
    };
    requestAnimationFrame(move);
  }
}
```

Функциональность кнопки «Пауза»:

```
pauseBtn.addEventListener('click', () => {
  isPaused = !isPaused;
  if (isPaused) {
    clearInterval(spawnTimer);
    pausedTime = Date.now() - gameStartTime;
    pauseBtn.textContent = '▶ Продолжить';
    pauseBtn.style.background = '#4CAF50';
  } else {
    spawnTimer = setInterval(spawnBubble, Math.max(300, SPAWN_INTERVAL
- (level - 1) * 50));
    startTimer();
    pauseBtn.textContent = '⏸ Пауза';
    pauseBtn.style.background = '#9c27b0';
  }
});
```

Функциональность кнопки «Подсказка» — сканирует всё поле, находит все валидные группы, выбирает случайную и отображает пунктирную рамку на 2 секунды (рис. 6).:

```
function showHint() {
  if (isAnimating || isGameOver || isPaused) return;
  const allGroups = [];
  for (let row = 0; row < ROWS; row++) {
    for (let col = 0; col < COLUMNS; col++) {
      const bubble = grid[row][col];
```

```

        if (bubble) {
            const group = findGroup(bubble);
            if (group.length >= MIN_GROUP_SIZE) {
                allGroups.push(group);}}}}
const oldHints = document.querySelectorAll('.hint');
oldHints.forEach(h => h.remove());
if (allGroups.length > 0) {
    const randomGroup = allGroups[Math.floor(Math.random() *
allGroups.length)];
    const bubble = randomGroup[0];
    const hint = document.createElement('div');
    hint.className = 'hint';
    hint.style.left = bubble.col * CELL_SIZE + 'px';
    hint.style.top = bubble.row * CELL_SIZE + 'px';
    gameEl.appendChild(hint);
    setTimeout(() => {
        hint.remove();
    }, 2000);}}}

```

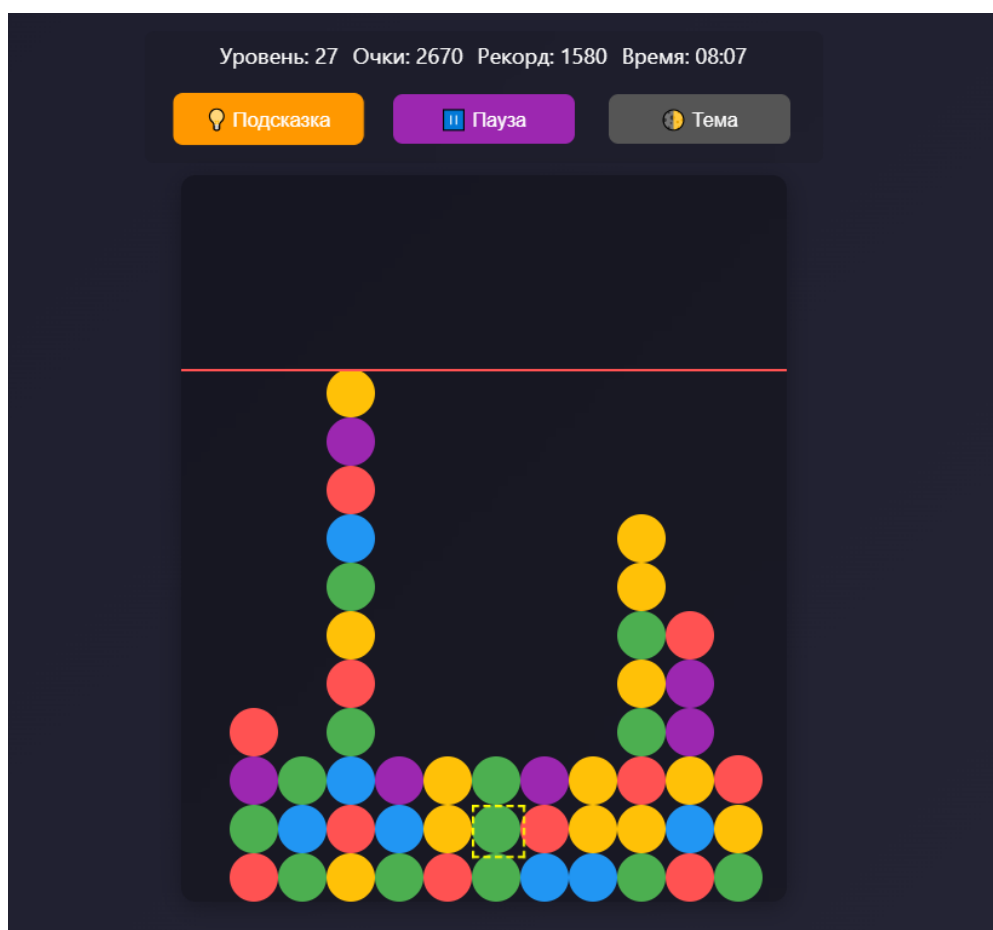


Рисунок 6 – Применение функции «Подсказка»

Рекорд игры сохраняется только при превышении 1-го места:

```

function loadLeaderboard() {
    const records =
JSON.parse(localStorage.getItem('bubblePopLeaderboard') || '[]');
    recordsList.innerHTML = '';
    records.forEach((record, i) => {
        const li = document.createElement('li');

```

```

        li.textContent = `${record.name} - ${record.score} очков  

        (${record.time})`;
        recordsList.appendChild(li);});
        updateHighScoreDisplay();}
        function getTopRecordScore() {
            const records =
JSON.parse(localStorage.getItem('bubblePopLeaderboard') || '[]');
            return records.length > 0 ? records[0].score : 0;}
        function updateHighScoreDisplay() {
            highScoreEl.textContent = getTopRecordScore();}
        function saveToLeaderboard(name, score, timeStr) {
            const records =
JSON.parse(localStorage.getItem('bubblePopLeaderboard') || '[]');
            const newRecord = { name: name.trim() || 'Аноним', score, time: timeStr
        };
            records.push(newRecord);
            records.sort((a, b) => b.score - a.score);
            if (records.length > 10) records.length = 10;
            localStorage.setItem('bubblePopLeaderboard',
JSON.stringify(records));
            loadLeaderboard();}

```

Управление темой, переключение фона игры на светлый и темный режим:

```

function toggleTheme() {
    currentTheme = currentTheme === 'dark' ? 'light' : 'dark';
    localStorage.setItem('bubblePopTheme', currentTheme);
    document.body.className = `theme-${currentTheme}`;}

```

Все визуальные эффекты реализованы через CSS-переходы и requestAnimationFrame, что обеспечивает плавность и высокую производительность на любых устройствах.

4 Выводы

В ходе исследования было успешно разработано веб-приложение «Лопай шарики», демонстрирующее широкие возможности нативных веб-технологий для создания динамичных, визуально привлекательных и интерактивных игр. Архитектура проекта отличается модульностью, читаемостью кода и отсутствием внешних зависимостей. Все ключевые компоненты — от плавной гравитационной анимации и визуальных частиц до системы рекордов, и переключения тем — реализованы эффективно и соответствуют современным требованиям пользовательского опыта. Приложение может использоваться как методическое пособие при изучении основ клиентской веб-разработки, а также как самостоятельный игровой продукт для развития внимания, стратегического мышления и скорости реакции.

Библиографический список

1. Смирнов И.В. Использование CSS3 для создания анимированных веб-интерфейсов // Веб-разработка и дизайн. 2023. № 3. С. 12-18.

2. Петрова А.С. Разработка интерактивных игр на JavaScript: методы управления событиями мыши // Программирование для начинающих. 2022. № 7. С. 34-40.
3. Кузнецов Д.А. Программная реализация системы управления мультимедийным контентом в браузере // Мультимедиа технологии. 2021. № 5. С. 56-61.
4. Григорьева Е.М., Лебедев М.П. Создание адаптивных веб-приложений с использованием Flexbox и Grid // Современные веб-технологии. 2020. № 9. С. 22-27.
5. Тихонов П.Н. Оптимизация производительности веб-анимаций // Производительность веб-приложений. 2024. № 2. С. 45-50.
6. Материалы разработки интерактивного веб-приложения «Лопай шарики» URL:
<https://drive.google.com/drive/folders/1AqkXmwjk0VYHAnbdUZylYd4jp8kRyZZH?usp=sharing> (дата обращения: 26.11.2025)