

Разработка интерактивного веб-приложения для украшения новогодней елки 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, имитирующего процесс украшения новогодней елки. В работе подробно описаны архитектура приложения, используемые технологии (HTML5, CSS3, JavaScript) и основные функциональные возможности: добавление и перетаскивание различных украшений, управление фоновым режимом (день/ночь), включение анимированного снега и фейерверков, а также воспроизведение рождественской музыки. Приложение создает погружение в праздничную атмосферу и может быть использовано как образовательный инструмент для изучения основ веб-разработки или как развлекательное средство для создания индивидуальных новогодних композиций.

Ключевые слова: HTML, CSS, JavaScript, веб-приложение, новогодняя елка, интерактивность, анимация.

Implementation of an interactive web application for decorating a christmas tree 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 that simulates the process of decorating a Christmas tree. The paper describes in detail the application architecture, technologies used (HTML5, CSS3, JavaScript), and main functional capabilities: adding and dragging various decorations, managing the background mode (day/night), enabling animated snow and fireworks, as well as playing Christmas music. The application creates immersion in a festive atmosphere and can be used as an educational tool for learning the basics of web development or as an entertainment tool for creating individual New Year's compositions.

Keywords: HTML, CSS, JavaScript, web application, Christmas tree, interactivity, animation.

1 Введение

1.1 Актуальность

В современной цифровой среде все большее значение приобретают интерактивные веб-приложения, способные не только передавать информацию, но и создавать уникальный пользовательский опыт. Новогодние праздники, являясь важной частью культурного наследия многих стран, служат отличным поводом для создания тематических цифровых продуктов, которые помогают пользователям погрузиться в праздничное настроение. Разработка приложений для украшения виртуальной елки позволяет сочетать творческое самовыражение с техническими навыками программирования.

Актуальность темы обусловлена растущим интересом к веб-технологиям и потребностью в простых, но эстетически привлекательных интерактивных решениях. Такие приложения могут использоваться в образовательных целях для демонстрации возможностей HTML, CSS и JavaScript, а также в качестве развлекательного контента для семейного досуга. Использование современных стандартов веб-разработки обеспечивает кроссплатформенность и доступность приложения для широкого круга пользователей.

1.2 Обзор исследований

И.В. Смирнов в своей статье рассмотрел применение HTML5 и CSS3 для создания анимированных веб-интерфейсов, подчеркнув важность использования ключевых кадров для реализации сложных анимаций, таких как падающий снег [1]. А.С. Петрова провела исследование, посвященное разработке интерактивных игр на JavaScript, где подробно описала методы управления событиями мыши для реализации функций перетаскивания объектов и двойного клика для их удаления [2]. Д.А. Кузнецов представил программную реализацию системы управления мультимедийным контентом в браузере, включая воспроизведение аудио и управление громкостью с помощью элементов управления `<input type="range">`, 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, что позволило им эффективно организовать макеты для различных устройств, включая панели управления и игровые поля [4]. П.Н. Тихонов представил исследование по оптимизации производительности веб-анимаций, где рекомендовал использовать `requestAnimationFrame` и ограничивать количество DOM-элементов для обеспечения плавной работы, особенно при работе с множеством анимированных объектов, таких как фейерверки [5].

1.3 Цель исследования

Цель исследования – разработка интерактивного веб-приложения на основе технологий HTML, CSS и JavaScript.

2 Материалы и методы

В данном исследовании для разработки интерактивного веб-приложения использовалась среда разработки Visual Studio Code (VS Code), которая предоставляет мощные инструменты для написания, отладки и тестирования кода на языках HTML, CSS и JavaScript. Выбор этих технологий обусловлен их универсальностью, поддержкой всеми современными браузерами и возможностью создания богатых пользовательских интерфейсов без необходимости установки дополнительных плагинов или программ.

Для создания визуального и аудио оформления приложения были использованы внешние файлы с расширением .jpg и .png, аудиофайлы с расширением .mp3 [6].

Основным методом исследования является практическая разработка приложения с использованием следующих технологий:

- HTML5: Для структурирования содержимого страницы, создания элементов управления (кнопок, чекбоксов, слайдеров) и контейнеров для игрового поля.
- CSS3: Для оформления интерфейса, задания стилей, анимаций (падающий снег, мерцание украшений) и управления состоянием (переключение между дневным и ночным режимом).
- JavaScript: Для реализации логики приложения, включая обработку событий пользователя (клики, перетаскивание, изменение состояния), динамическое создание и удаление элементов, управление анимациями и воспроизведение аудио.

3 Результаты

Разработанное интерактивное веб-приложение "Новогодняя Елка — Украшаем вместе!" представляет собой полноценное приложение, запускаемое в веб-браузере. Оно позволяет пользователю виртуально украшать новогоднюю елку, добавляя разнообразные элементы декора, управлять атмосферными эффектами и музыкальным сопровождением.

3.1 Архитектура приложения

Приложение имеет модульную архитектуру, разделенную на три основные части: структуру (HTML), стиль (CSS) и поведение (JavaScript). Основные компоненты:

- Приветственное окно: Отображается при первом запуске, содержит описание и кнопку для начала игры.
- Панель управления: Расположена над игровым полем и содержит все элементы управления.
- Игровое поле: Центральная область с фоновым изображением, елкой и динамически создаваемыми украшениями.
- Логика JavaScript: Реализует всю интерактивную функциональность приложения.

Приветственное окно приложения представляет собой полноэкранный блок с градиентным фоном, который создает праздничную атмосферу (рис. 1). В центре расположены заголовок, описание функционала и инструкция по использованию, а также кнопка "Начать украшать →", которая запускает основную часть приложения.

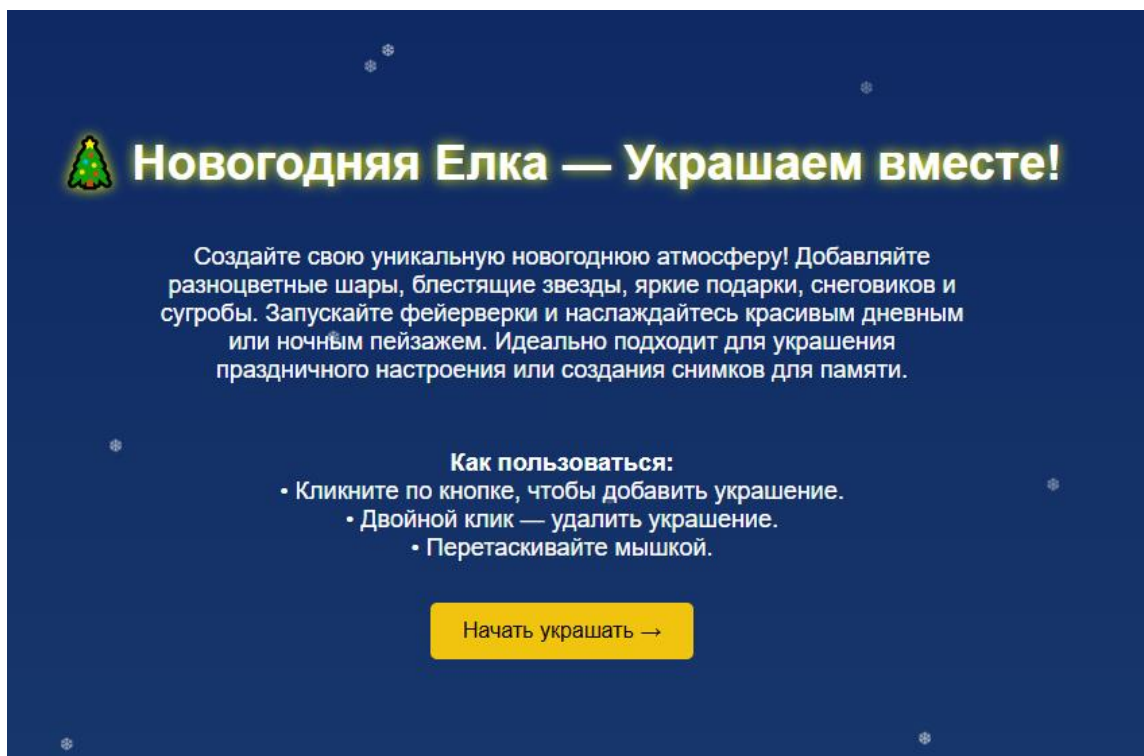


Рисунок 1 – Приветственное окно приложения

Для усиления визуального эффекта в этом окне реализована анимация падающих снежинок, которая создается с помощью JavaScript-функции `createSnowflakes()`, вызываемой при загрузке страницы (`window.onload`). Эта функция генерирует 20 случайных снежинок, каждая из которых имеет свою скорость и прозрачность, что придает окну динамичность:

```
function createSnowflakes() {  
  for (let i = 0; i < 20; i++) {  
    const flake = document.createElement('div');  
    flake.className = 'snowflake';  
    flake.innerHTML = '❄';  
    flake.style.left = Math.random() * 100 + 'vw';  
    flake.style.top = Math.random() * 100 + 'vh';  
    flake.style.animationDuration = (Math.random() * 5 + 5) + 's';  
    flake.style.opacity = Math.random() * 0.5 + 0.3;  
    document.getElementById('welcome-screen').appendChild(flake);  
  }  
}
```

Панель управления расположена над игровым полем и состоит из двух частей: верхней строки с кнопками для добавления украшений и нижней строки с переключателями и элементами управления музыкой (рис. 2).

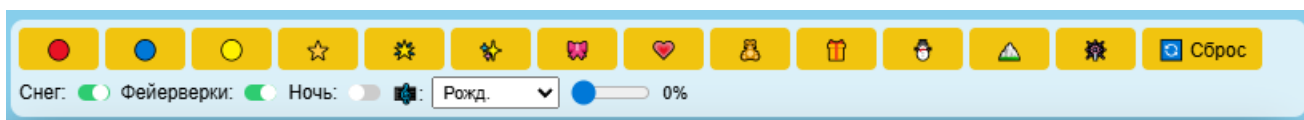


Рисунок 2 – Панель управления

Каждая кнопка связана с соответствующей функцией JavaScript через атрибут onclick. Это реализовано в HTML-разметке панели управления (рис. 3).

```
<div id="controls-container">
  <div id="controls">
    <button onclick="addOrnament('🔴')">🔴</button>
    <button onclick="addOrnament('🔵')">🔵</button>
    <button onclick="addOrnament('🟡')">🟡</button>
    <button onclick="addOrnament('★')">★</button>
    <button onclick="addOrnament('☆')">☆</button>
    <button onclick="addOrnament('✨')">✨</button>
    <button class="no-blink" onclick="addOrnament('💖')">💖</button>
    <button onclick="addOrnament('❤️')">❤️</button>
    <button onclick="addBearGift()">🧸</button>
    <button onclick="addGift()">📺</button>
    <button onclick="addSnowman()">🧊</button>
    <button onclick="addSnowdrift()">🏔️</button>
    <button onclick="launchFirework()">💣</button>
    <button onclick="resetTree()">🔄 Сброс</button>
    <div class="switch">
      <label>Снег:</label>
      <input type="checkbox" id="snowToggle" onchange="toggleSnow(this.checked)">
    </div>
    <div class="switch">
      <label>Фейерверки:</label>
      <input type="checkbox" id="fireworksToggle" onchange="toggleFireworks(this.checked)">
    </div>
    <div class="switch">
      <label>Ночь:</label>
      <input type="checkbox" id="dayNightToggle" onchange="toggleDayNight(this.checked)">
    </div>
    <!-- Музыкальные контролы -->
    <div id="music-controls">
      <label>🎵:</label>
      <select id="music-selector" onchange="changeMusic()" style="font-size: 11px; padding: 2px;">
        <option value="jingle">Рожд.</option>
        <option value="sleigh">Перв. снег</option>
        <option value="winter">Новог.</option>
      </select>
      <input type="range" id="music-slider" min="0" max="1" step="0.1" value="0.5" oninput="setVolume()" style="width: 50px;">
      <span id="volume-display">50%</span>
    </div>
  </div>
</div>
```

Рисунок 3 – Код панели управления

Переключатели выполнены с помощью стилизованных checkbox-элементов, как показано в CSS-стиле для класса .switch:

```
.switch {
  display: inline-flex;
  align-items: center;
  gap: 3px;
  font-size: 12px;
}
.switch input[type="checkbox"] {
  width: 20px;
  height: 10px;
```

```
    appearance: none;
    background: #ccc;
    border-radius: 10px;
    position: relative;
    cursor: pointer;
  }
  .switch input[type="checkbox"]:checked {
    background: #2ecc71;
  }
  .switch input[type="checkbox"]:after {
    content: '';
    position: absolute;
    width: 10px;
    height: 10px;
    background: white;
    border-radius: 50%;
    transition: transform 0.2s;
  }
  .switch input[type="checkbox"]:checked:after {
    transform: translateX(10px);
  }
}
```

Игровое поле представляет собой прямоугольный контейнер с закругленными углами и тенью, что придает ему объем (рис. 4). Это задается в CSS-стиле для #game-container:

```
#game-container {
  position: relative;
  width: 800px;
  height: 600px;
  border-radius: 15px;
  overflow: hidden;
  box-shadow: 0 0 30px rgba(0, 0, 0, 0.6);
  background: transparent; /* Прозрачный фон */
}
```

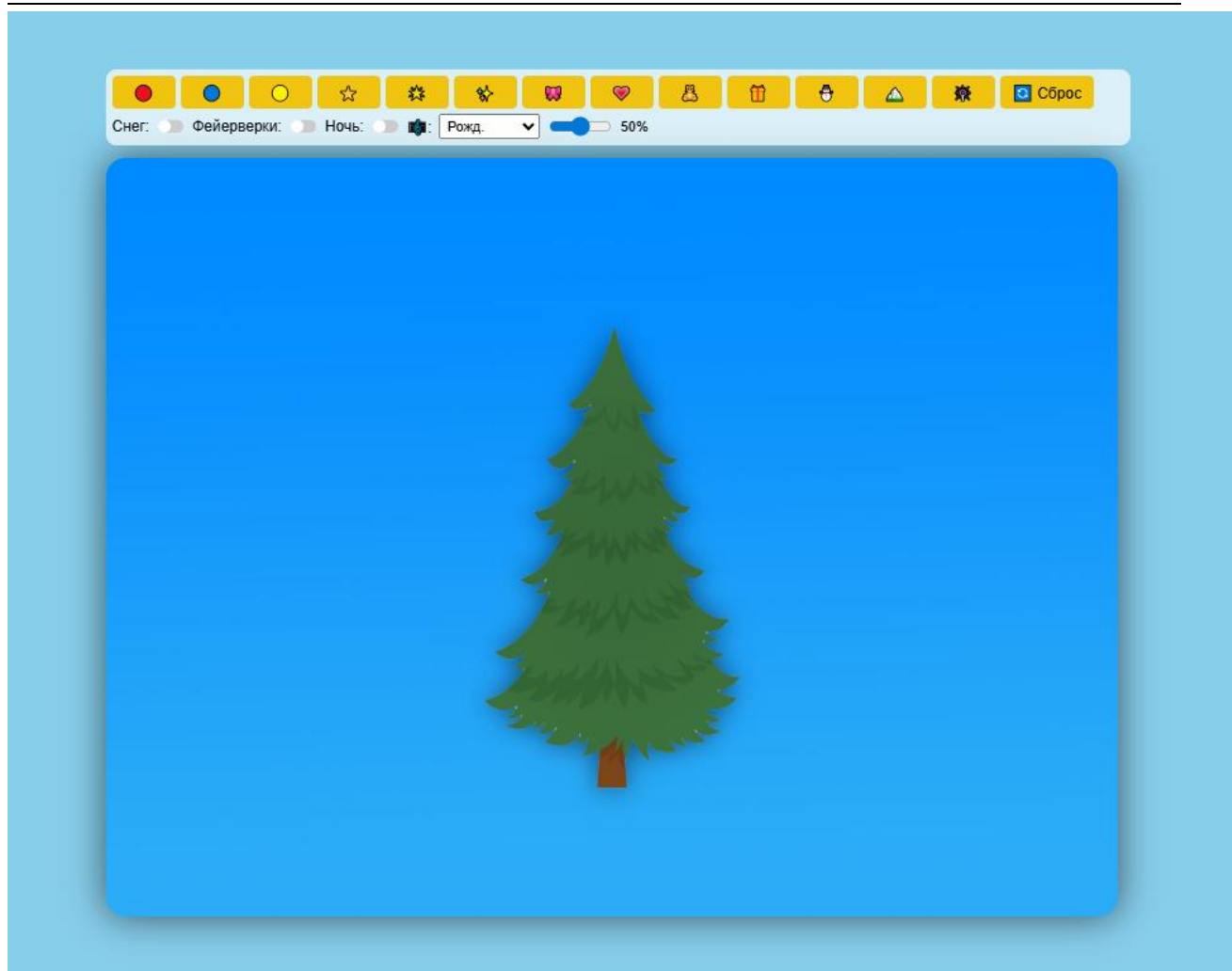


Рисунок 4 – Игровое поле

Внутри этого контейнера находятся два фоновых изображения — дневное (`#day-bg`) и ночное (`#night-bg`), которые переключаются в зависимости от состояния переключателя. В центре расположен контейнер `#tree`, содержащий изображение самой елки. Этот контейнер является родителем для всех динамически создаваемых украшений, что позволяет легко управлять их позиционированием относительно елки:

```
<div id="game-container">
  <!-- Фоновые изображения -->
  
  
  <div id="snow"></div>
  <div id="tree">
    
  </div>
</div>
```

Также в игровом поле находится контейнер `#snow`, в котором создаются анимированные снежинки, и он имеет более низкий `z-index`, чтобы не мешать взаимодействию с украшениями:

```
#snow {  
    position: absolute;  
    top: 0;  
    left: 0;  
    width: 100%;  
    height: 100%;  
    pointer-events: none;  
    z-index: 5;  
}
```

3.2 Основные функции и их реализация

Приложение начинается с показа приветственного окна (#welcome-screen). После нажатия кнопки "Начать украшать →" вызывается функция startGame(), которая скрывает это окно и инициализирует начальное состояние: устанавливает дневной фон и запускает выбранную музыку. Эта функция также проверяет состояние чекбокса снега и, если он активен, запускает анимацию снега:

```
// Запуск игры  
function startGame() {  
    document.getElementById('welcome-screen').style.display = 'none';  
    // Устанавливаем начальный фон на день  
    document.getElementById('day-bg').style.display = 'block';  
    document.getElementById('night-bg').style.display = 'none';  
    // Устанавливаем начальную тему фона тела на день  
    document.body.classList.remove('night-theme');  
    document.body.classList.add('day-theme');  
    // Снег по умолчанию выключен, запускаем только если чекбокс активен  
    if (document.getElementById('snowToggle').checked) {  
        startSnow();  
    }  
    // Запуск музыки (по умолчанию первая)  
    changeMusic();  
}
```

Для управления цветом фона используется классовая система в CSS. Функция toggleDayNightTheme(isNight) добавляет или удаляет классы day-theme и night-theme у элемента body, что позволяет плавно менять цвет фона согласно требованиям к единому стилю. Это решение позволяет отделить логику от стиля, что является хорошей практикой в веб-разработке:

```
/* Цвета для дня и ночи */  
body.day-theme {background-color: #87CEEB;}  
body.night-theme {background-color: #1e3a8a;}  
// Функция для переключения темы дня/ночи  
function toggleDayNightTheme(isNight) {  
    const body = document.body;  
    if (isNight) {  
        body.classList.remove('day-theme');  
        body.classList.add('night-theme');  
    } else {  
        body.classList.remove('night-theme');  
        body.classList.add('day-theme');  
    }  
}
```

Центральная функция `addElement(className, content, blink)` отвечает за создание нового украшения. Она принимает параметры класса, контента (например, эмодзи) и флага мерцания, создает элемент `div`, устанавливает ему случайное положение, добавляет в DOM и делает его интерактивным:

```
function addElement(className, content, blink = true) {  
  const element = document.createElement('div');  
  element.className = className;  
  if (!blink) element.classList.add('no-blink');  
  element.innerHTML = content;  
  element.style.left = (Math.random() * 180 + 10) + 'px';  
  element.style.top = (Math.random() * 180 + 50) + 'px';  
  document.getElementById('tree').appendChild(element);  
  makeDraggable(element);  
  makeDoubleClickable(element);  
}
```

Функция `makeDraggable(element)` реализует механизм перетаскивания через обработку событий `mousedown`, `mousemove` и `mouseup`. Это позволяет пользователю свободно перемещать любое украшение по экрану. При нажатии на украшение (`mousedown`) фиксируются координаты курсора относительно элемента, а затем при движении мыши (`mousemove`) элемент перемещается, сохраняя это смещение. При отпускании кнопки мыши (`mouseup`) перемещение прекращается.

Функция `makeDoubleClickable(element)` реализует удаление украшения по двойному клику. При втором клике вызывается диалог `confirm`, и при положительном ответе элемент удаляется из DOM. Это обеспечивает простой и интуитивно понятный способ очистки игрового поля.

DOM (Document Object Model — Объектная модель документа) — это программный интерфейс, представляющий структуру HTML- или XML-документа в виде иерархического дерева объектов (узлов), с которыми может взаимодействовать скриптовый язык, такой как JavaScript. Это связующее звено между разметкой документа и программным кодом, позволяющее динамически читать, изменять, добавлять или удалять содержимое, структуру и стили веб-страницы в реальном времени.

Анимированные эффекты:

1) Падающий снег

Реализован с помощью функции `startSnow()`, которая каждые 300 миллисекунд создает новый элемент-снежинку (`<div>❄</div>`) внутри контейнера `#snow`. CSS-анимация `@keyframes fall` заставляет снежинку двигаться вниз по экрану. Снежинки имеют случайную скорость, прозрачность и время жизни, что создает реалистичный эффект. Функция `stopSnow()` используется для остановки анимации при отключении переключателя:

```
function startSnow() {
  stopSnow();
  const snowContainer = document.getElementById('snow');
  snowInterval = setInterval(() => {
    const flake = document.createElement('div');
    flake.innerHTML = '❄';
    flake.className = 'snowflake-game';
    flake.style.left = Math.random() * 100 + 'vw';
    flake.style.animationDuration = (Math.random() * 5 + 5) + 's';
    flake.style.opacity = Math.random() * 0.5 + 0.3;
    snowContainer.appendChild(flake);
    setTimeout(() => { if (flake.parentNode) flake.remove(); }, 10000);
  }, 300);
}
function stopSnow() {
  if (snowInterval) clearInterval(snowInterval);
}
```

2) Фейерверки

Функция `launchFirework()` создает ракету, которая летит вверх по экрану. По достижении случайной высоты она исчезает, и в этой точке запускается функция `explodeFirework(x, yTop, colors)`, которая создает 50 разлетающихся в разные стороны пикселей, имитируя взрыв. Каждый пиксель получает случайное направление и скорость, а затем постепенно исчезает, создавая эффект рассеивания. Эта функция использует рекурсивный подход с `setInterval` для управления анимацией каждого пикселя:

```
function launchFirework() {
  const container = document.getElementById('game-container');
  const colors = ['#e74c3c', '#f1c40f', '#2ecc71', '#3498db', '#9b59b6', '#1abc9c'];
  const rocket = document.createElement('div');
  rocket.className = 'firework-pixel';
  rocket.style.width = '8px';
  rocket.style.height = '8px';
  rocket.style.backgroundColor = colors[Math.floor(Math.random() * colors.length)];
  rocket.style.left = Math.random() * 800 + 'px';
  rocket.style.bottom = '50px';
  container.appendChild(rocket);
  let y = 50;
  const targetY = 100 + Math.random() * 400;
  const interval = setInterval(() => {
    y += 8;
    rocket.style.bottom = y + 'px';
    if (y > targetY) {
      clearInterval(interval);
      const xCoord = rocket.style.left;
      rocket.remove();
      explodeFirework(xCoord, (600 - y) + 'px', colors);
    }
  }, 30);
}
function explodeFirework(x, yTop, colors) {
  const container = document.getElementById('game-container');
  for (let i = 0; i < 50; i++) {
```

```

const pixel = document.createElement('div');
pixel.className = 'firework-pixel';
pixel.style.width = '5px';
pixel.style.height = '5px';
pixel.style.backgroundColor = colors[Math.floor(Math.random() *
colors.length)];
pixel.style.left = x;
pixel.style.top = yTop;
container.appendChild(pixel);
const angle = Math.random() * Math.PI * 2;
const speed = Math.random() * 6;
const dx = Math.cos(angle) * speed;
const dy = Math.sin(angle) * speed;
let opacity = 1;
const fade = setInterval(() => {
  opacity -= 0.02;
  if (opacity <= 0) {
    clearInterval(fade);
    pixel.remove();
  } else {
    pixel.style.opacity = opacity;
    pixel.style.transform = `translate(${dx * 5}px, ${dy * 5}px)`;
  }
}, 50);
setTimeout(() => {
  clearInterval(fade);
  pixel.remove();
}, 1500);

```

Музыкальное сопровождение

Пользователь может выбрать одну из трех рождественских мелодий с помощью выпадающего списка (#music-selector). Функция changeMusic() создает новый объект Audio, настраивает его (зацикливание, громкость) и запускает воспроизведение. Также реализована обработка блокировки автовоспроизведения браузером: если игра не запускается автоматически, он будет запущен после первого клика пользователя на страницу:

```

function changeMusic() {
  if (currentMusic) {
    currentMusic.pause();
    currentMusic = null;
  }
  const selected = document.getElementById('music-selector').value;
  currentMusic = new Audio(musicFiles[selected]);
  currentMusic.loop = true;
  currentMusic.volume = parseFloat(document.getElementById('music-
slider').value);
  const playPromise = currentMusic.play();
  if (playPromise !== undefined) {
    playPromise.catch(error => {
      console.log("Автовоспроизведение заблокировано:", error);
      const playOnInteraction = () => {currentMusic.play();
      document.removeEventListener('click', playOnInteraction);};
      document.addEventListener('click', playOnInteraction);});
  }
}

```

Приложение включает функции для добавления подарков (addGift()), медведей (addBearGift()), снеговиков (addSnowman()) и сугробов (addSnowdrift()). Эти элементы создаются вне контейнера елки и размещаются

в нижней части игрового поля. Размеры подарков и сугробов генерируются случайным образом, что добавляет визуального разнообразия. Например, функция `addGift()` использует функцию `getRandomGiftSize()` для получения случайного размера подарка из массива предопределенных значений:

```
function getRandomGiftSize() {  
    const sizes = ['30px', '35px', '40px', '45px'];  
    return sizes[Math.floor(Math.random() * sizes.length)];  
}  
function addGift() {  
    const giftSize = getRandomGiftSize();  
    const gift = document.createElement('div');  
    gift.className = 'gift';  
    gift.innerHTML = '🎁';  
    gift.style.width = giftSize;  
    gift.style.height = giftSize;  
    gift.style.fontSize = `calc(${giftSize} * 0.8)`;  
    const leftOffset = Math.random() * 100 - 50;  
    gift.style.left = `calc(50% - (${giftSize} / 2) + ${leftOffset}px)`;  
    gift.style.bottom = '50px';  
    document.getElementById('game-container').appendChild(gift);  
    makeDraggable(gift);  
    makeDoubleClickable(gift);  
}
```

Аналогичным образом реализованы функции для добавления других декоративных элементов на игровое поле: `addBearGift()`, `addSnowman()` и `addSnowdrift()`. Каждая из этих функций создает соответствующий DOM-элемент (`div`), задает ему уникальные CSS-классы (`gift`, `snowman`, `snowdrift`), определяет размеры (в том числе случайные для подарков и сугробов) и начальное положение, а также делает элемент интерактивным, присоединяя обработчики событий перетаскивания и удаления через вызов функций `makeDraggable(element)` и `makeDoubleClickable(element)`.

Финальный результат работы пользователя с приложением демонстрирует возможности интерактивного дизайна (рис. 5). В ночном режиме, с активированным эффектом падающего снега и фейерверков, пользователь создал насыщенную праздничную сцену. На елке размещены разноцветные шары и звезды, а также сердечки в качестве особого украшения. У подножия елки находятся подарки и игрушечные медведи, что создает атмосферу щедрости и веселья. По бокам от елки расположены два снеговика, добавляющие сказочности сцене. Этот пример наглядно показывает, как комбинация всех функций приложения — от простого добавления элементов до управления атмосферными эффектами — позволяет пользователю реализовать свой творческий замысел и создать уникальную, визуально привлекательную композицию.

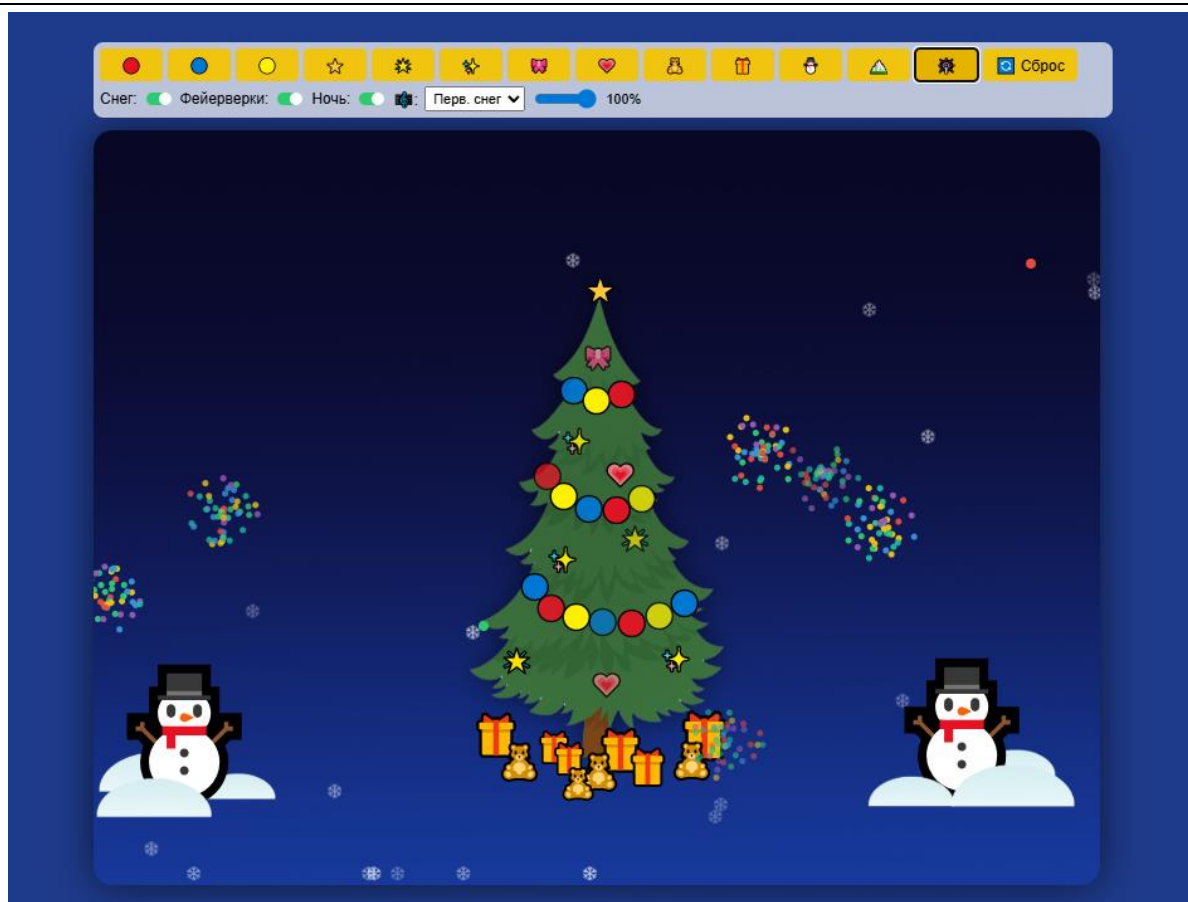


Рисунок 5 – Пример завершенной композиции после украшения елки

4 Выводы

В ходе исследования было разработано полнофункциональное интерактивное веб-приложение для украшения новогодней елки. Проект успешно демонстрирует, как сочетание HTML, CSS и JavaScript позволяет создавать сложные, визуально насыщенные и интерактивные приложения прямо в браузере.

Ключевые особенности реализации включают использование модульного подхода к коду, где каждая функция (добавление, перетаскивание, анимация) вынесена в отдельный блок, что обеспечивает читаемость и легкость поддержки. Визуальные эффекты, такие как снег и фейерверки, реализованы с учетом производительности, путем динамического создания и удаления DOM-элементов.

Разработанное приложение может служить ценным примером для студентов и начинающих разработчиков, желающих углубить свои знания в области клиентской веб-разработки. Его функциональность и архитектура могут быть легко расширены для создания более сложных проектов.

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.