

Разработка веб-игры «Поле чудес» с использованием HTML, CSS и JavaScript: методика создания и анализ пользовательского взаимодействия

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

Статья посвящена разработке интерактивной веб-версии игры «Поле чудес» с использованием современных технологий фронтенд-разработки: HTML5, CSS3 и JavaScript. Рассматривается поэтапное создание приложения — от базовой структуры HTML-документа до реализации игровой логики, включая обработку пользовательских событий, визуальное оформление и адаптацию под мобильные устройства. Особое внимание уделено механизму обратной связи при неправильном выборе буквы: система динамически обновляет интерфейс, подсвечивая ошибки и выводя текстовые уведомления. Результатом является полнофункциональное веб-приложение, сочетающее образовательную ценность с развлекательной составляющей.

Ключевые слова: веб-разработка, HTML5, CSS3, JavaScript, интерактивные игры, образовательные технологии, пользовательский интерфейс.

Development of the Field of Wonders web game using HTML, CSS and JavaScript: a methodology for creating and analyzing user interaction

Ekimova Yana Sergeevna

Sholom Aleichem Priamurskiy State University

Student

Abstract

The article is devoted to the development of an interactive web version of the Field of Wonders game using modern frontend development technologies: HTML5, CSS3 and JavaScript. The step—by-step creation of an application is considered - from the basic structure of an HTML document to the implementation of game logic, including processing user events, visual design and adaptation to mobile devices. Special attention is paid to the feedback mechanism in case of incorrect letter selection: the system dynamically updates the interface, highlighting errors and displaying text notifications. The result is a fully functional web application that combines educational value with an entertainment component.

Keywords: web development, HTML5, CSS3, JavaScript, interactive games, educational technologies, user interface.

1 Введение

1.1 Актуальность

В условиях роста популярности онлайн-обучения и интерактивных образовательных платформ разработка веб-игр, направленных на развитие лингвистических навыков, приобретает особую значимость. Классическая игра «Поле чудес» адаптирована в цифровой формат с сохранением узнаваемого стиля телешоу, что делает её привлекательной для широкой аудитории. Использование чистого JavaScript без дополнительных фреймворков демонстрирует базовые принципы веб-разработки, что полезно для обучения начинающих программистов. Актуальность работы также подчеркивается применением адаптивного дизайна, обеспечивающего корректное отображение на различных устройствах.

1.2 Обзор исследований

В статье С.В. Рогатина, В.А. Антошкина рассматривается методика разработки компьютерных игр по технологии HTML5 с помощью CASE-средства GameMakerStudio компании YoYoGamesLtd. Проводится тестирование производительности работы компьютерных игр в браузере в зависимости от сложности использованных спрайтов игровых объектов [1]. Разработали игровое приложение «league of football heads» с использованием html5, css3, javascript И.А. Фёдоров, Е.Ю. Кузьменкова[2]. Королук Я. А., Куликова Н. Ю. показали опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике [3]. В работе рассмотрели возможности языка разметки гипертекста HTML5 для создания макета электронного учебника Д.М. Размологов, А.С. Панина, П.В. Захаров. Привели разработанный макет, а также часть таблицы стилей для дизайна макета [4]. И.С. Шегаев, А.Г. Полозов разработали новейший политологический глоссарий: основные категории, понятия, термины [5].

1.3 Цель исследования

Целью исследования является разработка веб-игры «Поле чудес» с использованием HTML, CSS и JavaScript.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовым редактор Notepad++, оптимизированным для языков программирования.

3 Результаты

Открываем Notepad++ и создаём файл index.html, для этого в меню выбираем файл → новый. Далее нажимаем сочетание клавиш "Ctrl+S" и в поле "Имя файла" вводим название файла "index" с расширением .html, сохраняем данный файл на рабочем столе.

Создадим основную структуру HTML-документа для игры "Поле чудес", которая включает: декларацию DOCTYPE, HTML-разметку с русской локализацией, мета-теги для корректного отображения на мобильных устройствах, заголовок страницы, а также разделы для стилей (в теге style) и JavaScript-кода (в теге script). В теле документа размещены два основных блока - стартовый экран с выбором категории (животные, страны, фильмы или еда) и кнопкой начала игры, а также игровой экран, содержащий таймер, счетчик попыток, область для отображения загаданного слова, виртуальную клавиатуру и кнопку перезапуска игры. Игровой экран изначально скрыт с помощью класса "hidden" и активируется при нажатии кнопки "Начать игру".

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Поле чудес</title>
  <style>

</style>
</head>
<body>
  <div class="game-container">
    <h1>Поле чудес</h1>

    <div id="start-screen">
      <p>Добро пожаловать в игру "Поле чудес"! Угадывайте буквы в загаданном слове.</p>
      <div class="category-select">
        <h3>Выберите категорию:</h3>
        <select id="category-select">
          <option value="animals">Животные</option>
          <option value="countries">Страны</option>
          <option value="movies">Фильмы</option>
          <option value="food">Еда</option>
        </select>
      </div>
      <button id="start-btn">Начать игру</button>
    </div>

    <div id="game-screen" class="hidden">
      <div class="game-info">
        <div class="info-box">Осталось попыток: <span id="attempts">5</span></div>
        <div class="info-box">Время: <span id="timer">60</span> сек</div>
      </div>

      <div class="word-display" id="word-display"></div>

      <div class="message" id="message"></div>

      <div class="keyboard" id="keyboard"></div>

      <button id="restart-btn">Новая игра</button>
    </div>
  </div>

  <script>

</script>
```

</body>
</html>



Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей, который задает оформление игры в стиле телешоу "Поле чудес": темный фон с золотистыми акцентами и точечным градиентом, основной контейнер с закругленными углами и красной рамкой, стилизованный заголовок с текстовой тенью, выпадающий список категорий с желтой рамкой, область отображения слова с увеличенным шрифтом и подчеркиванием для каждой буквы, виртуальную клавиатуру с кнопками в 3D-стиле (меняющими цвет при наведении и нажатии, а также подсвечивающимися зеленым для правильных и красным для неправильных букв), информационные панели с таймером и счетчиком попыток, анимацию пульсации для элементов интерфейса и адаптивные стили для мобильных устройств, уменьшающие размер элементов при узком экране. Все элементы используют плавные переходы и тени для создания глубины и реалистичного взаимодействия.

Данный код пропишем на следующей строчке после тега <title>Поле чудес</title> в секции <head> HTML-документа.

```
body {  
  font-family: 'Arial', sans-serif;  
  background-color: #1a1a2e;  
  color: white;  
  text-align: center;  
  padding: 20px;  
  background-image: radial-gradient(circle, #2d2d44 1px, transparent 1px);  
  background-size: 20px 20px;  
}  
  
.game-container {  
  max-width: 800px;  
  margin: 0 auto;  
  background-color: #16213e;  
  padding: 30px;  
  border-radius: 15px;  
  box-shadow: 0 10px 25px rgba(0, 0, 0, 0.5),  
             inset 0 0 15px rgba(255, 215, 0, 0.2);  
  border: 2px solid #e94560;
```

```
}

h1 {
  color: #f9c74f;
  text-shadow: 0 0 10px rgba(249, 199, 79, 0.5),
    2px 2px 0px rgba(233, 69, 96, 0.8);
  font-size: 2.5rem;
  margin-bottom: 20px;
}

.category-select {
  margin: 25px 0;
  padding: 15px;
  background-color: rgba(0, 0, 0, 0.3);
  border-radius: 10px;
  border: 1px solid #f9c74f;
}

.category-select h3 {
  color: #f9c74f;
  margin-top: 0;
}

select {
  padding: 8px 15px;
  border-radius: 5px;
  background-color: #0f3460;
  color: white;
  border: 1px solid #e94560;
  font-size: 16px;
  cursor: pointer;
}

.word-display {
  font-size: 2.5rem;
  letter-spacing: 15px;
  margin: 40px 0;
  min-height: 60px;
  font-weight: bold;
  color: #f9c74f;
  text-shadow: 0 0 5px rgba(249, 199, 79, 0.7);
}

.letter {
  display: inline-block;
  width: 40px;
  position: relative;
}

.letter::after {
  content: "";
  display: block;
  width: 100%;
  height: 3px;
  background: linear-gradient(90deg, transparent, #f9c74f, transparent);
  position: absolute;
  bottom: -10px;
  left: 0;
  opacity: 0.7;
}

.keyboard {
  display: flex;
```

```
flex-wrap: wrap;
justify-content: center;
gap: 8px;
margin: 30px 0;
padding: 20px;
background-color: rgba(15, 52, 96, 0.5);
border-radius: 15px;
border: 2px solid #e94560;
box-shadow: inset 0 0 20px rgba(0, 0, 0, 0.5);
}

.key {
width: 50px;
height: 50px;
font-size: 1.5rem;
font-weight: bold;
background: linear-gradient(145deg, #2d4059, #1a2a3a);
color: #f9c74f;
border: none;
border-radius: 8px;
cursor: pointer;
transition: all 0.2s;
box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3),
            inset 0 0 5px rgba(255, 255, 255, 0.1);
text-transform: uppercase;
}

.key:hover {
transform: translateY(-3px);
box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.4);
background: linear-gradient(145deg, #3a516d, #223344);
}

.key:active {
transform: translateY(1px);
}

.key.used {
opacity: 0.5;
transform: scale(0.95);
cursor: not-allowed;
box-shadow: none;
}

.key.correct {
background: linear-gradient(145deg, #4caf50, #2e7d32);
color: white;
}

.key.wrong {
background: linear-gradient(145deg, #f44336, #c62828);
color: white;
}

.game-info {
display: flex;
justify-content: space-around;
margin: 30px 0;
gap: 20px;
}

.info-box {
background: linear-gradient(145deg, #0f3460, #08203e);
```

```
padding: 15px 25px;
border-radius: 10px;
font-size: 1.1rem;
flex: 1;
max-width: 200px;
box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3);
border: 1px solid #e94560;
}

.info-box span {
  font-weight: bold;
  color: #f9c74f;
  font-size: 1.3rem;
}

.message {
  margin: 25px 0;
  font-size: 1.2rem;
  min-height: 30px;
  color: #f9c74f;
  text-shadow: 0 0 5px rgba(249, 199, 79, 0.5);
  font-weight: bold;
}

button {
  background: linear-gradient(145deg, #e94560, #c2334d);
  color: white;
  border: none;
  padding: 12px 25px;
  font-size: 1.1rem;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s;
  font-weight: bold;
  text-transform: uppercase;
  letter-spacing: 1px;
  box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3);
  margin-top: 10px;
}

button:hover {
  background: linear-gradient(145deg, #ff5c77, #d93a56);
  transform: translateY(-2px);
  box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.4);
}

button:active {
  transform: translateY(1px);
}

.hidden {
  display: none;
}

/* Анимации */
@keyframes pulse {
  0% { transform: scale(1); }
  50% { transform: scale(1.05); }
  100% { transform: scale(1); }
}

.pulse {
  animation: pulse 1s infinite;
```

```
}

/* Адаптивность */
@media (max-width: 600px) {
  .key {
    width: 40px;
    height: 40px;
    font-size: 1.2rem;
  }

  .word-display {
    font-size: 1.8rem;
    letter-spacing: 8px;
  }

  .game-info {
    flex-direction: column;
    align-items: center;
  }

  .info-box {
    max-width: 100%;
    width: 100%;
    margin-bottom: 10px;
  }
}
```

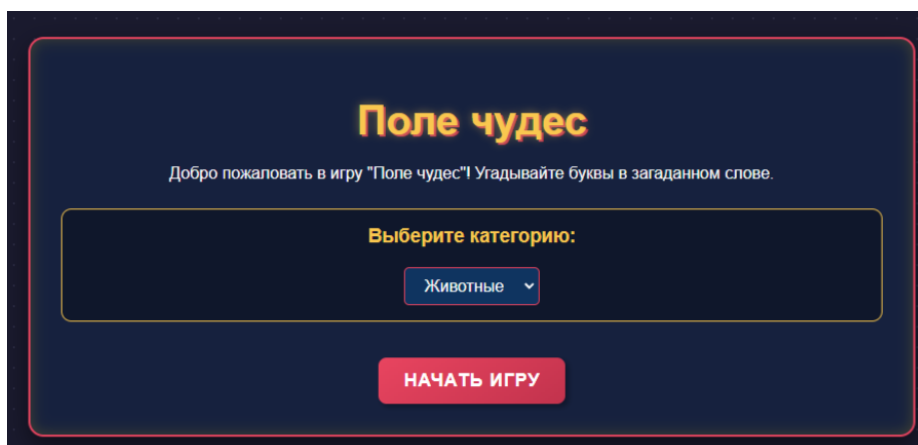


Рисунок 2. Добавление CSS-стилей

Теперь напишем код, который определяет логику игры: создаем объект `wordsByCategory` с массивами слов по четырем тематикам (животные, страны, фильмы и еда), получаем ссылки на DOM-элементы интерфейса (экраны, кнопки, таймер и т.д.), объявляем игровые переменные для хранения текущего слова, угаданных букв, количества ошибок и состояния игры, задаем константы (максимальное число попыток, время на игру и русский алфавит), а также реализуем функцию `initKeyboard()` для динамического создания виртуальной клавиатуры - она генерирует кнопки для каждой буквы алфавита, добавляет обработчики кликов и сохраняет буквы в `data`-атрибуты. Этот код формирует основу для последующей реализации игровой механики угадывания слов.

Вставьте данный код после закрывающего тега `</div>` и после открывающего тега `<script>`:

```

// Слова по категориям
const wordsByCategory = {
  animals: ['ТИГР', 'СЛОН', 'ЖИРАФ', 'КРОКОДИЛ', 'ПАНДА', 'ЗЕБРА', 'НОСОРОГ',
'БЕГЕМОТ'],
  countries: ['РОССИЯ', 'КИТАЙ', 'БРАЗИЛИЯ', 'КАНАДА', 'ИСПАНИЯ', 'ИНДИЯ',
'АВСТРАЛИЯ', 'ЯПОНИЯ'],
  movies: ['ТИТАНИК', 'АВАТАР', 'ГАРРИПОТТЕР', 'МАТРИЦА', 'ПИРАТЫ КАРИБСКОГО
МОРЯ', 'ТЕРМИНАТОР', 'ФОРСАЖ'],
  food: ['ПИЦЦА', 'СУШИ', 'ПЕЛЬМЕНИ', 'ШАШЛЫК', 'ПАСТА', 'БОРЩ', 'ХАЧАПУРИ',
'ПЛОВ']
};

// Элементы интерфейса
const startScreen = document.getElementById('start-screen');
const gameScreen = document.getElementById('game-screen');
const startBtn = document.getElementById('start-btn');
const restartBtn = document.getElementById('restart-btn');
const categorySelect = document.getElementById('category-select');
const wordDisplay = document.getElementById('word-display');
const keyboard = document.getElementById('keyboard');
const messageEl = document.getElementById('message');
const attemptsEl = document.getElementById('attempts');
const timerEl = document.getElementById('timer');

// Игровые переменные
let selectedWord = '';
let guessedLetters = [];
let wrongAttempts = 0;
const maxAttempts = 5;
let timer;
let timeLeft = 60;
let gameActive = false;

// Русский алфавит для клавиатуры
const russianAlphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ';

// Инициализация клавиатуры
function initKeyboard() {
  keyboard.innerHTML = '';
  for (let letter of russianAlphabet) {
    const key = document.createElement('button');
    key.className = 'key';
    key.textContent = letter;
    key.dataset.letter = letter;
    key.addEventListener('click', () => handleLetterClick(letter));
    keyboard.appendChild(key);
  }
}

```

Затем реализуем функцию `handleLetterClick(letter)`, которая обрабатывает нажатия на буквы клавиатуры: сначала проверяет активность игры и неиспользованность буквы, затем помечает кнопку как использованную и проверяет наличие буквы в загаданном слове - при правильном ответе добавляет класс `correct`, обновляет отображение слова, запускает анимацию пульсации и проверяет условие победы через `checkWin()`, а при ошибке добавляет класс `wrong`, уменьшает количество попыток, воспроизводит анимацию покачивания кнопки, проверяет условие проигрыша

и выводит сообщение об ошибке, при этом все изменения интерфейса синхронизированы с игровой логикой через обновление DOM-элементов.

Вставьте данный код в продолжение тега `<script>`:

```
// Обработка нажатия на букву
function handleLetterClick(letter) {
  if (!gameActive) return;

  const key = document.querySelector(`.key[data-letter="${letter}"]`);
  if (key.classList.contains('used')) return;

  key.classList.add('used');

  if (selectedWord.includes(letter)) {
    // Правильная буква
    key.classList.add('correct');
    guessedLetters.push(letter);
    updateWordDisplay();

    // Анимация при правильном ответе
    key.classList.add('pulse');
    setTimeout(() => key.classList.remove('pulse'), 1000);

    // Проверка на победу
    if (checkWin()) {
      endGame(true);
    }
  } else {
    // Неправильная буква
    key.classList.add('wrong');
    wrongAttempts++;
    attemptsEl.textContent = maxAttempts - wrongAttempts;

    // Анимация при неправильном ответе
    key.style.transform = 'translateX(5px)';
    setTimeout(() => {
      key.style.transform = 'translateX(-5px)';
      setTimeout(() => key.style.transform = "", 100);
    }, 100);

    // Проверка на проигрыш
    if (wrongAttempts >= maxAttempts) {
      endGame(false);
    } else {
      showMessage(`Буквы "${letter}" нет в слове!`);
    }
  }
}
```

Теперь реализуем вспомогательные функции для работы игры: `updateWordDisplay()` динамически формирует отображение загаданного слова, заменя неугаданные буквы подчеркиваниями, а угаданные - зелеными буквами; `checkWin()` проверяет победу, сравнивая все буквы слова с массивом угаданных; `showMessage()` выводит временное сообщение на 2 секунды; `startTimer()` запускает 60-секундный обратный отсчет с визуальным предупреждением (красный цвет и пульсация) при остатке менее 10 секунд,

автоматически завершая игру по истечении времени. Эти функции обеспечивают визуальную обратную связь и контроль игрового процесса.

Вставьте данный код в продолжение тега <script>:

```
// Обновление отображения слова
function updateWordDisplay() {
  wordDisplay.innerHTML = '';
  for (let letter of selectedWord) {
    const letterSpan = document.createElement('span');
    letterSpan.className = 'letter';
    if (guessedLetters.includes(letter) || letter === ' ') {
      letterSpan.textContent = letter;
      letterSpan.style.color = '#4caf50';
    } else {
      letterSpan.textContent = '_';
    }
    wordDisplay.appendChild(letterSpan);
  }
}

// Проверка на победу
function checkWin() {
  return selectedWord.split('').every(letter =>
    guessedLetters.includes(letter) || letter === ' '
  );
}

// Отображение сообщения
function showMessage(msg) {
  messageEl.textContent = msg;
  setTimeout(() => {
    if (messageEl.textContent === msg) {
      messageEl.textContent = '';
    }
  }, 2000);
}

// Запуск таймера
function startTimer() {
  timeLeft = 60;
  timerEl.textContent = timeLeft;

  clearInterval(timer);
  timer = setInterval(() => {
    timeLeft--;
    timerEl.textContent = timeLeft;

    if (timeLeft <= 10) {
      timerEl.style.color = '#f44336';
      timerEl.classList.add('pulse');
    }

    if (timeLeft <= 0) {
      endGame(false);
    }
  }, 1000);
}
```

Переходим к реализации основных игровых функций: startGame() выбирает случайное слово из выбранной категории, сбрасывает игровые

параметры (список угаданных букв, счетчик ошибок), активирует игровой процесс, инициализирует клавиатуру, обновляет отображение слова, запускает таймер и переключает экраны; endGame() останавливает игру и таймер, отображая соответствующее сообщение о победе (с анимацией пульсации слова) или поражении; также настраиваем обработчики событий для кнопок старта и рестарта, а при загрузке страницы сразу инициализируем клавиатуру, обеспечивая готовность игры к запуску. Эти функции связывают все компоненты в единую игровую систему.

```
// Начало игры
function startGame() {
  const category = categorySelect.value;
  const words = wordsByCategory[category];
  selectedWord = words[Math.floor(Math.random() * words.length)];

  guessedLetters = [];
  wrongAttempts = 0;
  gameActive = true;

  attemptsEl.textContent = maxAttempts;
  messageEl.textContent = "";
  timerEl.style.color = '#f9c74f';
  timerEl.classList.remove('pulse');

  initKeyboard();
  updateWordDisplay();
  startTimer();

  startScreen.classList.add('hidden');
  gameScreen.classList.remove('hidden');
}

// Завершение игры
function endGame(isWin) {
  gameActive = false;
  clearInterval(timer);

  if (isWin) {
    showMessage(`Поздравляем! Вы угадали слово: ${selectedWord}`);
    wordDisplay.classList.add('pulse');
  } else {
    showMessage(`Игра окончена! Загаданное слово: ${selectedWord}`);
  }
}

// Обработчики событий
startBtn.addEventListener('click', startGame);
restartBtn.addEventListener('click', () => {
  wordDisplay.classList.remove('pulse');
  gameScreen.classList.add('hidden');
  startScreen.classList.remove('hidden');
});

// Инициализация при загрузке
initKeyboard();
```

Теперь игра полностью завершена и готова к использованию: реализован полный цикл игрового процесса от выбора категории до завершения раунда с выводом результатов, включая интерактивную клавиатуру, таймер с визуальными эффектами, систему подсчета очков и механизм проверки слов, а также плавные переходы между экранами и адаптивный интерфейс, что делает игру функциональной, удобной и визуально привлекательной для пользователей, которые могут сразу начать играть, угадывая слова в классическом стиле телешоу "Поле чудес".

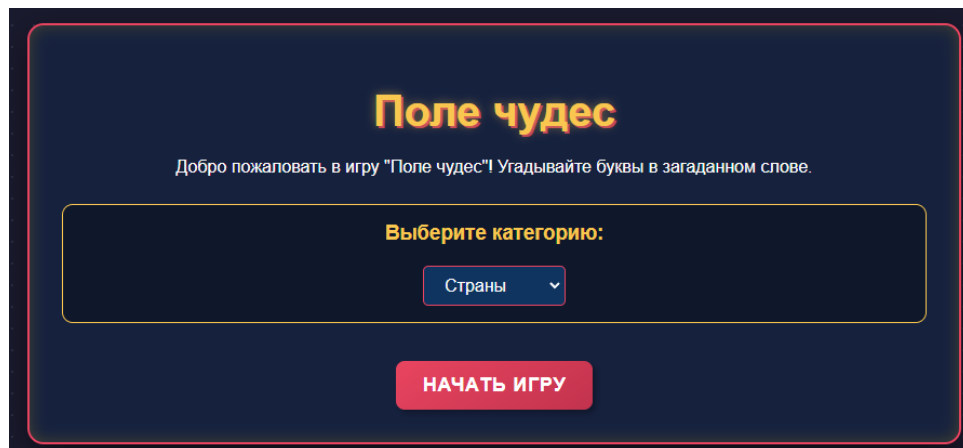


Рисунок 3. Игра

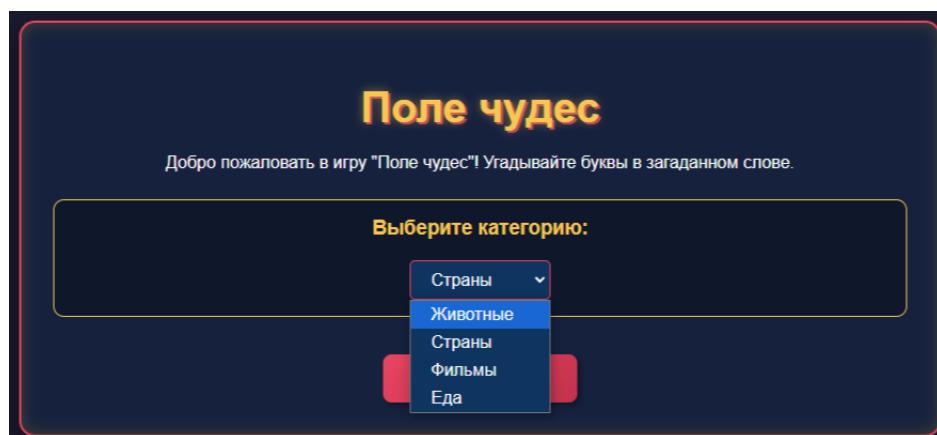


Рисунок 4. Выбор категорий

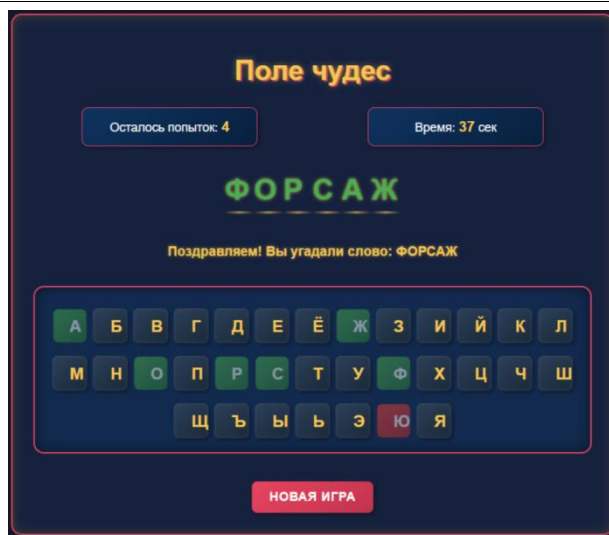


Рисунок 5. Готовое слово

Если игрок выбирает неправильную букву, система сразу реагирует визуально и текстово — кнопка буквы подсвечивается красным, дрожит для усиления эффекта, а над игровым полем появляется сообщение "Буквы "[буква]" нет в слове!", которое автоматически исчезает через 2 секунды, одновременно уменьшая счетчик оставшихся попыток и проверяя, не достигнут ли лимит ошибок для завершения игры.

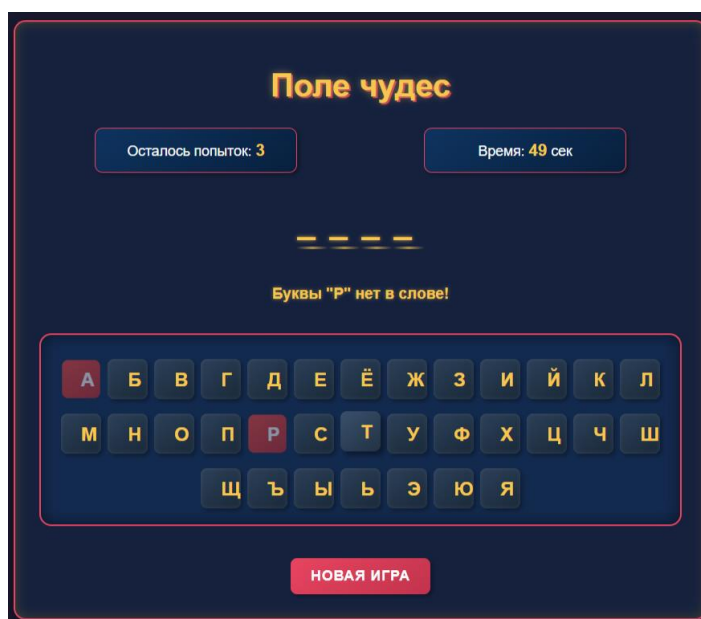


Рисунок 6. Ошибка

4 Выводы

В данной работе реализована веб-версия игры «Поле чудес», включающая ключевые интерактивные элементы: виртуальную клавиатуру, таймер с обратным отсчётом, систему ограниченных попыток и тематические категории слов. Особенности проекта стали: модульная архитектура с

чётким разделением HTML-структуры, CSS-оформления и JavaScript-логики, что обеспечивает лёгкость поддержки и развития; высокая интерактивность за счёт анимаций и мгновенной визуальной обратной связи; полная адаптивность интерфейса для различных устройств; а также образовательная ценность как демонстрационного проекта для изучения веб-технологий и инструмента развития лингвистических навыков. Предложенное решение ориентировано на преподавателей и студентов, заинтересованных в применении современных веб-технологий для повышения эффективности образовательного процесса и превращения обучения в более увлекательный и интерактивный опыт.

Библиографический список

1. Рогатин С. В., Антошкин В. А. Разработка компьютерных игр по технологии html5 //Информатика и прикладная математика. 2015. №. 21. С. 12-16.
2. Фёдоров И. А., Кузьменкова Е. Ю. Разработка игрового приложения «league of football heads» с использованием html5, css3, javascript //Новые математические методы и компьютерные технологии в проектировании, производстве и научных исследованиях. 2018. С. 311.
3. Королюк Я. А., Куликова Н. Ю. Опыт разработки и использования интерактивных образовательных ресурсов на основе html5 для онлайн-обучения школьников информатике //XXV Всероссийская студенческая научно-практическая конференция Нижневартковского государственного университета. 2023. С. 48-55.
4. Размологов Д. М., Панина А. С., Захаров П. В. Использование html5 для разработки электронного учебника //Современные тенденции развития науки и технологий. 2016. №. 4-4. С. 92-94.
5. Шегаев И. С., Полозов А. Г. Новейший политологический глоссарий: основные категории, понятия, термины. Общество с ограниченной ответственностью Издательство Перо, 2015. 242 с..

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Поле чудес</title>
  <style>
    body {
      font-family: 'Arial', sans-serif;
      background-color: #1a1a2e;
      color: white;
      text-align: center;
      padding: 20px;
      background-image: radial-gradient(circle, #2d2d44 1px, transparent 1px);
      background-size: 20px 20px;
    }

    .game-container {
      max-width: 800px;
      margin: 0 auto;
      background-color: #16213e;
      padding: 30px;
      border-radius: 15px;
      box-shadow: 0 10px 25px rgba(0, 0, 0, 0.5),
        inset 0 0 15px rgba(255, 215, 0, 0.2);
      border: 2px solid #e94560;
    }

    h1 {
      color: #f9c74f;
      text-shadow: 0 0 10px rgba(249, 199, 79, 0.5),
        2px 2px 0px rgba(233, 69, 96, 0.8);
      font-size: 2.5rem;
      margin-bottom: 20px;
    }

    .category-select {
      margin: 25px 0;
      padding: 15px;
      background-color: rgba(0, 0, 0, 0.3);
      border-radius: 10px;
      border: 1px solid #f9c74f;
    }

    .category-select h3 {
      color: #f9c74f;
      margin-top: 0;
    }

    select {
      padding: 8px 15px;
      border-radius: 5px;
      background-color: #0f3460;
      color: white;
      border: 1px solid #e94560;
      font-size: 16px;
      cursor: pointer;
    }

    .word-display {
      font-size: 2.5rem;
```

```
letter-spacing: 15px;
margin: 40px 0;
min-height: 60px;
font-weight: bold;
color: #f9c74f;
text-shadow: 0 0 5px rgba(249, 199, 79, 0.7);
}

.letter {
  display: inline-block;
  width: 40px;
  position: relative;
}

.letter::after {
  content: "";
  display: block;
  width: 100%;
  height: 3px;
  background: linear-gradient(90deg, transparent, #f9c74f, transparent);
  position: absolute;
  bottom: -10px;
  left: 0;
  opacity: 0.7;
}

.keyboard {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 8px;
  margin: 30px 0;
  padding: 20px;
  background-color: rgba(15, 52, 96, 0.5);
  border-radius: 15px;
  border: 2px solid #e94560;
  box-shadow: inset 0 0 20px rgba(0, 0, 0, 0.5);
}

.key {
  width: 50px;
  height: 50px;
  font-size: 1.5rem;
  font-weight: bold;
  background: linear-gradient(145deg, #2d4059, #1a2a3a);
  color: #f9c74f;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.2s;
  box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3),
    inset 0 0 5px rgba(255, 255, 255, 0.1);
  text-transform: uppercase;
}

.key:hover {
  transform: translateY(-3px);
  box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.4);
  background: linear-gradient(145deg, #3a516d, #223344);
}

.key:active {
  transform: translateY(1px);
```

```
}

.key.used {
  opacity: 0.5;
  transform: scale(0.95);
  cursor: not-allowed;
  box-shadow: none;
}

.key.correct {
  background: linear-gradient(145deg, #4caf50, #2e7d32);
  color: white;
}

.key.wrong {
  background: linear-gradient(145deg, #f44336, #c62828);
  color: white;
}

.game-info {
  display: flex;
  justify-content: space-around;
  margin: 30px 0;
  gap: 20px;
}

.info-box {
  background: linear-gradient(145deg, #0f3460, #08203e);
  padding: 15px 25px;
  border-radius: 10px;
  font-size: 1.1rem;
  flex: 1;
  max-width: 200px;
  box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3);
  border: 1px solid #e94560;
}

.info-box span {
  font-weight: bold;
  color: #f9c74f;
  font-size: 1.3rem;
}

.message {
  margin: 25px 0;
  font-size: 1.2rem;
  min-height: 30px;
  color: #f9c74f;
  text-shadow: 0 0 5px rgba(249, 199, 79, 0.5);
  font-weight: bold;
}

button {
  background: linear-gradient(145deg, #e94560, #c2334d);
  color: white;
  border: none;
  padding: 12px 25px;
  font-size: 1.1rem;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s;
  font-weight: bold;
  text-transform: uppercase;
```

```
        letter-spacing: 1px;
        box-shadow: 3px 3px 5px rgba(0, 0, 0, 0.3);
        margin-top: 10px;
    }

    button:hover {
        background: linear-gradient(145deg, #ff5c77, #d93a56);
        transform: translateY(-2px);
        box-shadow: 5px 5px 8px rgba(0, 0, 0, 0.4);
    }

    button:active {
        transform: translateY(1px);
    }

    .hidden {
        display: none;
    }

    /* Анимации */
    @keyframes pulse {
        0% { transform: scale(1); }
        50% { transform: scale(1.05); }
        100% { transform: scale(1); }
    }

    .pulse {
        animation: pulse 1s infinite;
    }

    /* Адаптивность */
    @media (max-width: 600px) {
        .key {
            width: 40px;
            height: 40px;
            font-size: 1.2rem;
        }

        .word-display {
            font-size: 1.8rem;
            letter-spacing: 8px;
        }

        .game-info {
            flex-direction: column;
            align-items: center;
        }

        .info-box {
            max-width: 100%;
            width: 100%;
            margin-bottom: 10px;
        }
    }
</style>
</head>
<body>
    <div class="game-container">
        <h1>Поле чудес</h1>

        <div id="start-screen">
            <p>Добро пожаловать в игру "Поле чудес"! Угадывайте буквы в загаданном слове.</p>
            <div class="category-select">
```

```

    <h3>Выберите категорию:</h3>
    <select id="category-select">
      <option value="animals">Животные</option>
      <option value="countries">Страны</option>
      <option value="movies">Фильмы</option>
      <option value="food">Еда</option>
    </select>
  </div>
  <button id="start-btn">Начать игру</button>
</div>

<div id="game-screen" class="hidden">
  <div class="game-info">
    <div class="info-box">Осталось попыток: <span id="attempts">5</span></div>
    <div class="info-box">Время: <span id="timer">60</span> сек</div>
  </div>

  <div class="word-display" id="word-display"></div>

  <div class="message" id="message"></div>

  <div class="keyboard" id="keyboard"></div>

  <button id="restart-btn">Новая игра</button>
</div>
</div>

<script>
  // Слова по категориям
  const wordsByCategory = {
    animals: ['ТИГР', 'СЛОН', 'ЖИРАФ', 'КРОКОДИЛ', 'ПАНДА', 'ЗЕБРА', 'НОСОРОГ',
'БЕГЕМОТ'],
    countries: ['РОССИЯ', 'КИТАЙ', 'БРАЗИЛИЯ', 'КАНАДА', 'ИСПАНИЯ', 'ИНДИЯ',
'АВСТРАЛИЯ', 'ЯПОНИЯ'],
    movies: ['ТИТАНИК', 'АВАТАР', 'ГАРРИПОТТЕР', 'МАТРИЦА', 'ПИРАТЫ КАРИБСКОГО
МОРЯ', 'ТЕРМИНАТОР', 'ФОРСАЖ'],
    food: ['ПИЦЦА', 'СУШИ', 'ПЕЛЬМЕНИ', 'ШАШЛЫК', 'ПАСТА', 'БОРЩ', 'ХАЧАПУРИ',
'ПЛОВ']
  };

  // Элементы интерфейса
  const startScreen = document.getElementById('start-screen');
  const gameScreen = document.getElementById('game-screen');
  const startBtn = document.getElementById('start-btn');
  const restartBtn = document.getElementById('restart-btn');
  const categorySelect = document.getElementById('category-select');
  const wordDisplay = document.getElementById('word-display');
  const keyboard = document.getElementById('keyboard');
  const messageEl = document.getElementById('message');
  const attemptsEl = document.getElementById('attempts');
  const timerEl = document.getElementById('timer');

  // Игровые переменные
  let selectedWord = '';
  let guessedLetters = [];
  let wrongAttempts = 0;
  const maxAttempts = 5;
  let timer;
  let timeLeft = 60;
  let gameActive = false;

  // Русский алфавит для клавиатуры
  const russianAlphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ';

```

```
// Инициализация клавиатуры
function initKeyboard() {
  keyboard.innerHTML = "";
  for (let letter of russianAlphabet) {
    const key = document.createElement('button');
    key.className = 'key';
    key.textContent = letter;
    key.dataset.letter = letter;
    key.addEventListener('click', () => handleLetterClick(letter));
    keyboard.appendChild(key);
  }
}

// Обработка нажатия на букву
function handleLetterClick(letter) {
  if (!gameActive) return;

  const key = document.querySelector(`.key[data-letter="${letter}"]`);
  if (key.classList.contains('used')) return;

  key.classList.add('used');

  if (selectedWord.includes(letter)) {
    // Правильная буква
    key.classList.add('correct');
    guessedLetters.push(letter);
    updateWordDisplay();

    // Анимация при правильном ответе
    key.classList.add('pulse');
    setTimeout(() => key.classList.remove('pulse'), 1000);

    // Проверка на победу
    if (checkWin()) {
      endGame(true);
    }
  } else {
    // Неправильная буква
    key.classList.add('wrong');
    wrongAttempts++;
    attemptsEl.textContent = maxAttempts - wrongAttempts;

    // Анимация при неправильном ответе
    key.style.transform = 'translateX(5px)';
    setTimeout(() => {
      key.style.transform = 'translateX(-5px)';
      setTimeout(() => key.style.transform = "", 100);
    }, 100);

    // Проверка на проигрыш
    if (wrongAttempts >= maxAttempts) {
      endGame(false);
    } else {
      showMessage(`Буквы "${letter}" нет в слове!`);
    }
  }
}

// Обновление отображения слова
function updateWordDisplay() {
  wordDisplay.innerHTML = "";
  for (let letter of selectedWord) {
    const letterSpan = document.createElement('span');
```

```
        letterSpan.className = 'letter';
        if (guessedLetters.includes(letter) || letter === ' ') {
            letterSpan.textContent = letter;
            letterSpan.style.color = '#4caf50';
        } else {
            letterSpan.textContent = '_';
        }
        wordDisplay.appendChild(letterSpan);
    }
}

// Проверка на победу
function checkWin() {
    return selectedWord.split("").every(letter =>
        guessedLetters.includes(letter) || letter === ' '
    );
}

// Отображение сообщения
function showMessage(msg) {
    messageEl.textContent = msg;
    setTimeout(() => {
        if (messageEl.textContent === msg) {
            messageEl.textContent = "";
        }
    }, 2000);
}

// Запуск таймера
function startTimer() {
    timeLeft = 60;
    timerEl.textContent = timeLeft;

    clearInterval(timer);
    timer = setInterval(() => {
        timeLeft--;
        timerEl.textContent = timeLeft;

        if (timeLeft <= 10) {
            timerEl.style.color = '#f44336';
            timerEl.classList.add('pulse');
        }

        if (timeLeft <= 0) {
            endGame(false);
        }
    }, 1000);
}

// Начало игры
function startGame() {
    const category = categorySelect.value;
    const words = wordsByCategory[category];
    selectedWord = words[Math.floor(Math.random() * words.length)];

    guessedLetters = [];
    wrongAttempts = 0;
    gameActive = true;

    attemptsEl.textContent = maxAttempts;
    messageEl.textContent = "";
    timerEl.style.color = '#f9c74f';
    timerEl.classList.remove('pulse');
```

```
        initKeyboard();
        updateWordDisplay();
        startTimer();

        startScreen.classList.add('hidden');
        gameScreen.classList.remove('hidden');
    }

    // Завершение игры
    function endGame(isWin) {
        gameActive = false;
        clearInterval(timer);

        if (isWin) {
            showMessage('Поздравляем! Вы угадали слово: ${selectedWord}');
            wordDisplay.classList.add('pulse');
        } else {
            showMessage('Игра окончена! Загаданное слово: ${selectedWord}');
        }
    }

    // Обработчики событий
    startBtn.addEventListener('click', startGame);
    restartBtn.addEventListener('click', () => {
        wordDisplay.classList.remove('pulse');
        gameScreen.classList.add('hidden');
        startScreen.classList.remove('hidden');
    });

    // Инициализация при загрузке
    initKeyboard();
</script>
</body>
</html>
```