

Образовательная игра «Водопровод» как инструмент развития логического мышления

Екимова Яна Сергеевна

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается методика создания интерактивной логической игры «Водопровод» с применением технологий HTML5, CSS3 и JavaScript. Разработанное решение позволяет воссоздать реалистичный процесс прокладки трубопроводов с механизмом соединения элементов, системой проверки корректности пути жидкости и динамическим таймером. Особое внимание уделено реализации алгоритмов поиска пути по принципу обхода в ширину (BFS), визуализации игрового процесса с анимацией водяного потока, организации взаимодействия элементов трубопровода и обработке пользовательских действий. Представленное решение демонстрирует эффективное сочетание современных веб-технологий с алгоритмическими подходами для создания развивающих логических приложений. Материал предназначен для преподавателей и студентов, заинтересованных в применении игровых механик для развития пространственного мышления и изучения практической реализации алгоритмов теории графов в веб-среде.

Ключевые слова: Водопровод, HTML5, CSS, JavaScript.

Educational game "Plumbing" as a tool for developing logical thinking

Ekimova Yana Sergeevna

Sholom-Aleichem Priamurskiy State University

Student

Abstract

The article discusses the technique of creating an interactive logic game "Aqueduct" using HTML5, CSS3 and JavaScript technologies. The developed solution allows you to recreate a realistic process of laying pipelines with a mechanism for connecting elements, a system for checking the correctness of the liquid path and a dynamic timer. Special attention is paid to the implementation of pathfinding algorithms based on the breadth-first traversal (BFS) principle, visualization of the gameplay with animation of a water stream, organization of interaction of pipeline elements and processing of user actions. The presented solution demonstrates an effective combination of modern web technologies with algorithmic approaches to create developing logical applications. The material is intended for teachers and students interested in using game mechanics to develop spatial thinking and study the practical implementation of graph theory algorithms in a web environment.

Keywords: Plumbing, HTML5, CSS, JavaScript.

1 Введение

1.1 Актуальность

В современном мире веб-технологий разработка игровых приложений на HTML5 и JavaScript приобретает особую значимость, поскольку позволяет создавать кроссплатформенные проекты, работающие непосредственно в браузере без необходимости установки дополнительного программного обеспечения. Актуальность реализации классической игры Тетрис заключается в том, что она служит идеальным учебным полигоном для освоения ключевых аспектов программирования: работы с DOM-структурами, обработки пользовательского ввода, реализации игровой логики и анимации, а также демонстрирует эффективное применение алгоритмов для проверки коллизий и оптимизации производительности. Кроме того, возрождение интереса к ретро-играм и их адаптация под современные веб-стандарты отвечает запросам широкой аудитории пользователей, обеспечивая доступность и сохранение культурного наследия игровой индустрии.

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]. Я. А. Королук, Н.Ю. Куликова 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 создание полнофункциональной веб-версии логической игры "Водопровод" с использованием языков HTML5, CSS3 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: объявим doctype, создадим корневой элемент html с русской языковой разметкой, в разделе head добавим метатеги для корректного отображения и адаптивности, установим заголовок страницы. В теле документа разместим игровой интерфейс с контейнерами для заголовка, панели инструментов с выбором типов труб, игрового поля для размещения элементов водопровода, информационной панели с таймером и счетчиком результатов, а также блока управления с кнопками действий и всплывающего окна для отображения сообщений, оставив области для CSS-стилей и JavaScript-кода пустыми для последующей реализации игровой логики соединения труб от источника к стоку.

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

</style>
</head>
<body>
  <div class="container">
    <header>
      <h1>ВОДОПРОВОД</h1>
      <div class="instructions">
        <p>Соедините трубы от источника (И) к стоку (С). Вода может
подключаться с любой стороны!</p>
      </div>
    </header>

    <div class="game-area">
      <div class="tools-panel">
        <h2>Выбор трубы</h2>

        <div class="pipe-option" data-type="horizontal">
          <div class="pipe"><div class="straight-horizontal"></div></div>
          <div class="pipe-name">Горизонтальная —</div>
        </div>

        <div class="pipe-option" data-type="vertical">
          <div class="pipe"><div class="straight-vertical"></div></div>
          <div class="pipe-name">Вертикальная |</div>
        </div>


```

```
<div class="pipe-option" data-type="corner">
  <div class="pipe"><div class="corner corner-0"></div></div>
  <div class="pipe-name">Угол ↘ (поворот кликом)</div>
</div>

<div class="game-board" id="game-board"></div>

<div class="info-panel">
  <div class="timer">
    <h2>Осталось времени</h2>
    <div class="timer-value" id="timer">01:00</div>
  </div>

  <div class="score">
    <h2>Результат</h2>
    <div>Подключено:      <span      id="completed">0</span></div>
    <div>id="total">2</div></div>

  <div class="controls">
    <button id="start-btn">▶ Начать игру</button>
    <button id="check-btn" disabled>⬇ Проверить путь</button>
    <button id="reset-btn">↺ Сбросить</button>
    <button id="undo-btn" disabled>⏮ Отменить</button>
  </div>
</div>

<div class="message" id="message">
  <h2 id="message-title">Результат</h2>
  <p id="message-text"></p>
  <button id="restart-btn">Играть снова</button>
</div>

<script>

</script>
</body>
</html>
```

ВОДОПРОВОД

Соедините трубы от источника (И) к стоку (С). Вода может подключаться с любой стороны!

Выбор трубы

Горизонтальная —
 Вертикальная |
 Угол > (поворот кликом)

Осталось времени

01:00

Результат

Подключено: 0/2

Начать игру
Проверить путь
Сбросить
Отменить

Результат

Играть снова

Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей: зададим базовые настройки сброса отступов и модели box-sizing, определим градиентный фон для всего документа и стилизуем основной контейнер с полупрозрачным темным фоном. Реализуем адаптивную сетку игрового поля с клетками для размещения труб, создадим визуальное представление различных типов трубопроводов (прямых и угловых) с цветовым кодированием и эффектами анимации воды, оформим панели инструментов и информации с использованием современных CSS-свойств для теней, градиентов и плавных переходов, а также разработаем стили для специальных элементов (источника, стока) и всплывающих сообщений с фиксированным позиционированием для создания интуитивного и визуально привлекательного интерфейса игры.

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

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background: linear-gradient(135deg, #1a2a6c, #b21f1f, #fdbb2d);
  color: #fff;
  min-height: 100vh;
  padding: 20px;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.container {
```

```
    max-width: 1200px;
    width: 100%;
    background: rgba(0, 0, 0, 0.7);
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);
  }

  header {
    text-align: center;
    margin-bottom: 25px;
  }

  h1 {
    font-size: 2.5rem;
    margin-bottom: 10px;
    color: #4fc3f7;
    text-shadow: 0 2px 5px rgba(0, 0, 0, 0.5);
  }

  .instructions {
    background: rgba(255, 255, 255, 0.1);
    padding: 15px;
    border-radius: 10px;
    margin-bottom: 20px;
    text-align: center;
    font-size: 1rem;
  }

  .game-area {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
    justify-content: center;
  }

  .tools-panel {
    flex: 1;
    min-width: 250px;
    background: rgba(0, 75, 135, 0.8);
    border-radius: 10px;
    padding: 15px;
    display: flex;
    flex-direction: column;
    gap: 15px;
  }

  .pipe-option {
    background: rgba(255, 255, 255, 0.9);
    border-radius: 8px;
    padding: 10px;
    display: flex;
```

```
        flex-direction: column;
        align-items: center;
        cursor: pointer;
        transition: all 0.3s;
    }

    .pipe-option:hover {
        transform: scale(1.05);
        box-shadow: 0 0 15px #4fc3f7;
    }

    .pipe-option.selected {
        background: #4fc3f7;
        box-shadow: 0 0 15px #4fc3f7;
    }

    .pipe-name {
        font-weight: bold;
        color: #1565c0;
        margin-top: 5px;
        font-size: 0.9rem;
    }

    .game-board {
        flex: 2;
        min-width: 500px;
        display: grid;
        grid-template-columns: repeat(8, 1fr);
        grid-template-rows: repeat(8, 1fr);
        gap: 3px;
        background: #0d47a1;
        padding: 10px;
        border-radius: 10px;
    }

    .cell {
        width: 60px;
        height: 60px;
        background: rgba(255, 255, 255, 0.1);
        border-radius: 5px;
        display: flex;
        justify-content: center;
        align-items: center;
        cursor: pointer;
        transition: background 0.3s;
        position: relative;
    }

    .cell:hover {
        background: rgba(255, 255, 255, 0.2);
    }
```

```
.info-panel {  
  flex: 1;  
  min-width: 250px;  
  background: rgba(0, 75, 135, 0.8);  
  border-radius: 10px;  
  padding: 15px;  
  display: flex;  
  flex-direction: column;  
  gap: 15px;  
}
```

```
.timer, .score {  
  background: rgba(0, 0, 0, 0.3);  
  padding: 15px;  
  border-radius: 10px;  
  text-align: center;  
}
```

```
.timer-value {  
  font-size: 2rem;  
  font-weight: bold;  
  color: #4fc3f7;  
}
```

```
.controls {  
  display: flex;  
  flex-direction: column;  
  gap: 10px;  
}
```

```
button {  
  padding: 12px;  
  border: none;  
  border-radius: 8px;  
  background: #4fc3f7;  
  color: #000;  
  font-weight: bold;  
  cursor: pointer;  
  transition: all 0.3s;  
}
```

```
button:hover {  
  background: #29b6f6;  
  transform: translateY(-2px);  
}
```

```
button:disabled {  
  background: #78909c;  
  cursor: not-allowed;  
  transform: none;  
}
```

```
/* Стили труб */
.pipe {
  width: 50px;
  height: 50px;
  position: relative;
}

.straight-horizontal {
  width: 50px; height: 18px;
  background: #ff9800;
  border: 3px solid #e65100;
  border-radius: 9px;
  position: absolute;
  top: 21px; left: 0;
}

.straight-vertical {
  width: 18px; height: 50px;
  background: #ff9800;
  border: 3px solid #e65100;
  border-radius: 9px;
  position: absolute;
  left: 21px; top: 0;
}

.corner {
  width: 30px; height: 30px;
  background: #ff9800;
  border: 3px solid #e65100;
  position: absolute;
  border-radius: 15px;
}

.corner-0 { /* ↘ */
  border-top: 3px solid #e65100;
  border-left: 3px solid #e65100;
  border-top-right-radius: 0;
  border-bottom-left-radius: 15px;
  top: 0; left: 0;
}

.corner-1 { /* ↙ */
  border-top: 3px solid #e65100;
  border-right: 3px solid #e65100;
  border-top-left-radius: 0;
  border-bottom-right-radius: 15px;
  top: 0; right: 0;
}

.corner-2 { /* ↖ */
  border-bottom: 3px solid #e65100;
  border-right: 3px solid #e65100;
```

```
border-top-left-radius: 15px;
border-bottom-right-radius: 0;
bottom: 0; right: 0;
}

.corner-3 { /* ↗ */
border-bottom: 3px solid #e65100;
border-left: 3px solid #e65100;
border-top-right-radius: 15px;
border-bottom-left-radius: 0;
bottom: 0; left: 0;
}

.source, .drain {
width: 40px;
height: 40px;
border-radius: 50%;
display: flex;
justify-content: center;
align-items: center;
font-weight: bold;
color: white;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
}

.source {
background: radial-gradient(circle at 30% 30%, #2196F3, #0d47a1);
}

.drain {
background: radial-gradient(circle at 30% 30%, #4CAF50, #1b5e20);
}

.water {
position: absolute;
background: rgba(33, 150, 243, 0.8);
z-index: 1;
border-radius: 5px;
animation: ripple 1s infinite;
}

@keyframes ripple {
0% { opacity: 0.6; }
50% { opacity: 1; }
100% { opacity: 0.6; }
}

.message {
position: fixed;
top: 50%;
left: 50%;
transform: translate(-50%, -50%);
```

```

background: rgba(0, 0, 0, 0.9);
padding: 30px;
border-radius: 15px;
text-align: center;
box-shadow: 0 0 30px rgba(79, 195, 247, 0.8);
z-index: 100;
display: none;
}

.message h2 {
margin-bottom: 20px;
color: #4fc3f7;
}

.message p {
margin-bottom: 20px;
font-size: 1.2rem;
}

```

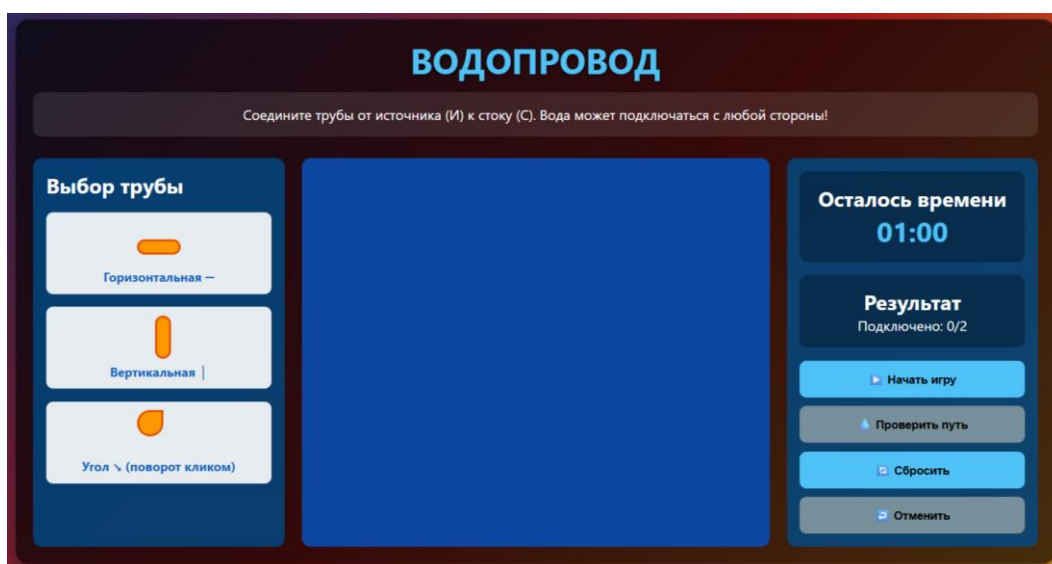


Рисунок 2. Добавление CSS-стилей

Затем инициализируем игровую логику: объявим конфигурационные параметры размера поля, лимита времени и типов труб с их соединениями, получим ссылки на DOM-элементы управления и отображения. Создадим объект `gameState` для хранения текущего состояния игры, включая выбранный тип трубы, матрицу игрового поля, расположение источников и стоков, оставшееся время, статус игрового процесса, счетчик выполненных соединений и историю действий для реализации функции отмены, что обеспечит структурированное управление данными в процессе игры.

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

```

document.addEventListener('DOMContentLoaded', function() {
  const config = {
    boardSize: 8,

```

```

    timeLimit: 60,
    pipeTypes: {
      horizontal: { connections: ['left', 'right'] },
      vertical: { connections: ['top', 'bottom'] },
      corner: {
        rotations: [
          ['top', 'left'], // 0: ↘
          ['top', 'right'], // 1: ↗
          ['bottom', 'right'], // 2: ↖
          ['bottom', 'left'] // 3: ↙
        ]
      }
    }
  }
};

const gameBoard = document.getElementById('game-board');
const timerElement = document.getElementById('timer');
const startBtn = document.getElementById('start-btn');
const checkBtn = document.getElementById('check-btn');
const resetBtn = document.getElementById('reset-btn');
const undoBtn = document.getElementById('undo-btn');
const messageElement = document.getElementById('message');
const messageTitle = document.getElementById('message-title');
const messageText = document.getElementById('message-text');
const restartBtn = document.getElementById('restart-btn');
const completedElement = document.getElementById('completed');
const totalElement = document.getElementById('total');
const pipeOptions = document.querySelectorAll('.pipe-option');

let gameState = {
  selectedPipe: null,
  board: [],
  sources: [],
  drains: [],
  timeLeft: config.timeLimit,
  timerInterval: null,
  isPlaying: false,
  completed: 0,
  actionHistory: []
};

```

Далее реализуем основные игровые функции: создадим методы инициализации игрового поля с очисткой состояния, размещения источников и стоков в случайных позициях, установки и поворота труб с сохранением истории действий для отмены. Реализуем алгоритм поиска пути воды через анализ соединений между клетками с использованием обхода в ширину (BFS), добавим визуальную анимацию потока воды с последовательным заполнением труб и систему завершения игры с проверкой достижения стоков. Организуем работу таймера с обратным отсчетом и визуальной индикацией, разработаем механизм отмены последнего действия через стек истории, а

также создадим функции отображения результатов игры и управления состоянием кнопок интерфейса.

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

```
// === Инициализация игры ===
function initGame() {
  createGameBoard();
  placeSourcesAndDrains();
  resetState();
}

function resetState() {
  gameState.selectedPipe = null;
  gameState.timeLeft = config.timeLimit;
  gameState.isPlaying = false;
  gameState.completed = 0;
  gameState.actionHistory = [];
  updateTimerDisplay();

  pipeOptions.forEach(option => option.classList.remove('selected'));
  startBtn.disabled = false;
  checkBtn.disabled = true;
  undoBtn.disabled = true;
  completedElement.textContent = '0';
}

// === Создание поля ===
function createGameBoard() {
  gameBoard.innerHTML = "";
  gameState.board = Array(config.boardSize).fill().map(() =>
    Array(config.boardSize).fill(null));

  for (let row = 0; row < config.boardSize; row++) {
    for (let col = 0; col < config.boardSize; col++) {
      const cell = document.createElement('div');
      cell.className = 'cell';
      cell.dataset.row = row;
      cell.dataset.col = col;

      cell.addEventListener('click', () => {
        if (!gameState.isPlaying) return;

        const cellData = gameState.board[row][col];

        // Поворот угла
        if (cellData && cellData.type === 'corner') {
          rotateCorner(row, col);
          return;
        }

        // Установка новой трубы
        if (gameState.selectedPipe) {
```

```

        const prev = JSON.parse(JSON.stringify(cellData));
        placePipe(row, col);
        gameState.actionHistory.push({ row, col, prev });
        undoBtn.disabled = false;
    }
});

    gameBoard.appendChild(cell);
}
}
}

// === Размещение источников и стоков ===
function placeSourcesAndDrains() {
    gameState.sources = [];
    gameState.drains = [];

    for (let i = 0; i < 2; i++) {
        let row;
        do {
            row = Math.floor(Math.random() * config.boardSize);
        } while (gameState.board[row][0]);
        setCell(row, 0, 'source', '<div class="source">И</div>', 'source-cell');
        gameState.sources.push({ row, col: 0 });
    }

    for (let i = 0; i < 2; i++) {
        let row;
        do {
            row = Math.floor(Math.random() * config.boardSize);
        } while (gameState.board[row][config.boardSize - 1]);
        setCell(row, config.boardSize - 1, 'drain', '<div class="drain">C</div>',
'drain-cell');
        gameState.drains.push({ row, col: config.boardSize - 1 });
    }

    totalElement.textContent = '2';
}

function setCell(row, col, type, html, className = "") {
    const cell = getCellElement(row, col);
    cell.innerHTML = html;
    if (className) cell.classList.add(className);
    gameState.board[row][col] = { type };
}

function getCellElement(row, col) {
    return document.querySelector(`.cell[data-row="${row}"][data-
col="${col}"]`);
}

function getConnections(type, rotation = 0) {

```

```

    if (type === 'source' || type === 'drain') return ['top', 'right', 'bottom', 'left'];
    if (type === 'horizontal') return ['left', 'right'];
    if (type === 'vertical') return ['top', 'bottom'];
    if (type === 'corner') return config.pipeTypes.corner.rotations[rotation];
    return [];
  }

  // === Размещение трубы ===
  function placePipe(row, col) {
    const cell = getCellElement(row, col);
    const current = gameState.board[row][col];
    if (current && ['source', 'drain'].includes(current.type)) return;

    Array.from(cell.children).forEach(ch => ch.remove());

    const pipe = gameState.selectedPipe;
    const pipeVisual = document.createElement('div');
    pipeVisual.className = 'pipe';

    if (pipe === 'horizontal') {
      pipeVisual.innerHTML = '<div class="straight-horizontal"></div>';
      gameState.board[row][col] = { type: 'horizontal', connections: ['left', 'right'] };
    } else if (pipe === 'vertical') {
      pipeVisual.innerHTML = '<div class="straight-vertical"></div>';
      gameState.board[row][col] = { type: 'vertical', connections: ['top', 'bottom'] };
    } else if (pipe === 'corner') {
      pipeVisual.innerHTML = '<div class="corner corner-0"></div>';
      const con = getConnections('corner', 0);
      gameState.board[row][col] = { type: 'corner', rotation: 0, connections: con };
    }

    cell.appendChild(pipeVisual);
  }

  // === Поворот угла ===
  function rotateCorner(row, col) {
    const cell = getCellElement(row, col);
    const current = gameState.board[row][col];
    if (current.type !== 'corner') return;

    const prev = JSON.parse(JSON.stringify(current));
    const newRot = (current.rotation + 1) % 4;
    const con = getConnections('corner', newRot);

    cell.innerHTML = "";
    const pipeVisual = document.createElement('div');
    pipeVisual.className = 'pipe';
    pipeVisual.innerHTML = `<div class="corner corner-${newRot}"></div>`;
    cell.appendChild(pipeVisual);

    gameState.board[row][col] = { type: 'corner', rotation: newRot, connections: con
  };

```

```

        gameState.actionHistory.push({ row, col, prev });
        undoBtn.disabled = false;
    }

    // === Таймер ===
    function updateTimerDisplay() {
        const min = Math.floor(gameState.timeLeft / 60).toString().padStart(2, '0');
        const sec = (gameState.timeLeft % 60).toString().padStart(2, '0');
        timerElement.textContent = `${min}:${sec}`;
        timerElement.style.color = gameState.timeLeft <= 10 ? '#ff5252' : '#4fc3f7';
    }

    // === Начать игру ===
    function startGame() {
        gameState.isPlaying = true;
        gameState.timeLeft = config.timeLimit;
        gameState.completed = 0;
        completedElement.textContent = '0';
        gameState.actionHistory = [];

        startBtn.disabled = true;
        checkBtn.disabled = false;
        undoBtn.disabled = true;

        gameState.timerInterval = setInterval(() => {
            gameState.timeLeft--;
            updateTimerDisplay();
            if (gameState.timeLeft <= 0) endGame();
        }, 1000);
    }

    // === Проверить путь (можно в любой момент!) ===
    function checkWaterFlow() {
        clearAllWater();
        gameState.completed = 0;

        gameState.sources.forEach(source => {
            const path = findPath(source);
            animateWater(path);
            if (reachesDrain(path)) gameState.completed++;
        });

        completedElement.textContent = gameState.completed;
    }

    // === Найти путь ===
    function findPath(start) {
        const visited = Array(config.boardSize).fill().map(() =>
        Array(config.boardSize).fill(false));
        const queue = [{ row: start.row, col: start.col }];
        const path = [start];
        visited[start.row][start.col] = true;

```

```

const directions = {
  top: { dr: -1, dc: 0, opp: 'bottom' },
  right: { dr: 0, dc: 1, opp: 'left' },
  bottom: { dr: 1, dc: 0, opp: 'top' },
  left: { dr: 0, dc: -1, opp: 'right' }
};

while (queue.length > 0) {
  const { row, col } = queue.shift();
  const cell = gameState.board[row][col];
  if (!cell) continue;

  const connections = cell.connections || getConnections(cell.type);

  for (const [dir, { dr, dc, opp }] of Object.entries(directions)) {
    if (connections.includes(dir)) {
      const nr = row + dr;
      const nc = col + dc;
      if (nr >= 0 && nr < config.boardSize && nc >= 0 && nc <
config.boardSize && !visited[nr][nc]) {
        const next = gameState.board[nr][nc];
        if (!next) continue;

        const      nextConnections      =      next.connections      ||
getConnections(next.type);
        if (nextConnections.includes(opp)) {
          visited[nr][nc] = true;
          path.push({ row: nr, col: nc });
          queue.push({ row: nr, col: nc });
        }
      }
    }
  }
}

return path;
}

function reachesDrain(path) {
  const last = path[path.length - 1];
  return gameState.drains.some(d => d.row === last.row && d.col === last.col);
}

// === Анимация воды ===
function animateWater(path) {
  path.forEach((pos, i) => {
    setTimeout(() => {
      const cell = getCellElement(pos.row, pos.col);
      const water = document.createElement('div');
      water.className = 'water';

```

```

        const pipe = gameState.board[pos.row][pos.col];
        if (pipe.type === 'horizontal') {
            water.style.width = '50px'; water.style.height = '18px';
            water.style.top = '21px'; water.style.left = '0';
        } else if (pipe.type === 'vertical') {
            water.style.width = '18px'; water.style.height = '50px';
            water.style.left = '21px'; water.style.top = '0';
        } else if (pipe.type === 'corner') {
            water.style.width = '30px'; water.style.height = '30px';
            water.style.borderRadius = '15px';
        }

        cell.appendChild(water);
    }, i * 250);
});
}

function clearAllWater() {
    document.querySelectorAll('.water').forEach(el => el.remove());
}

// === Завершение игры (вызывается при таймере или при "Проверить") ===
function endGame() {
    clearInterval(gameState.timerInterval);
    gameState.isPlaying = false;
    checkBtn.disabled = true;
    checkWaterFlow();
    showMessage();
}

function showMessage() {
    messageTitle.textContent = gameState.completed === 2 ? '🎉 Поздравляем!' :
    '🕒 Время вышло!';
    messageText.textContent = `Вы подключили ${gameState.completed} из 2
стоков.`;
    messageElement.style.display = 'block';
}

// === Отмена действия ===
function undoAction() {
    if (gameState.actionHistory.length === 0) return;
    const action = gameState.actionHistory.pop();
    const cell = getCellElement(action.row, action.col);
    cell.innerHTML = "";

    if (action.prev) {
        gameState.board[action.row][action.col] =
JSON.parse(JSON.stringify(action.prev));
        if (action.prev.type === 'source') {
            cell.innerHTML = '<div class="source">И</div>';
            cell.classList.add('source-cell');
        } else if (action.prev.type === 'drain') {

```

```

        cell.innerHTML = '<div class="drain">C</div>';
        cell.classList.add('drain-cell');
    } else if (action.prev.type === 'corner') {
        const rot = action.prev.rotation;
        const div = document.createElement('div');
        div.className = 'pipe';
        div.innerHTML = `<div class="corner corner-${rot}"></div>`;
        cell.appendChild(div);
    } else {
        const div = document.createElement('div');
        div.className = 'pipe';
        cell.appendChild(div);
    }
} else {
    gameState.board[action.row][action.col] = null;
}

undoBtn.disabled = gameState.actionHistory.length === 0;
}

```

Последнее подключим обработчики событий: назначим кнопкам старта, проверки, сброса и отмены соответствующие функции управления игровым процессом, реализуем выбор типа трубы через выделение элемента на панели инструментов и сохранение выбора в состоянии игры, а также добавим обработчик для кнопки перезапуска, скрывающий сообщение о результате и инициализирующий новую игровую сессию. Завершим код вызовом функции `initGame` для первоначальной настройки игрового поля и подготовки к старту, обеспечив полную функциональность всех интерактивных элементов интерфейса и их согласованную работу с игровой логикой.

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

```

// === Обработчики событий ===
startBtn.addEventListener('click', startGame);
checkBtn.addEventListener('click', () => {
    if (gameState.isPlaying) {
        endGame(); // Останавливает таймер и запускает проверку
    }
});
resetBtn.addEventListener('click', initGame);
undoBtn.addEventListener('click', undoAction);
restartBtn.addEventListener('click', () => {
    messageElement.style.display = 'none';
    initGame();
});

pipeOptions.forEach(option => {
    option.addEventListener('click', () => {
        if (gameState.isPlaying) {
            pipeOptions.forEach(opt => opt.classList.remove('selected'));
            option.classList.add('selected');
        }
    });
});

```

```
        gameState.selectedPipe = option.dataset.type;
    }
});

// Запуск
initGame();
});
```



Рисунок 3. Процесс игры

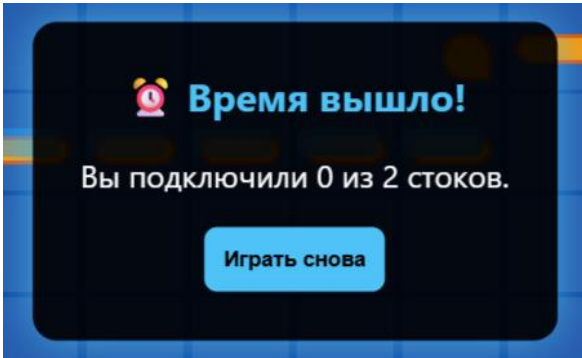


Рисунок 4. Проигрыш

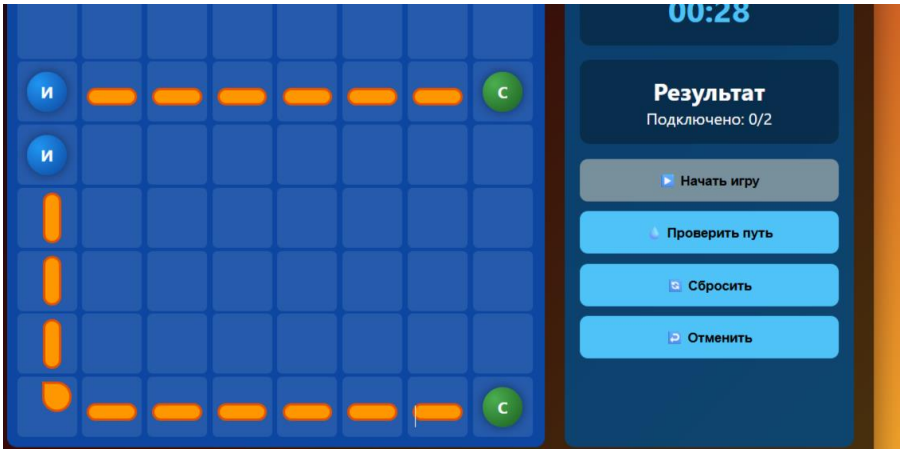


Рисунок 5. Проигрыш

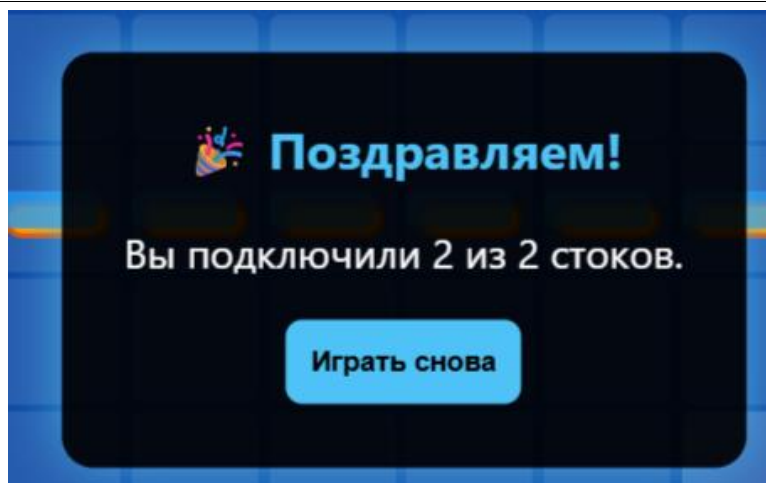


Рисунок 6. Выигрыш

4 Выводы

В данной работе подробно рассмотрено создание логической игры "Водопровод" на современном веб-стеке с использованием технологий HTML5, CSS3 и JavaScript, которая включает реалистичную механику соединения трубопроводов, систему проверки корректности пути жидкости, таймер обратного отсчета и адаптивный интерфейс для различных устройств. Реализованы ключевые игровые механизмы: генерация случайных расположений источников и стоков, визуализация различных типов труб с возможностью поворота угловых элементов, алгоритм поиска пути по принципу обхода в ширину (BFS), а также система анимации потока воды с последовательным заполнением. Разработан интуитивно понятный игровой интерфейс с визуальным представлением элементов водопровода, панелью инструментов для выбора труб и информационной панелью с отслеживанием прогресса.

Предложенное решение демонстрирует эффективное применение алгоритмов теории графов и современных веб-технологий для создания развивающих логических игр, ориентированных на развитие пространственного мышления и решение инженерных задач. Материал предназначен для преподавателей и студентов, заинтересованных в применении игровых механик для развития логического мышления и изучения практического применения алгоритмов поиска пути в интерактивных веб-приложениях.

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 онлайнобучения 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. М.: Перо, 2015. 242 с.