

Разработка браузерной аркадной игры "Студент-бегун" на основе HTML, CSS и JavaScript

Екимова Яна Сергеевна

*Приамурский государственный университет имени Шолом-Алейхема
Студент*

Аннотация

В статье представлена методика разработки аркадной браузерной игры «Студент-бегун» с использованием технологий HTML5, CSS3 и чистого JavaScript без привлечения сторонних библиотек. Реализовано полноценное игровое приложение с динамической анимацией персонажа, системой генерации препятствий, проверкой коллизий, таймером, индикатором прогресса и поддержкой нескольких уровней сложности. Особое внимание уделено построению интерактивного пользовательского интерфейса, реализации плавного прыжка, обработке событий с клавиатуры и мыши, а также организации игрового цикла на основе requestAnimationFrame. Предложенное решение демонстрирует эффективное применение базовых веб-технологий для создания кроссплатформенных интерактивных приложений, пригодных для использования в образовательной практике. Материал ориентирован на преподавателей информатики и студентов, изучающих веб-разработку, и направлен на популяризацию геймификации как средства повышения мотивации и наглядности в процессе обучения программированию.

Ключевые слова: Студент-бегун, анимация, HTML5, CSS, JavaScript.

Development of a browser-based arcade game "Student Runner" based on HTML, CSS and JavaScript

Ekimova Yana Sergeevna

*Sholom Aleichem Priamurskiy State University
Student*

Abstract

The article presents a methodology for developing the arcade browser game "Student Runner" using HTML5, CSS3 and pure JavaScript technologies without involving third-party libraries. A full-fledged game application has been implemented with dynamic character animation, an obstacle generation system, collision checking, a timer, a progress indicator, and support for multiple difficulty levels. Special attention is paid to the construction of an interactive user interface, the implementation of a smooth jump, the processing of keyboard and mouse events, as well as the organization of a game cycle based on requestAnimationFrame. The proposed solution demonstrates the effective use of basic web technologies to create

cross-platform interactive applications suitable for use in educational practice. The material is aimed at computer science teachers and students studying web development, and is aimed at popularizing gamification as a means of increasing motivation and visibility in the learning process of programming.

Keywords: Student runner, animation, HTML5, CSS, JavaScript.

1 Введение

1.1 Актуальность

В современном образовательном и цифровом пространстве разработка интерактивных приложений с использованием веб-технологий HTML5, CSS3 и JavaScript приобретает всё большую актуальность, поскольку позволяет создавать кроссплатформенные, легко распространяемые и доступные в любом браузере проекты без необходимости установки дополнительного программного обеспечения. Актуальность создания игры «Студент-бегун» заключается в её педагогической и практической ценности: она выступает эффективным учебным инструментом для освоения фундаментальных принципов веб-программирования — управления DOM, обработки событий, анимации с использованием CSS и реализации игрового цикла на чистом JavaScript. Проект демонстрирует применение ключевых концепций, таких как проверка коллизий, динамическая генерация элементов, работа с таймерами и состояние игры, что делает его ценным примером для студентов и начинающих разработчиков. Кроме того, использование игровой механики, знакомой по жанру бесконечных раннеров, повышает мотивацию к обучению, превращая процесс освоения кода в увлекательное и наглядное занятие. В условиях растущего интереса к геймификации образования подобные проекты способствуют повышению вовлечённости, развитию логического мышления и формированию практических навыков программирования в контексте реальных приложений.

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 аркадной браузерной игры «Студент-бегун» с использованием современных веб-технологий — HTML5, CSS3 и чистого JavaScript.

2 Материалы и методы

Для создания приложения используется язык программирования HTML5. В качестве среды разработки используется текстовый 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 браузерной игры «Студент-бегун» — аркадного приложения, в котором игрок управляет студентом, преодолевающим препятствия на беговой дорожке; документ соответствует стандартам HTML5 и включает корректные метаданные: указание кодировки UTF-8, адаптивный viewport и заголовок страницы, а вся игровая сцена организована внутри контейнера #game-container, включающего панель интерфейса с отображением дистанции и времени, визуальный индикатор прогресса, игровое поле с персонажем, построенным из модульных CSS-элементов (голова, тело, руки, ноги, рюкзак), а также экраны старта с выбором сложности и завершения игры с финальным счётом и кнопкой перезапуска — всё это создаст семантически правильную, гибкую и подготовленную к стилизации и программной логике основу для реализации интерактивной 2D-игры на чистом JavaScript.

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

</style>
</head>
<body>
  <div id="game-container">
    <div id="game-ui">
      <div id="score">Дистанция: 0%</div>
      <div id="timer">Время: 60с</div>
    </div>
    <div id="progress-container">
      <div id="progress-bar"></div>
```

```
</div>
<div id="game-canvas">
  <div id="ground"></div>
  <div id="player">
    <div class="student">
      <div class="student-hair"></div>
      <div class="student-head"></div>
      <div class="student-glasses">
        <div class="student-eye"></div>
        <div class="student-eye"></div>
      </div>
      <div class="student-body"></div>
      <div class="student-backpack"></div>
      <div class="student-arm left"></div>
      <div class="student-arm right"></div>
      <div class="student-leg left"></div>
      <div class="student-leg right"></div>
    </div>
  </div>

  <div id="start-screen">
    <h1>Студент-бегун</h1>
    <p>Прыгай через препятствия на земле!</p>
    <button class="difficulty-btn" id="easy">Легко</button>
    <button class="difficulty-btn" id="medium">Средне</button>
    <button class="difficulty-btn" id="hard">Тяжело</button>
  </div>

  <div id="end-screen">
    <h1>Игра окончена!</h1>
    <p>Результат: <span id="final-score">0</span>%</p>
    <button class="difficulty-btn" id="restart">Заново</button>
  </div>
</div>

<script>

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

Дистанция: 0%
Время: 60с

Студент-бегун

Прыгай через препятствия на земле!

Легко Средне Тяжело

Игра окончена!

Результат: 0%

Заново

Рисунок 1. Основная структура HTML-документа

Далее добавим блок CSS-стилей, который формирует визуальное оформление и динамическое поведение игры «Студент-бегун»: сброс стандартных отступов, единый стиль шрифтов и применение `box-sizing: border-box` обеспечивают кроссбраузерную согласованность, а гибкая разметка на основе `Flexbox` и абсолютное позиционирование создают адаптивную и чётко структурированную игровую сцену; фон страницы выполнен в мягких голубых тонах, игровое поле выделено с закруглёнными углами и тенью, а интерфейс включает стилизованные панели счёта и таймера, а также плавно анимируемую полосу прогресса с градиентным заполнением; персонаж студента детально проработан с помощью CSS-элементов — от головы и очков до рюкзака и конечностей, каждая из которых анимирована с использованием `@keyframes` для имитации бега; реализован плавный прыжок через анимацию `transform: translateY`, препятствия (книга, кофе, ноутбук, сон) корректно позиционируются на земле и движутся справа налево, а экраны старта и завершения игры перекрывают сцену, обеспечивая пользовательскую навигацию с помощью стилизованных кнопок с градиентной заливкой и интерактивными эффектами.

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

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
  font-family: 'Arial Rounded MT Bold', 'Arial', sans-serif;  
}  
  
body {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
  min-height: 100vh;  
  background: #a0d8f1;  
  overflow: hidden;  
}
```

```
#game-container {
  width: 800px;
  position: relative;
}

#game-ui {
  display: flex;
  justify-content: space-between;
  margin-bottom: 10px;
  background: rgba(255, 255, 255, 0.9);
  padding: 10px 20px;
  border-radius: 20px;
  box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
}

#progress-container {
  width: 100%;
  height: 20px;
  background: rgba(255, 255, 255, 0.8);
  border-radius: 10px;
  margin-bottom: 10px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}

#progress-bar {
  height: 100%;
  width: 0%;
  background: linear-gradient(to right, #4facfe, #00f2fe);
  border-radius: 10px;
  transition: width 0.3s;
}

#game-canvas {
  width: 800px;
  height: 400px;
  background: #e0f7fa;
  border-radius: 15px;
  position: relative;
  overflow: hidden;
  box-shadow: 0 8px 25px rgba(0, 0, 0, 0.15);
}

#ground {
  position: absolute;
  bottom: 0;
  width: 100%;
  height: 30px;
  background: linear-gradient(to bottom, #81C784, #4CAF50);
  z-index: 2;
}
```

```
/* === Увеличенный игрок === */
#player {
  position: absolute;
  bottom: 30px;
  left: 120px;
  width: 70px;
  height: 110px;
  z-index: 5;
}

/* ===== УВЕЛИЧЕННЫЙ СТУДЕНТ ===== */
.student {
  position: relative;
  width: 100%;
  height: 100%;
}

.student-head {
  position: absolute;
  top: 0;
  left: 50%;
  transform: translateX(-50%);
  width: 40px;
  height: 40px;
  background: #FFCCBC;
  border-radius: 50%;
}

.student-hair {
  position: absolute;
  top: -7px;
  left: 50%;
  transform: translateX(-50%);
  width: 45px;
  height: 20px;
  background: #5D4037;
  border-radius: 50% 50% 0 0;
}

.student-glasses {
  position: absolute;
  top: 12px;
  left: 50%;
  transform: translateX(-50%);
  width: 36px;
  height: 16px;
  border: 2px solid #263238;
  border-radius: 12px;
  display: flex;
  justify-content: space-between;
  padding: 0 5px; /* Увеличили — глаза не сливаются */
}
```

```
background: rgba(255, 255, 255, 0.3);
box-sizing: border-box;
}

.student-eye {
width: 6px;
height: 6px;
background: #000;
border-radius: 50%;
}

.student-body {
position: absolute;
top: 38px;
left: 50%;
transform: translateX(-50%);
width: 45px;
height: 40px;
background: #0288D1;
border-radius: 12px 12px 6px 6px;
}

.student-backpack {
position: absolute;
top: 42px;
right: 6px;
width: 20px;
height: 32px;
background: #7B1FA2;
border-radius: 6px 0 0 6px;
}

.student-arm {
position: absolute;
top: 42px;
width: 13px;
height: 32px;
background: #0288D1;
}

.student-arm.left {
left: 0;
transform-origin: top center;
animation: swingLeft 0.3s infinite alternate-reverse;
}

.student-arm.right {
right: 0;
transform-origin: top center;
animation: swingRight 0.3s infinite alternate;
}
```

```
.student-leg {
    position: absolute;
    top: 75px;
    width: 16px;
    height: 35px;
    background: #0066CC;
}

.student-leg.left {
    left: 14px;
    transform-origin: top center;
    animation: legLeft 0.3s infinite alternate;
}

.student-leg.right {
    right: 14px;
    transform-origin: top center;
    animation: legRight 0.3s infinite alternate;
}

@keyframes legLeft { 0% { transform: rotate(-5deg); } 100% { transform:
rotate(5deg); } }
@keyframes legRight { 0% { transform: rotate(5deg); } 100% { transform: rotate(-
5deg); } }
@keyframes swingLeft { 0% { transform: rotate(20deg); } 100% { transform: rotate(-
20deg); } }
@keyframes swingRight { 0% { transform: rotate(-20deg); } 100% { transform:
rotate(20deg); } }

/* === ЛЕГЧЕ ПРЫГАТЬ === */
.jumping {
    animation: jump 0.5s ease-out;
}

@keyframes jump {
    0% { transform: translateY(0); }
    50% { transform: translateY(-180px); } /* Выше прыжок */
    100% { transform: translateY(0); }
}

/* ===== ПРЕПЯТСТВИЯ (Только на земле!) =====
*/

.obstacle {
    position: absolute;
    bottom: 30px; /* Только это значение */
    left: 800px;
    z-index: 3;
    display: flex;
    justify-content: center;
    align-items: center;
    color: white;
    font-size: 18px;
```

```
border-radius: 5px;
/* Убрали rotateX — чтобы не ломать позиционирование */
box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2); /* Тень вместо 3D */
}

.book {
  background: #7B1FA2;
  width: 35px;
  height: 45px;
}

.coffee {
  background: #795548;
  width: 25px;
  height: 35px;
  border-radius: 5px 5px 0 0;
}

.sleep {
  background: #FFD54F;
  width: 40px;
  height: 30px;
  border-radius: 50%;
  font-size: 20px;
}

.laptop {
  background: #90A4AE;
  width: 45px;
  height: 25px;
  border-radius: 5px;
}

/* ===== ЭКРАНЫ ===== */
#start-screen, #end-screen {
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: rgba(255, 255, 255, 0.95);
  border-radius: 15px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  z-index: 10;
}

#end-screen {
  display: none;
}
```

```
h1 {  
  color: #0288D1;  
  margin-bottom: 10px;  
}  
  
.difficulty-btn {  
  padding: 12px 25px;  
  margin: 10px;  
  border: none;  
  border-radius: 50px;  
  background: linear-gradient(to right, #4facfe, #00f2fe);  
  color: white;  
  font-size: 18px;  
  cursor: pointer;  
}  
  
#score, #timer {  
  font-size: 18px;  
  font-weight: bold;  
  color: #0288D1;  
}
```

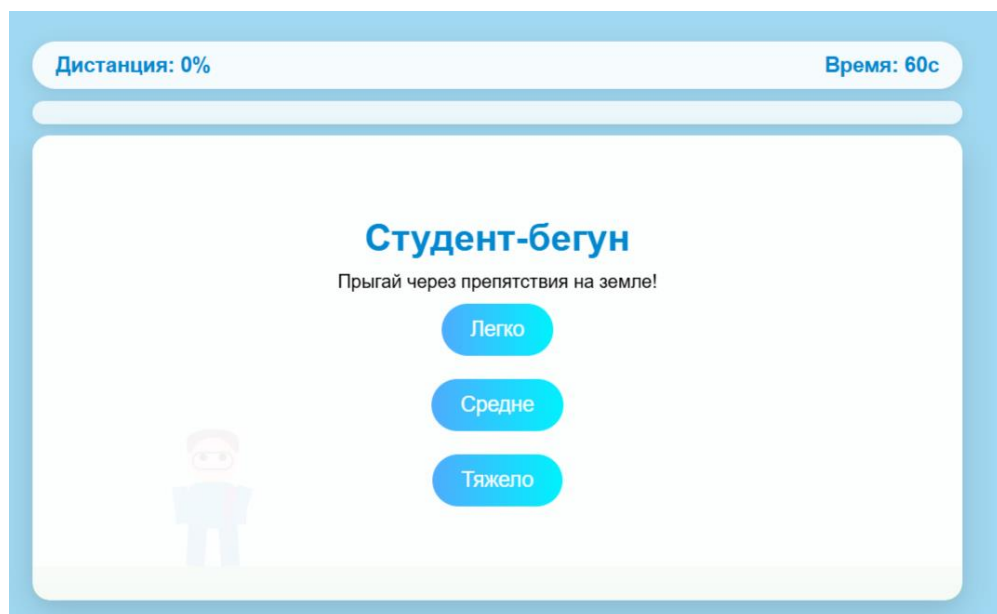


Рисунок 2. Добавление CSS-стилей

Затем перейдём к реализации JavaScript-логики, инициализируем основные элементы интерфейса — игровое поле, персонажа, экраны старта и завершения, индикаторы прогресса, счёта и времени — и определим переменные состояния игры, включая флаг активности, прыжка, скорость движения, частоту появления препятствий и таймер; после полной загрузки DOM-структуры с помощью DOMContentLoaded на кнопки выбора сложности навешиваются обработчики, которые при клике запускают игру с уникальными параметрами скорости и частоты препятствий для уровней «Легко», «Средне» и «Тяжело», а кнопка перезапуска возвращает игрока на

стартовый экран, скрывая финальное окно; при этом все параметры гибко настраиваются в объекте `difficultyBtns`, что обеспечивает чистоту кода и простоту масштабирования, а также закладывается основа для основной игровой функции `startGame`, которая будет управлять циклом анимации, генерацией препятствий, обновлением прогресса и временем.

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

```
document.addEventListener('DOMContentLoaded', () => {
  const gameCanvas = document.getElementById('game-canvas');
  const player = document.getElementById('player');
  const startScreen = document.getElementById('start-screen');
  const endScreen = document.getElementById('end-screen');
  const progressBar = document.getElementById('progress-bar');
  const scoreDisplay = document.getElementById('score');
  const timerDisplay = document.getElementById('timer');
  const finalScoreDisplay = document.getElementById('final-score');
  const restartBtn = document.getElementById('restart');

  const difficultyBtns = {
    easy: document.getElementById('easy'),
    medium: document.getElementById('medium'),
    hard: document.getElementById('hard')
  };

  let gameActive = false;
  let isJumping = false;
  let obstacles = [];
  let gameSpeed = 4; // медленнее
  let obstacleFrequency = 1800; // реже
  let score = 0;
  let gameTimeLeft = 60;
  let gameTimer;
  let animationFrame;
  let lastObstacleTime = 0;

  Object.keys(difficultyBtns).forEach(key => {
    difficultyBtns[key].addEventListener('click', () => {
      const [speed, freq] = {
        easy: [4, 1800],
        medium: [6, 1400],
        hard: [8, 1100]
      }[key];
      startGame(speed, freq);
    });
  });

  restartBtn.addEventListener('click', () => {
    endScreen.style.display = 'none';
    startScreen.style.display = 'flex';
  });
});
```

Теперь осталось прописать основную игровую логику, реализованную через функции `startGame`, `endGame`, `gameLoop` и вспомогательные модули обновления, генерации и проверки столкновений: при запуске игры устанавливаются выбранные параметры сложности, сбрасываются счёт и таймер, очищается игровое поле от предыдущих препятствий и запускается игровой цикл через `requestAnimationFrame`, обеспечивающий плавную анимацию; прогресс игрока обновляется в реальном времени в зависимости от прошедшего времени, а полоса прогресса динамически изменяется в процентах до достижения 100%; препятствия — книги, чашки кофе, символы сна и ноутбуки — создаются с заданной частотой, движутся слева направо с постоянной скоростью и удаляются при выходе за пределы экрана; проверка столкновений осуществляется через сравнение прямоугольных границ игрока и препятствий с учётом только нижней части персонажа (ног), что делает игру более справедливой; прыжок реализован по нажатию пробела или клику на экран, с использованием CSS-анимации и флага `isJumping` для предотвращения двойных прыжков; при окончании времени или столкновении вызывается `endGame`, который останавливает все процессы и показывает финальный экран с результатом, завершая тем самым полнофункциональный цикл работы браузерной игры.

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

```
function startGame(speed, freq) {
  gameSpeed = speed;
  obstacleFrequency = freq;
  startScreen.style.display = 'none';
  endScreen.style.display = 'none';
  gameActive = true;
  score = 0;
  gameTimeLeft = 60;
  obstacles = [];
  progressBar.style.width = '0%';
  scoreDisplay.textContent = 'Дистанция: 0%';
  timerDisplay.textContent = 'Время: 60с';
  finalScoreDisplay.textContent = '0';

  obstacles.forEach(o => o.element.remove());
  obstacles = [];
  clearInterval(gameTimer);
  cancelAnimationFrame(animationFrame);

  gameTimer = setInterval(() => {
    gameTimeLeft--;
    timerDisplay.textContent = `Время: ${gameTimeLeft}с`;
    if (gameTimeLeft <= 0) endGame(false);
  }, 1000);

  gameLoop();
}
```

```
function endGame() {
  gameActive = false;
  clearInterval(gameTimer);
  cancelAnimationFrame(animationFrame);
  obstacles.forEach(o => o.element.remove());
  finalScoreDisplay.textContent = score;
  endScreen.style.display = 'flex';
}

function gameLoop() {
  if (!gameActive) return;
  updateScore();
  updateObstacles();
  checkCollisions();
  animationFrame = requestAnimationFrame(gameLoop);
}

function updateScore() {
  score = Math.min(100, Math.floor(((60 - gameTimeLeft) / 60) * 100));
  progressBar.style.width = `${score}%`;
  scoreDisplay.textContent = `Дистанция: ${score}%`;
  if (score >= 100) endGame();
}

function updateObstacles() {
  const now = Date.now();
  if (now - lastObstacleTime > obstacleFrequency) {
    createObstacle();
    lastObstacleTime = now;
  }

  for (let i = obstacles.length - 1; i >= 0; i--) {
    const ob = obstacles[i];
    ob.x -= gameSpeed;
    ob.element.style.left = `${ob.x}px`;
    if (ob.x < -60) {
      ob.element.remove();
      obstacles.splice(i, 1);
    }
  }
}

function createObstacle() {
  const types = ['book', 'coffee', 'sleep', 'laptop'];
  const type = types[Math.floor(Math.random() * types.length)];

  const el = document.createElement('div');
  el.className = `obstacle ${type}`;
  el.textContent = { book: '📖', coffee: '☕', sleep: '😴', laptop: '💻' }[type];
  el.style.left = '800px';
  el.style.bottom = '30px'; // гарантированно на земле
```

```
gameCanvas.appendChild(el);

obstacles.push({ element: el, x: 800 });
}

function checkCollisions() {
  const playerRect = player.getBoundingClientRect();
  // Игнорируем верх головы
  const feetTop = playerRect.top + 30; // начинаем с корпуса
  const feetBottom = playerRect.bottom;

  for (const ob of obstacles) {
    const obRect = ob.element.getBoundingClientRect();
    if (
      playerRect.left < obRect.right &&
      playerRect.right > obRect.left &&
      feetTop < obRect.bottom &&
      feetBottom > obRect.top
    ) {
      endGame();
      return;
    }
  }
}

document.addEventListener('keydown', e => {
  if (gameActive && e.code === 'Space' && !isJumping) {
    jump();
  }
});

gameCanvas.addEventListener('click', () => {
  if (gameActive && !isJumping) {
    jump();
  }
});

function jump() {
  if (isJumping) return;
  isJumping = true;
  player.classList.add('jumping');
  setTimeout(() => {
    isJumping = false;
    player.classList.remove('jumping');
  }, 500); // совпадает с анимацией
}
});
```

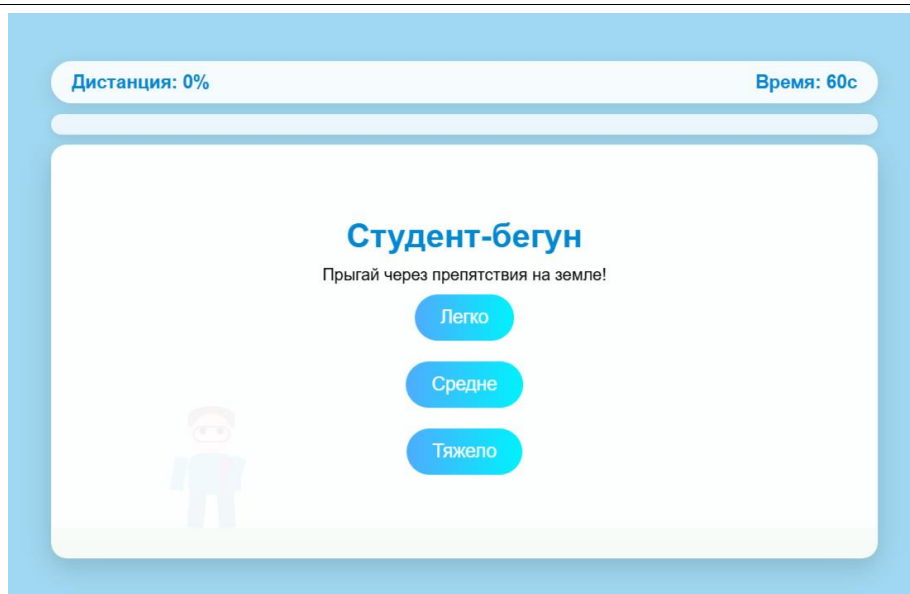


Рисунок 3. Главная страница

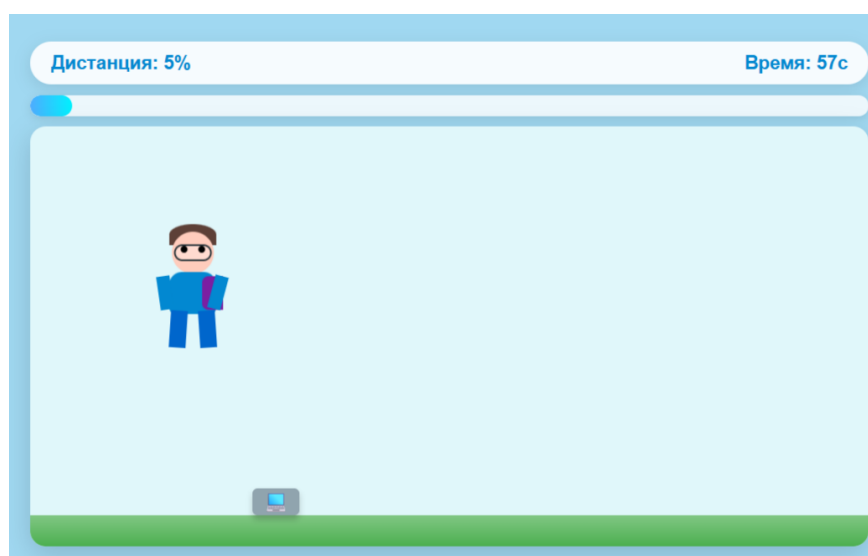


Рисунок 4. Процесс игры

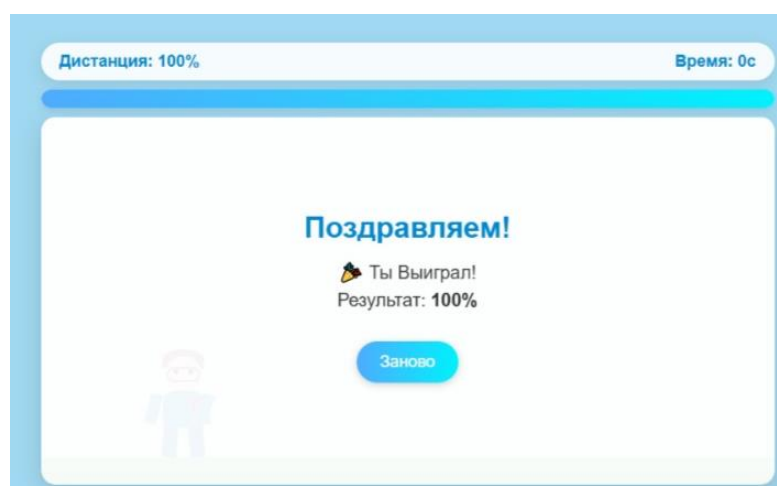


Рисунок 5. Конец игры

4 Выводы

В данной работе подробно рассмотрено создание аркадной браузерной игры «Студент-бегун» с использованием современных веб-технологий — HTML5, CSS3 и чистого JavaScript, без привлечения сторонних библиотек. Была реализована полноценная игровая механика, включающая плавный прыжок персонажа, динамическую генерацию препятствий, систему коллизий, отслеживание прогресса и времени, а также визуальную индикацию счёта и уровня сложности. Особое внимание уделено пользовательскому интерфейсу: реализованы интуитивные экраны старта и завершения игры, анимированная полоса прогресса, градиентные стили и плавные CSS-анимации, создающие эффект непрерывного движения. Поддержка управления как с помощью клавиатуры (пробел), так и через клик мышью делает игру доступной на различных устройствах. Разработанная система уровней сложности позволяет гибко настраивать скорость и частоту препятствий, что повышает игровую реиграбельность. Представленное решение может быть эффективно использовано в образовательной практике для демонстрации принципов веб-программирования, создания интерактивных приложений и формирования у студентов практических навыков работы с DOM, CSS-анимациями и событийной моделью JavaScript, превращая процесс обучения в увлекательный и наглядный 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 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.