Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра АСУ

ЛАБОРАТОРНАЯ РАБОТА №4

по организации графических систем и систем мультимедиа

Обработка потокового видео/аудио

Студент Лапшова А.Г.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк 2020г.

Задание кафедры

Реализовать обработку потокового видео/аудио. Обработкой является внесение изменений, применение фильтров и т.п., а не только визуализация.

Цель работы

Освоить на практике обработку потокового видео/аудио.

1 Теоретические сведения

Для реализации программного продукта была использована технология WebRTC и стандартные функции JavaScript.

WebRTC (Web Real-Time Communications) - это технология, которая позволяет Web-приложениям и сайтам захватывать и выборочно передавать аудио и/или видео медиа-потоки, а также обмениваться произвольными данными между браузерами, без обязательного использования посредников. Набор стандартов, которые включает в себя технология WebRTC, позволяет обмениваться данными и проводить пиринговые телеконференции, без необходимости пользователю устанавливать плагины или любое другое стороннее программное обеспечение.

WebRTC состоит из нескольких взаимосвязанных программных интерфейсов (API) и протоколов, которые работают вместе. Документация, которую вы здесь найдете, поможет вам понять основы WebRTC, как настроить и использовать соединение для передачи данных и медиа-потока, и многое другое [1].

WebRTC является многоцелевым и вместе с Media Capture and Streams API, предоставляют мощные мультимедийные возможности для Web, включая поддержку аудио и видео конференций, обмен файлами, захват экрана, управление идентификацией и взаимодействие с устаревшими телефонными системами, включая поддержку передачи сигналов тонового набора DTMF. Соединения между узлами могут создаваться без использования специальных драйверов или плагинов, и часто без промежуточных сервисов.

Работу с WebRTC можно описать следующим образом:

- Пользователь открывает страницу, содержащую HTML5 тег <video>.
- Браузер запрашивает доступ к веб-камере и микрофону пользователя.
- JavaScript код на странице пользователя контролирует параметры соединения (IP-адреса и порты сервера WebRTC или других WebRTC клиентов) для обхода NAT и Firewall.

- При получении информации о собеседнике или о потоке со смикшированной на сервере конференцией, браузер начинает согласование используемых аудио и видео кодеков.
- Начинается процесс кодирования и передача потоковых данных между WebRTC клиентами.

Технология WebRTC базируется на трех основных API:

1. MediaStream (отвечает за принятие веб-браузером аудио и видеосигнала от камер или рабочего стола пользователя).
2. RTCPeerConnection (отвечает за соединение между браузерами для “обмена” полученными от камеры, микрофона и рабочего стола, медиаданными. Также в “обязанности” этого API входит обработка сигнала (очистка его от посторонних шумов, регулировка громкости микрофона) и контроль над используемыми аудио и видеокодеками).
3. RTCData Channel (обеспечивает двустороннюю передачу данных через установленное соединение) [2].

2 Основная часть

2.1 Описание работы программы

В разметке html содержится тег <video>, который будет получать поток с видеокамеры.

В начале работы программы инициализируется переменные, которые отвечают за ширину и высоту изображения. С помощью стандартных JavaScript функций получаем элементы DOM по их идентификатору.

При нажатии на кнопку «Начать эфир» срабатывает обработчик, в котором происходит получение медиапотока. Метод mediaDevices.getUserMedia() запрашивает сам медиапоток и возвращает промис(стандартный объект js для работы с асинхронным кодом), при успешном выполнении(then) возвращается объект потока, который присваивается свойству srcObject элемента <video>, направляя в него данный поток. Как только объект получен, запускается воспроизведение методом .play(). Если получение потока окажется неудачным, обработчик ошибок выведет соответствующую ошибку в консоль.

После вызова метода play() возникает промежуток времени до начала воспроизведения видеопотока, для недопущения блокирования интерфейса устанавливается обработчик события ‘canplay’, который сработает, когда элемент <video> начнет воспроизведение.

Переменная streaming отслеживает повторный запуск обработчика (устанавливается значение true, чтобы предотвратить случайное повторное выполнение установочного кода). Если это первый запуск устанавливаются соответствующие параметры изображения.

Так как захватываемое изображение находится в элементе <video>, к нему можно применить любые CSS-фильтры, которые применяются с помощью свойства filter. Для этого в html-разметке были установлены соответствующие value для элементов option. Пример:

```
<option value="grayscale(100%)">Черно-белое</option>
```

Также был добавлен функционал загрузки своего видеоролика и применения к нему заданных фильтров.

2.2 Листинг кода

```
let width = 500, // ширина изображения
    height = 0, // вычисляется на основе входящего потока
    filter = 'none',
    streaming = false; // текущая активность видеопотока

// DOM элементы
const video = document.getElementById('video');
const clearButton = document.getElementById('clear-button');
const photoFilter = document.getElementById('photo-filter');
const startButton = document.getElementById('start-button');
const uploadButton = document.getElementById('upload-button');

uploadButton.addEventListener('change', function(e) {
  video.srcObject = null;
  video.src = null;

  const path = (window.URL ||
window.webkitURL).createObjectURL(uploadButton.files[0]);

  video.src = path;
  video.play();
});

startButton.addEventListener('click', function(e) {
// Получение медиапотока
navigator.mediaDevices.getUserMedia({video: true, audio: false})
  .then(function(stream) {
    video.src = null;
    //Направление потока в элемент <video>
    video.srcObject = stream;
    // Запуск видео
    video.play();
  })
  .catch(function(err) {
    console.log(`Error: ${err}`);
  });
});

  // Обработчик события момента воспроизведения видеопотока
  video.addEventListener('canplay', function(e) {
    if(!streaming) {
      // Установка размеров видео
      height = video.videoHeight / (video.videoWidth / width);

      video.setAttribute('width', width);
      video.setAttribute('height', height);

      // Флаг предотвращения повторного выполнения
      streaming = true;
    }
  }, false);

  // Обработчик фильтрации
  photoFilter.addEventListener('change', function(e) {
    // Установка выбранным фильтров
    filter = e.target.value;
    // Установка фильтра для видео
```

```
  video.style.filter = filter;

  e.preventDefault();
});

// Сброс фильтров
clearButton.addEventListener('click', function(e) {
  // Изменение на "Без фильтров"
  filter = 'none';
  // Установка фильтра
  video.style.filter = filter;
  // Сбрасываем индекс селектов
  photoFilter.selectedIndex = 0;
});
```

3 Результат работы программы

Репозиторий проекта находится на GitHub [3].

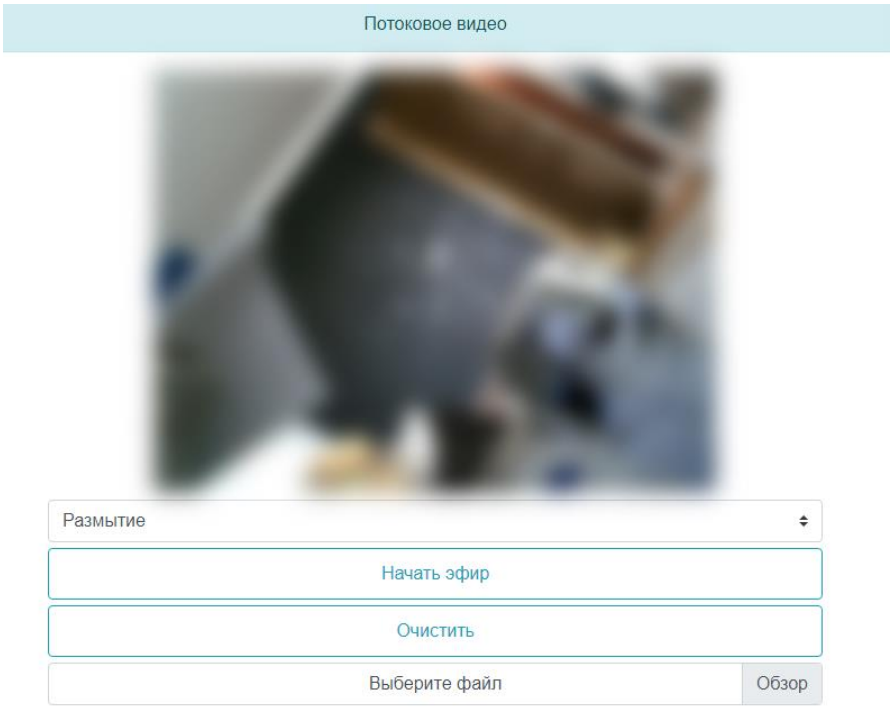

Рисунок 1 – Применение фильтра «Размытие»

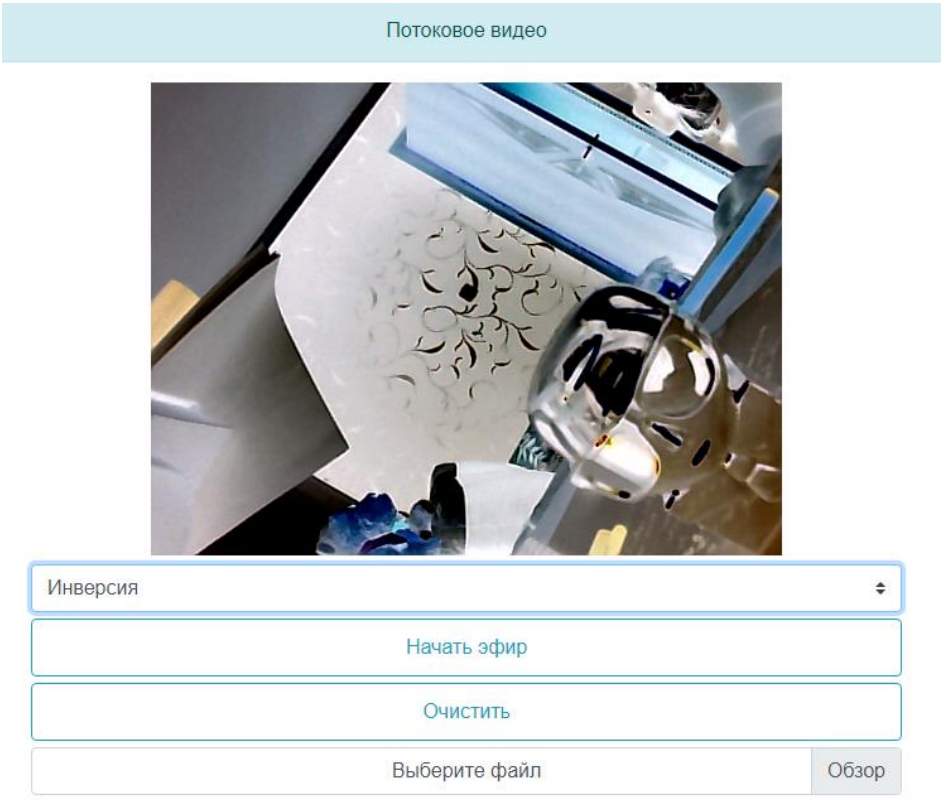

Рисунок 2 – Применение фильтра «Инверсия»

Вывод

В ходе выполнения данной лабораторной работы была изучена технология WebRTC, а также применение на практике воспроизведение потокового видео с использованием возможности редактирования данного потока, а именно наложение различных фильтров, которые реализованы с помощью стандартных фильтров CSS.

Список источников

1. WebRTC Интерфейсы веб API [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://developer.mozilla.org/ru/docs/Web/API/WebRTC_API, свободный.

2. WebRTC преимущества, недостатки, секреты [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://trueconf.ru/webrtc.html, свободный.

3. Репозиторий проекта [Электронный ресурс]. – Электрон. текст. дан. – режим доступа: https://github.com/nwdles/Graphics-and-Multimedia, свободный.