Липецкий государственный технический университет

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

Организация графических систем

Лабораторная работа №1

«Исследование методов смешанной растрово-векторной обработки изображений»

Студентка Комаричева А.Г.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк 2020 г.

Задание кафедры

Реализовать наложение фильтров на изображения как растровые, так векторные. Количество фильтров – не меньше 2. Язык реализации - произвольный. Вычисления, связанные с обработкой изображения, необходимо производить на GPU.

Оглавление

Введение

Важнейшая функция компьютера - обработка информации. Особо можно выделить обработку информации, связанную с изображениями. Она разделяется на три основные направления: компьютерная графика (КГ), обработка и распознавание изображений. Задача компьютерной графики (Computer Graphics) - визуализация, то есть создание изображения. Визуализация выполняется, исходя из описания (модели) того, что нужно отображать. Существует много методов и алгоритмов визуализации, которые различаются между собою в зависимости от того что и как отображать. Например, отображение того, что может быть только в воображении человека — график функций, диаграмма, схема, карта. Или наоборот, имитация трехмерной реальности — изображение сцен в компьютерных играх, художественных фильмах, тренажерах, в системах архитектурного проектирования. Важными и связанными между собою факторами здесь являются: скорость изменения кадров, насыщенность сцены объектами, качество изображения, учет особенностей графического устройства. Обработка изображений (Computer Vision) — это преобразования изображений. Входными данными является изображение, и результат обработки — тоже изображение. Примерами обработки изображений могут служить: повышение контраста, чѐткости, коррекция цветов, редукция цветов, сглаживание, уменьшение шумов и так далее. В качестве материала для обработки могут использоваться космические снимки, сканированные изображения, радиолокационные, инфракрасные изображения и т. п. Задачей обработки изображений может быть как улучшение в зависимости от определенного критерия (реставрация, восстановление), так и специальное преобразование, кардинально меняющее изображения. В последнем случае обработка изображений может быть промежуточным этапом для дальнейшего распознавания изображения. Например, перед распознаванием часто необходимо выделять контуры, создавать бинарное изображение, разделять по цветам. Методы обработки

изображений могут существенно отличаться в зависимости от того, каким путем получено изображение — синтезировано системой КГ либо это результат оцифровки черно-белой или цветной фотографии.

В данной лабораторной работе мы реализуем наложение фильтров на изображения таких как: Grayscale, Sepia, Negativ, Add Noise. Языком реализации в данной работе выступает JavaScript.

1 Теоретические сведения

1.1 Виды компьютерной графики

Графические программы – программное обеспечение, позволяющее создавать, редактировать или просматривать графические файлы.

Графический редактор – программа (или пакет программ), позволяющая создавать и редактировать двумерные изображения с помощью компьютера.

Компьютерную графику можно разделить на три категории:

— растровая графика;

— векторная графика;

— трёхмерная графика.

Растровое изображение – это файл данных или структура, представляющая прямоугольную сетку пикселей. Пиксель – наименьшая единица двухмерного цифрового изображения в растровой графике. Пиксель представляет собой неделимый объект прямоугольной (обычно квадратной) формы, обладающий определённым цветом. Растровое компьютерное изображение состоит из пикселов, расположенных по строкам и столбцам. На компьютерном мониторе, бумаге и других отображающих устройствах и материалах.

При использовании растровой графики важным элементом является:

— размер полотна (canvas);

— цветовое пространство (например, RGB);

— количество используемых цветов.

Растровую графику редактируют с помощью растровых графических редакторов. Создается растровая графика фотоаппаратами, сканерами, непосредственно в растровом редакторе, также путем экспорта из векторного редактора или в виде скриншотов.

Достоинства:

1. Растровая графика позволяет создать (воспроизвести) практически любой рисунок, вне зависимости от сложности, в отличие, например, от векторной, где невозможно точно передать эффект перехода от одного цвета к другому (в теории, конечно, возможно, но файл размером 1 МБ в формате BMP будет иметь размер 200 МБ в векторном формате).

2. Распространённость: растровая графика используется сейчас практически везде: от маленьких значков до плакатов.

3. Высокая скорость обработки сложных изображений, если не нужно масштабирование.

Недостатки:

1. Большой размер файлов с простыми изображениями.

2. Невозможность идеального масштабирования.

Из-за этих недостатков для хранения простых рисунков рекомендуют вместо даже сжатой растровой графики использовать векторную графику.

Растровый графический редактор – специализированная программа, предназначенная для создания и обработки изображений. Подобные программные продукты нашли широкое применение в работе художников-иллюстраторов, при подготовке изображений к печати типографским способом или на фотобумаге, публикации в Интернете.

Растровые графические редакторы позволяют пользователю рисовать и редактировать изображения на экране компьютера, а также сохранять их в различных растровых форматах.

В противоположность векторным редакторам, растровые используют для представления изображений матрицу точек (bitmap). Однако, большинство современных растровых редакторов содержат векторные инструменты редактирования в качестве вспомогательных.

1.2 Векторная графика

Векторная графика – это использование геометрических примитивов, таких как точки, линии, сплайны и многоугольники, для представления изображений в компьютерной графике. Термин используется в противоположность к растровой графике, которая представляет изображения как матрицу пикселей. Современные компьютерные видеодисплеи отображают информацию в растровом формате. Для отображения векторного формата на растровом используются преобразователи, программные или аппаратные, встроенные в видеокарту.

Кроме этого, существует узкий класс устройств, ориентированных исключительно на отображение векторных данных. К ним относятся мониторы с векторной развёрткой, графопостроители, а также некоторые типы лазерных проекторов.

Термин «векторная графика» используется в основном в контексте двухмерной компьютерной графики.

Способ хранения изображения

Рассмотрим, к примеру, окружность радиуса r. Список информации, необходимой для полного описания окружности, таков:

— радиус r;

— координаты центра окружности;

— цвет и толщина контура (возможно прозрачный);

— цвет заполнения (возможно прозрачный).

Достоинства:

— Минимальное количество информации передаётся намного меньшему размеру файла (размер не зависит от величины объекта). Соответственно, можно бесконечно увеличить, например, дугу окружности, и она останется гладкой. С другой стороны, если кривая представлена в виде ломаной линии, увеличение покажет, что она на самом деле не кривая.

— При увеличении или уменьшении объектов толщина линий может быть постоянной.

— Параметры объектов хранятся и могут быть изменены. Это означает, что перемещение, масштабирование, вращение, заполнение и т. д. не ухудшат качества рисунка. Более того, обычно указывают размеры в аппаратно-независимых единицах (англ. device-independent unit), которые ведут к наилучшей возможной растеризации на растровых устройствах.

Недостатки:

— Не каждый объект может быть легко изображен в векторном виде. Кроме того, количество памяти и времени на отображение зависит от числа объектов и их сложности.

— Перевод векторной графики в растр достаточно прост. Но обратного пути, как правило, нет – трассировка растра обычно не обеспечивает высокого качества векторного рисунка.

Типичные примитивные объекты:

1. Линии и ломаные линии.
2. Многоугольники.
3. Окружности и эллипсы.
4. Кривые Безье.
5. Безигоны.
6. Текст (в компьютерных шрифтах, таких как TrueType, каждая буква создаётся из кривых Безье).

Этот список неполон. Есть разные типы кривых, которые используются в различных приложениях.

Также возможно рассматривать растровое изображение как примитивный объект, ведущий себя как прямоугольник.

Векторные операции

Векторные графические редакторы, типично, позволяют вращать, перемещать, отражать, растягивать, скашивать, выполнять основные аффинные

преобразования над объектами, изменять и комбинировать примитивы в более сложные объекты.

Более изощрённые преобразования включают булевы операции на замкнутых фигурах: объединение, дополнение, пересечение и т. д.

Векторная графика идеальна для простых или составных рисунков, которые должны быть аппаратно-независимыми или не нуждаются в фотореализме.

1.3 Трёхмерная графика

Трёхмерная графика (3D, 3 Dimensions, русск. 3 измерения) – раздел компьютерной графики, охватывающий алгоритмы и программное обеспечение для оперирования объектами в трёхмерном пространстве, а также результат работы таких программ. Больше всего применяется для создания изображений в архитектурной визуализации, кинематографе, телевидении, компьютерных играх, печатной продукции, а также в науке.

Трёхмерное изображение отличается от плоского построением геометрической проекции трёхмерной модели сцены на экране компьютера с помощью специализированных программ.

При этом модель может как соответствовать объектам из реального мира (автомобили, здания, ураган, астероид), так и быть полностью абстрактной (проекция четырёхмерного фрактала).

Для получения трёхмерного изображения требуются следующие шаги:

— моделирование – создание математической модели сцены и объектов в ней;

— рендеринг – построение проекции в соответствии с выбранной физической моделью.

Моделирование

В сцене могут участвовать следующие типы объектов:

— источники света;

— геометрические примитивы – сфера, куб, конус, а также тела, описываемые квадратными и кубическими уравнениями;

— каркасы (англ. mesh) – группы связанных между собой «встык» треугольников, образующих иллюзию тела или поверхности среды;

— среды жидкости в стаканах, газы, например, воздух в атмосфере, дымы;

Есть и концептуально более сложные типы, как, например, искажения пространства или системы частиц.

Задача трёхмерного моделирования – описать эти объекты и разместить их на сцене с помощью геометрических преобразований в соответствии с требованиями к будущему изображению.

1.4 Язык реализации

JavaScript — динамический, интерпретируемый язык со слабой типизацией, обычно используемый для написания скриптов на стороне клиента. JavaScript поддерживается всеми современными браузерами и позволяет выполнять на стороне клиента достаточно сложные вычисления. Для JS существует большое количество фреймворков для простого рендеринга 3D-сцен, отрисовки пользовательских интерфейсов и т.д. Один из таких фреймворков – Vue.js. Vue.js — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле. В отличие от фреймворков-монолитов, Vue создан пригодным для постепенного внедрения. Его ядро в первую очередь решает задачи уровня представления (view), что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue полностью подходит и для создания сложных одностраничных приложений

(SPA, Single-Page Applications), если использовать его совместно с современными инструментами и дополнительными библиотеками.

Кроме того, современные браузеры также поддерживают и использование GPU API WebGL. WebGL предполагает использование языка шейдеров GLSL, который имеет много общего с C. GLSL (OpenGL Shading Language, Graphics Library Shader Language) — язык высокого уровня для программирования шейдеров. Разработан для выполнения математики, которая обычно требуется для выполнения растеризации графики. Синтаксис языка базируется на языке программирования ANSI C, однако, из-за его специфической направленности, из него были исключены многие возможности, для упрощения языка и повышения производительности. В язык включены дополнительные функции и типы данных, например для работы с векторами и матрицами. Основное преимущество GLSL перед другими шейдерными языками — переносимость кода между платформами и ОС. Язык GLSL используется в OpenGL, в OpenGL ES и WebGL используется язык GLSL ES (OpenGL ES Shading Language).

2 Исходный код программы

Файл «index.html »

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
            <!--Определение стилей HTML страницы редактора-->
  <style>
    h1 {
      font-family: arial;
      color: dimgray;
    }
    body {
      font-family: arial;
```

```
            margin: 30px;
        }
        canvas {
            height: 200px;
            border: 1px solid lightgray;
        }
        input {
            font-size: 14pt;
        }
    </style>
    <title>Photo Editor</title>
</head>
<body>
    <script
src="https://www.dukelearntoprogram.com/course1/common/js/image/SimpleImage.js"></script>
    <h1>Photo Editor</h1>
    <canvas id="can"></canvas>
<!--Создание кнопок-->
    <p>
        <input type="file" multiple="false" accept="image/*" id="finput" onchange="upload()">
    </p>
    <p>
        <input type="button" value="Make Grayscale" onclick="imageProcessor.makeGray()">
        <input type="button" value="Make Sepia" onclick="imageProcessor.sepia()">
        <input type="button" value="Make Negativ" onclick="imageProcessor.negativ()">
        <input type="button" value="Add Noise" onclick="imageProcessor.addNoise()">
    </p>
<!--Подключение дополнительных файлов для полноценной работы, определение языка
содежащегося в файле-->
    <script type="text/javascript" src="./ImageProcessor.js"></script>
    <script type="text/javascript">
        let imageProcessor = new ImageProcessor();
<!--Функция загрузки фотографии-->
        function upload() {
            var fileinput = document.getElementById("finput");
```

```
        imageProcessor.image = new SimpleImage(fileinput);
        imageProcessor._drawImage();
      }
  </script>
</body>
</html>
```

Файл “ImageProcessor.js”

```
// Объявление класса "Процессор обработки изображения"
class ImageProcessor {

  constructor(){
    this.image = null;
  }
//Фильтер "Градация серого"
  makeGray() {
    for (var pixel of this.image.values()) {
      var avg = pixel.getRed() * 0.3 + pixel.getGreen() * 0.59 + pixel.getBlue() * 0.11;
       pixel.setRed(avg);
       pixel.setGreen(avg);
       pixel.setBlue(avg);
    }

    this._drawImage();
  }
//Фильтер "Сепия"
  sepia() {
    for (var pixel of this.image.values()) {
       var avg = pixel.getRed() * 0.3 + pixel.getGreen() * 0.59 + pixel.getBlue() * 0.11;
       pixel.setRed(avg + 100);
       pixel.setGreen(avg + 50);
       pixel.setBlue(avg);
    }
```

```
        this._drawImage();
    }
//Фильтер "Негатив"
    negativ() {
        for (var pixel of this.image.values()) {
            pixel.setRed(255 - pixel.getRed());
            pixel.setGreen(255 - pixel.getGreen());
            pixel.setBlue(255 - pixel.getBlue());
        }

        this._drawImage();
    }
//Фильтер "Добавление шума"
    addNoise() {
        for (var pixel of this.image.values()) {
            var rand =  (0.5 - Math.random()) * 100; // 100 - случайный фактор
            pixel.setRed(pixel.getRed()+rand);
            pixel.setGreen(pixel.getGreen() + rand);
            pixel.setBlue(pixel.getBlue()+rand);
        }

        this._drawImage();
    }

    _drawImage(){
        let canvas = document.getElementById("can");
        this.image.drawTo(canvas);
    }
}
```

3 Пример работы программы

Пользователь может как загружать изображения для их обработки в различных форматах JPEG, png, screenshot. Для запуска приложения необходимо открыть файл index.html в любом из удобных браузеров и в открывшемся окне загрузить фотографию.

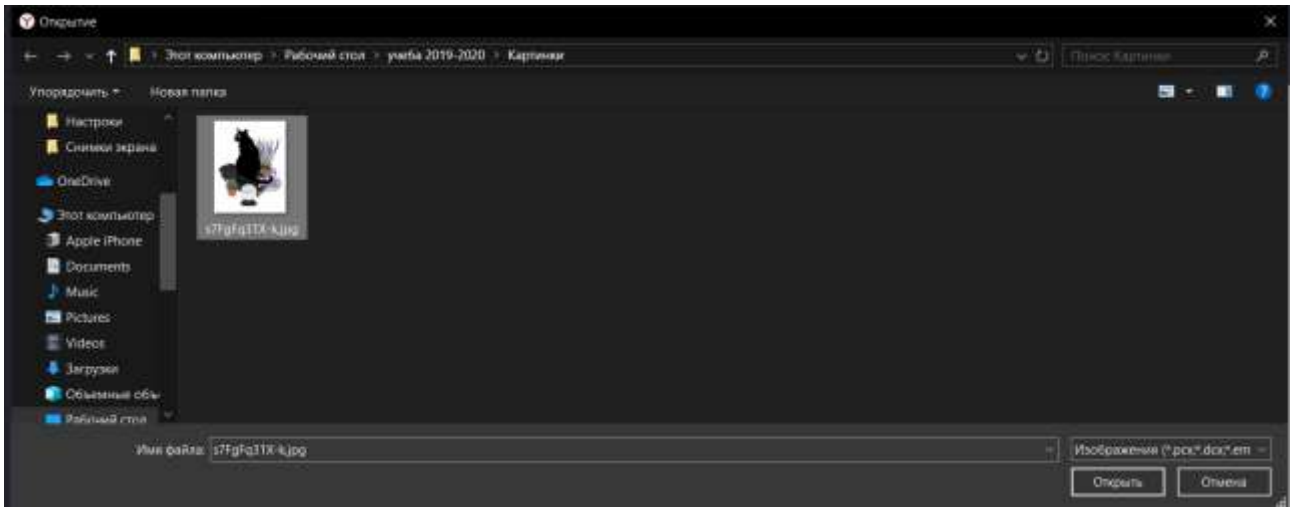

Рисунок 1 – Загрузка изображения

Рисунок 2 – Исходное изображение

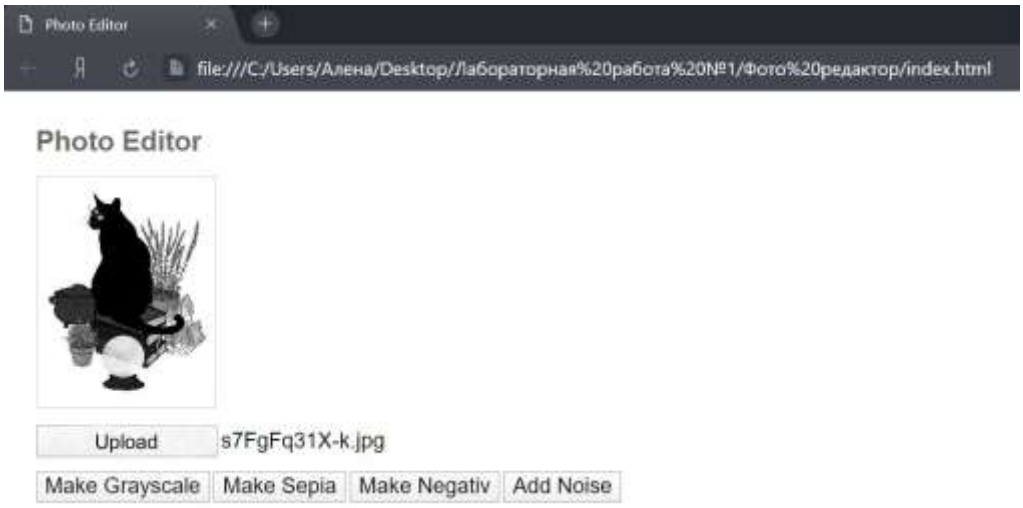

Рисунок 3 – Применение фильтра «Градация серого»

Рисунок 4 – Применение фильтра «Сепия»

Рисунок 5 – Применение фильтра «Негатив»

Рисунок 6 – Применение фильтра «Добавление шума»

Выводы

В ходе выполнения данной работы были освоены основы языка разметки HTML и языка программирования JavaScript, также были изучены и применены на практике способы обработки изображений на видеоадаптере. При выполнении задания была написана веб-страница, на которой можно загрузить и обработать изображение. Для пользователя был создан простой интерфейс для загрузки и обработки изображений.

Список литературы

1. Графические редакторы [Электронный ресурс]: статья / – **Режим доступа:** https://studme.org/54389/informatika/graficheskie_redaktory, дата обращения: 28.02.2020.

2. Графические редакторы [Электронный ресурс]: лекционный материал / – **Режим доступа:** http://inf.susu.ac.ru/Klinachev/lc_sga_14.htm, дата обращения: 28.02.2020.

3. Графические редакторы [Электронный ресурс]: статья / – **Режим доступа:** https://www.internet-technologies.ru/articles/graficheskie-programmy-dlya-dosuga-i-professionalnoy-deyatelnosti.html, дата обращения: 28.02.2020.

4. Векторные графические редакторы [Электронный ресурс]: статья / – **Режим доступа:** http://csaa.ru/vektornye-graficheskie-redaktory-2/, дата обращения: 28.02.2020.

5. Виды компьютерной графики и графических файлов [Электронный ресурс]: лекционный материал / – **Режим доступа:** https://knowledge.allbest.ru/programming/2c0a65625b3bc68b4c43b89521306d37_0.html, дата обращения: 28.02.2020.

6. https://github.com/Fire-Phoenix07/Progects