Весна 2021

Системное программное обеспечение

Онлайн-лекции

Лекция №13: Графические файлы. Алгоритм Брезенхэма. Требования к оформлению РЗ

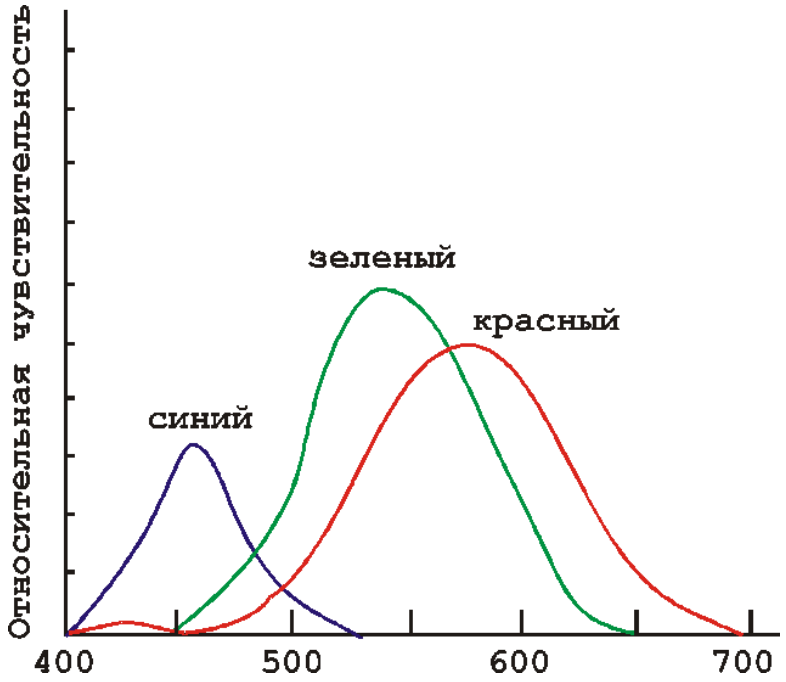
Доцент, к.т.н. ГОЛЬЦОВ Александр Геннадьевич

Цвет

- Цвет не является физическим явлением, цвет существует в сознании зрителя и формируется на основе возбуждения зрительных рецепторов.
- Монохроматическом свету определенной длины волны соответствует определенный цвет, однако одно и то же зрительное ощущение может быть получено при разном спектральном составе излучения.
- В спектре отсутствуют пурпурные цвета ("между" синим и красным: сиреневый, малиновый и т.п.)
- Цвет кодируется не менее чем тремя независимыми числовыми параметрами: RGB, CMY, CMYK, HLS, YUV.

Параметры цвета

- Традиционно цвет описывается:
  - яркостью
  - насыщенностью
  - оттенком
- Одна из теорий цветового зрения гласит, что есть рецепторы трех типов: красного, зеленого и синего
- Подавая смесь монохроматического света красного, зеленого и синего цвета можно формировать произвольное цветовое ощущение:
  желтый может быть "настоящим" 620 нм, а может быть смесью красного и зеленого



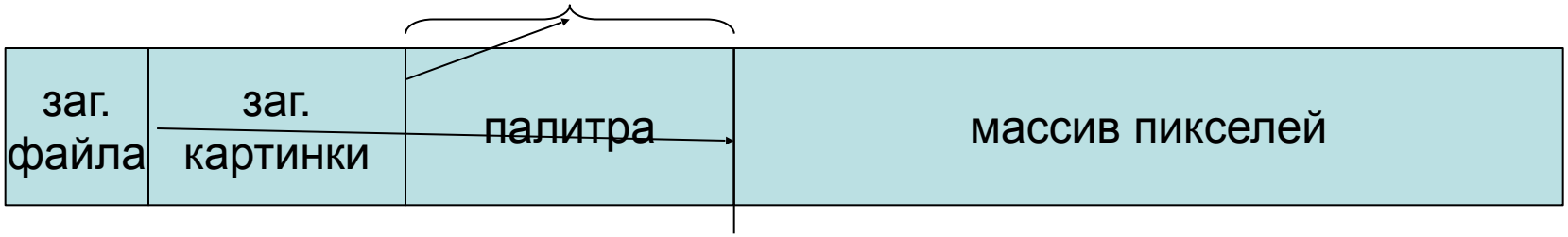
Изменение яркости и контраста изображения

- Пусть цвет пикселя кодируется яркостью трех лучей по схеме RGB, на каждый луч по 1 байту
- Изменить яркость = увеличить или уменьшить все три составляющие "с насыщением" (без перехода через 255 или 0 при увеличении или уменьшении)
- Изменить контраст = изменить расстояние каждой цветовой координаты от "точки серого" - значения 128. Тоже с насыщением.
- Можно менять как сложением, так и умножением
- При этом может поменяться цветовой баланс
- При этом обратное преобразование не восстановит исходную картинку
- Существуют более тонкие способы изменения яркости и контраста, но эти - достаточны в рамках РЗ.

Файлы BMP

- Стандартный формат представления растровой графики в MS Windows (поддерживается на уровне графической подсистемы).
- Подразумевает простое сжатие, но реально используется без сжатия.
- Поддерживается практически всеми современными графическими редакторами.
- Стандартная глубина цвета:
  - 1 бит (монохром)
  - 4 бита (16 цветов, произвольная палитра)
  - 8 бит (256 цветов, произвольная палитра)
  - 16 бит (RGB - 1555)
  - 24 бита (RGB - 888)
  - 32 бита (aRGB - 8888)

Структура файла BMP



- Заголовок файла включает размер файла и ссылку на массив пикселей, размер заголовка фиксирован
- Размер заголовка картинки прописан в его начале
- Заголовок картинки содержит размер палитры (или 0, если нет)
- Массив пикселей идет после палитры
- Формат палитры: ARGB - 4 байта на цвет, отличается от PCX

Особенности представления

- Заголовок картинки содержит ширину и высоту в пикселях.
- По умолчанию развертка в массиве пикселей снизу вверх.
- Если задать отрицательную высоту - развертка будет выполняться сверху вниз как обычно и как в PCX.
- Одна строка пикселей представляется количеством байт, кратным четырем. Последние байты в представлении строки могут содержать мусор.
- 8 бит/пиксель, ширина 150 ➔ 152 байта (150+2)
- 24 бит/пиксель, ширина 150 ➔ 452 байта (450+2)
- Последовательность RGB не такая, как в PCX

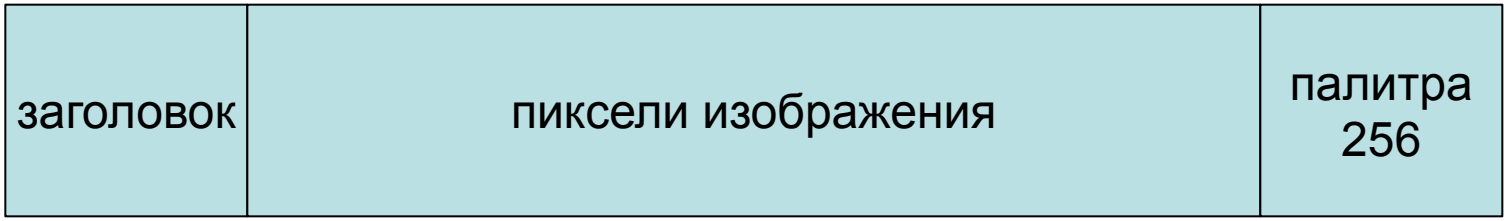
Задания с BMP

- Нужно работать с файлами блоками байтов - например, читать/писать целый заголовок или одну или несколько строк пикселей.
- Нужно помнить о мусоре в конце строки пикселей.
- При зеркальном отображении 24-битных изображений меняются местами тройки байт в строке, байты внутри тройки не переставляются.
- При преобразовании в PCX следует помнить, что RGB-цвет кодируется в другом порядке, другой формат палитры для 8-битных изображений.
- Варианты с отображением на экране - только монохромных (1 бит на пиксель) файлов. Вполне удобно в режиме 13h (320x200, 256 цветов)

Файлы PCX

- PC exchange format - ZSoft PC PaintBrush
- Затем PaintBrush купила Microsoft и он стал Paint
- Поддержка формата постепенно из Paint улетучилась
- Формат предназначен для изображений глубиной цвета 1, 4, 8 и 24 бита.
- Подразумевает простейшее сжатие повторяющихся по цвету пикселей.
- Поддерживается Adobe Photoshop

Структура файла PCX



- Заголовок включает палитру для 16-цветного изображения, мы с такими не работаем (сложно, они организованы в битовые плоскости, как аппаратная графика в EGA)
- Палитра 256-цветной картинки занимает 768 байт в конце файла, перед ней идет байт с кодом 0Ch по смещению (size-768-1)
- Формат палитры: 3 байта на цвет, отличается от BMP

Сжатие RLE

- RLE - run length encoding, кодирование повторяющихся последовательностей.
- Если в несжатом представлении пикселей изображения (независимо от глубины цвета) имеются повторяющиеся байты, их последовательность заменяется на 2 байта: повторитель-значение.
- В повторителе взведены 2 старших бита: 11ссссcc, шесть младших бит ссссcc образуют счетчик повторов ➔
  - максимальная длина последовательности 63 бита
  - байты-значения с кодами больше 192 (11000000) требуют кодировки двумя байтами: единичным повторителем и значением
- ➔ Такое сжатие может вести не к сжатию, а к раздуванию файла до 2 раз.
- Такое сжатие эффективно для "графики презентаций" без полутонов в 16- и 256-битовом цвете, для представления фотографий неэффективно.

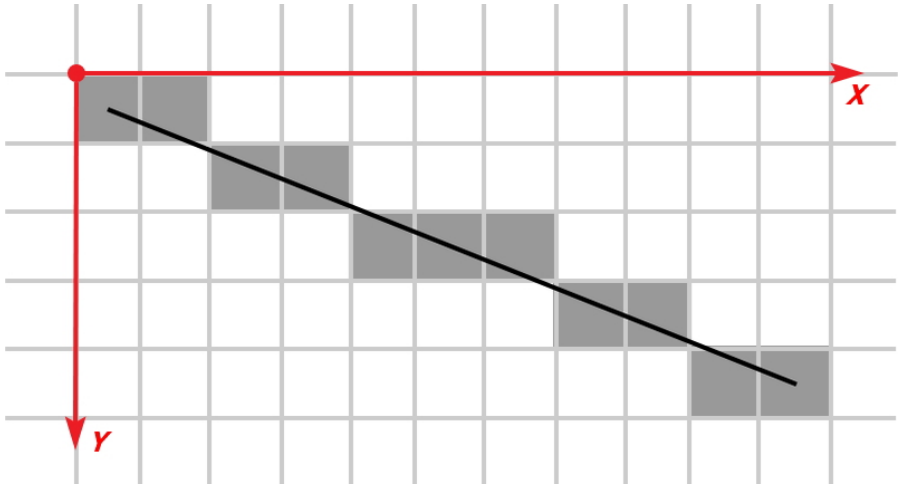
Особенности представления

- Заголовок картинки содержит координаты начала и конца строки по X, номер первой и последней строки по Y.
- Развертка обычная, сверху вниз.
- Теоретически нет запрета на то, чтобы одной RLE-порцией кодировались пиксели нескольких строк, но обычно через границу строки перехода нет.
- Последовательность RGB не такая, как в BMP.

Задания с PCX

- Нужно работать с файлами блоками байтов.
- При зеркальном отображении 24-битных изображений меняются местами тройки байт в строке, байты внутри тройки не переставляются.
- Нужно ли при зеркальном преобразовании распаковывать и вновь упаковывать RLE - хорошой вопрос, правильный ответ - да.
- При преобразовании в BMP следует помнить, что RGB-цвет кодируется в другом порядке, другой формат палитры для 8-битных изображений.
- Варианты с отображением на экране - только монохромных (1 бит на пиксель) файлов. Вполне можно в режиме 13h (320x200, 256 цветов)

Алгоритм Брезенхэма



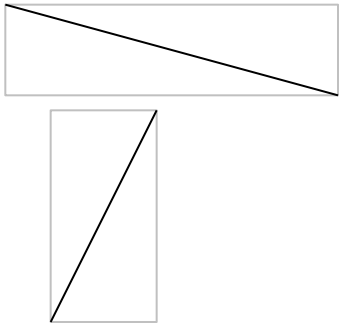
- Существует алгоритм рисования линий, окружностей и в целом кривых 2-го порядка.
- 1962 год.

• Необходимо закрасить точки растра, через которые проходит линия.

• Вычислять координаты следующей точки через уравнение прямой - неприемлемо долго.

• Хотелось бы рисовать каждую точку левее (или ниже) предыдущей, периодически смещаясь по другой оси, без операций умножения и в целых числах.

Идея алгоритма



- "Горизонтальная" линия: |DX| > |DY|
- "Вертикальная" линия: |DX| < |DY|
- DX=0 или DY=0 - разрешенные случаи
- Всего шагов рисования S = max(|DX|, |DY|)
- Делается S шагов по 1 пикселю в направлении "длинной" координаты
- На каждом шаге проверяется, не превысила ли накопленная ошибка по "короткой" координате порог
- Если порог превышен - модифицируется "короткая" координата
- Накопленный порог не сбрасывается в 0, но модифицируется с учетом произошедшего сдвига по "короткой" координате

Реализация на Паскале


```
Procedure Line(x1,y1,x2,y2,color:integer);
var dx,dy,i,sx,sy,e,x,y:integer;
    d1, d2: integer; {d1 > d2 - длина по "длинной" и "короткой" координате}
    check:boolean;
begin
    dx:=abs(x1-x2);
    dy:=abs(y1-y2);
    sx:=Sign(x2-x1);
    sy:=Sign(y2-y1);
    x:=x1;
    y:=y1;
    check:=dy>dx; {признак основного движения по Y}
    if check then begin
        d1:=dy; d2:=dx;
    end
    else begin
        d1:=dx; d2:=dy;
    end;
    e:= 2*d2 - d1;
    for i:=0 to d1 do begin
        PutPixel(x,y,color);
        if e>=0 then begin
            if check then inc(x,sx) else inc(y,sy);
            dec(e,2*d1);
        end;
        if check=1 then inc(y,sy) else inc(x,sx);
        inc(e,2*d2);
    end;
end;
```

Когда модифицировать короткую координату?

- Если длинная координата изменяется на 1, то короткая (вдоль линии) на d2/d1 (с учетом знака + или -)
- На каждом шаге надо бы прибавлять к суммарной дельте по короткой координате d2/d1
- Когда дельта короткой координаты превысит 1/2, координату надо изменить (+/–), а дельту - уменьшить на 1.
- Накапливать сумму этих d2/d1 в вещественном формате и следить, когда она превысит 1/2, неудобно, накладно и долго.
- Если умножить все на d1*2

  d2/d1*d1*2 ➔ d2*2; 1/2*d1*2 ➔ d1; 1*d1*2 ➔ d1*2

  то рассчитывать смену короткой координаты станет проще, и обойдемся без вещественной арифметики

Дополнительные замечания

- Удобно использовать видеорежим 13h 320x200 пикселей.
- Видеопамять в одном сегменте, одна точка = 1 байт.
- Точка справа = +1 к адресу, слева = -1
- Точка снизу = прибавить ширину экрана к адресу, сверху = вычесть ширину экрана.

Требования к оформлению

- См. файл с кафедральными требованиями КР, КП, выпускных работ.
- Оформление библиграфических ссылок - по ГОСТ на библиографию.

Библиография

"Старый" ГОСТ (СССР):

1. Иванов А.В., Петров И.К. Руководство по программированию. – М.: "Высшая школа", 1995. – 252 с.

2. Семенов В.Я. и др. Некоторые вопросы реализации операций умножения // 3-я конференция энтузиастов программирования. – Новосибирск, 1997. – сс. 75-79.

Современный ГОСТ Р 7.0.100-2018 :

1. **Иванов, А.В.** Руководство по программированию / А.В. Иванов, И.К. Петров. – М.: "Высшая школа", 1995. – 252 с.

2. **Семенов, В.Я.** Некоторые вопросы реализации операций умножения / Владимир Семенов, Николай Ушаков, У Лин, Валерий Сумской // Программист. – 1997. – №2. – сс. 75-79.

3. eLIBRARY.RU: научная электронная библиотека: сайт. – Москва, 2000. – URL: https://elibrary.ru (дата обращения: 01.09.2020). – Режим доступа: для зарегистрир. пользователей.

- Разделитель разделов - "точка тире".
- Ссылка должна давать возможность идентифицировать издание (например, найти в каталоге) и проверить достоверность ссылки.

Спасибо за внимание.