

Информатика

Курячий Георгий Владимирович

*Преподаватель Вечерней математической школы
факультета ВМК МГУ им. М.В. Ломоносова.*



ХРМ: растровое изображение своими руками

Для наглядной демонстрации свойств растровой графики можно использовать текстовый формат ХРМ, представляющий собой фрагмент кода на языке Си.

Формат растровых графических файлов ХРМ изобретён довольно давно. Это полностью текстовый формат, то есть графические ХРМ-файлы можно редактировать в текстовом редакторе. Что ещё интереснее, ХРМ-файл представляет собой фрагмент кода на языке Си, который можно включать в программу с помощью `#include "..."`. Всё это делает ХРМ удобным инструментом для начального изучения темы.

О чём по теме «Растровое изображение» стоит разговаривать?

- О матричном представлении изображения, в том числе о том, что одной только матрицы недостаточно: размеры, способ кодирования цвета, другая дополнительная информация.

- О кодировании цветов и цветовых пространствах (как минимум – RGB, CMYK и HLS/HLV).

- Об упаковке изображения и об упаковке «с потерями».

- О проблемах геометрических преобразований.

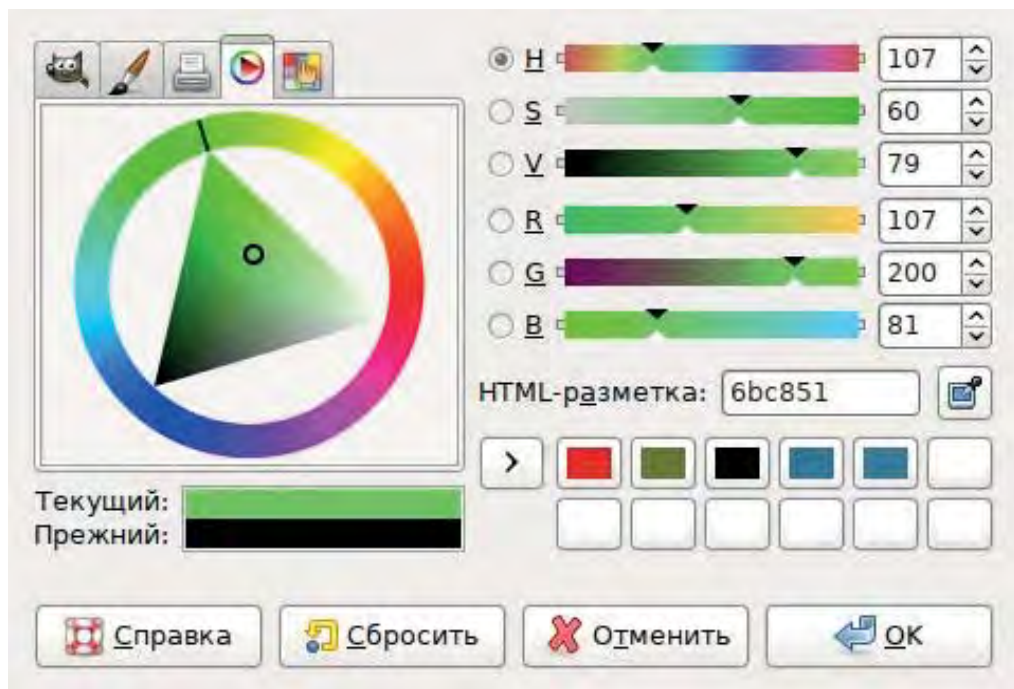
Ставить точки руками или обрабатывать области магией?

Среди перечисленных ранее тем некоторые явно выше того, что можно в школе объяснить до конца (та же упаковка с потерями, например). Не-

которые темы можно осветить только поверхностно. Можно запустить любой приличный редактор растровой графики и показать диалог выбора

цвета в различных цветовых пространствах – какой при изменении параметров получается в результате цвет. Однако объяснять, почему цветовые пространства именно

такие, придётся всё равно «на пальцах» (колбочки, палочки, складывание цветов, вычитание, а на освещённости/светлоте начинается уже математика).



Диалог выбора цвета в редакторе GIMP. Довольно поучительно начать изменять один параметр из пространства HSV и посмотреть, как движется «прицел» и вращается маркер цвета.

Геометрические преобразования стоит исследовать уже после близкого знакомства с предыдущими темами.

А вот теме «матричное представление изображения», как это ни странно, не хватает *программной* наглядности. С одной стороны, имеется растровый редактор, в котором всё вполне наглядно, своими руками делается. А с другой стороны, со

стороны программиста, – разнообразные прекрасные библиотеки для работы с разнообразными прекрасными растровыми форматами.

И вот эта пропасть слабо преодолима. Потому что библиотеки, в конечном счёте, описывают совсем не то, что человек делает в простейшем растровом редакторе, а если описывают, то на очень общем, действительно сложном уровне. Для короткого, но познавательного изучения растровых форматов на занятиях по *программированию* большинство библиотек непригодны – получится изучение самих библиотек.

Просто строки

Чего не хватает – это наглядного представления изображения в виде двумерного массива байтов и сопутствующей информации. Например, в «проекте рисования графика» для этой цели используется... двумерный массив байтов! Более точно – массив строк, каждый символ которых соответствует отдельной точке «экрана». Обновление такого «экрана» – просто

вывод всех строк на текстовый терминал.

Довольно интересный компромисс в этом плане – формат X Pixmap (сокращенно – XPM).

Вот, например, маленькая кар-

тинка в формате xpm:  А вот *тот же самый файл* непосредственно:

```

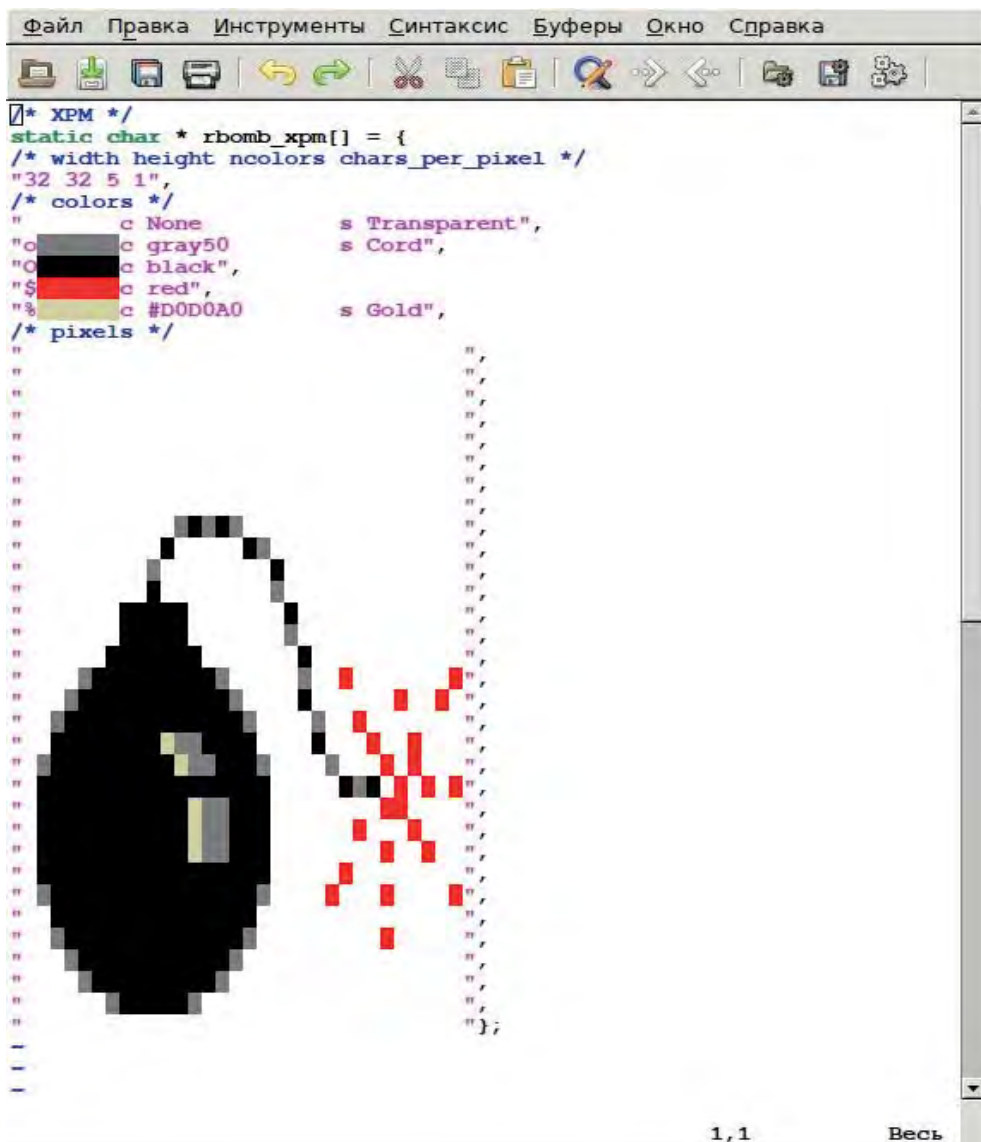
1 static char * rbomb_xpm[] = {
2 /* width height ncolors chars_per_pixel */
3 "32 32 5 1",
4 /* colors */
5 "      c None          s Transparent",
6 "o      c gray50       s Cord",
7 "0      c black",
8 "$      c red",
9 "%      c #D0D0A0      s Gold",
10 /* pixels */
11 "
12 "
13 "
14 "
15 "
16 "
17 "
18 "
19 "          o0o0o
20 "          0          0o
21 "          0          0
22 "          0          o
23 "        00000        0
24 "        00000        o
25 "       0000000       0
26 "      o0000000000o   o $
27 "      o0000000000o   0 $ $
28 "      o00000000000o  o $ $
29 "      00000000%o0000 0 $ $
30 " o0000000000%o0000o  o $ $
31 " 000000000000000000 0o0 $ $
32 " 00000000000%o0000  ",
33 " 00000000000%o0000   $ $
34 " 00000000000%o0000   $ $
35 " 000000000000000000   $
36 " o00000000000000000  $ $

```

<u>37</u>	"	0000000000000000		"	,
<u>38</u>	"	0000000000000000	\$	"	,
<u>39</u>	"	0000000000000000		"	,
<u>40</u>	"	0000000000000000		"	,
<u>41</u>	"	0000000000000000		"	,
<u>42</u>	"	00000000		"	};

То есть сам формат – **строго текстовый**. Зная структуру XPM-файла, его не-
сложно создать и ещё проще – редактировать в произвольном текстовом редакторе.

Вот так выглядит «подсветка синтаксиса», когда редактируешь XPM-файл с помощью редактора gvim:



Нетрудно заметить, что XPM-файл – это просто массив строк, записанный на языке программирования Си. Первая строка массива содержит (в строковом виде) метainформацию: ширину и высоту картинки в точках раstra, количество цветов в палитре и «количество символов на цвет». Вот этот последний параметр особенно забавен: дело в том, что последние строки XPM-файла представляют собственно изображение, причём одна точка раstra соответствует фиксированному числу символов (удобнее всего, когда одному). При этом сами символы задаются произвольно в палитре, начинающейся со второй строки массива. В частности, в примере чёрному цвету соответствует символ 'O', а красному – '\$'.

Цвета в палитре можно задавать в пространстве RGB (24 бита на точку), как это сделано для золотистого цвета, а можно по именам. Имена цветов XOrg (графической подсистеме, используемой в большинстве дистрибутивов Linux) заданы примерно в том же формате в файле `/usr/share/X11/rgb.txt`. Среди них встречаются весьма поэтично названные, например LavenderBlush или PeachPuff. Формат этого файла текстовый:

```
...
255 218 185    PeachPuff
...
255 240 245    LavenderBlush
...
```

Программа на Си

Рассмотрим простую учебную программу на языке Си. В программе

- XPM-файл с картинкой просто включается в исходный текст (он же тоже написан на Си!),
- затем создаётся доступная на запись копия (заметим, что исходный код имеет тип `const char **`),

Ещё одна особенность XPM – «значимые» имена цветов (задаются в форме «s имя»); в примере значимые имена даны всем цветам, кроме красного и чёрного. Соответствующая библиотека (`libxpm`) позволяет программам работать с этими значащими именами вместо непосредственно символов, так что нет необходимости пользоваться ровно теми символами, которые заданы в палитре. Например, при разработке графического интерфейса можно выделить цвет фона, цвет печати, цвета обычной, затенённой и высветленной областей границы и т. п. «Кнопочка» будет нарисована независимо от того, какая в действительности задана палитра. Но и без библиотеки такие имена полезны – они указывают на назначение цвета. При описании цвета можно ещё задавать, как отображать точки этого цвета в монохромном и чёрно-белом пространстве (когда-то давно это было насущной необходимостью, т. к. далеко не все графические устройства были цветными).

Одно из имён цвета – «None» – особенное. Это не значащее имя, а именно задание цвета (значащее имя для цвета None в примере – Transparent). При обработке изображения точки цвета None считаются прозрачными, выводятся и обрабатываются соответственно.

- в этой копии модифицируется содержимое растровой матрицы,

- результат записывается в новый XPM-файл (формат XPM при этом соблюдается вручную, это несложно 😊).

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include "face.xpm"
5
6 int main(int argc, char *argv[]) {
7     int Width, Height, PSize, BPP;
8     int size;
9     int beg, end, y, x, len;
10    char **newface;
11    FILE *fout;
12
13    /* Извлекаем размеры картинки и палитры */
14    /* ВНИМАНИЕ! Для простоты считаем, что BPP=1 */
15    sscanf(face[0], "%d %d %d %d", &Width, &Height, &PSize,
&BPP);
16
17    /* Область картинки */
18    beg = PSize+1;
19    end = PSize+Height+1;
20
21    /* Копия картинки, доступная для модификации */
22    newface = calloc(end, sizeof(char *));
23    for(y=size=0; y<end; y++) {
24        len = strlen(face[y]);
25        strcpy(newface[y]=malloc(len+1), face[y]);
26    }
27
28    /* Удаляем все '*' (красные точки) */
29    for(y=beg; y<end; y++)
30        for(x=0; x<Width; x++)
31            if(newface[y][x] == '*')
32                newface[y][x] = ' ';
33
34    /* Посмотрим на безротую картинку */
35    for(y=beg; y<end; y++)
36        puts(newface[y]);
37
38    /* Перевернём рот. ВНИМАНИЕ! Мы заранее знали, в каких он
строках */
39    for(y=0; y<4; y++)
40        for(x=0; x<Width; x++)
41            if(face[end-8+y][x] == '*')
42                newface[end-7-y][x] = '*';
43
44    /* Создадим XPM-файл. ВНИМАНИЕ! Идентификатор картинки
задан заранее */
45    fout = fopen("sadface.xpm", "w");
46    fputs("/* XPM */", fout);
47    fputs("static char * sadface[] = {", fout);
48    for(y=0; y<end-1; y++)
```

```

49     fprintf(fout, "\\\"%s\\\",\\n\",newface[y]);
50     fprintf(fout, "\\\"%s\\\"};\\n\",newface[y]);
51     fclose(fout);
52
53     return 0;
54 }

```

Обратите внимание на то, что программа несложная, однако затрагивает сразу несколько тем:

- работа со строками (включая преобразование числовых форматов),

- работа с памятью и указателями,
- работа с файлами,
- работа с растровой графикой 😊.

Вот такой файл включается в программу:

```

1  /* XPM */
2  static char * face[] = {
3  "32 32 5 1",
4  "      c none s mask",
5  "#      c #55aad4 s HiShadow",
6  "*"      c #fff0000",
7  "+"      c #4488aa s Foreground",
8  "."      c #33667f s LoShadow",
9  "
10 "      .....
11 "      .....
12 "      ..      ..
13 "      ....      ..
14 "      ..      ..
15 "      ..      .#
16 "      .   ###      ###   #
17 "      ..  ##  ##      ##  ##  ##
18 "      ..  #   .      #   .   ##
19 "      .   .   .      .   .   #
20 "      .   .   .      .   .   #
21 "      .   .   .      ##
22 "      .   .   .      #
23 "      .   .   .      #
24 "      .   .   .      #
25 "      .   .   .      #
26 "      .   .   .      #
27 "      .   .   .      #
28 "      .   .   .      #
29 "      .   .   .      ##
30 "      .   .   .      #
31 "      .   .   .      #
32 "      .   .   .      ##
33 "      .   **      **      ##
34 "      .   ***      ***      #
35 "      .#   **      **      ##
36 "      ##      *****      ##

```

```
37 "      ###          ###      ",
38 "      ##          ##        ",
39 "      #####        #####    ",
40 "      #####        #####    "};
```

Вот такой файл получается в результате:

```
1 /* XPM */static char * sadface[] = {"32 32 5 1",
2 "      c none s mask",
3 "#      c #55aad4 s HiShadow",
4 "*      c #ff0000",
5 "+      c #4488aa s Foreground",
6 "."      c #33667f s LoShadow",
7 "      ",
8 "      .....      ",
9 "      ....      ",
10 "      ..      ",
11 "      ...      ",
12 "      ..      ",
13 "      .#      ",
14 "      .###      .###      #      ",
15 "      .. ## ##      ## ##      ##      ",
16 "      .#      .#      .##      ",
17 "      .#      .#      .#      ",
18 "      .#      .#      .#      ",
19 "      .#      .#      .#      ",
20 "      .#      .#      .#      ",
21 "      .#      .#      .#      ",
22 "      .#      .#      .#      ",
23 "      .#      .#      .#      ",
24 "      .#      .#      .#      ",
25 "      .#      .#      .#      ",
26 "      .#      .#      .#      ",
27 "      .#      .#      .#      ",
28 "      .#      .#      .#      ",
29 "      .#      .#      .#      ",
30 "      .#      .#      .#      ",
31 "      .#      .#      .#      ",
32 "      .#      .#      .#      ",
33 "      .#      .#      .#      ",
34 "      .#      .#      .#      ",
35 "      .#      .#      .#      ",
36 "      .#      .#      .#      ",
37 "      .#      .#      .#      ",
38 "      .#      .#      .#      "};
```

Что в итоге

При всей своей «игрушечности» формат XPM:

- На самом деле никакой не игрушечный.

- Долгое время был чуть ли не де-факто стандартом на маленькие изображения в графической системе X.org, правда, браузеры его показывать отказываются 😊.

- Позволяет сразу зацепить несколько тем:

1. Растр как матрица точек + *дополнительная информация*.

2. Представление цветов в пространстве RGB и именование.

3. «Палитрованное изображение».

4. Абстракции «прозрачность», «цвет фона», «цвет рисования».

5. Работа с точками растра при преобразовании.

- Неплохо подходит для изучения в курсе программирования на Си.

Калейдоскоп

Калейдоскоп

Калейдоскоп

Второй суперкран в России встаёт на колёса

По всей России идёт строительство. Строятся жилые дома, бизнес-центры, заводы, мосты и многое другое. Для всего этого нужна подъёмная техника. Автокраны требуются на любой строительной площадке. В городе Галич Костромской области находится завод строительно-дорожного машиностроения «Галичский автокрановый завод», который входит в Ассоциацию «Росспецмаш».

В этом году завод планирует выпустить второй кран КС-85713 с самой большой грузоподъёмностью – 100 тонн. У этого крана самая длинная в России телескопическая стрела – 51 метр. При помощи специального решётчатого удлинителя её можно нарастить ещё на 18 метров. Первый автокран этой модели уже передан для строительства многоэтажек в Воронеже.

В этом году запланирован выпуск двух таких машин. Дело в том, что рынок автокранов высокой грузоподъёмности довольно специфичен, поэтому тут массовости не бывает. В России этот рынок растёт вместе с ростом строительства. По словам пресс-службы завода, есть все перспективы для развития производства таких автокранов. Пока завод планирует поставлять КС-85713 лишь внутри России, но ведётся поиск потенциальных зарубежных заказчиков.

Новый автокран оснащён интеллектуальной системой управления крановыми операциями, что делает работу оператора удобной и безопасной. По желанию заказчика кран может быть смонтирован на любом подходящем автомобильном шасси отечественного или зарубежного производства. В прошлом году завод в Галиче выпустил в общей сложности 48 автокранов грузоподъёмностью от 50 до 100 тонн. Самыми востребованными оказались краны, способные поднять 50 тонн груза, которых было выпущено 42 штуки. В планах предприятия стоят задачи по расширению модельного ряда автокранов грузоподъёмностью 100 тонн.

