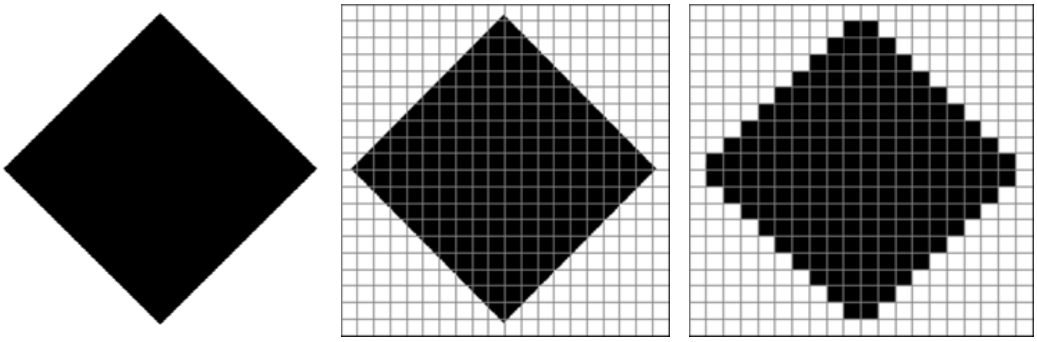
КОДИРОВАНИЕ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

Как и все виды информации, изображения в компьютере закодированы в виде двоичных последовательностей. Используют два принципиально разных метода кодирования, каждый из которых имеет свои достоинства и недостатки.

Растровое кодирование

И линия, и область состоят из бесконечного числа точек. Цвет каждой из этих точек нам нужно закодировать. Если их бесконечно много, мы сразу приходим к выводу, что для этого нужно бесконечно много памяти. Поэтому «поточечным» способом изображение закодировать не удастся. Однако, эту все-таки идею можно использовать.

Начнем с черно-белого рисунка. Представим себе, что на изображение ромба наложена сетка, которая разбивает его на квадратики. Такая сетка называется*растром*. Теперь для каждого квадратика определим цвет (черный или белый). Для тех квадратиков, в которых часть оказалась закрашена черным цветом, а часть белым, выберем цвет в зависимости от того, какая часть (черная или белая) больше.



У нас получился так называемый *растровый рисунок*, состоящий из квадратиков-пикселей.

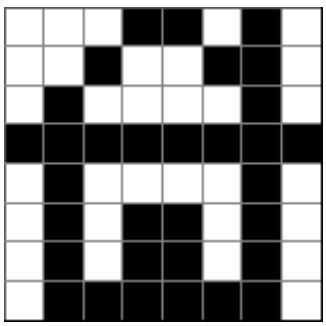
Пиксель (англ. *pixel = picture element*, элемент рисунка) – это наименьший элемент рисунка, для которого можно задать свой цвет.

Разбив «обычный» рисунок на квадратики, мы выполнили его *дискретизацию* – разбили единый объект на отдельные элементы. Действительно, у нас был единый и неделимый рисунок – изображение ромба. В результаты мы получили *дискретный*объект – набор пикселей.

Двоичный код для черно-белого рисунка, полученного в результате дискретизации можно построить следующим образом:

- заменяем белые пиксели нулями, а черные – единицами;
- выписываем строки полученной таблицы одну за другой.

Покажем это на простом примере:



0	0	0	1	1	0	1	0
0	0	1	0	0	1	1	0
0	1	0	0	0	0	1	0
1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	0
0	1	0	1	1	0	1	0
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0

Ширина этого рисунка – 8 пикселей, поэтому каждая строчка таблицы состоит из 8 двоичных разрядов – бит. Чтобы не писать очень длинную цепочку нулей и единиц, удобно использовать шестнадцатеричную систему счисления, закодировав 4 соседних бита (тетраду) одной шестнадцатеричной цифрой. Например, для первой строки получаем код $1A_{16}$:

0	0	0	1	1	0	1	0
			1			A	

а для всего рисунка: $1A2642FF425A5A7E_{16}$.

Очень важно понять, что мы приобрели и что потеряли в результате дискретизации. Самое важное – мы смогли закодировать рисунок в двоичном коде. Однако при этом рисунок исказился - вместо ромба мы получили набор квадратиков. Причина искажения в том, что в некоторых квадратиках части исходного рисунка были закрашены разными цветами, а в закодированном изображении каждый пиксель обязательно имеет один цвет. Таким образом, часть исходной информации при кодировании была потеряна. Это проявится, например, при увеличении рисунка - квадратики увеличиваются, и рисунок еще больше искажается. Чтобы уменьшить потери информации, нужно уменьшать размер пикселя, то есть увеличивать*разрешение*.

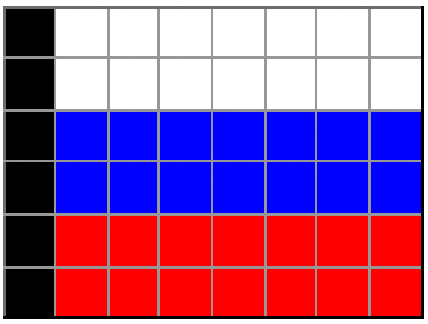
Разрешение – это количество пикселей, приходящихся на дюйм размера изображения.

Разрешение обычно измеряется в пикселях на дюйм (используется английское обозначение *ppi = pixels per inch*). Например, разрешение 254 ppi означает, что на дюйм (25,4 мм) приходится 254 пикселя, так что каждый пиксель «содержит» квадрат исходного изображения размером 0,1×0,1 мм. Если провести дискретизацию рисунка размером 10×15 см с разрешением 254 ppi, высота закодированного изображения будет 100/0,1 = 1000 пикселей, а ширина – 1500 пикселей.

Чем больше разрешение, тем точнее кодируется рисунок (меньше информации теряется), однако одновременно растет и объем файла.

Кодирование цвета

Что делать, если рисунок цветной? В этом случае для кодирования цвета пикселя уже не обойтись одним битом. Например, в показанном на рисунке изображении российского флага 4 цвета: черный, синий, красный и белый. Для кодирования одного из четырех вариантов нужно 2 бита, поэтому код каждого цвета (и код каждого пикселя) будет состоять из двух бит. Пусть 00 обозначает черный цвет, 01 – красный, 10 – синий и 11 – белый. Тогда получаем такую таблицу:



00	11	11	11	11	11	11	11
00	11	11	11	11	11	11	11
00	10	10	10	10	10	10	10
00	10	10	10	10	10	10	10
00	01	01	01	01	01	01	01
00	01	01	01	01	01	01	01

Проблема только в том, что при выводе на экран нужно как-то определить, какой цвет соответствует тому или другому коду. То есть информацию о цвете нужно выразить в виде числа (или набора чисел).

Человек воспринимает свет как множество электромагнитных волн. Определенная длина волны соответствуют некоторому цвету. Например, волны длиной 500-565 нм – это зеленый цвет. Так называемый «белый» свет на самом деле представляет собой смесь волн, длины которых охватывают весь видимый диапазон.

Согласно современному представлению о **цветном зрении** (теории Юнга-Гельмгольца) глаз человека содержит чувствительные элементы трех типов. Каждый из них воспринимает весь поток света, но первые наиболее чувствительны в области красного цвета, вторые – области зеленого, а третьи – в области синего цвета. Цвет – это результат возбуждения всех трех типов рецепторов. Поэтому считается, что любой цвет (то есть ощущения человека, воспринимающего волны определенной длины) можно имитировать, используя только три световых луча (красный, зеленый и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближенно раскладывается на три составляющих – красную, зеленую и синюю. Меняя силу этих составляющих, можно составить любые цвета. Эта модель цвета получила название RGB по начальным буквам английских слов *red* (красный), *green* (зеленый) и *blue*(синий).

В **модели RBG** яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом от 0 до 255. При этом код цвета – это тройка чисел (R,G,B), яркости отдельных каналов. Цвет (0,0,0) – это черный цвет, а (255,255,255) – белый. Если все составляющие имеют равную яркость, получаются оттенки серого цвета, от черного до белого.

Чтобы сделать светло-красный (розовый) цвет, нужно в красном цвете (255,0,0) одинаково увеличить яркость зеленого и синего каналов, например, цвет (255, 150, 150) – это розовый.

Равномерное уменьшение яркости всех каналов делает темный цвет, например, цвет с кодом (100,0,0) – тёмно-красный.

При кодировании цвета на веб-страницах также используется модель RGB, но яркости каналов записываются в шестнадцатеричной системе счисления (от 00_{16} до FF_{16}), а перед кодом цвета ставится знак #. Например, код красного цвета записывается как #FF0000, а код синего - как #0000FF.

Вот коды некоторых цветов:

Цвет	Код (R,G,B)	Код на веб-странице
Красный	(255,0,0)	#FF0000
Зеленый	(0,255,0)	#00FF00
Синий	(0,0,255)	#0000FF
Белый	(255,255,255)	#FFFFFF
Черный	(0,0,0)	#000000
Серый	(128,128,128)	#808080
Фиолетовый	(255,0,255)	#FF00FF
Голубой	(0,255,255)	#00FFFF
Желтый	(255,255,0)	#FFFF00
Тёмно-фиолетовый	(128,0,128)	#800080
Светло-желтый	(255,255,128)	#FFFF80

Всего есть по 256 вариантов яркости каждого из трех цветов. Это позволяет закодировать 256^3= 16 777 216 оттенков, что более чем достаточно для человека. Так как $256 = 2^8$, каждая из трех составляющих занимает в памяти 8 бит или 1 байт, а вся информация о каком-то цвете – 24 бита (или 3 байта). Эта величина называется*глубиной цвета*.

Глубина цвета – это количество бит, используемое для кодирования цвета пикселя.

24-битное кодирование цвета часто называют режимом **истинного цвета** (англ. *True Color* – истинный цвет). Для вычисления объема рисунка в байтах при таком кодировании нужно определить общее количество пикселей (перемножить ширину и высоту) и умножить результат на 3, так как цвет каждого пикселя кодируется тремя байтами. Например, рисунок размером 20×30 пикселей, закодированный в режиме истинного цвета, будет занимать 20×30×3 = 1800 байт. Конечно, здесь не учитывается сжатие, которое применяется во всех современных форматах графических файлов. Кроме того, в реальных файлах есть *заголовок*, в котором записана служебная информация (например, размеры рисунка).

Кроме режима истинного цвета используется также 16-битное кодирование (англ. *High Color* – «высокий» цвет), когда на красную и синюю составляющую отводится по 5 бит, а на зеленую, к которой человеческий глаз более чувствителен – 6 бит. В режиме*High Color* можно закодировать 2^{16} = 65 536 различных цветов. В мобильных телефо 12-битное кодирование цвета (4 бита на канал, 4096 цветов).

Как правило, чем меньше цветов используется, тем больше будет искажаться цветное изображение. Таким образом, при кодировании цвета тоже есть неизбежная потеря информации, которая «добавляется» к потерям, вызванным дискретизацией. Однако при увеличении количества

используемых цветов одновременно растет объем файла. Например, в режиме истинного цвета файл получится в два раза больше, чем при 12-битном кодировании.

Очень часто (например, в схемах, диаграммах и чертежах) количество цветов в изображении невелико (не более 256). В этом случае применяют **кодирование с палитрой.**

Цветовая палитра – это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

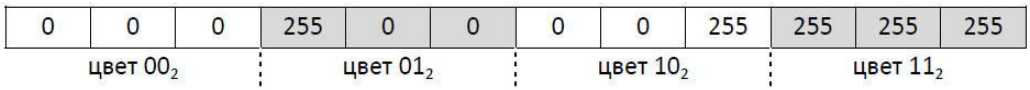
Кодирование с палитрой выполняется следующим образом:

- выбираем количество цветов *N* (как правило, не более 256);
- из палитры истинного цвета (16 777 216 цветов) выбираем любые *N* цветов и для каждого
- из них находим составляющие в модели RGB;
- каждому из цветов присваиваем номер (код) от 0 до *N*–1;
- составляем палитру, записывая сначала RGB-составляющие цвета, имеющего код 0, затем составляющие цвета с кодом 1 и т.д.
- цвет каждого пикселя кодируется не в виде значений RGB-составляющих, а как номер цвета в палитре.

Например, при кодировании изображения российского флага (см. выше) были выбраны 4 цвета:

- черный: RGB-код (0,0,0); двоичный код 002;
- красный: RGB-код (255,0,0); двоичный код 012;
- синий: RGB-код (0,0,255); двоичный код 102;
- белый: RGB-код (255,255,255); двоичный код 112;

Поэтому палитра, которая обычно записывается в специальную служебную область в начале файла (ее называют *заголовком файла*), представляет собой четыре трехбайтных блока:



Код каждого пикселя занимает всего два бита.

Чтобы примерно оценить объем рисунка с палитрой, включающей *N* цветов (без учета сжатия), нужно

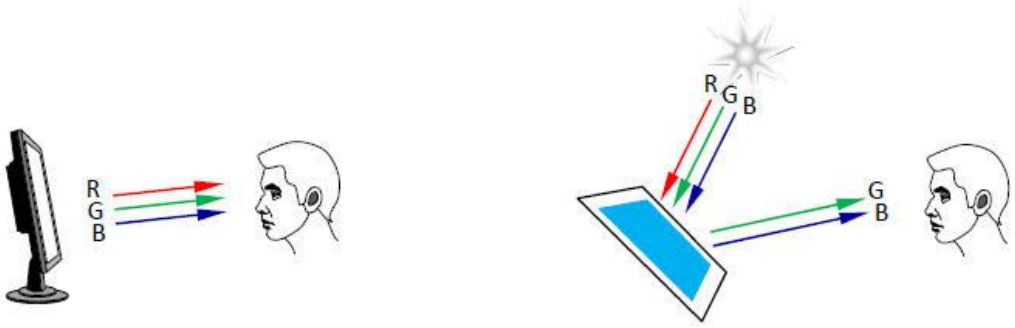
- определить размер палитры, $3 \times N$ байт или $24 \times N$ бит;
- определить *глубину цвета* (количество бит на пиксель), то есть найти наименьшее
- натуральное число k, такое что $2^k \geq N$;
- вычислить общее количество пикселей *M*, перемножив размеры рисунка;
- определить информационный объем основной части $M \times k$ бит.

В таблице приведены данные по некоторым вариантам кодирования с палитрой:

Количество цветов	Размер палитры (байт)	Глубина цвета (бит на пиксель)
2	6	1
4	12	2
16	48	4
256	768	8

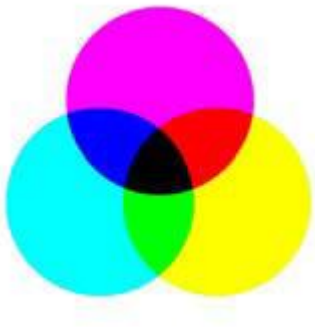
Палитры с количеством цветом более 256 на практике не используются.

RGB-кодирование лучше всего описывает цвет, который *излучается* некоторым устройством, например, монитором или экраном ноутбука. Когда же мы смотрим на изображение, отпечатанное на бумаге, ситуация совершенно другая. Мы видим не прямые лучи источника, попадающие в глаз, а *отраженные* от поверхности. «Белый свет» от какого-то источника (солнца, лампочки), содержащий волны во всем видимом диапазоне, попадает на бумагу, на которой нанесена краска. Краска поглощает часть лучей (их энергия уходит на нагрев), а оставшиеся попадают в глаз, это и есть тот цвет, который мы видим.



Например, если краска поглощает красные лучи, остаются только синие и зеленые – мы видим голубой цвет. В этом смысле красный и голубой цвета дополняют друг друга, так же, как и пары зеленый – фиолетовый и синий – желтый. Действительно, если из белого цвета (его RGB - код #FFFFFF) «вычесть» зеленый, то получится цвет #FF00FF (фиолетовый, пурпурный), а если «вычесть» синий, то получится цвет #FFFF00 (желтый).

На трех дополнительных цветах – голубом, фиолетовом и желтом – строится цветовая модель CMY (англ. ***C**yan* – голубой, ***M**agenta* – фиолетовый,***Y**ellow* – желтый), которая применяется для вывода на печать. Значения C=M=Y=0 говорят о том, что на белую бумагу не наносится никакая краска, поэтому все лучи отражаются, это белый цвет.



Если добавить голубого цвета, красные лучи поглощаются, остаются только синие и зеленые. Если сверху нанести еще желтую краску, которая поглощает синие лучи, остается только зеленый.

При наложении голубой, фиолетовой и желтой красок теоретически должен получиться черный цвет, все лучи поглощаются. Однако на практике все не так просто. Краски не идеальны, поэтому вместо черного цвета получается грязно-коричневый. Кроме того, при печати черных областей приходится «выливать» тройную порцию краски в одно место. Нужно также учитывать, что обычно на принтерах часто распечатывают черный текст, а цветные чернила значительно дороже черных.

Чтобы решить эту проблему, в набор красок добавляют черную, это так называемый*ключевой цвет* (англ. ***K**ey color*), поэтому получившуюся модель обозначают CMYK. Изображение, которое печатает большинство принтеров, состоит из точек этих четырех цветов, которые расположены в виде узора очень близко друг к другу. Это создает иллюзию того, что в рисунке есть разные цвета.

Кроме цветовых моделей RGB и CMY (CMYK), существуют и другие. Наиболее интересная из них – модель HSB (англ. ***H**ue* – тон, оттенок; ***S**aturation* – насыщенность, ***B**rightness* – яркость), которая ближе всего к естественному восприятию человека. Тон – это, например, синий, зеленый, желтый. Насыщенность – это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет светлый или темный. Любой цвет при снижении яркости до нуля превращается в черный.

Строго говоря, цвет, кодируемый в моделях RGB, CMYK и HSV, зависит от устройства, на котором этот цвет будет изображаться. Для кодирования «абсолютного» цвета применяют модель Lab (англ. *Lightness* – светлота, *a* и *b* – параметры, определяющие тон и насыщенность цвета), которая является международным стандартом. Эта модель используется, например, для перевода цвета из RGB в CMYK и обратно.

Обычно изображения, предназначенные для печати, готовятся на компьютере (в режиме RGB), а потом переводятся в цветовую модель CMYK. При этом стоит задача получить при печати такой же цвет, что и на мониторе. И вот тут возникают проблемы. Дело в том, что при выводе пикселей на экран монитор получает некоторые числа (RGB-коды), на основании которых нужно «выкрасить» пиксели тем или иным цветом. Отсюда следует важный вывод.

Цвет, который мы видим на мониторе, зависит от характеристик и настроек монитора.

Это значит, что, например, красный цвет (R=255, G=B=0) на разных мониторах будет разным. Наверняка вы видели этот эффект в магазине где продают телевизоры и мониторы – одна и та же картинка на каждом из них выглядит по-разному. Что же делать?

Во-первых, выполняется калибровка монитора – настройка яркости, контрастности, белого, черного и серого цветов. Во-вторых, профессионалы, работающие с цветными изображениями, используют *цветовые профили* мониторов, сканеров, принтеров и других устройств. В профилях хранится информация о том, каким реальным цветам соответствуют различные RGB-коды или CMYK-коды. Для создания профиля используют специальные приборы – *калибраторы* (*колориметры*), которые «измеряют» цвет с помощью трех датчиков, принимающих лучи в красном, зеленом и синем диапазонах. Современные форматы графических файлов (например, формат **.PSD** программы *Adobe Photoshop*) вместе с кодами пикселей содержат и профиль монитора, на котором создавался рисунок.

Для того, чтобы результат печати на принтере был максимально похож на изображение на мониторе, нужно (используя профиль монитора) определить «абсолютный» цвет (например, в модели *Lab*), который видел пользователь, а потом (используя профиль принтера) найти CMYK-код, который даст при печати наиболее близкий цвет.

Проблема состоит в том, что не все цвета RGB-модели могут быть напечатаны. В первую очередь это относится к ярким и насыщенным цветам. Например, ярко-красный цвет (R=255, G=B=0) нельзя напечатать, ближайший к нему цвет в модели CMYK (C=0, M=Y=255, K=0) при обратном переводе в RGB может дать значения в районе R=237, G=28, B=26. Поэтому при преобразовании ярких цветов в модель CMYK (и при печати ярких рисунков) они становятся тусклее. Это обязательно должны учитывать профессиональные дизайнеры.

Растровое кодирование

Итак, при растровом кодировании рисунок разбивается на пиксели (дискретизируется). Для каждого пикселя определяется единый цвет, который чаще всего кодируется с помощью RGB-кода. На практике эти операции выполняет *сканер*(устройство для ввода изображений) и цифровой фотоаппарат.

Растровое кодирование имеет **достоинства:**

- универсальный метод (можно закодировать любое изображение);
- единственный метод для кодирования и обработки размытых изображений, не имеющих
- четких границ, например, фотографий;

и **недостатки:**

- при дискретизации всегда есть *потеря информации*;
- при *изменении размеров* изображения искажается цвет и форма объектов на рисунке, поскольку при увеличении размеров надо как-то восстановить недостающие пиксели, а при уменьшении – заменить несколько пикселей одним;
- *размер файла* не зависит от сложности изображения, а определяется только разрешением и глубиной цвета; как правило, растровые рисунки имеют большой объем.

Существует много разных форматов растровых рисунков. Чаще всего встречаются следующие:

BMP (англ. *bitmap* – битовая карта, файлы с расширением **.bmp**) – стандартный формат в операционной системе *Windows*; поддерживает кодирование с палитрой и в режиме истинного цвета;

JPEG (англ. *Joint Photographic Experts Group* – объединенная группа фотографов-экспертов, файлы с расширением **.jpg** или **.jpeg**) – формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объема файла используется сильное сжатие, при котором изображение немного искажается, поэтому не рекомендуется использовать его для рисунков с четкими границами;

GIF (англ. *Graphics Interchange Format* – формат для обмена изображениями, файлы с расширением **.gif**) – формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов, части рисунка могут быть прозрачными, то есть на веб-странице через них будет «просвечивать» фон; в современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь, то есть при сжатии

изображение не искажается;

PNG (англ. *Portable Network Graphics* – переносимые сетевые изображения, файлы с расширением **.png**) – формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полупрозрачными (32-битное кодирование RGBA, где четвертый байт задает прозрачность); изображение сжимается без искажения; анимация не поддерживается.

Свойства рассмотренных форматов сведены в таблицу:

Формат	Истинный цвет	С палитрой	Прозрачность	Анимация
BMP	да	да	-	–
JPEG	да	–	–	–
GIF	–	да	да	да
PNG	да	да	да	–

Вы уже знаете, что все виды информации хранятся в памяти компьютера в виде двоичных кодов, то есть цепочек из нулей и единиц. Получив такую цепочку, абсолютно невозможно сказать, что это – текст, рисунок, звук или видео. Например, код 11001000_2 может обозначать число 200, букву 'И', одну из составляющих цвета пикселя в режиме истинного цвета, номер цвета в палитре для рисунка с палитрой 256 цветов, цвета 8 пикселей черно-белого рисунка и т.п. Как же компьютер разбирается в двоичных данных? В первую очередь нужно ориентироваться на расширение имени файла. Например, чаще всего файлы с расширением **.txt** содержат текст, а файлы с расширениями **.bmp**, **.gif**, **.jpg**, **.png** – рисунки.

Однако расширение файла можно менять как угодно. Например, можно сделать так, что текстовый файл будет иметь расширение **.bmp**, а рисунок в формате JPEG – расширение **.txt.** Поэтому в начало всех файлов специальных форматов (кроме простого текста, **.txt**) записывается *заголовок*, по которому можно «узнать» тип файла и его характеристики. Например, файлы в формате BMP начинаются с символов «BM», а файлы в формате GIF – с символов «GIF».

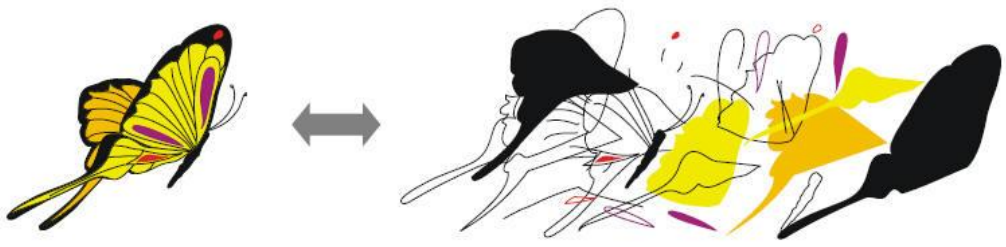
Кроме того, в заголовке указывается размер рисунка и его характеристики, например, количество цветов в палитре, способ сжатия и т.п. Используя эту информацию, программа «расшифровывает» основную часть файла и выводит его на экран.

Векторное кодирование

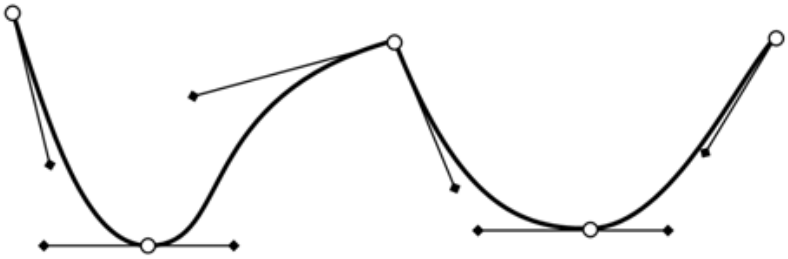
Для чертежей, схем, карт применяется другой способ кодирования, который позволяет не терять качество при изменении размеров изображения. Рисунок хранится как набор простейших геометрических фигур (*графических примитивов*): линий, многоугольников, сглаженных кривых, окружностей, эллипсов. Такой рисунок называется *векторным*.

Векторный рисунок – это рисунок, который закодирован в виде набора простейших геометрических фигур, параметры которых (размеры, координаты вершин, углы наклона, цвет контура и заливки) хранятся в виде чисел.

Векторный рисунок можно «разобрать» на части, растащив мышкой его элементы, а потом снова собрать полное изображение:



При векторном кодировании для отрезка хранятся координаты его концов, для прямоугольников и ломаных – координаты вершин. Окружность и эллипс можно задать координатами прямоугольника, в который вписана фигура. Сложнее обстоит дело со сглаженными кривыми. На рисунке изображена линия с опорными точками.



У каждой из этих точек есть «рукоятки» (*управляющие линии*), перемещая концы этих рукояток можно регулировать наклон касательной и кривизну всех участков кривой. Если обе рукоятки находятся на одной прямой, получается сглаженный узел, если нет – то угловой узел. Таким образом, форма этой кривой полностью задается координатами опорных точек и координатами рукояток. Кривые, заданные таким образом, называют *кривыми Безье* в честь их изобретателя французского инженера Пьера Безье.

Векторный способ кодирования рисунки обладает значительными **преимуществами**в сравнении с растровым тогда, когда изображение может быть полностью разложено на простейшие геометрические фигуры (например, чертеж, схема, карта, диаграмма). В этом случае при кодировании нет потери информации.

Объем файлов напрямую зависит от сложности рисунка – чем меньше элементов, тем меньше места занимает файл. Как правило, векторные рисунки значительно меньше по объему, чем растровые.

При изменении размера векторного рисунка не происходит никакого искажения формы элементов, при увеличении наклонных линий не появляются «ступеньки», как при растровом кодировании:

Самый главный **недостаток** этого метода – он практически непригоден для кодирования размытых изображений, например, фотографий.

Среди форматов векторных рисунков отметим следующие:

- **WMF** (англ. *Windows Metafile* – метафайл *Windows*, файлы с расширением **.wmf** и**.emf**) – стандартный формат векторных рисунков в операционной системе *Windows*;
- **CDR** (файлы с расширением **.cdr**) – формат векторных рисунков программы*CorelDRAW*;
- **AI** (файлы с расширением .ai) – формат векторных рисунков программы *Adobe Illustrator*;
- **SVG** (англ. *Scalable Vector Graphics* – масштабируемые векторные изображения, файлы с расширением **.svg**) – векторная графика для веб-страниц.