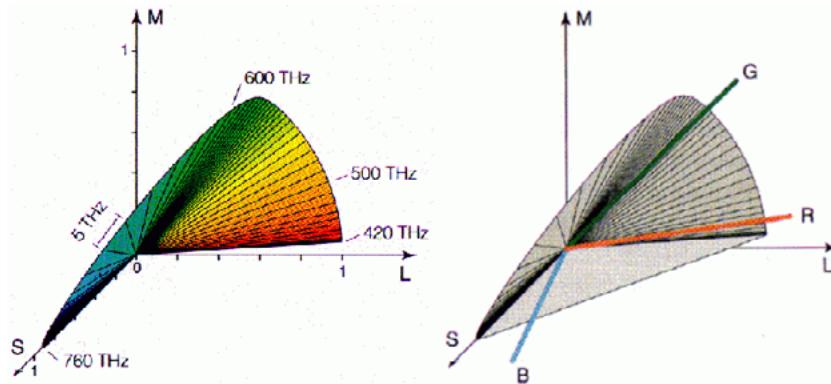


Эта модель появилась ещё в 1931 году. В компьютере её место занимает ранее описанный Lab. А почему было не обойтись такой устоявшейся за годы моделью, зачем понадобилось что-то изобретать? Оказалось, что модель хоть и эталонная, но не очень удобная для практического вычисления цветов: она имеет неравномерное изменение цвета по периметру, например, зажатую жёлтую область и растянутую зеленую. Тем не менее, если вы откроете описание любой цветовой модели, то там приводятся формулы пересчета цветов именно из XYZ, как эталонной модели человеческого зрения.

Это не единственная модель такого рода, например, существовала модель LMS – по названию диапазонов, в которых воспринимают цвет колбочки (длинные, средние и короткие волны видимого спектра), но используется именно XYZ, в которой координаты опираются так же на спектральную чувствительность колбочек L, M и S типа.



Модуль 2. Урок 1. Компьютерная графика.

Компьютерная графика – очень широкое понятие:

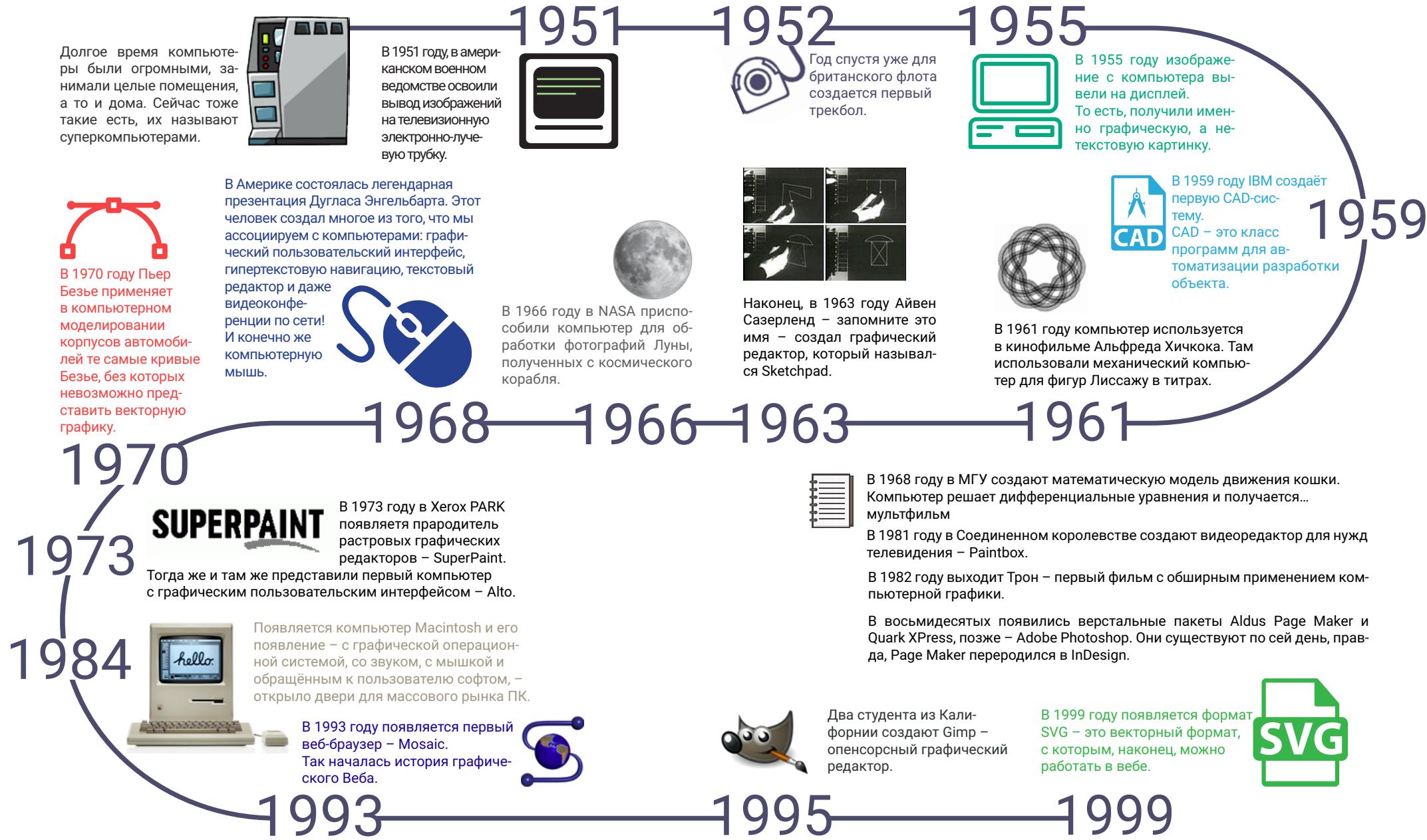
в телевидении это оформление кадра, субтитры, плашки, заставки	в кино – спецэффекты и генерированные сцены	в учебных курсах – начиная от основ и работы с фотографиями до 3D графики
Трёхмерная графика – это обычно (но не всегда) векторная графика в трёхмерном пространстве, да ещё и с растровыми текстурами.	Векторную графику называют так, потому что она оперирует кривыми и заливками.	В отличие от растровой графики, которая создается упорядоченным набором точек – растром.

Если обратить внимание на начертание букв, то текст тоже можно назвать графикой, это довольно древнее искусство каллиграфии.

Нам понадобится понимать, как устроены шрифты и как текст взаимодействует с изображениями.

abcdefghijklmnopqrstuvwxyz

История компьютерной графики



Аналого-цифровое преобразование

Свет, который мы видим, и который так долго разбирали в предыдущем модуле, имеет аналоговую природу, то есть, непрерывный и, если смотреть применительно к количеству информации, которое нам предстоит обрабатывать – бесконечно подробный.

Например: наши глаза считывают то, что они видят, при помощи отдельных рецепторов – колбочек и палочек. Если не вдаваться в подробности, о которых мы говорили в первом модуле, то можно считать, что каждый отдельный рецептор передает в мозг сигнал, который является функцией интенсивности излучения, поступающего в данное место, где расположен этот рецептор, в определенном диапазоне спектра от времени.

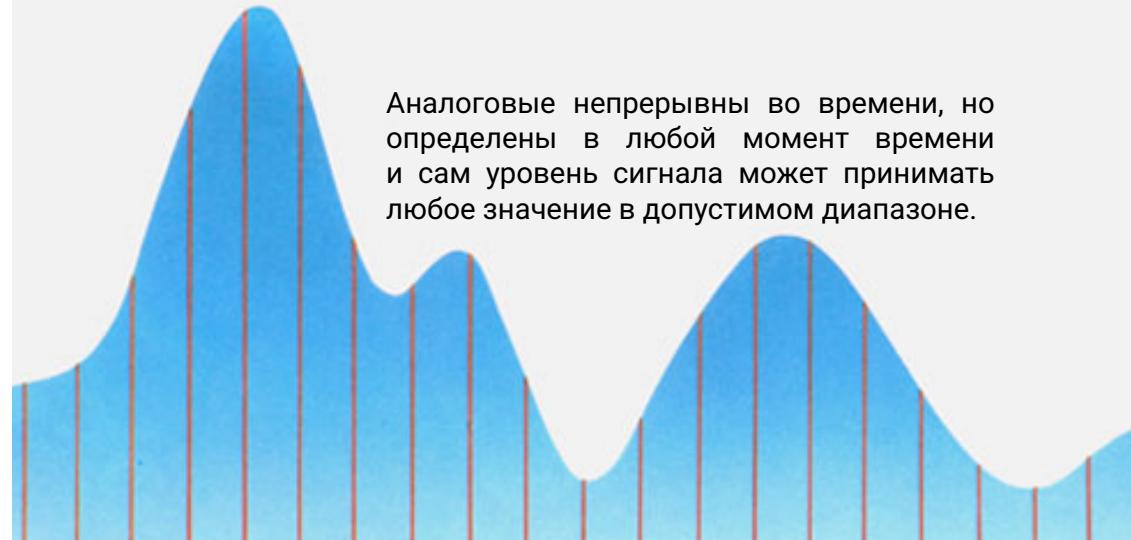
Совсем рядом может быть расположена рецептор, принимающий излучение в другом диапазоне и он может давать совсем другие уровни интенсивности. То есть, в один момент времени, практически в одном месте мы получили информацию об интенсивности в двух областях спектра. Например, красной и синей. Если добавить третий – зеленый – получим полную информацию о цвете в данной точке.

Но здесь мы получили непрерывный поток света и преобразовали его в непрерывный сигнал. Это аналоговая система. Нам же предстоит работать с компьютером – у него вся обработка, передача и хранение – в цифровом виде.

Сигналы бывают трёх типов:

- аналоговые
- дискретные
- цифровые

Аналоговые непрерывны во времени, но определены в любой момент времени и сам уровень сигнала может принимать любое значение в допустимом диапазоне.

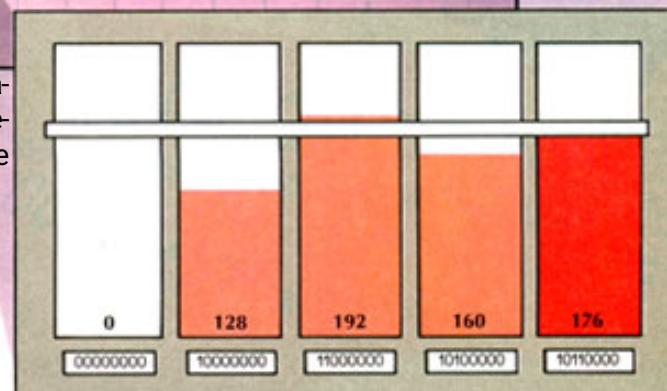


Дискретные сигналы имеют значение в конкретные моменты времени.

Их называют отсчётами. При этом уровень сигнала все так же принимает значение из непрерывного диапазона.



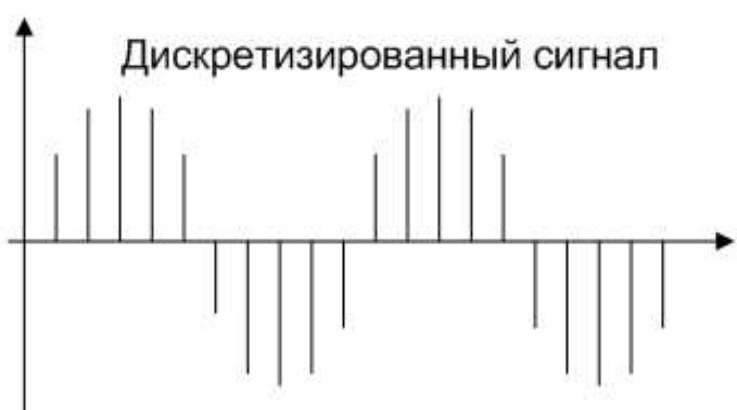
Цифровые – это сигналы, дискретные по времени и квантованные по уровню.



По сути, речь идет о том, что в аналоговом сигнале всё непрерывно. Мы можем сколь угодно часто интересоваться уровнем этого сигнала и сколь угодно точно его измерять. Вот две оси – время и уровень сигнала.



Теперь разделим время на дискретные отсчеты. Эта процедура называется дискретизацией. Так вот, дискретный сигнал – это такой же аналоговый сигнал, но который мы можем измерять лишь в определенные моменты времени. Эти моменты времени называются отсчетами, а уровень сигнала как был величиной непрерывной, так и остался. То есть, мы всё ещё можем измерять его сколь угодно точно.

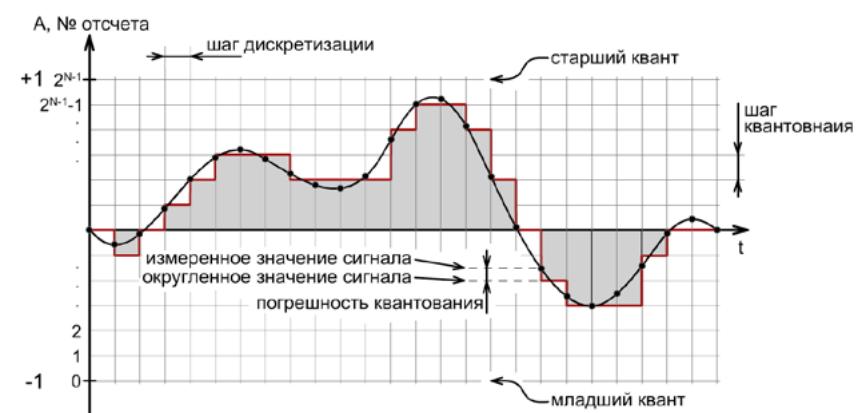


Итак, чтобы преобразовать сигнал из аналоговой формы в цифровую, производят две операции: дискретизацию и квантование.

Дискретизация бывает по времени или пространству. Например, оцифровывая звук, мы производим дискретизацию по времени. А оцифровывая изображение – дискретизацию в пространстве. Если дело дойдет до захвата видео, то будет дискретизация и по времени, и в пространстве. В данном случае по времени мы делим запись на кадры, например, 25 кадров в секунду, по пространству – на пиксели, например, 640x480.

Второй процесс – это квантование. Мы в каждом отсчете измеряем уровень сигнала. Мы поступаем следующим образом: делим весь диапазон измерений на определенное число дискретных уровней.

Сколько будет этих уровней – это зависит от того, сколько мы выделим памяти для записи одного отсчета. Дискретизация определяет количество отсчетов, то есть, измерений сигнала. А квантование – количество уровней, которые мы можем различить при каждом измерении. Все это вместе определяет, сколько памяти будет занято для записи сигнала.



Урок 2. Виды компьютерной графики. Векторная графика

В этом фрагменте мы разберём те виды графики, с которыми нам предстоит работать. Это растровая, векторная графика и текст в его графическом представлении.

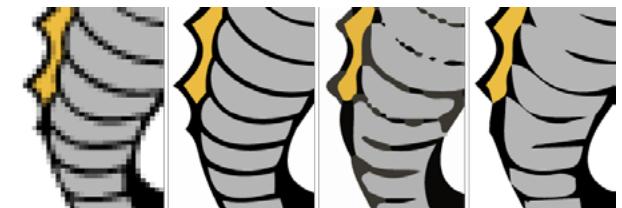
Векторная графика оперирует кривыми и заливками, которые составляют объекты в пространстве, одномерном, двухмерном или трёхмерном. В отличие от растровой, которая обходится точками – пикселями. Это и определяет их разное предназначение, возможности и затраты. Что-то удобнее описывать и хранить точками, что-то – кривыми.

Пиксель – это от английского picture element или picture cell, то есть, элемент изображения или клетка изображения. Каждый пиксель имеет определенный цвет и координату.

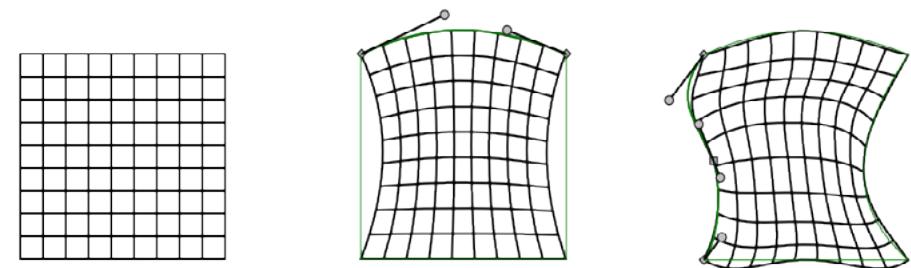
Векторная графика существует в математическом пространстве и работает с идеальными объектами. То есть, не привязана к чему-то вещественному. А следовательно – аппаратно-независима. То есть, мы можем создавать сколь угодно подробное изображение, сколь угодно большое, можем растягивать и сжимать – все это лишь выражается в координатах в некоем пространстве.



Описание объекта в виде кривых удобно тем, что это, как правило, требует не так много памяти, как описание его же – точками. С другой стороны, если мы увлечёмся и опишем объекты слишком подробно, то есть шанс, что такой векторный рисунок станет занимать больше места, чем даже несжатое изображение. Но это исключительный случай. Гораздо раньше возникнет проблема интерпретации такого рисунка – одно дело поставить на экране точку, а другое – рассчитать множество довольно сложных формул, которые описывают объекты на экране. В любом случае, здесь память расходуется на описание кривых и заливок, а не на описание каждой точки.



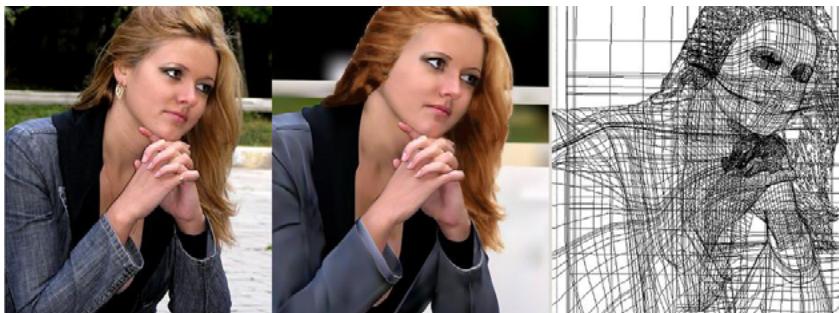
Ещё одна важная особенность векторной графики вытекает из её математической природы. Если мы применим какую-нибудь операцию сжатия, поворота или другую деформацию, то объекты не станут рассыпаться на части и не будут выглядеть потрёпанными даже после нескольких таких операций. Если же попытаться искажать, уменьшать, потом опять увеличивать растровое изображение, то бесследно это не пройдет.



Векторная графика не только не предназначена для автоматического ввода, то есть, сканирования или фотографирования, но и для фотorealистических изображений она тоже непригодна. То есть, перевести фотографию в более-менее похожую внешне картинку из кривых и заливок можно, но зачем – это большой вопрос.

Здесь есть два аспекта:

- Первый – это сложность описания. Пока мы рисовали какую-нибудь фигуру, векторная графика была очень уместна: мы получали объект и могли работать с ним. А фотография изобилует подробностями начиная с шума матрицы до размытого фона. Как все это описывать в кривых? Получается невероятное количество малозначащих деталей.
- Второй – о логической бесполезности такого ввода. Ведь для чего бы он мог понадобиться? Сэкономить место? При высокой детализации место мы можем ещё и потерять. Ещё мы можем хотеть работать с объектами отдельно. Например, человек на фоне города – отдельно человек, отдельно объекты фона. Векторные редакторы, как правило, имеют возможность трассировки – того самого преобразования растра и линии и заливки. Но при этом мы не получим возможность, например, снять человека в кадре очки, отделить его от фона или подвинуть что-нибудь. Для редактирования такая фотография совершенно непригодна.



Растровая графика

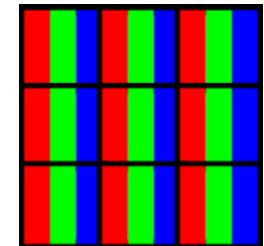
Растр – это упорядоченный набор точек, в нашем случае – это двухмерный массив, таблица или карта. Их так и называют – битовой картой – bitmap. Если есть карта, то должны быть и координаты. А по этим координатам – какие-нибудь значения. В нашем случае – значения цвета.



Здесь он записывается в координатах, про которые мы говорили в первом модуле. Например, RGB 8 бит – это три байта (по байту на каждый цвет). CMYK – соответственно, по четыре байта каждый пиксель. Так несложно вычислить размер картинки. Например, 100x100 пикселей в RGB будут занимать 30 000 байт. А в CMYK – уже 40 000. Но это без сжатия.

Особенностью растровой графики является независимость пикселей. Если в векторной графике элементом изображения является кривая, а вернее – составленный из кривых объект, то здесь нет никаких объектов, только независимые, расставленные по определенным координатам точки определенного цвета.

Если векторное изображение ввести в компьютер сложно, то растровое – значительно проще. С матрицы фотокамеры или сканера мы получаем набор цветных точек, который и составляет битовую карту, то есть, наше растровое изображение. Каждый пиксель состоит из трех субпикселей – соответственно, красного, зелёного и синего цветов. Чем мельче будет каждая отдельная точка, тем их больше нужно для описания того же изображения. И тем выше его разрешающая способность – разрешение. Это ключевое для растровой графики.

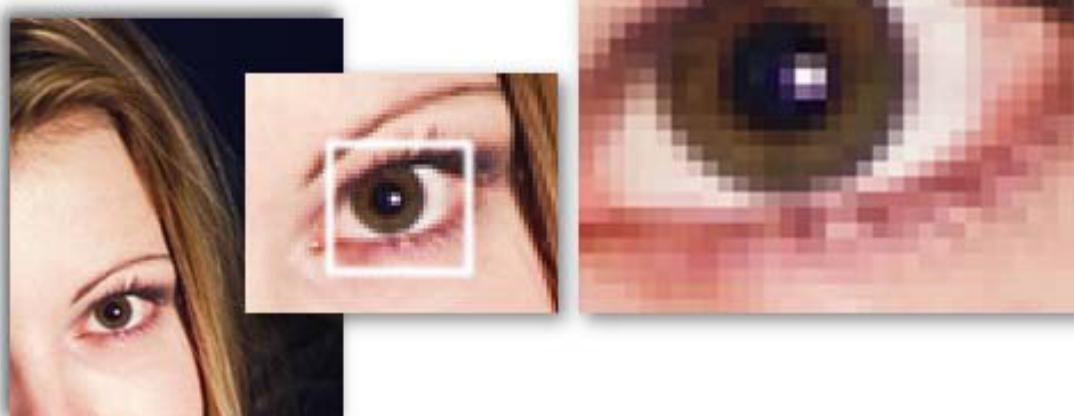


Тот факт, что растровая графика удобно вводится в компьютер, удачно дополняется тем, что она прекрасно передаёт фотorealистичные изображения. Правда, это приходится платить расходом памяти.

Если в векторном изображении количество точек, описывающих кривые, прямо задаёт размер файла, то в растровом изображении, что бы там ни было изображено, количество памяти, занятое данным количеством пикселей, определяется лишь числом этих пикселей и глубиной цвета, то есть, тем, сколько для каждого пикселя выделено памяти. В случае с CMYK – ещё и добавившимся четвёртым каналом. В остальных моделях по три канала, но вот сколько занимает каждый канал – в этом и выражается глубина цвета.

В отличие от векторной графики, растровая вполне однозначно интерпретируется разными программами. Можно создать изображение при помощи фотокамеры, отредактировать в любом удобном вам растровом редакторе, потом посмотреть в любой программе просмотра и, наконец, напечатать на любом принтере.

Главное, что приводят в укор растровым изображениям – это то, что они рассыпаются на квадратики при увеличении. Если же взять исходную картинку более высокого разрешения, то её можно разглядывать поближе – больше пикселей содержат больше информации об изображении. Но и занимают больше места.



Классический пример пикселизации – «лесенка», которая получается при наклоне оцифрованной горизонтальной линии. От неё избавиться путем повышения разрешения будет сложно – слишком много пикселей потребуется, чтобы глаз не замечал структуру изображения. А ведь такие проблемы будут возникать с любым изображением на экране. Вот текст, например, будет выглядеть неопрятно.

И здесь нам поможет простой метод, который называется Антиалайзинг. Суть его заключается в том, что в местах, где образуется такая «лесенка», соседние пиксели подкрашиваются в промежуточный цвет. Так, с одной стороны, размыивается контур изображения, но с другой – глаз не замечает неровности края. А именно это и требуется.

Aliased



Anti-Aliased

Для наглядности сведём всё вышесказанное о растровой и векторной графике в таблицу.

		
представление информации	с помощью кривых и заливок	с помощью точек – пикселей
ввод информации	предназначена, как правило, для ручного ввода	легко ввести автоматически с помощью фотокамеры или сканера
хранение информации	описание объекта кривыми занимает меньше памяти	в памяти хранится информация о каждом пикселе, чем больше пикселей – тем больше памяти требуется
вопроизведение информации	изображения существуют в математическом пространстве, поэтому разные производители программ интерпретируют их по-разному	однозначно интерпретируются разными программами
деформация	растягивание и сжатие изображения не влияют на хранимую информацию	растягивание и сжатие приводят к потерями информации
увеличение	изображение не распадается на пиксели, его можно увеличивать бесконечно, но это никак не влияет на информативность	при увеличении изображение распадается на пиксели, предел увеличения есть даже у изображений с очень высоким разрешением

Текст.

Между растровыми и векторными изображениями есть ещё такая категория, как текст. Начертание задаётся шрифтом – это упорядоченный набор изображений, в котором есть определенные места для каждого символа.

Ӯү	Кк	Ӣи	Мм	Ӣн
Ӫо	Ӣн	Ӫр	Ҫс	Ӣт
Ӳү	ӪՓ	Ӣх	Ӯи	Ӳر

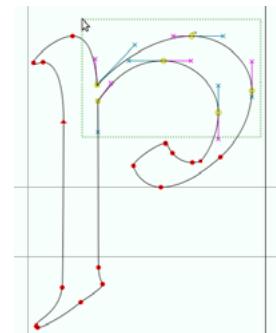
Фактически, текст представляет собой ссылки на рисунки, которые хранятся в файле шрифта. Замена шрифта меняет начертания символов, но обычно сохраняет их значение, если, конечно, в новом шрифте есть нужные символы.

Когда-то шрифты были простые, там помещалось лишь 256 символов, и тогда один символ кодировался одним байтами, потому что 256 – это два в восьмой степени, а в байте как раз восемь бит. Этого хватало, чтобы поместить два алфавита (например, латиницу и кириллицу), цифры, знаки препинания и еще какие-то спецсимволы.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
1234567890
!@#\$%^&*()_+=[]{};:'"\|/.,
АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЫЪЭЮЯ
абвгдежзиклмнопрстуфхцчшшыъэюя

Потом стало ясно, что нужно больше символов – иначе не помещаются все возможные алфавиты и начинается путаница.

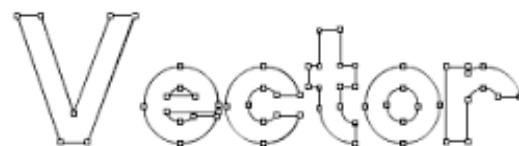
В графических редакторах, когда работают с символами шрифта и нужно быть уверенным, что ничто никуда не денется, текст преобразуют в кривые, тогда они теряют свойства редактируемого текста, зато уже свое начертание сохраняют окончательно и бесповоротно и у таких символов, которые становятся просто набором кривых, можно поправить что-нибудь в начертании.



У каждого символа есть свой номер. Например, есть такие полезные коды: **Alt-0150** – тире, **Alt-0181/0187** – кавычки «ёлочки». Таким же образом можно вызвать любой символ шрифта.

Когда вы набираете текст на клавиатуре, то вы передаёте программе лишь эти номера символов, а программа, уточнив у вас желаемый шрифт, ходит туда за начертаниями. Если вы вдруг решили, что шрифт должен быть другой, то она заменит все начертания, взяв их из нового файла.

Шрифт может содержать как растровые, так и векторные символы. Не одновременно, конечно. Большинство шрифтов в наше время – векторные. Это удобно тем, что один и тот же символ может быть представлен в любом размере. Когда вы меняете размер шрифта, скажем, в Ворде, символы просто растягиваются. Если увеличить символы до размера страницы, то мы всё равно будем видеть гладкие контуры и чёткие границы.

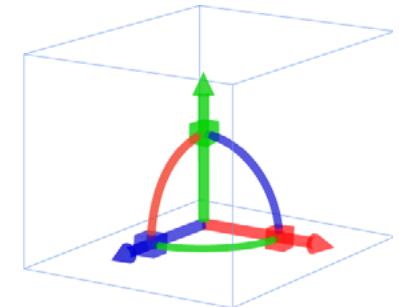


Но растровые шрифты всё же существуют. Они нужны там, где устройство вывода имеет низкое разрешение и интерпретация кривых, применительно к десятку точек по высоте – это сложная задача. То есть, что-то получится, но будет ли это узнаваемая буква – большой вопрос. Поэтому для какой-нибудь бегущей строки, например, используют точечные шрифты – каждый символ задан отдельными точками.



Трёхмерная графика.

Трёхмерная графика существует в математическом пространстве. Возьмите любые оси x,y,z и назначьте любую цену делениям на них. Постройте какой-нибудь шарик или кубик – вот и объект в пространстве.



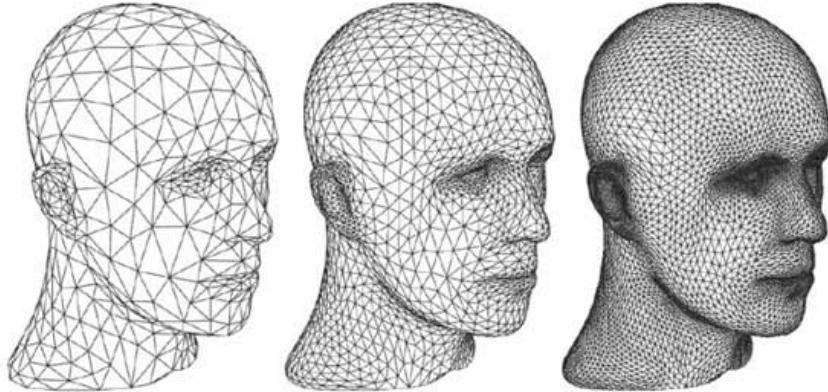
Первым шагом в создании трехмерной графики является моделирование. Если в растровой графике мы выделяем поле точек и начинаем менять их цвет, в векторной задаём плоскость и рисуем там кривые, то в трехмерной графике в трехмерном же пространстве нам предстоит создавать объекты. Можно и не обязательно трёхмерные, плоские тоже встречаются.



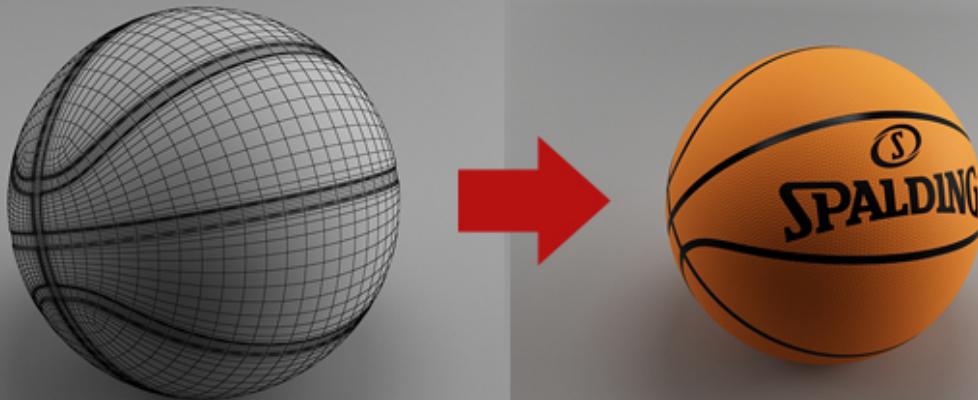
Модели задаются полигонами, что в переводе на русский значит – многоугольниками. Если посмотреть на трехмерные модели, то они обычно заданы сеткой многоугольников, а какой самый простой многоугольник? Правильно – треугольник. Три точки задают плоскость, этого достаточно. Поэтому даже четырехугольные полигоны легко превращаются в треугольные.

Если детализация растрового изображения зависит от количества точек, описывающих цвет в данном месте, векторного – от количества точек, описывающих кривые, то в трехмерной графике за детализацию отвечает количество полигонов.

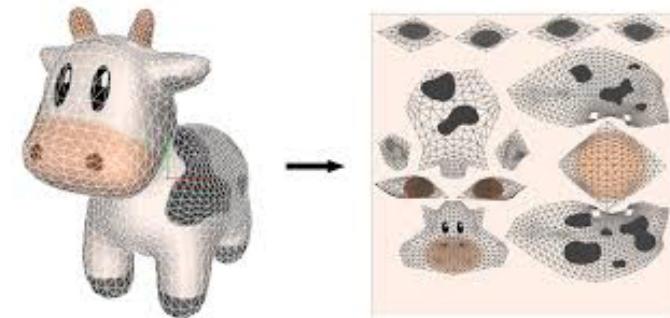
Начнём с того, как сделать много полигонов. Изначально рисуется довольно грубая модель, потом производится так называемая **тесселяция**, это разбиение существующих полигонов на несколько, чтобы они могли более плавно описывать поверхность. В отличие от кривых, используемых в двухмерной графике, здесь практикуются прямые, поэтому тема плавности изгибов остаётся актуальной.



Второй шаг на пути к успеху у зрителей – это **текстурирование**. Чтобы объект обрёл вид какого-то материала, будь то серый пластик или человеческое лицо, нужно наложить этот материал на объект. Допустим, мы накладываем однородную текстуру, фотография которой у нас есть. Здесь в пространстве встречаются растровая и векторная графика – мы фактически накладываем фотографию на плоскость. Хорошо, что текстура однородная, можно клесть как ляжет.



Теперь усложняем задачу. Нам нужно наложить текстуру точно по месту. И здесь встает вопрос **маппинга**, то есть, установления соответствия координат текстуры и объекта. Да ещё и с учётом рельефа плоскости, что бывает не так просто задать в изображении, которое размещается на объекте. Для этого создают так называемую **развёртку текстуры**.



На этом моделирование заканчивается. Чего не хватает? Не хватает света. Свет бывает общий – это как бы фоновый уровень света, не создающий теней и исходящий отовсюду. И направленный разной степени сфокусированности. От направленного уже появляются тени. Обработка теней – это вообще отдельная сложность, поэтому иногда ими пренебрегают или вместо настоящей тени рисуют что-то условное.

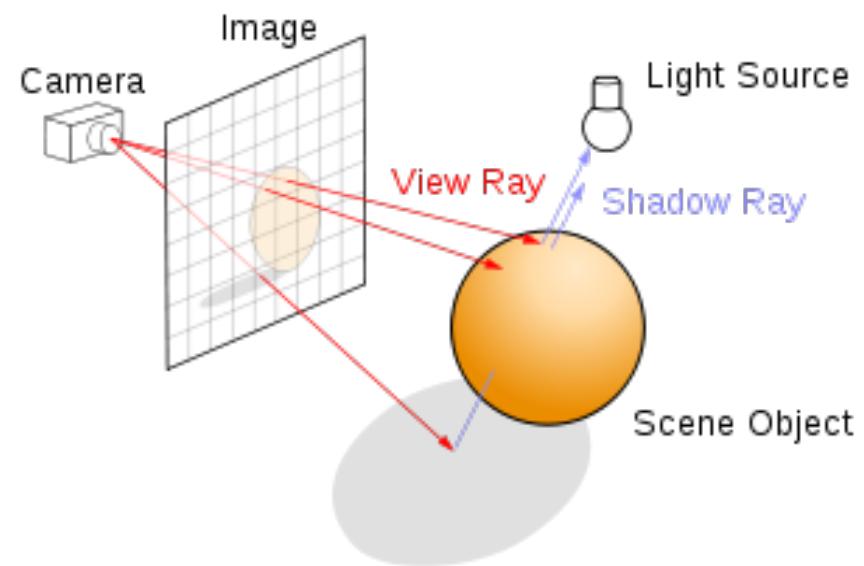


Допустим, фонари поставили, теперь вроде бы всё. Или нет? Опять, оказывается, не всё. Собственно, глаза забыли. Ну, не глаза, а камеру. Ведь всю эту красоту, которую мы сотворили, нужно в том виртуальном пространстве чем-то увидеть. Для этого создаются камеры, у них есть координаты, направление, свойства объектива, например, зум, и они могут двигаться.



Поставить камеры – это полбеды. В обычной жизни лучи сами прилетают, а в компьютерном мире их нужно ещё собрать и при этом не очень напрячь процессор. В общем, нужно как-то смоделировать ход лучей и все сопутствующие этому процессы.

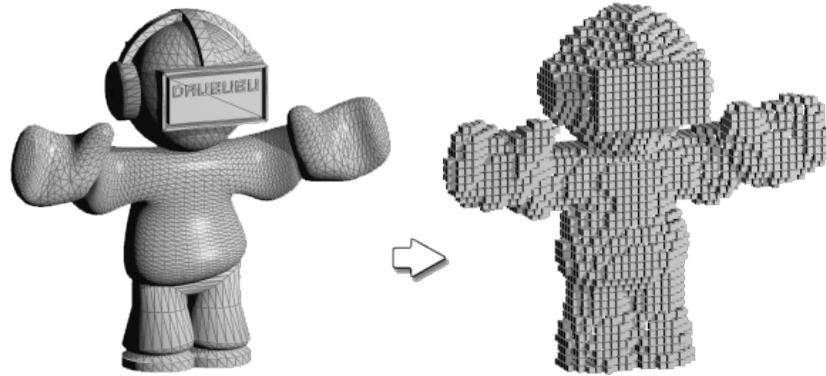
Если мы будем рассчитывать все возможные лучи, исходящие от источников света, то этим увлекательным делом можно заниматься до бесконечности. Поэтому куда разумнее было бы выделить те лучи, которые нас интересуют и посчитать именно их. Этот метод называется обратной трасировкой луча, *ray tracing*. Это не единственный метод, но он самый распространенный в рендеринге.



Для этого от каждого пикселя результирующей картинки проводится обратный ход луча. С виду это простая задача, но на самом деле, луч мог преломляться, отражаться и складываться с другими – в общем, это довольно трудоёмкая для процессора задача. Мало того, что нужно вычислять положения всех полигонов, накладывать на них текстуры, так ещё и нужно с учётом всех свойств материалов считать ход лучей.

Сложно не заметить, что трехмерная графика имеет корни в векторной графике. Отличие в том, что здесь строятся точки, по ним прямые, по ним многоугольники, по ним поверхности в трехмерном пространстве. Потом на поверхности накладываются текстуры подобно тому, как в векторных рисунках качестве заливки можно поместить растровое изображение.

Аналог растровой графики в трёхмерном пространстве тоже существует – это так называемая воксельная графика. От слов *Volume* и *Pixel*, то есть, объёмный пиксель. Если растровое изображение – это двухмерный массив точек, имеющих определенное значение цвета, то воксельное изображение – уже уже трёхмерный массив таких же точек. Можно представить, сколько памяти требуется для таких изображений.



Векторную графику тяжело вводить в компьютер, а что с трехмерной? Надо полагать, её вводить ещё тяжелее?

И да и нет. Я думаю, вам доводилось видеть, как создают современные мультфильмы или фильмы с нарисованными трёхмерными героями. Для этого используется технология *Motion Capture* – на человека вешают маркеры, в помещении ставят множество камер, которые считывают положение этих маркеров в каждый момент времени. Дальше по этим данным вычисляются координаты каждого маркера, к этим точкам привязываются части виртуального скелета, на него натягиваются объекты и текстуры.

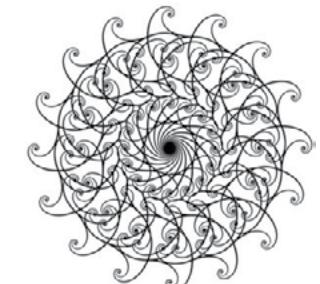
Это, конечно, не сканирование трёхмерного объекта, это сканирование движений.

Дальше объект нужно создать и привязать к траектории движений актера.



Фрактальная графика

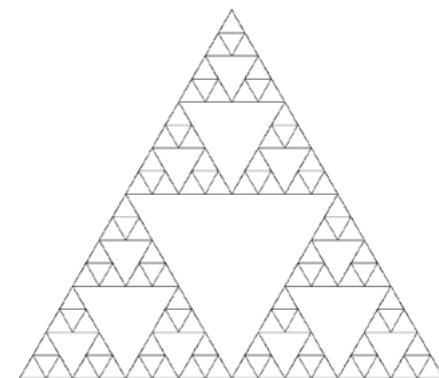
Фрактал – это бесконечно самоподобная геометрическая фигура, каждый фрагмент которой повторяется при уменьшении масштаба.



Термин был предложен Бенуа Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Рождение фрактальной геометрии принято связывать с выходом в 1977 году книги Мандельброта «Фрактальная геометрия природы».

Фрактальная графика, как и векторная – вычисляемая, но отличается от нее тем, что никакие объекты в памяти компьютера не хранятся. Фрактальное изображение строится по уравнению (или по системе уравнений), поэтому ничего, кроме формулы, хранить не надо. Изменив коэффициенты в уравнении, можно получить совершенно другую фрактальную картину.

Простейшим фрактальным объектом является фрактальный треугольник. Нужно взять равносторонний треугольник, разделить каждую из его сторон на три отрезка и на среднем отрезке каждой из сторон снова построить равносторонний треугольник со стороной, равной одной третьей стороны исходного треугольника, а на других отрезках постройте равносторонние треугольники со стороной, равной одной девятой. С полученными треугольниками повторить те же операции, пока не надоест. Треугольники последующих поколений наследуют свойства своих родительских фрактальных структур. Так рождается фрактальная фигура.



Фрактальными свойствами обладают многие объекты живой и неживой природы. Взгляните на ветку папоротника и вы увидите, что каждая дочерняя ветка во многом повторяет свойства ветки более высокого фрактального уровня.

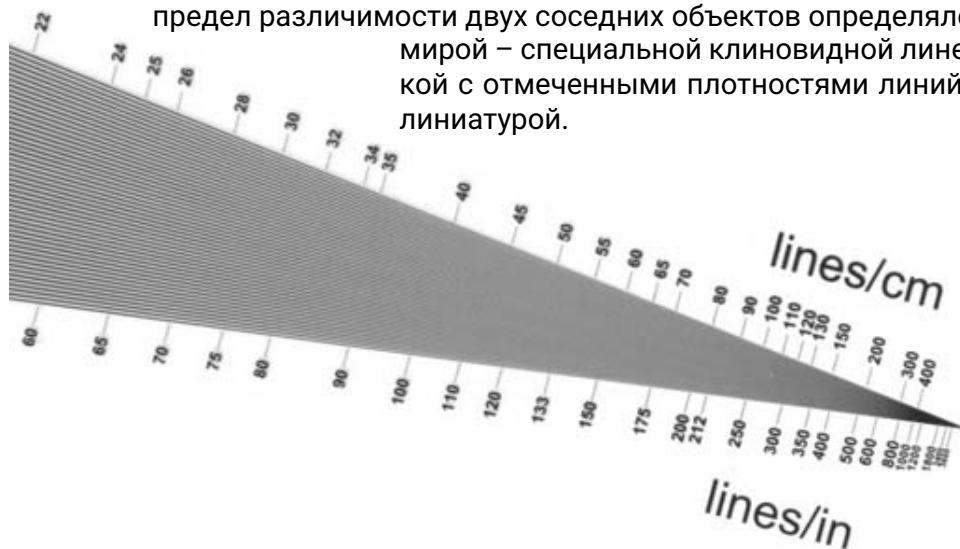
Способность фрактальной графики моделировать образы живой природы вычислительным путем часто используют для автоматической генерации необычных фрактальных иллюстраций.



Урок 3. Основные понятия компьютерной графики. Разрешение.

Разрешающая способность или разрешение изображения это величина, определяющая количество элементов изображения на единицу длины. То есть это плотность точек (пикселей).

Понятие разрешающей способности пришло из оптики, где предел различимости двух соседних объектов определялся мирой – специальной клиновидной линейкой с отмеченными плотностями линий – линиатурой.



Там, где соседние линии переставали различаться, наступал предел оптической разрешающей способности и соответствующее количество линий на дюйм (обычно в таких единицах записывают линиатуру) становилось характеристикой оптической системы.

Мы в первом уроке этого модуля уже рассматривали пример с уменьшением растрового изображения и видели, чем это кончается, если перейти определенный предел.

Представим себе путь, который проделывает изображение, полученное цифровой камерой, до отпечатка:



Помимо **объектива** и **матрицы** на изображение влияет также **процессор камеры**. Подробнее на этом мы остановимся позже.

Объектив фотокамеры имеет своё оптическое разрешение. Для измерения линиатуры используется стандартная таблица. Это нужно, поскольку объектив дает неоднородное по качеству изображение возле оптической оси и на периферии.

Матрица фотокамеры регистрирует лишь определенное число пикселей. Это число называют разрешением матрицы, хотя это и некорректно: ведь разрешение – это не количество, а плотность пикселей.

Монитор, на котором отображается фотография при обработке, напрямую не влияет на само изображение, но неправильно откалиброванный монитор искажает восприятие оператора, отчего он может некорректно редактировать изображение.

Так же, **дисплей фотокамеры** обычно имеет меньше точек, чем демонстрируемые им снимки, что скрывает часть деталей.



Полутон передается частотой или размером точек полиграфического растра, принтер же не умеет печатать «слегка пурпурным», он либо ставит точку, либо нет.



Дисплей – это электронное устройство, предназначенное для визуального отображения информации.

Монитор – это более общий термин, обозначающий в технике устройство для слежения и контроля.

Экран – это в данном случае часть дисплея, отображающая выводимую визуальную информацию.

При **подготовке изображения к печати** происходит его **растрирование** – создание четырех одноцветных изображений без полутонов, какие будут воспроизведены **принтером**.

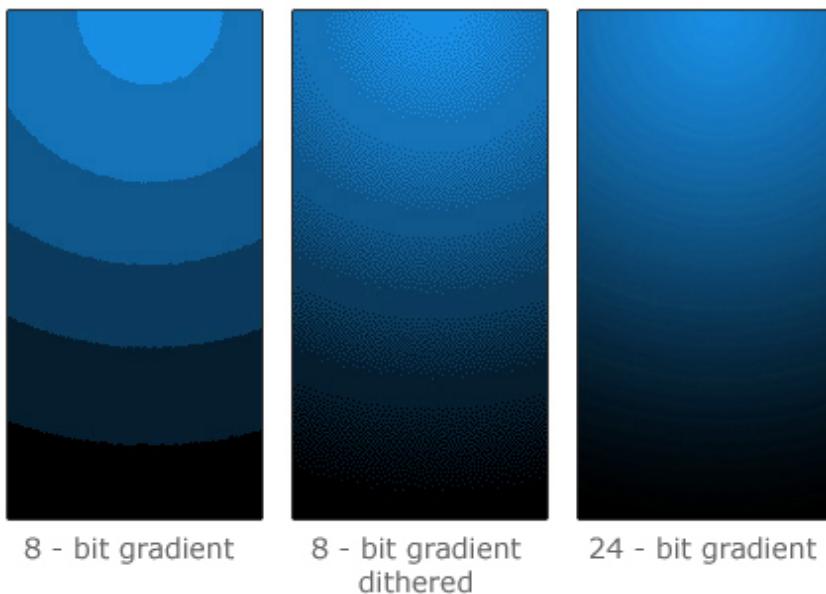
Чтобы напечатать даже черно-белую полутоновую фотографию, требуется каждому пикслю выделить значительное число точек принтера. Поэтому для печати обычно используют фотоизображения от 300 до 600 точек на дюйм, а принтеры при этом имеют разрешение порой превышающее 2000 точек на дюйм.



С разрешением дисплея существует такая сложность: мы говорим, например, «видео высокого разрешения», подразумевая количество точек по вертикали и горизонтали. Или говорим, что разрешение монитора – 1280x1024. Но это на самом деле лишь количество точек. Про разрешение дисплея уместнее было бы говорить в таком контексте: современный смартфон зачастую отображает такое же количество точек, как и огромный телевизор – 1920x1080. При этом, разумеется, размер одной точки и плотность их расположения отличаются многократно. Вот это разговор про разрешение, но в быту, и так уж прижилось, описывая характеристики дисплеев, количество точек называют разрешением. Так же, как и количество пикселей матрицы фото или видео-камеры, только там обычно называют общее количество пикселей, например, 8 мегапикселей.

В итоге, можно считать, что количество точек, выводимых дисплеем, допустимо называть разрешением, но про себя помните, что разрешение – это плотность, а не количество.

Глубина цвета



Глубина цвета – это количество градаций, допустимых для каждого пикселя. Указывается в количестве бит на пиксель или бит на канал для данной цветовой модели.

Например, обычные фотографии на сайтах – это восьмибитный RGB, то есть, в каждом канале RGB выделяется по 8 бит на пиксель. Поскольку каналов три, то один пиксель в итоге представлен 24 битами. Важно понимать, что эти цифры указываются для итогового представления пользователю, внутри файла изображение хранится в гораздо более компактном виде. Но если сжатие к изображению не применять, то его размер будет вычисляться именно так: 1 пиксель – это 24 бита, то есть, 3 байта. Умножаем на количество пикселей по вертикали и горизонтали и получаем почти точный размер файла.

Количество памяти, выделяемой для каждого пикселя, может быть различным. Например, если изображение содержит только черный и белый цвета (цвета могут быть любыми, на самом деле, важно, что их два), то для его описания достаточно выделять 1 бит на пиксель, это существенно сократит объём файла.

С другой стороны, как мы уже упоминали ранее, в ряде случаев цвет кодируют, например, 10 или 16 битами на канал. Это делается там, где важно взять максимум исходной информации, чтобы потом при редактировании выловить из нее нужную. Так записывается формат RAW цифровых фотокамер и так обычно передают информацию в компьютер сканеры. Так же, с более «глубоким» цветом работают профессиональные видеоформаты и использующая их видеотехника.

Говоря о каналах в изображении полезно помнить, что прозрачность обычно представляет собой дополнительный канал. Например, прозрачный фон в PNG – это четвертый, дополнительный к основным RGB, канал с маской прозрачности, которая представляет собой просто черно-белое изображение в 256 градаций тона. Такой дополнительный канал прозрачности в графике и видео называется Альфа-каналом.



Динамический диапазон и оптическая плотность

Говоря о цветовом разрешении, уместно вспомнить такие понятия, как динамический диапазон и оптическая плотность. Представим себе типовую задачу: нужно отсканировать фотографию и записать ее в файл. Помимо пространственного разрешения мы выберем цветовое разрешение – сколько оттенков мы хотим видеть в итоговом файле. Но тут встает вопрос, а сколько целесообразно? Сколько информации об оттенках содержится в самом сканируемом оригинале и сколько способен передать сканер?

Характеристикой сканера, описывающей его возможности по получению оттенков от черного до белого является **динамический диапазон**. Это величина, представляющая логарифм отношения максимального и минимального возможных значений величины входного параметра устройства, в данном случае – яркости по каждому цветовому каналу.

Минимальное значение обычно определяется уровнем собственных шумов или внешних помех, а максимальное – перегрузочной способностью или, в нашем случае – яркостью лампы подсветки, которая не просто так именна такая.

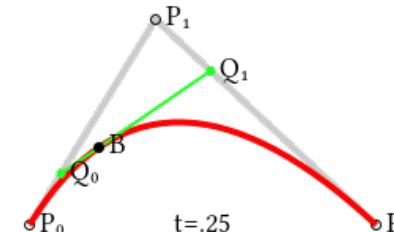
Для сканера, **оптический диапазон** характеризует интервал воспринимаемых им оптических плотностей. **Оптическая плотность** – это характеристика оригинала (носителя изображения) – мера ослабления света прозрачными объектами или отражения света непрозрачными объектами.

Кривые Безье

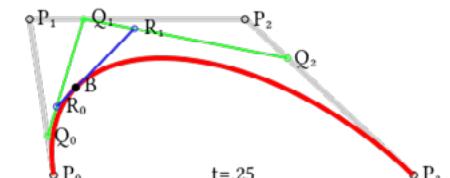
Для иллюстрации того, как работают кривые Безье, вернёмся к теме текста и шрифта. Шрифты бывают разных форматов: TrueType, PostScript и OpenType. OpenType появился относительно недавно, а True Type существовал в царстве Windows, PostScript – в царстве Macintosh. И жителям царства Windows было удивительно, почему это в соседнем царстве всё как-то лучше получается в работе с текстом.

Итак, векторные изображения, которыми являются символы шрифтов TrueType и PostScript, описываются кривыми Безье. Но разными, TrueType использует кривые второго порядка, а PostScript – третьего.

Кривая Безье второго порядка (квадратичная). Имеет три контрольные точки: две опорные – начало, конец, и одну управляемую точку.



Кривая Безье третьего порядка (кубическая). Имеет две пары точек: начало, конец и по одной управляемой точке для начала и конца.

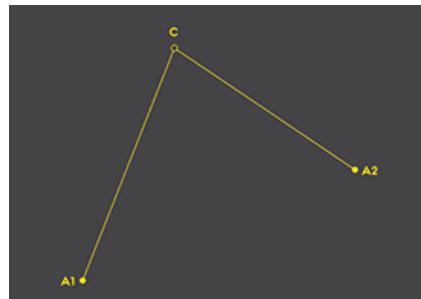


То есть, чтобы нарисовать круг, для квадратичной кривой понадобится вдвое больше опорных точек, хотя управляющих будет столько же. Итого, 16 точек для квадратичной и 12 – для кубической кривой, чтобы нарисовать окружность.

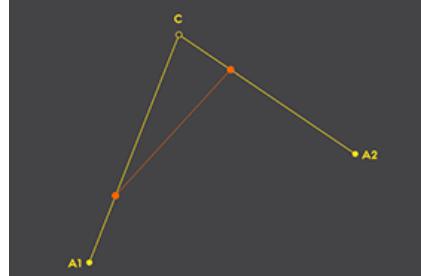
Отсюда два важных следствия:

- Чем меньше точек нужно хранить для описания изображения, тем меньше будет размер файла, здесь кубические кривые выигрывают.
- Индивидуальное управление, которое мы видим в кубических кривых, позволяет более тонко настраивать ход кривой, что, в свою очередь, дает больше свободы в формах и требует меньше опорных точек, что, в свою очередь, усложняет формирование кривых.

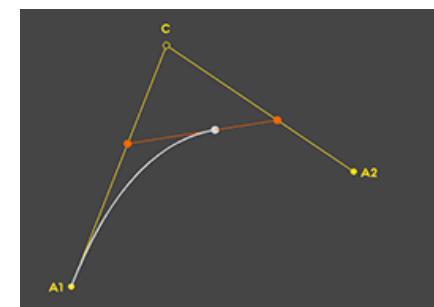
Теперь посмотрим, как рассчитываются кривые Безье. Начнем с квадратичных. Начало и конец назовём A1 и A2 соответственно, управляющая точка – С, она одна, так что без индекса. Соединим концы с управляющей точкой, получим два отрезка.



Теперь пустим по этим отрезкам по одной точке – пусть перемещаются синхронно: одна от A1 в сторону С, другая от С в сторону A2, проделывая в каждый момент времени одинаковую долю пути. Начнут и закончат они путь одновременно.



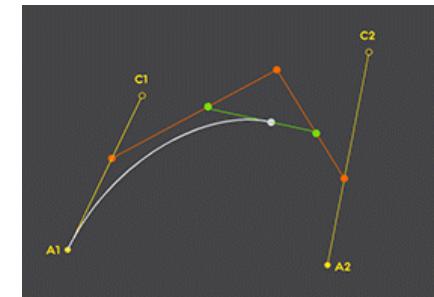
Соединим эти две бегающие точки отрезком и пустим по нему бегать третью такую точку, пусть тоже путешествует по тем же правилам. Вот её-то координаты и будем записывать, она-то и будет описывать нашу кривую Безье второго порядка.



Программы, работающие с большим количеством текста, используют относительно быстрые алгоритмы растеризации. От этого текст выглядит, возможно, не так гладко, как мог бы, но более тонкие алгоритмы требуют нескольких расчетов кривых и комбинирования результатов.

Кубические кривые растеризуются примерно таким же образом, просто построения требуют на одну промежуточную операцию больше: фактически, разница состоит в том, что управляющая точка, которая в квадратичной кривой была одна для обеих опорных точек, разделена и у каждой опорной точки – своя.

Чтобы визуализировать кривую в этом случае, сначала нужно соединить управляющие точки, пустить бегунки по полученным трем отрезкам, а на этих движущихся точках построить два движущихся вместе с ними отрезка, после чего решить задачу построения квадратичной кривой – по этим двум отрезкам снова пустить бегунки, соединить их и на полученной линии, наконец, пустить бегунок, который и будет отмечать положение кривой.



Форматы файлов

Здесь мы рассмотрим только форматы двухмерной графики, причем, разберем лишь наиболее показательные и используемые. Устаревшие и экзотические форматы оставим для энтузиастов, благо в интернете вы легко можете найти информацию о любом из них.

Итак, традиционная классификация: рассматриваем растровые и векторные форматы, универсальные и оригинальные. Большинство оригинальных форматов – смешанные, то есть, могут работать как с растровой, так и с векторной графикой, несмотря на то, что редактор, который создает эти файлы, считается, например, только растровым (как Photoshop) или векторным, как Corel Draw.

Существуют так же форматы, которые вроде бы для документов, но не рассмотреть их в нашем курсе просто нельзя – это PDF и Djvu. Запишем их отдельной категорией.

	Растровые	Векторные
Универсальные	BMP, GIF, JPEG, JPEG2000 TIFF, WebP	WMF, EMF, PS, EPS, SVG
Оригинальные	PSD	AI, CDR, CMX, FLA, SWF, DWG

BMP

Файл состоит из заголовка фиксированной длины, каталога информации об изображениях и непосредственно изображений. Может хранить растровое изображение от однобитного до полноцветного, но только 8 бит на канал, при этом поддерживается только модель RGB. Бывают 32-битные BMP, но четвертый канал зарезервирован, прозрачность здесь не поддерживается. Из сжатия там поддерживается RLE.

GIF

Формат GIF был одним из первых, хранивших цветные изображения в более-менее сжатом виде. К тому же, он имел некое подобие прозрачности (один из этих 256 цветов объявлялся прозрачным). А чуть позже у него появилась возможность анимации. Правда, он мог поддерживать только восьмibитную палитру – любые цвета из RGB, но не более 256 штук. В GIF хорошо записывать малоцветные изображения, критичные к потере четкости контуров, поскольку в нем применяется сжатие без потерь. Если надо представить текст картинкой, нарисовать схему проезда – это лучший вариант.

JPEG

Формат JPEG, как несложно догадаться, основан на алгоритме сжатия JPEG. Этим характеризуются его основные особенности как носителя графической информации – поддержка полноцветных изображений, сжатие с потерями и регулируемая степень сжатия. Формат не поддерживает прозрачность, зато, в отличие от GIF, поддерживает различные цветовые модели, знает, что такое разрешение и имеет поддержку метаданных в формате EXIF2, что дает возможность сопровождать изображение, например, с фотокамеры, подробнейшей информацией о дате и параметрах съемки, а при редактировании туда могут быть записаны самые разные данные, включая имена людей в кадре, данные об авторе и владельце изображения и много чего еще. Со второй половины 90х годов и по сей день JPEG – безусловный лидер среди графических форматов при хранении и передаче фотоизображений. В веб это второй и самый распространенный графический формат.

JPEG2000

Недостатки формата JPEG – его ярко выраженные артефакты сжатия, отсутствие поддержки прозрачности. Их попытались устранить в новой его версии – JPEG2000, но формат этот, хоть и технически более совершенный, поддержки не получил. Сохранить в него можно, но быть уверенным, что файл откроется у произвольно взятого пользователя в любой программе, которая открывает JPEG, увы, нельзя.

PNG

Отчасти, печальную судьбу JPEG 2000 можно объяснить тем, что его нововведения не были новыми. Например, еще в середине 90х фирма Macromedia предложила формат PNG, который, правда, тоже не потесnil ни GIF, ни JPEG – тогда говорили, что он опоздал. К тому же, сжатые PNG изображения имели больший размер, чем JPEG. Но здесь стоит посмотреть его характеристики: сжатие PNG – не деструктивное, притом относительно эффективное. При этом, поддерживаются и большая глубина цвета и ограниченные палитры и полноценная прозрачность (она называется альфа-каналом), а так же метаданные и даже служебная разметка на слайсы – это для создания элементов веб-страниц.

TIFF

Ещё один классический формат растровой графики – это TIFF. Если все, что мы до сих пор обсуждали, так или иначе относилось к компьютерному воспроизведению графики, то TIFF интересен тем, что это формат, с которым работают еще и в полиграфии. Формат TIFF позволяет работать с CMYK, поддерживает цветовые профили. Вообще, в TIFF можно записать не только графику, это более универсальный формат. Например, некоторые фотокамеры записывали фотографии в TIFF с JPEG сжатием, а формат этот выбириали, чтобы заодно дописать туда звук. Ещё одно применение этого формата – факсы. Причем, в исполнении факс-программ TIFFы получались многостраничными, что было удивительным для Photoshopа, который упорно видел только первый лист.

При записи в TIFF можно выбирать алгоритм сжатия: можно ограничиться стандартной внутренней архивацией ZIP, можно применить LZW, если содержимое позволяет, а можно в этом контейнере хранить самый настоящий JPEG, но надо понимать, что от этого потерь меньше не будет. Хотя, прозрачность и другие возможности формата при этом поддерживаются.

Форматы-негативы

Теперь посмотрим на семейство форматов-фотонегативов, здесь мы возьмем сразу и универсальные и оригинальные форматы, так как они тесно связаны. Цифровыми негативами называют RAW файлы, получаемые с фотокамер. Напрямую работать с ним нельзя, это просто информация, полученная с камеры до обработки. По той же причине при съёмке можно не задумываться о настройке баланса белого или о смещении экспозиции (есть в камерах такая настройка).

При «печати» такого «цифрового негатива» можно тонко настроить параметры цветопередачи и экспозиции, применить фильтры шумоподавления и резкости. Все это применяется не уже сжатому изображению, а к чистому массиву пикселей, полученному с фотокамеры. RAW – это не формат файла. Разные камеры пишут разные форматы, есть также универсальный формат цифрового негатива, он так и называется Digital Negative, DNG.

WebP

Google выпустила сразу два медиаформата – WebP, WebM. Первый хранит статические изображения (Web Pictures), второй – видео (Web Motion). Причем, статические изображения сжимаются так же, как ключевые кадры в видеоформате. Это веб-ориентированный формат, поддерживающийся преимущественно программными средствами Google (браузер Chrome, система Android) и браузером Opera. Остальные браузеры пока вынуждены использовать дополнительный модуль Flash для просмотра этих изображений.

PSD

Формат PSD – это изначально растровый формат. Но если в 4 версии Photoshop был полностью растровым, потом появилась возможность хранить текст, потом – векторные объекты, то сейчас там есть и 3д и видеодорожки и много чего еще. Разумеется, этот формат не используется, например, в веб или полиграфии, он предназначен исключительно для хранения проектов Photoshop. Это, правда, не мешает другим программам с разной степенью успешности его импортировать. Например, Corel Draw понимает этот формат, причем, может разобрать по слоям и так далее, но с определенными ограничениями.

WMF, EMF

Метафайлы – поддерживаемые в Windows векторные форматы. Это верный способ вставить векторное изображение в документы MS Office, они используются в интерфейсах программ. Каких-то других полезных применений у них не замечено.

PS

Postscript – это язык общения с принтером. Не единственный, кстати. Например, есть PCL (это язык бытовых принтеров) или, например, HPGL – для плоттеров. В файлы PS выводятся команды для печатающих устройств. Обычно мы посылаем документ на принтер, который подключен к компьютеру или находится в локальной сети. А если принтер где-то в другом месте – как быть? Вот для этого «печатают в файл» и это как раз файл Postscript.

Ещё язык Postscript интересен тем, что может при печати подменять часть изображения, не меняя все остальное, что позволяет удобно печатать персонифицированные документы, если это поддерживается печатной машиной.

EPS

Формат EPS – это разновидность PostScript. Он не привязан к конкретной печатной машине, хранит только один лист, зато может использоваться в качестве формата для обмена векторными изображениями между программами. На практике, правда, это хорошо работает, например, между программами Adobe, чего не скажешь о Кореле.

SVG

Векторные форматы до относительно недавнего времени не поддерживались в веб-среде. Это недоразумение исправил формат Scalable Vector Graphics – в названии даже заложено основное свойство векторной графики. Формат поддерживает анимацию, может отслеживать события на странице и поддерживает множество полезных функций, а так же является открытым.

По записи это формат – текстовый, там записаны функции и их параметры, от этого он несколько избыточный и файлы получаются неоправданно большими. Это недоразумение устраняется версией формата SVGZ – такой же SVG, но сжатый архиватором ZIP. Правда, до сих пор встречаются проблемы его поддержки в различных программах. Стоит отметить, что основные браузеры и графические редакторы в современных версиях работают с SVG и можно его безбоязненно использовать.

AI

Adobe Illustrator. Если Photoshop – это главный растровый редактор, то его сосед по пакету программ Adobe Illustrator – это наиболее популярный векторный редактор. Действительно, эти программы достаточно универсальны и пригодны для широкого круга задач. Но, скажем, для рисования Photoshop не самая подходящая программа, да и для работы с фотографиями, как ни странно, – тоже. Аналогично и с Иллюстратором: прекрасный инструмент дизайнера, но в допечатной подготовке технологичнее работать в CorelDraw, хотя он и не лишен недостатков. Формат Иллюстратора поддерживает хранение всего, с чем работает эта программа, то есть и растровая и векторная графика там уживаются.

CDR, CMX

Альтернативой Иллюстратору является CorelDraw. С точки зрения форматов файлов тут отличий мало – они жестко привязаны каждый к своей программе, их можно импортировать, но тут традиционно Corel показывает большую гибкость – он может импортировать и экспортить формат AI, а Иллюстратор про формат Корела не в курсе. Корел изначально поддерживал многостраничные документы, для Иллюстратора это относительно новая возможность. И главное, что нужно помнить про эти форматы – их версионная совместимость. Файлы Корела на протяжении всей своей истории упорно несовместимы с предыдущими версиями версиями редактора. То есть, сохранив файл в Кореле версии X6, вы не сможете его открыть в версии X5, если даже в этом файле нарисовали один квадратик. Иллюстратор тоже запрашивает версию при сохранении. Формат CMX используется в клипартах – библиотеках векторных изображений CorelDraw

FLA, SWF

Следующей программой, внесшей значительный вклад в компьютерную графику, является Adobe Flash, в прошлом – Macromedia Flash. Это векторная графика и анимация, причем формат редактора FLA исключительно рабочий, чтобы посмотреть результат, нужно создать файл SWF, он уже размещается на сайте. С конца 90х и примерно до 2011 года Флеш имел множество применений и порой неочевидных. Начиналось все с анимированных векторных кнопочек на сайтах, а выросло в целый веер направлений вплоть до интернет-телевещания и программирования на языке Action Script. Однако, мобильные платформы погубили Флеш, он перестал поддерживаться на основных системах и стремительно потерял популярность, хотя адекватная замена ему нашлась не во всех областях.

DWG

И последний в нашем обзоре формат – это Autodesk Autocad DWG. И программа и формат – долгожители мира компьютерной графики. Автокад, строго говоря, – трехмерная программа, но его формат обычно поддерживается в двухмерной графике и импортируется основными редакторами. Без нужды с ним связываться нет смысла, но часто бывает, что либо где-то требуют предоставить материалы в DWG, либо наоборот, присыпают их в этом формате. Нет причин для беспокойства, ранее упомянутый Корел их откроет. Одно но: если там трехмерный чертеж, то откроется векторное изображение в той проекции, которая сохранена как рабочий вид.

И в завершение обзора уделим внимание двум форматам документов.

PDF

Строго говоря, PDF – это формат программы Adobe Acrobat. Но на самом деле его создают практически все программы без участия Акробата, и есть всевозможные библиотеки по генерации PDF. Интересно в нем вот что: Во-первых, это лучший способ показать кому-нибудь документ (картинку, верстку) так, как его видели вы.

Во-вторых вывод в PDF – это, зачастую, лучший способ передать информацию между программами. PDF – это ни что иное, как электронная распечатка. Интересно, как хранится информация внутри файла: различные виды текста и графики складываются по-разному. Можно даже настроить вид кодирования для каждого типа информации.

PDF можно редактировать, этим занимается Adobe Acrobat. Но отредактировать выведенный из Ворда документ в Акробате – это совсем не то же, что в ворде. Тем не менее, добавить-переставить страницы, подвинуть что-то и даже заменить шрифт можно, но верстка там будет вести себя как ей заблагорассудится. Например, текст в PDF хранится не блоками, а строками, и то, как-то по частям.

Зато PDF поддерживает интерактивные элементы, видео, звук и шифрование. Шифрование, конечно, ломается, но в целом очень интересный формат и я очень советую подружиться с ним.

DJVU

И последний формат – DJVU. В отличие от PDF, он довольно жестко привязан к своей программе, использовать его где-то еще крайне затруднительно.

Здесь, как и в PDF, хранение разнородной информации разделено, но интересно, что вообще-то в DJVU хранят сканированные книги. И при кодировании текст, иллюстрации и фон отделяются программой, кодируются оптимальным образом.