ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физический факультет
Кафедра автоматизации физико-технических исследований

И. Г. Таранцев

КОМПЬЮТЕРНАЯ ГРАФИКА

Учебно-методическое пособие

Новосибирск
2010

Таранцев И. Г. Компьютерная графика: Учебно-методическое пособие / Новосиб. гос. ун-т. Новосибирск, 2010. 71 с.

Данное учебное пособие предназначено для подготовки специалистов в области разработки программных средств современных информационных технологий. Представленный материал позволяет ознакомится с базовыми знаниями и современным технологиям для работы с двумерной и трехмерной компьютерной графикой.

Пособие объединяет знания по двумерной и трехмерной компьютерной графике, позволяя за короткое время получить знания по самым различным областям. Ведь без четкого понимания принципов визуализации графической информации, как двумерной, так и трехмерной, невозможно адекватно использовать весь потенциал современных технических средств, предоставляемых разработчиками операционных систем и аппаратных средств визуализации.

Рецензент
доц., зам. зав. кафедрой АФТИ ФФ НГУ М. Ю. Шадрин

Учебно-методическое пособие подготовлено в рамках реализации программы развития НИУ-НГУ на 2009-2018 гг.

ОГЛАВЛЕНИЕ

1 ДВУМЕРНАЯ ГРАФИКА

1.1 Физиология цветового зрения.

Видимая часть спектра включает излучения с разной длиной волны, воспринимаемые глазом в виде различных цветов. Цветовое зрение обусловлено совместной работой нескольких светоприёмников, т. е. фоторецепторов сетчатки разных типов, отличающихся спектральной чувствительностью. Фоторецепторы преобразуют энергию излучения в физиологическое возбуждение, которое воспринимается нервной системой как различные цвета, т.к. излучения возбуждают приёмники в неодинаковой степени. Спектральная чувствительность фоторецепторов разного типа различна и определяется спектром поглощения зрительных пигментов. Каждый светоприёмник в отдельности не способен различать цвета: все излучения для него отличаются лишь одним параметром – видимой яркостью, или светлотой, т.к. свет любого спектрального состава оказывает качественно одинаковое физиологическое воздействие на каждый из фотопигментов. В связи с этим любые излучения при определённом соотношении их интенсивностей могут быть полностью неразличимы друг от друга одним приёмником. Если в сетчатке есть несколько приёмников, то условия равенства для каждого из них будут различными. Поэтому для сочетания нескольких приёмников многие излучения не могут быть уравнены никаким подбором их интенсивностей.

Основы современных представлений о цветовом зрении человека разработаны в 19 веке английским физиком Т. Юнгом и немецким учёным Г. Гельмгольцем в виде так называемой трёхкомпонентной, или трихроматической, теории цветовосприятия. Согласно этой теории, в сетчатке глаза человека имеются три типа фоторецепторов (колбочковых клеток), чувствительных в разной степени к красному, зелёному и синему свету. Однако физиологический механизм цветовосприятия позволяет различать не все

излучения. Так, смеси красного и зелёного в определённых соотношениях неотличимы от жёлто-зелёного, жёлтого и оранжевого излучений; смеси синего с оранжевым могут быть уравнены со смесями красного с голубым или с сине-зелёным. У некоторых людей наследственно отсутствует один или два светоприёмника из трёх, в последнем случае цветовое зрение отсутствует, что приводит к дальтонизму.

Цветовое зрение свойственно многим видам животных. У позвоночных (обезьяны, многие виды рыб, земноводные), а из насекомых у пчёл и шмелей Цветовое зрение трихроматическое, как и у человека. У сусликов и многих видов насекомых Цветовое зрение дихроматическое, т. е. основано на работе двух типов светоприёмников, у птиц и черепах, возможно, - четырёх. Для насекомых видимая область спектра смещена в сторону коротковолновых излучений и включает ультрафиолетовый диапазон. Поэтому мир красок насекомого существенно отличается от нашего.

Основное биологическое значение цветового зрения для человека и животных, существующих в мире несамосветящихся объектов, – правильное узнавание их окраски, а не просто различение излучений. Спектральный состав отражённого света зависит как от окраски предмета, так и от падающего света и поэтому подвержен значительным изменениям при перемене условий освещения. Способность зрительного аппарата правильно узнавать (идентифицировать) окраску предметов по их отражательным свойствам в меняющихся условиях освещения называются константностью восприятия окраски. Цветовое зрение – важный компонент зрительной ориентации животных. В ходе эволюции многие животные и растения приобрели разнообразные средства сигнализации, рассчитанные на способность животных-«наблюдателей» воспринимать цвета. Таковы ярко окрашенные венчики цветков растений, привлекающие насекомых и птиц-опылителей; яркая окраска плодов и ягод, привлекающая животных – распространителей

семян; предупреждающая и отпугивающая окраска ядовитых животных и видов, им подражающих; «плакатная» раскраска многих тропических рыб и ящериц, имеющая сигнальное значение в территориальных взаимоотношениях; яркий брачный наряд, носящий сезонный или постоянный характер, свойственный множеству видов рыб, птиц, пресмыкающихся, насекомых; наконец, специальные средства сигнализации, облегчающие у рыб и птиц взаимоотношения между родителями и потомством.

Физиологические свойства палочек и колбочек

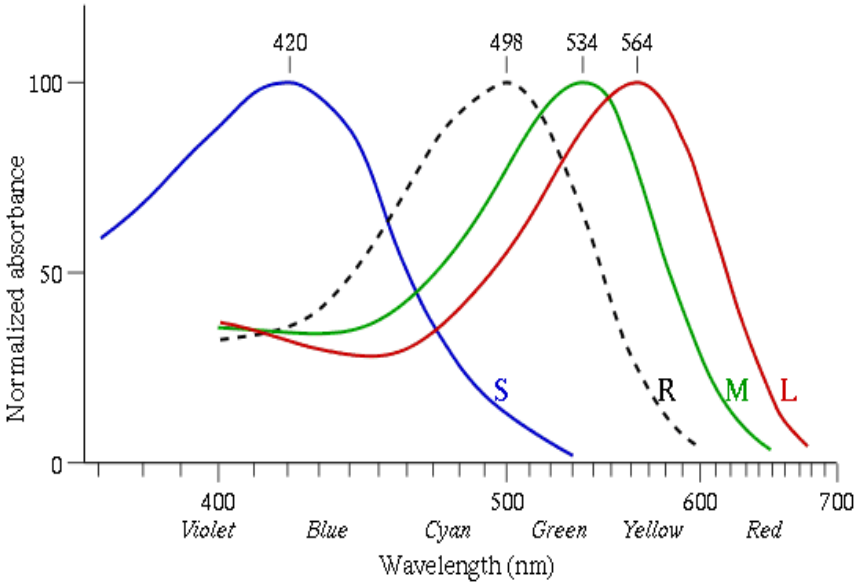
	Палочки	**Колбочки**
Светочувствительный пигмент	Родопсин	Йодопсин
Максимум поглощения пигмента	Имеет два максимума – один в видимой части спектра (500 нм), другой – в ультрафиолетовой (350 нм)	Существуют 3 вида йодопсинов, которые имеют различные максимумы поглощения: 440 нм (синий), 520 нм (зеленый) и 580 нм (красный)
Классы клеток	Нет	Каждая колбочка содержит только один пигмент. Соответственно, существуют 3 класса колбочек, чувствительных свету с разной длиной волны

Распределение по сетчатке	В центральной части сетчатки плотность палочк составляет около 150 000 на мм2 , по направлению к периферии она снижается до 50 000 на мм2. В центральной ямке и слепом пятне палочки отсутствуют.	Плотность колбочек в центральной ямке достигает 150 000 на мм 2 , в слепом пятне они отсутствуют, а на всей остальной поверхности сетчатки плотность колбочек не превышает 10 000 на мм 2 .
Чувствительность к свету	У палочек примерно в 500 раз выше, чем у колбочек	
Функция	Обеспечивают черно-белое (скототопическое зрение)	Обеспечивают цветное (фототопическое зрение)

Наличие двух фоторецепторных систем (колбочки и палочки), различающихся по световой чувствительности, обеспечивает подстройку к изменчивому уровню внешнего освещения. В условиях недостаточной освещенности восприятие света обеспечивается палочками, цвета при этом неразличимы (скототопическое зрение). При ярком освещении зрение обеспечивается главным образом колбочками, что позволяет хорошо различать цвета (фототопическое зрение).

Равномерное раздражение всех трёх типов колбочек, соответствующее средневзвешенному дневному свету, вызывает ощущение белого цвета. Очень сильный свет также возбуждает все 3 типа колбочек и потому воспринимается как излучение слепяще-белого цвета.

Спектральная чувствительность колбочек и палочек



Нормализованные графики светочувствительности колбочек человеческого глаза S, M, L. Пунктиром показана сумеречная, «чёрно-белая» восприимчивость палочек

Кроме трехкомпонентной теории цветового зрения есть еще теория оппонентных цветов, которая предполагает, что любой цвет можно однозначно описать, указав его положение на двух шкалах - «синий-желтый», «красный-зеленый». Цвета, лежащие на полюсах этих шкал, называют оппонентными. Эта теория подтверждается тем, что в сетчатке, глазном нерве и коре головного мозга существуют нейроны, которые активируются, если их рецептивное поле освещают красным светом и тормозятся, если свет зеленый. Другие нейроны возбуждаются при действии желтого цвета и тормозятся при действии синего. Предполагается, что сравнивая степень

возбуждения нейронов «красно-зеленой» и «желто-синей» системы, зрительная сенсорная система может вычислить цветовые характеристики света. Авторы теории - Max, Геринг.

Существуют экспериментальные доказательства обеих теорий цветового зрения. В настоящее время считается, что трехкомпонентная теория адекватно описывает механизмы цветовосприятия на уроне фоторецепторов сетчатки, а теория оппонентных цветов – механизмы цветовосприятия на уровне нейронных сетей.

Поскольку размеры палочек гораздо меньше размеров колбочек, то человеческий глах обладает разным пространственным разрешением для цветоного и черно-белого изображений. А именно, угловое разрешение фототопического (цветового) зрения в разы хуже углового разрешения скототопического (черно-белого или серого) зрения.

Также, в процессе обработки зрительной информации информация о больших областях с одинаковым цветом объединяется в один средний цвет и границы области. Две соприкасающиеся области с разными цветами будут четко различаться друг от друга по их общей границе. Если же области не имеют общей границы, то цветовое различение резко падает. Аналогично, человек хорошо замечает резкое изменение цвета области. То есть при попеременном отображении на одном и том же месте двух почти одинаковых изображений, человек очень тонко чувствует различия цветов. Таким образом можно говорить про высокую дифференциальную чувствительность человеческого зрения, как во временном представлении, так и в пространственном.

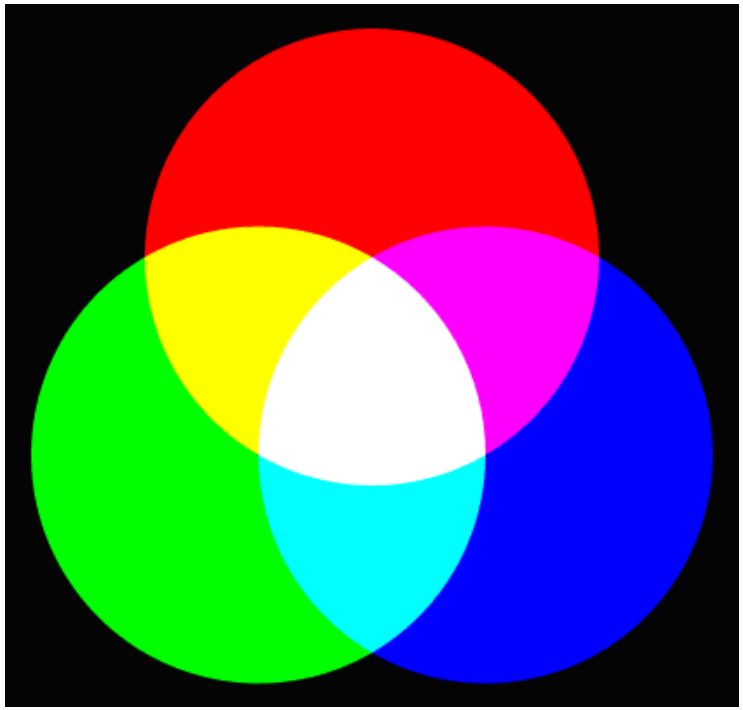
1.2 Цветовые системы координат.

Аддитивная

RGB (аббревиатура английских слов Red, Green, Blue – красный, зелёный, синий) – аддитивная цветовая модель,

описывающая способ синтеза цвета путем путём добавления (англ. addition) к черному.

Выбор основных цветов обусловлен особенностями физиологии восприятия цвета сетчаткой человеческого глаза. Цветовая модель RGB нашла широкое применение в технике, поскольку в телевизорах и мониторах применяются три электронные пушки (светодиода, светофильтра) для красного, зеленого и синего каналов. А при выключенном телевизоре его экран черный.



Если цвет экрана, освещённого цветным прожектором, обозначается в RGB как (r1, g1, b1), а цвет того же экрана, освещенного другим прожектором, — (r2, g2, b2), то при освещении двумя прожекторами цвет экрана будет обозначаться как (r1+r2, g1+g2, b1+b2).

Изображение в данной цветовой модели состоит из трёх каналов. При смешении основных цветов (основными цветами считаются красный, зелёный и синий) – например, синего (B) и красного (R), мы получаем пурпурный (M magenta), при смешении зеленого (G) и красного (R) – жёлтый (Y yellow), при смешении зеленого (G) и синего (B) – циановый (C cyan). При смешении всех трёх цветовых компонентов мы получаем белый цвет (W).

Цветовая модель RGB имеет по многим тонам цвета более широкий цветовой охват (может представить более насыщенные цвета), чем типичный охват цветов CMYK, поэтому иногда изображения, замечательно выглядящие в RGB, значительно тускнеют и гаснут в CMYK.

Субтрактивная

Четырёхцветная автотипия (CMYK: Cyan, Magenta, Yellow, Key color или blacK) – субтрактивная схема формирования цвета, используемая прежде всего в полиграфии для стандартной триадной печати. По-русски эти цвета часто называют так: голубой, пурпурный, жёлтый; но профессионалы подразумевают cyan, magenta и yellow. Печать четырьмя красками, соответствующими CMYK, также называют печатью триадными красками. Каждая краска (C,M,Y) поглощает соответствующую часть спектра. В результате из исходного белого цвета бумаги остается только часть цвета.

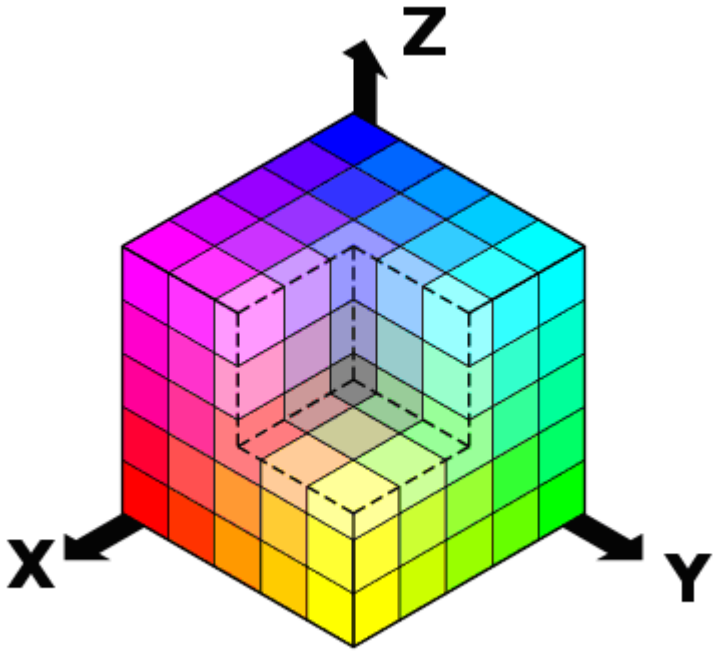
При использовании всех трех красок в равной пропорции цвет должен получиться черным (или серым). Однако точности изготовления красок недостаточно и цвет получается грязно-коричнеым или грязно-зеленым. Черная краска, изготавливаемая, например, из сажи, очень дешевая и дает точный черный цвет. Поэтому вводится дополнительный коэффициент K, указывающий количество черной краски. В результате в цетырехкомпонентной системе цветовых

координат CMYK одна из компонент всегда нулевая. В результате остается три независимых переменных.

Ясно, что цвет в CMYK зависит не только от спектральных характеристик красителей и от способа их нанесения, но и их количества, характеристик бумаги и других факторов. Фактически цифры CMYK являются лишь набором аппаратных данных для фотонаборного автомата или CTP и не определяют цвет однозначно.

Так, исторически в разных странах сложилось несколько стандартизованных процессов офсетной печати. Сегодня это американский, европейский и японский стандарты для мелованной и немелованной бумаг. Именно для этих процессов разработаны стандартизованные бумаги и краски. Для них же созданы соответствующие цветовые модели CMYK, которые используются в процессах цветоделения. Однако, многие типографии, в которых работают специалисты с достаточной квалификацией (или способные на время пригласить такого специалиста) нередко создают профиль описывающий печатный процесс конкретной печатной машины с конкретной бумагой. Этот профиль они предоставляют своим заказчикам.

Схема CMYK, как правило, обладает сравнительно небольшим цветовым охватом.



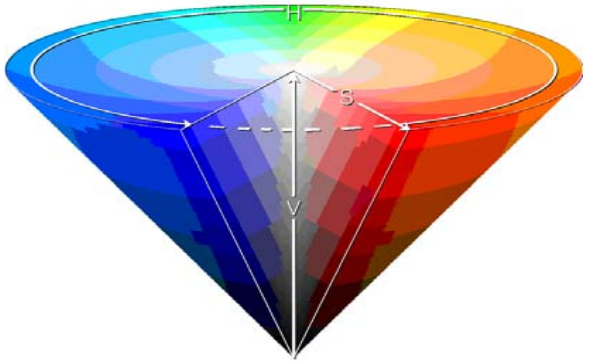
Если рассматривать цветовой куб, то видно, что системы координат RGB и CMY охватывают куб с разных сторон по главной диагонали Black – White. Если посмотреть на куб вдоль этой диагонали, то мы увидим шестиугольник с основными цветами Red-Yellow-Green-Cyan-Blue-Magenta. Это соответствует цветам радуги и мы приходим к ряду систем цветовых координат, удобных для человека, надо только шестиугольник развернуть в круг.

«Палитра художника»

HSV (англ. Hue, Saturation, Value — тон, насыщенность, значение) или HSB (англ. Hue, Saturation, Brightness — оттенок, насыщенность, яркость) — цветовая модель, в которой координатами цвета являются:

- Шкала оттенков — HueHue — цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1.
- Saturation — насыщенность. Варьируется в пределах 0—100 или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
- Value (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 и 0—1.

Модель была создана Элви Реем Смитом, одним из основателей Pixar, в 1978 году. Она является нелинейным преобразованием модели RGB.



Следует отметить, что HSV (HSB) и HSL — две разные цветовые модели.

Hue = (Alpha-arctan((2*Red-Green-Blue)/((Green-Blue)*(3^0.5))))/(2*PI)

где { Alpha=PI/2 if Green>Blue

{ Aplha=3*PI/2 if Green<Blue

{ Hue=1 if Green=Blue

Saturation = (Red^2+Green^2-Red*Green-Red*Blue-Blue*Green)^0.5

Luminosity = (Red+Green+Blue)/3

Телевизионная

Используются яркостная Y и две цветоразностные компоненты B-Y и R-Y.

Y = 0.299*Red + 0.587*Green + 0.114*Blue

U = Blue - Y = - 0.299*Red - 0.587*Green + (1- 0.114)*Blue

V = Red - Y = (1- 0.299)*Red - 0.587*Green - 0.114*Blue

Red = V + Y

Green = (Y - 0.299*(V + Y) - 0.114*(U + Y)) / 0.587

Blue = U + Y

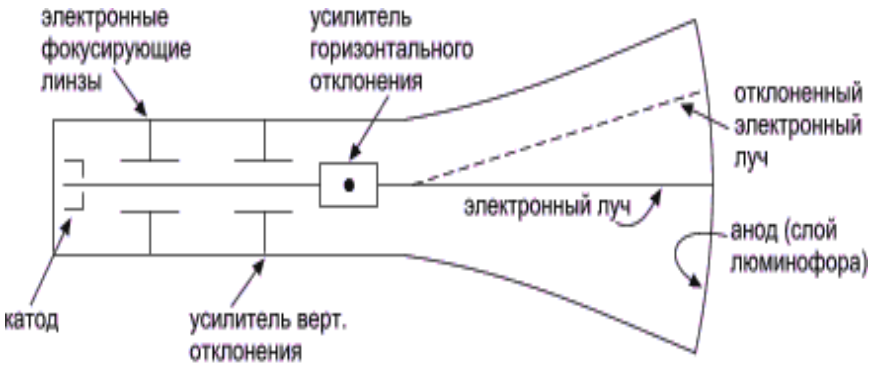
1.3 Векторная и растровая графика. Пространственное разрешение, размеры изображения, DPI.

Первыми появились устройства для отображения векторной графики. Они использовали набор геометрических примитивов, таких как точки, линии, сплайны и многоугольники, для представления изображений.

С развитием элементов памяти появились первые растровые устройства, у которых изображение строилось из отдельных точек - пикселей (pixel = picture element) и выводилось на телевизионный монитор. Изображение разбивалось на строки, а каждая строка – на отдельные пиксели. В результате получался прямоугольный растр.

Устройство электронно-лучевой трубки

Для отображения используется электронно-лучевая трубка – ЭЛТ.

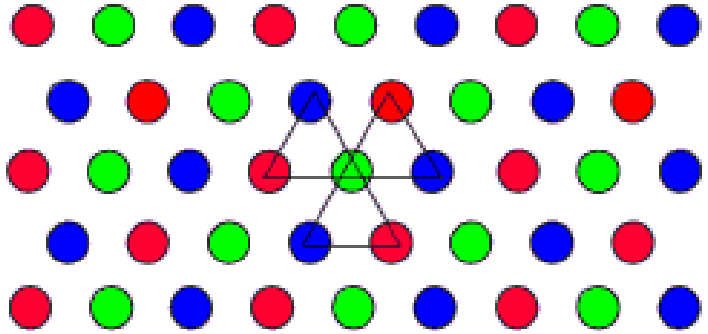


Отрицательно заряженный катод нагревают до тех пор, пока возбужденные электроны не создадут расширяющегося облака (электроны отталкиваются друг от друга, так как имеют одинаковый заряд). Эти электроны притягиваются к сильно заряженному положительному аноду. На внутреннюю сторону расширенного конца ЭЛТ нанесен люминофор. Если бы электронам ничто не препятствовало, то в результате их воздействия на люминофор весь экран ЭЛТ засветился бы ярким светом. Однако облако электронов с помощью электронных линз фокусируется в узкий, строго параллельный пучок. Теперь сфокусированный электронный луч дает одно яркое пятно в центре ЭЛТ. Луч отклоняется или позиционируется влево или вправо от центра и (или) выше или ниже центра с помощью усилителей горизонтального и вертикального отклонения – механизма развертки луча.

В растровом дисплее луч может отклоняться только в строго определенные позиции на экране, образующие своеобразную мозаику. Эта мозаика составляет видеоизображение. Люминофорное покрытие на экране растровой ЭЛТ тоже не непрерывно, а представляет собой множество тесно расположенных мельчайших точек, куда может позиционироваться луч, образуя мозаику.

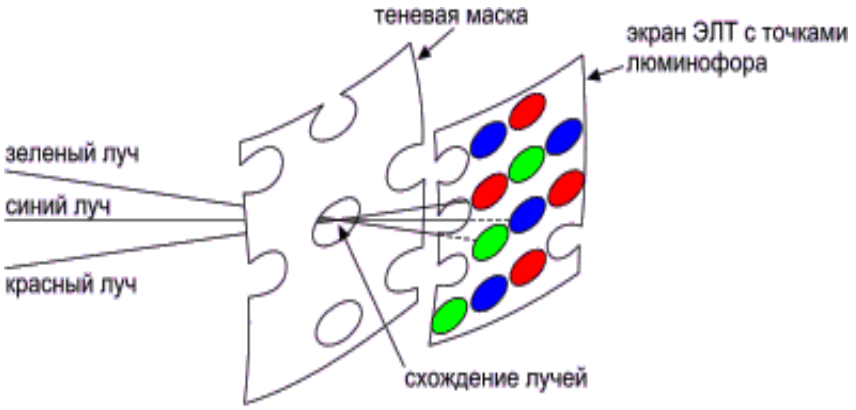
Устройство цветной растровой ЭЛТ

Цветная растровая ЭЛТ содержит три электронные пушки, по одной на каждый основной цвет: красный, зеленый и синий. Электронные пушки часто объединены в треугольный блок, соответствующий подобному треугольному блоку точек красного, зеленого и синего люминофоров на экране ЭЛТ.



Для того, чтобы электронные пушки возбуждали только соответствующие им точки люминофора (например, красная пушка возбуждала только точку красного люминофора), между электронными пушками и поверхностью экрана помещена перфорированная металлическая решетка – так называемая теневая маска. Отверстия в ней образуют такие же треугольные блоки, как и точки люминофора. Расстояния между отверстиями называются шагом. Цветовые пушки расположены таким образом, что их лучи сходятся и пересекаются в плоскости теневой маски. После прохождения через отверстие красный луч, например, защищен или маскирован от пересечения с зеленой или синей точкой люминофора. Он может пересечь лишь красную точку. Изменяя интенсивность электронного луча для каждого основного цвета, можно получить различные оттенки. Комбинация этих оттенков дает большое количество цветов для каждого пиксела. Обычно в дисплее с высоким

разрешением на каждый пиксел приходится от двух до трех цветовых триад.



Растровое изображение – это файл данных или структура, представляющая собой сетку пикселей или точек цветов (на практике прямоугольную), бумаге и других отображающих устройствах и материалах. Размерность сетки – количество строк (высота изображения) и количество столбцов (ширина изображения). Изображение в пикселе задается в каком-либо цветовом представлении: RGB, CMYK, YCbCr или другом. Также задается глубина цвета или разрядность цветовых компонент, обычно определяемая числом бит на пиксель (bit per pixel – BPP).

Следует четко различать физические и логические размеры изображения. Логические размеры – ширина и высота изображения в пикселях. Физические размеры – ширина и высота изображения в стандартных единицах длины (метры, сантиметры и т.д.). Соотношение между физическими и логическими размерами зависят от конкретного устройства отображения растровых изображений и измеряется в точках на дюйм (dot per inch – DPI). Так, характерные значения DPI для мониторов VGA – 96 DPI, а для современных принтеров –

300 DPI. Поэтому один и тот же файл (одно и то же растровое изображение) на экране монитора будет выглядеть очень крупным, а на принтере – очень маленьким. А если это изображение на принтере «растянуть» до таких же размеров, как и на мониторе, то изображение будет выглядеть нечетким, размытым.

Кроме того, различные устройства могут иметь разное разрешение (DPI) по горизонтали и вертикали. Например, старые струйные и матричные принтеры имели разрешение по горизонтали 150-300 DPI, а по вертикали – всего 72 DPI. Для таких устройств актуальным становится понятие Aspect Ratio – соотношение сторон пикселя – отношение физической ширины одного пикселя к его физической высоте. Не единичный Aspect Ratio означает, что круг, нарисованный в графическом редакторе, на устройстве будет показываться эллипсом.

1.4 Палитра, цветовое разрешение.

Память в первых растровых устройствах была очень медленной и дорогой, поэтому ее хватало только для отображения однобитовой информации. Изображение было черно-белым. Единичное значение бита включало сканирующий луч и на экране светилась точка. Нулевое значение выключало луч и экран был темным. Затем появились устройства с четырьмя банками памяти, что позволило задавать по 4 бита на пиксель. Появилась палитра из 16 цветов: шесть основных цветов (R,G,B,C,M,Y) полной интенсивности, шесть основных цветов половинной интенсивности, черный, белый, темно серый и светло серый.

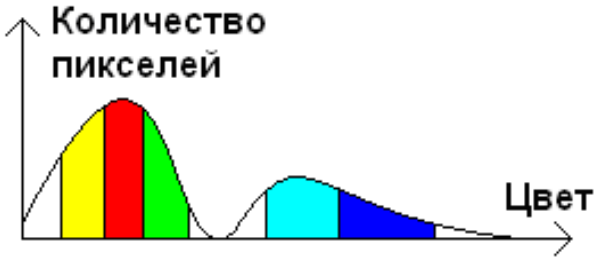
Затем появились устройства с одним байтом на пиксель – 8 BPP. Это соответствует 256 цветам в палитре. Такой стандарт используется до сих пор, поскольку в очень многих областях применений не требуется большого многообразия цветовых оттенков.

1.5 Методы уменьшения цветового разрешения (подбор палитры). Цветовой срез. Метод цветовых кубов.

Подбор палитры или уменьшение цветового разрешения (числа используемых цветов) означает выбор некоторого подмножества цветов, наиболее точно описывающих исходное полноцветное изображение.

Простейшим способом получения цветов в палитре является «цветовой срез», когда для отображения цвета используются старшие значащие биты каждой из цветовых компонент. Так, при использовании 256-цветной палитры 8 бит индекса в палитре формируются следующим образом: 3 старших бита компоненты R, затем 3 старших бита компоненты G и затем 2 старших бита компоненты B. Человеческий глаз обладает наименьшей спектральной чувствительностью к компоненте B, поэтому ей отводится меньшее число бит. В итоге для любого цвета пикселя быстро и однозначно формируется индекс в палитре. Этот способ хорош для аппаратной реализации, не зависит от изображения, но очень плохо передает информацию о цветах в исходном изображении и практически не применяется.

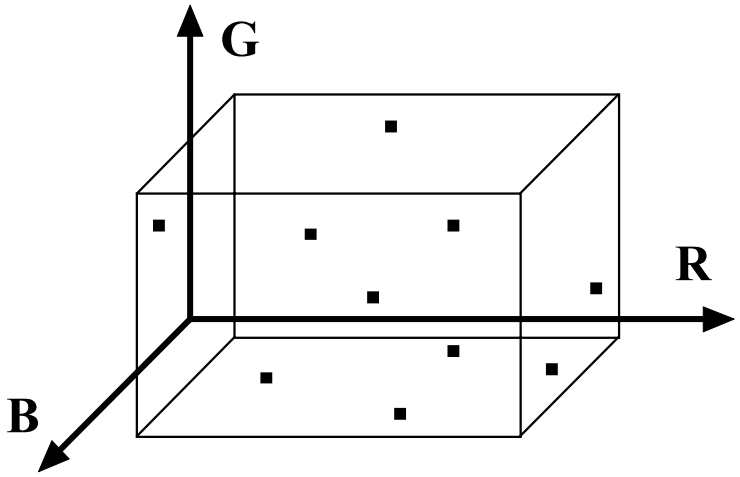
Простейшим способом эффективного подбора палитры является прямой способ построения статистики цветов в исходном изображении.



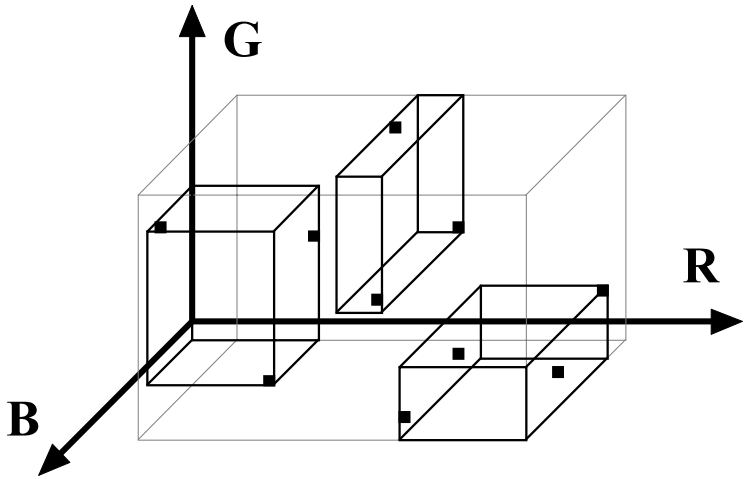
Пусть по горизонтальной оси отложены все цвета, а по вертикальной оси – количество пикселей с данным цветов, присутствующие в изображении. N – полное число пикселей в изображении. K – число цветов в палитре. Тогда N/K – число пикселей, приходящихся в среднем на один цвет в палитре. Найдя максимум статистики, можно выделить полосу цветов, суммарное количество пикселей в которой больше или равно N/K. Средневзвешенный цвет в полосе заносим в первый цвет в палитре, убираем все цвета полосы из статистики и продолжаем алгоритм. В результате мы разделим все цвета на не более чем K полос, что будет соответствовать K цветам в палитре.

Этот метод очень прост теоретически, но очень плох на практике. Во-первых, при очень высоком цветовом разрешении распределение цветов в статистике может быть слабо выраженным, без заметных максимумов. Можно специально сократить цветовое разрешение, чтобы укрупнить распределение и уменьшить полное число цветов в статистике, тем самым ускорив быстродействие алгоритма и уменьшив требования к памяти.

Метод деления цветового параллелепипеда.



В пространстве RGB строится минимальный параллелепипед, охватывающий все цвета исходной палитры (рис. вверху). Этот параллелепипед итеративно делится на подпараллелепипеды (рис. снизу), по которым строится конечная палитра: число цветов в палитре равно числу подпараллелепипедов, каждый цвет в палитре является среднеарифметическим цветом $\overline{C}$ соответствующего подпараллелепипеда.



N-ый шаг итерации. Для каждого подпараллелепипеда

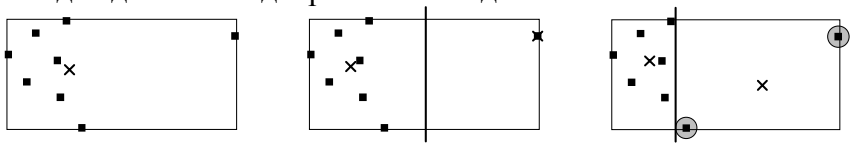
$$P = \sum_i w_i \bullet P_i$$

вычисляется общий вес , зависящий от нескольких параметов:

- максимальный размер подпараллелепипеда $L_{\max} = \max(L_R, L_G, L_B)$, где L_R, L_G и L_B являются размерами подпараллелепипеда вдоль осей R, G и B соответственно;
- число цветов в подпараллелепипеде n;

- отношение $\frac{L_{\max}}{n}$;
- отношение $\frac{L_{\max}}{L_{\min}}$, где $L_{\min} = \min(L_R, L_G, L_B)$.

Подпараллелепипед с максимальным общим весом делится на две части секущей плоскостью. В качестве секущей плоскости выбирается плоскость, нормаль которой совпадает с той осью, вдоль которой размер подпараллелепипеда максимален. Плоскость может проходить через центр подпараллелепипеда или через $\overline{C}$. При равномерном распределении цветов оба варианты практически эквивалентны. При значительной неравномерности распределения цветов (рис 3а) деление подпараллелепипеда по $\overline{C}$ может давать явные ошибки (рис. 3в). Кроме того, деление подпараллелепипеда пополам (рис. 3б) проще и быстрее в реализации - не нужно вычислять $\overline{C}$ на каждое деление подпараллелепипеда.



(а) (б) (в)

В процессе деления образуется два набора цветов, по которым вместо разделенного подпараллелепипеда строятся два новых подпараллелепипеда, причем те подпараллелепипеды, размеры которых не превышают максимально допустимой величины ошибки приближения цвета $Err_{\max}$, в дальнейшем не рассматриваются. Поскольку размер обрабатываемого изображения мал (8x8 пикселов), величину $Err_{\max}$ можно считать одинаковой для всех подпараллелепипедов.

0-ой шаг итерации. Поскольку вначале имеется только один набор цветов (исходная палитра), то для деления выбирается исходный параллелепипед (рис. 1). В остальном 0-ой шаг итерации подобен остальным шагам.

Итеративный цикл деления цветовых подпараллелепипедов завершается при достижении максимально допустимого числа цветов конечной палитры (ограничение количества подпараллелепипедов) или в том случае, когда $L_{\max}$ всех подпараллелепипедов не будут превышать $Err_{\max}$.

1.6 Дизеринг: метод упорядоченного возбуждения, распространение ошибки, комбинированные методы.

После построения палитры (или при использовании наперед заданной палитры) необходимо заменить каждый пиксель исходного полноцветного изображения некоторым индексом в палитре. Простейший способ – выбор наиболее близкого цвета в палитре. Однако он обладает рядом недостатков – плохо передает детали изображения, формирует большие одноцветные области, границы между которыми очень сильно бросаются в глаза из-за высокой дифференциальной чувствительности человеческого зрения. Если границы между одноцветными областями «размыть», сделать не такими заметными, то визуально качество результирующего изображения существенно возрастет. Аналогичная проблема встает при печати лазерными принтерами, у которых может быть только два цвета – черный (цвет краски) или белый (цвет бумаги). Используя палитру всего из двух цветов необходимо максимально достоверно отобразить произвольное изображение с 256 градациями серых цветов. Точки в принтере очень маленькие, гораздо меньше углового разрешения человеческого зрения при

рассматривании изображения в нормальных условиях. Объединяя несколько черных и белых точек в одну «общую» точку можно получить много градаций серого цвета. Собственно метод увеличения цветового разрешения за счет уменьшения пространственного разрешения и называется дизерингом.

Один из самых простых и достаточно эффективных методов дизеринга - метод упорядоченного возбуждения. В изображение вводится случайная ошибка, которая добавляется к интенсивности каждого пикселя до ее сравнения с выбранной пороговой величиной. Добавление совершенно произвольной ошибки не приводит к оптимальному результату. Тем не менее, существует оптимальная аддитивная матрица ошибки, минимизирующая эффекты появления фактуры на изображении. Матрица ошибки добавляется к изображению таким же способом, как расположены клетки на шахматной доске. Минимальная матрица упорядоченного возбуждения имеет размер 2 * 2. Оптимальная 2 * 2-матрица, которую первым предложил Лим, имеет вид:

$$|D_2| = \begin{vmatrix} 0 & 2 \\ 3 & 1 \end{vmatrix}$$

Матрицы 4 * 4, 8 * 8 и больших размеров получают с помощью рекуррентных соотношений (n => 4)

$$|D_n| = \begin{vmatrix} 4D_{n/2} & 4D_{n/2} + 2U_{n/2} \\ 4D_{n/2} + 3U_{n/2} & 4D_{n/2} + U_{n/2} \end{vmatrix}$$

где n — размер матрицы и

$$|U_n| = \begin{vmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & & \\ \dots & & & \\ 1 & & & \end{vmatrix}$$

Например, матрица возбуждения размера 4 * 4 имеет вид:

$$|D_4| = \begin{vmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{vmatrix}$$

Как показывают два этих примера, из матрицы возбуждения Dn можно породить n2 интенсивностей. С увеличением n изображение не теряет пространственного разрешения. Приведем алгоритм упорядоченного возбуждения.

Алгоритм упорядоченного возбуждения

```
Xmin, Xmax, Ymin, Ymax — пределы растра
Mod — функция, возвращающая остаток от
целого деления первого аргумента на второй
  for y = Ymax to Ymin step -1
    // для каждого пиксела на строке (слева
направо)
    for x = Xmin to Xmax
        // определяем позицию в матрице
возбуждения
        i = (x Mod n) + 1
        j = (y Mod n) + 1
        // определяем выводимое значение
пиксела
        if I(x, y) < D(i, j) then
```



```
            Пиксел(x, y) = Черный
        else
            Пиксел(x, y) = Белый
        end if
        изображаем пиксел
    Display Пиксел(x, y)
    next x
  next y
finish
```

Другим очень популярным методом является метод «распространения ошибки». В методе, разработанном Флойдом и Стейнбергом, эта ошибка распределяется на окружающие пиксели. Распределение ошибки происходит всегда вниз и вправо. Следовательно, при генерации изображения в порядке сканирования возвращаться обратно не нужно. В частности, в алгоритме Флойда-Стейнберга 3/8 ошибки распределяется вправо, 3/8 — вниз и 1/4 — по диагонали, как это показано на следующем рисунке.

Для порога, равного среднему между минимальной и максимальной интенсивностями, T = (Белый + Черный)/2, алгоритм формулируется следующим образом.

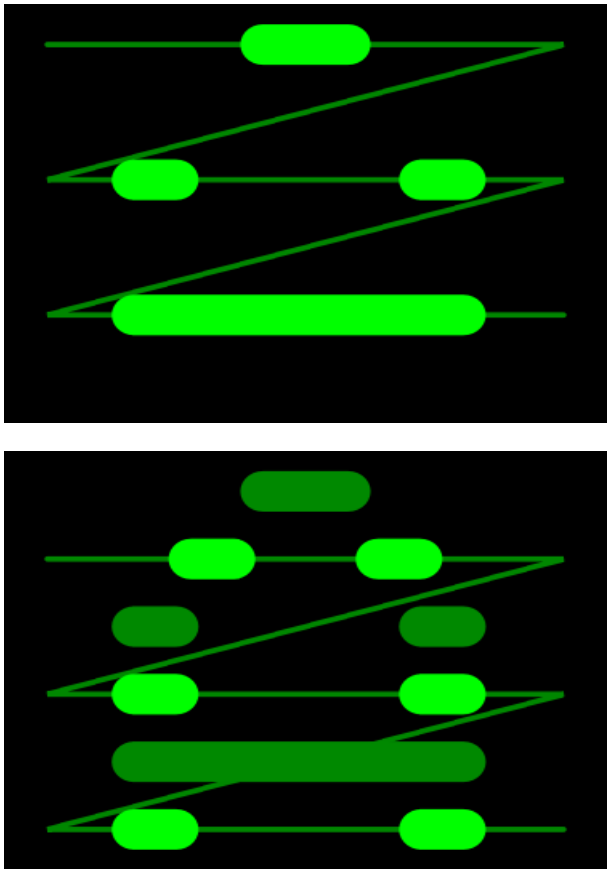
```
Xmin, Xmax, Ymin, Ymax — пределы растра
 T = (Черный + Белый)/2
 for y = Ymax to Ymin step -1
   // для каждого пиксела на строке (слева
направо)
   for x = Xmin to Xmax
         // определяем выводимое значение
пиксела для
         // пороговой величины T и вычисляем
ошибку
         if I(x, y) < T then
                Пиксел(x, y) = Черный
                Ошибка = I(x, y) - Черный
         else
                Пиксел(x, y) = Белый
                Ошибка = I(x, y) - Белый
         end if
         изображаем пиксел
         Display Пиксел(x, y)
         // распределяем ошибку на соседние
пикселы
         I(x + 1, y) = I(x + 1, y) + 3 *
0шибка/8
         I(x, y - 1) = I(x, y - 1) + 3 *
Ошибка/8
         I(x + 1, y - 1) = I(x + 1, y - 1) +
Ошибка/4
   next x
 next y
finish
```

Распределение ошибки на соседние пикселы улучшает вид деталей изображения, так как информация, заключенная в изображении, не теряется.

1.7 Телевизионная графика. Чересстрочная развертка. Уплотнение спектра (гребенчатый фильтр).

Телевидение начиналось с передачи черно-белого изображения, когда яркостный сигнал Y построчно сканировался и передавался в виде частотно модулированного сигнала в эфир. Телевизионный приемник принимал сигнал, демодулировал из него сигнал яркости и разворачивал его построчно на электронно-лучевой трубке. Модуль развертки в телевизионном приемнике смещал луч по вертикали и по горизонтали с кадровой и строчной частотами соответственно.

При разработке телевизионных стандартов использовалось большое число различных остроумных инженерных решений. Одним из таких решений является чересстрочная развертка, когда в последовательности изображений, передаваемых в эфир, четные изображения смещены вверх на пол строки, а нечетные – вниз на пол строки. В результате при передаче статического изображения телевизионный приемник отображает в два раза больше строк, чем их есть в одном изображении, поскольку люминофор монитора обладает высокой остаточной светимостью и строки соседних изображений не затирают, а дополняют друг друга.



На верхнем рисунке показано одно изображение, передаваемое в текущий момент времени. На нижнем рисунке показаны два соседних изображения – текущее и предыдущее, причем предыдущее изображение хотя и имеет меньшую светимость, но успешно воспринимается зрительным аппаратом человека. Два соседних изображения называют двумя полями одного кадра. В телевидении их называют четным и нечетным полями, но в разных телевизионных стандартах первое поле кадра бывает разным – четным или нечетным. Для пользователя проще использовать понятие Upper Field First или Lower Field First, т.е. верхнее поле первым или нижнее поле первым. Верхнее поле – это поле,

включающее самую верхнюю видимую строку кадра. Соответственно нижнее поле – это поле, включающее самую нижнюю видимую строку кадра.

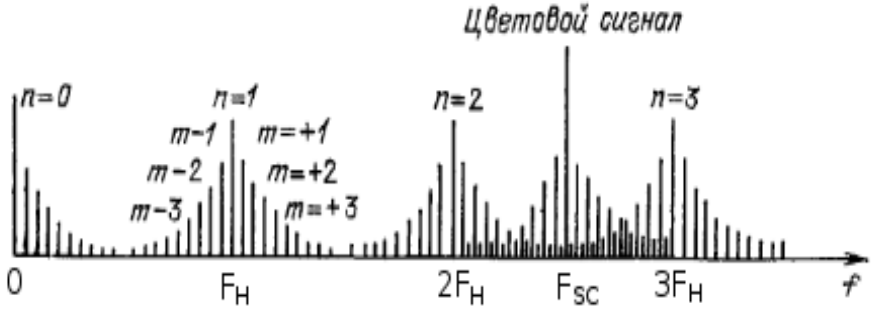
С другой стороны, при передаче динамических изображений, когда каждая следующая картинка отличается от предыдущей, половинного разрешения кадра достаточно. Ведь для того, чтобы зритель мог рассмотреть детали, он должен вглядываться в изображение, а для этого оно должно быть относительно неподвижным.

Если изображение не изменяется во времени (статический кадр), то механизм развертки формирует периодический сигнал, полностью повторяющийся с кадровой частотой.

Из Фурье анализа известно, что спектр такого сигнала состоит только из гармоник кадровой частоты. При изменении изображения во времени спектр будет расплываться. В ситуации, когда каждое следующее изображение будет абсолютно не похоже на предыдущее, спектры гармоник частоты кадров перекроются и в спектре результирующего сигнала будут присутствовать все частоты. Однако на практике изображения изменяются очень слабо. А при резкой смене изображений можно передать «любую» картинку, все равно зритель будет не в состоянии ее правильно распознать. Для надежного распознавания требуется некоторое время, существенно большее длительности одного кадра. Поэтому для подавляющего большинства случаев спектр телевизионного сигнала носит ярко выраженный гребенчатый характер.

1.8 Стандарты цветного телевидения: NTSC, PAL, SECAM.

Свойство гребенчатости спектра активно использовали разработчики цветного телевидения. Сигнал, несущий информацию о цвете обладает таким же гребенчатым характером, как и сигнал яркости, поскольку он сформирован тем же механизмом развертки. Если сместить спектры яркостного и цветоразостных сигналов на величину N*Fкадра + Fкадра/2, то пики одного спектра попадут в нули другого спектра.



При декодировании сигнала можно использовать гребенчатые фильтры и разделить спектры сигналов, а значит и сами сигналы.

Во всех трех стандартах NTSC, SECAM, PAL цветовая информация передается на поднесущей в спектре яркостного сигнала. Отличие состоит в способе модуляции поднесущей и выборе цветовых сигналов.

NTSC

Для получения цветоразностного сигнала используются квадратурная модуляция цветовой поднесущей с частотой 3.58 МГц, с использованием двух балансных модуляторов, питаемых одной и той же поднесущей с фазами, смещенными на 90°.

Параметры развертки: 480 видимых строк, 29,97 кадров в секунду.

PAL

Как и в системе NTSC, сигналы U и V используются для модуляции цветовых поднесущих используя два балансных модулятора, работающих со сдвигом по фазе на 90°. Основное отличие стандарта PAL от NTSC состоит сложении сигналов цветности двух соседних строк, что вместе с изменением фазы сигнала V на противоположную с началом каждой строки позволяет подавить цветовые искажения, возникающие в результате изменения сигналов U и V из-за фазовых помех. При этом основное подавление приходится на искажения вызванные ошибкой фазы, которая слабо меняется от строки к строке, а именно эти искажения наиболее заметны на экране.

Параметры развертки: 576 видимых строк, 50 кадров в секунду.

SECAM

Система SECAM (Séquentiel couleur à mémoire — последовательная цветная с памятью) исторически является первым стандартом цветного телевидения в Европе.

Отличительной чертой стандарта SECAM является поочередная передача двух цветоразностных сигналов через строку на частотно-модулированной поднесущей при непрерывной передаче сигнала яркости.

Параметры развертки: 576 видимых строк, 50 кадров в секунду.

1.9 Искажения, возникающие при передаче телевизионных изображений.

Основные искажения связаны с проблемами построения гребенчатых фильтров для разделения спектров яркостного и цветоразностных сигналов. В бытовых телевизионных приемниках вместо сложных и дорогих гребенчатых фильтров используют простые полосовые фильтры, что приводит к неполному разделению спектров и к

так называемым перекрестным помехам, когда высокочастотные составляющие сигнала яркости попадают в спектр сигнала цветности как низкочастотные помехи и наоборот, высокочастотные составляющие сигнала цветности попадают в спектр сигнала яркости как низкочастотные помехи. Поэтому на резких, контрастных цветовых переходах возникают низкочастотные яркостные помехи – по контрастной границе движутся маленькие точки, похожие на муравьев.

Для сигнала NTSC характерной проблемой является синфазная помеха, приводящая к изменению цветового тона. Есть даже расшифровка аббревиатуры NTSC как Never Twice the Same Color.

У сигнала SECAM характерной проблемой является формирование несуществующих цветов вдоль широкой горизонтальной границы разных цветов. Например, при полноэкранном отображении российского флага на границе синего и красного цветов возникают фиолетовый и черный цвета.

1.10 Простые алгоритмы сжатия изображений: RLE, LZW.

Алгоритм RLE

Алгоритм группового кодирования (RLE) позволяет сжимать данные любых типов независимо от содержащейся в них информации. Сама информация влияет лишь на коэффициент сжатия. Он является самым быстрым, простым и понятным алгоритмом компрессии и при этом иногда оказывается весьма эффективным.

Он оперирует группами данных – последовательностями, в которых один и тот же символ встречается несколько раз подряд. Суть его заключается в том, что при кодировании строка повторяющихся символов, составляющих группу, заменяется строкой, которая содержит

сам повторяющийся символ (значение группы) и количество его повторов (счетчик группы).

Из специфики работы алгоритма можно легко понять, для каких изображений коэффициент сжатия будет максимальным. Это изображения, содержащие минимальное количество цветов, без плавных переходов (с резкими границами) и с большими участками, заполненными одним цветом. Так как сканирование данных осуществляется построчно с лева на право, высокий коэффициент сжатия будет достигаться на изображениях, строки которых содержат достаточно длинные цепочки одинаковых пикселей. Время сжатия и распаковки существенно не различаются. Лучший, средний и худший коэффициенты сжатия для алгоритма RLE равны 32, 2, 0.5, соответственно.

Алгоритм группового кодирования применяется в форматах PCX, TIFF, BMP.

Алгоритм LZW

Алгоритм LZW – это универсальный алгоритм сжатия данных без потерь, созданный Абрахамом Лемпелем (Abraham Lempel), Якобом Зивом (Jacob Ziv) и Терри Велчем (Terry Welch). Он был опубликован Велчем в 1984 году, в качестве улучшенной реализации алгоритма LZ78, опубликованного Лемпелем и Зивом в 1978 году. Алгоритм разработан так, чтобы его можно было быстро реализовать, но он не обязательно оптимален, поскольку он не проводит никакого анализа входных данных.

Данный алгоритм при сжатии (кодировании) динамически создаёт таблицу преобразования строк: определённым последовательностям символов (словам) ставятся в соответствие группы бит фиксированной длины (обычно 12-битные). Таблица инициализируется всеми односимвольными строками (в случае 8-битных символов – это 256 записей). По мере кодирования, алгоритм просматривает текст символ за символом, и сохраняет каждую

новую, уникальную двухсимвольную строку в таблицу в виде пары код/символ, где код ссылается на соответствующий первый символ. После того как новая 2-символьная строка сохранена в таблице, на выход передаётся код первого символа. Когда на входе читается очередной символ, для него по таблице находится уже встречавшаяся строка максимальной длины, после чего в таблице сохраняется код этой строки со следующим символом на входе; на выход выдаётся код этой строки, а следующий символ используется в качестве начала следующей строки.

Алгоритму декодирования на входе требуется только закодированный текст, поскольку он может воссоздать соответствующую таблицу преобразования непосредственно по закодированному тексту.

Алгоритм

1. Инициализация словаря всеми возможными односимвольными фразами.
2. Инициализация входной фразы w первым символом сообщения.
3. Считать очередной символ K из кодируемого сообщения.
4. Если КОНЕЦ_СООБЩЕНИЯ, то выдать код для w, иначе
5. Если фраза wK уже есть в словаре, то присвоить входной фразе значение wK и перейти к Шагу 3, иначе выдать код w, добавить wK в словарь, присвоить входной фразе значение K и перейти к Шагу 3.

Конец

Алгоритм Хаффмана

Название алгоритм получил в честь разработчика - Дэвида Хаффмана. Метод Хаффмана строит таблицы кодов,

базирующихся на частоте повторения элементов. Чем чаще встречается тот или иной элемент, тем короче будет заменяющий его код.

По алгоритму Хаффмана для сжатия файла необходимо прочитать его полностью и подсчитать сколько раз встречается каждый элемент, подлежащий кодированию. Затем, последовательно объединяем два элемента с наименьшей частотой появления в новый составной элемент (узел), частота появления которого полагается равной сумме частот составляющих его элементов. В конечном итоге мы построим дерево, каждый узел которого имеет суммарную вероятность всех узлов, находящихся ниже него. Прослеживаем путь к каждому листу дерева, помечая направление к каждому узлу (например, направо - 1, налево - 0). Выполнив эту процедуру для каждого листа, получим бинарные коды для каждого изначального элемента – таблицу Хаффмана.

В чистом виде алгоритм используется для двуцветных (как правило, черно-белых) изображений, в которых преобладают большие блоки, заполненные одним цветом.

Время сжатия и распаковки существенно не различаются. Лучший, средний и худший коэффициенты сжатия для алгоритма Хаффмана равны 8, 1.5, 1, соответственно (без учета размера таблицы Хаффмана).

Кодирование по Хаффману используется в форматах JPEG и PNG.

1.11 Сложные алгоритмы сжатия изображений: JPEG, Wavelet.

JPEG (Joint Photographic Experts Group — объединённая группа экспертов в области фотографии) — является широкоиспользуемым методом сжатия фотоизображений.

При сжатии изображение переводится в цветовую систему YCbCr (YUV). Далее каналы изображения Cb и Cr,

отвечающие за цвет, уменьшаются в 2 раза (по линейному масштабу)- формат 2:1:1. Уже на этом этапе необходимо хранить только четверть информации о цвете изображения.

Реже используется уменьшение цветовой информации в 4 раза (4:1:1) или сохранение размеров цветовых каналов как есть (1:1:1). Количество программ, которые поддерживают сохранение в таком виде, относительно невелико.

Далее цветовые каналы изображения, включая черно-белый канал Y, разбиваются на блоки 8 на 8 пикселей. Каждый блок подвергается дискретно-косинусному преобразованию. Коэффициенты преобразования обходятся зигзагом с левого угла верхнего угла в правый нижний, в результате чего все низкочастотные компоненты, несущие основную информацию об изображении оказываются в начале строки, а все высокочастотные компоненты – в конце строки. Полученные коэффициенты подвергаются квантованию делением на строку такой же длины, у которой делители возрастают от начала к концу строки. Конкретный набор делителей задается параметром «качество». Чем ниже «качество», тем больше делители, и тем меньше оказываются коэффициенты после квантования. Далее строка коэффициентов сжимается вариантом алгоритма RLE (цепочка нулей заменяется длиной), а затем получившиеся пары {число нулей и первое ненулевое значение} упаковываются с помощью кодов Хаффмана.

Сторка, используемая для квантования коэффициентов, хранится вместе с изображением. Обычно она строится так, что высокочастотные коэффициенты подвергаются более сильному квантованию, чем низкочастотные. Это приводит к огрублению мелких деталей на изображении. Чем выше степень сжатия, тем более сильному квантованию подвергаются все коэффициенты.

В алгоритме JPEG2000 вместо дискретно-косинусного преобразования используется вейвлет-преобразование (WaveLet). Вейвлеты – это семейство функций, которые

локальны во времени и по частоте, и в которых все функции получаются из одной посредством её сдвигов и растяжений по оси времени (так что они «идут друг за другом»). Локальность базовых функций вейвлет-преобразования порождает ряд кардинальных преимуществ. В первую очередь свертка исходного изображения с конкретной базовой функцией не требует всего изображения – достаточно нескольких пикселей (по ширине окна базовой функции). Соответственно нет необходимости разбивать изображение на блоки – можно выполнять преобразование над всем изображением сразу, поскольку вычислительная сложность растет линейно от полного числа пикселей в изображении.

1.12 Алгоритмы сжатия последовательностей изображений: MPEG (1,2,4).

MPEG (Motion Picture Experts Group – экспертная группа по вопросам движущегося изображения) – группа специалистов, собирающаяся для выработки стандартов сжатия цифрового видео и аудио.

MPEG стандартизовала следующие стандарты сжатия и вспомогательные стандарты:

- MPEG-1: Исходный стандарт видео и аудио компрессии. Позднее использовался, как стандарт для Video CD, и включает в себя Layer 2 формат аудио сжатия.
- MPEG-2: Транспортные, видео и аудио стандарты для широковещательного телевидения. Используется в цифровом телевидении ATSC, DVB и ISDB, цифровых спутниковых ТВ службах, таких как Dish Network, цифровом кабельном телевидении, и (с небольшими изменениями) в DVD.
- MPEG-3: Изначально разрабатывался для HDTV, но от него отказались, когда обнаружилось, что MPEG-2 (с расширениями) вполне достаточно для HDTV. (Не

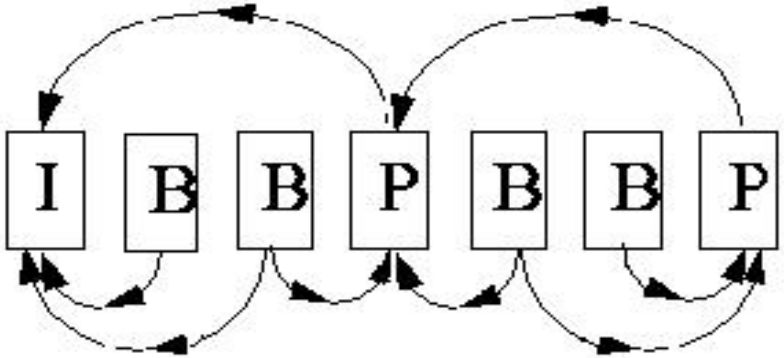
стоит его путать с MP3, который на самом деле MPEG-1 Layer 3.)

- MPEG-4: Расширяет MPEG-1 для поддержки видео/аудио «объектов», 3D контента, сжатия с низким битрейтом и DRM. В него включено несколько новых высоко эффективных видео стандартов (альтернатив MPEG-2), особо отметим: MPEG-4 Part 10 (или AVC, или H.264). MPEG-4 Part 10 может быть использован в HD DVD и Blu-Ray дисках.

MPEG-1 был разработан и оптимизирован для работы с разрешением 352 ppl (point per line -- точек на линии) * 240 (line per frame -- линий в кадре) * 30 fps (frame per second -- кадров в секунду), что соответствует скорости передачи CD звука высокого качества. Используется цветовая схема - YCbCr (где Y - яркостная плоскость, Cb и Cr - цветовые плоскости).

В зависимости от некоторых причин каждый frame (кадр) в MPEG может быть следующего вида:

- I (Intra) frame – кодируется аналогично алгоритму JPEG.
- P (Predicted) frame - при кодировании используется информация от предыдущих I или P кадров.
- B (Bidirectional) frame - при кодировании используется информация от одного или двух I или P кадров (один предшествующий данному и один следующий за ним)



Последовательность кадров может быть например такая: IBBPBBPBBPBBIBBPBBPB...

Последовательность декодирования: 0312645...

Нужно заметить, что прежде чем декодировать B кадр требуется декодировать два I или P кадра. Существуют разные стандарты на частоту, с которой должны следовать I кадры, приблизительно 1-2 в секунду, соответствующие стандарты есть и для P кадров (каждый 3 кадр должен быть P кадром).

Для применения алгоритмов кодировки происходит разбивка кадров на макроблоки по 16*16 пикселей исходного изображения.

Для большего сжатия в B и P кадрах используется алгоритм предсказания движения на выходе которого получается:

- Вектор смещения (вектор движения) блока который нужно предсказать относительно базового блока.
- Разница между блоками (которая затем и кодируется).

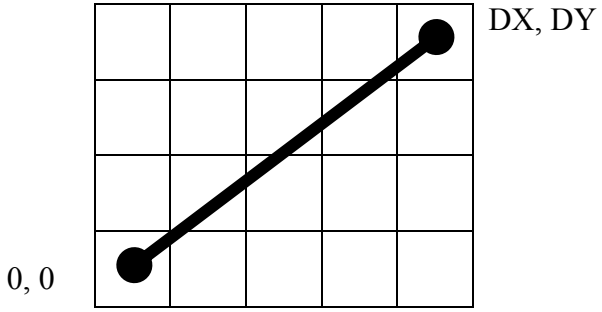
Так как не любой блок можно предсказать на основании информации о предыдущих, то в P и B кадрах могут находиться I блоки (блоки без предсказания движения).

1.13 Растрирование на плоскости. Алгоритм Брезенхема для растрирования отрезка и дуги. Кривые Безье. Векторизация кривых. Растрирование контуров.

Алгоритм Брезенхема

Алгоритм выбирает оптимальные растровые координаты для представления отрезка. В процессе работы одна из координат – либо x, либо y (в зависимости от углового коэффициента) – изменяется на единицу. Изменение другой координаты (либо на нуль, либо на единицу) зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. Такое расстояние мы назовем ошибкой. Алгоритм построен так, что требуется проверять лишь знак этой ошибки.

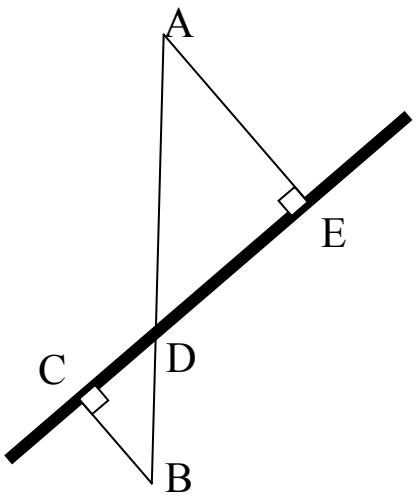
Приведем алгоритм Брезенхема для первого октанта, то есть для случая 0 <= DY <= DX.



Необходимо отрастрировать отрезок из точки {0,0} в точку {DX,DY}

Уравнение прямой $y = \frac{DY}{DX} \bullet x$

Рассмотрим прямую возле двух центров сетки растра:



Точки A и B – центры пикселей. Отрезки AE и BC соответствуют расстоянию от центра пикселя до прямой. Какое из расстояний меньше, такой пиксель и нужно выбрать для растрирования. Из подобия треугольников ADE и BDC следует, что вместо сравнения расстояний AE и BC можно сравнивать расстояния AD и BD. Величину BD можно назвать ошибкой аппроксимации.

Очевидно, что при работе в первом квадранте на каждом шаге алгоритма необходимо шагать либо вправо, либо вправо вверх. То есть нужно увеличивать значение x на единицу, а значение y оставлять старым или увеличивать на единицу.

В результате получаем следующий алгоритм:

```
// Инициализация переменных (начальный шаг
алгоритма):
x = 0,  y = 0,  err = 0

// Очередной шаг алгоритма:
x += 1;
err += DY/DX;
```

```
if( err > 0,5 ) {
   // нужно шагать вправо вверх
   y += 1;
   err -= 1;
} else {
   // нужно шагать вправо
}
```

Для перехода в целые числа умножим ошибку на 2*DX и установим начальное значение ошибке -0.5, чтобы сравнивать знак ошибки:

```
// Инициализация переменных (начальный шаг
алгоритма):
X = 0,  Y = 0,  err = -DX

// Очередной шаг алгоритма:
x += 1;
err += 2*DY;
if( err > 0 ) {
   // нужно шагать вправо вверх
   y += 1;
   err -= 2*DX;
} else {
   // нужно шагать вправо
}
```

1.14 Микширование в прямом и обратном порядке.

Традиционное микширование (смешивание) цветов: $C_{res} = C \bullet A + C_{back} \bullet (1 - A)$, где C_{back} - цвет фона, C - накладываемый сверху цвет, а A - его непрозрачность (вес). В этой формуле предполагается, что фон полностью непрозрачный. Аналогично можно обработать произвольное число слоев (граней), накладывая их один на другой, каждый

со своей прозрачностью. Обратите внимание, что при смене порядка микширования слоев результат может измениться.

А как быть, если у нас нет нижнего непрозрачного слоя?

Рассмотрим стопку из n слайдов, каждый из которых имеет цвет C_n и непрозрачность A_n. С учетом только верхнего слайда результирующий цвет стопки равен $C_{res} = C_1 \bullet A_1 + C_{back} \bullet (1 - A_1)$, где C_{back} - цвет фона. С учетом второго слайда $C_{res} = C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + C_{back} \bullet (1 - A_2) \bullet (1 - A_1)$.

Аналогично с учетом всех n слайдов результирующий цвет будет вычисляться по следующей формуле:

$$C_{res} = C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + \ldots + C_n \bullet A_n \bullet \prod_{k=1}^{n-1} (1 - A_k) + C_{back} \bullet \prod_{k=1}^{n} (1 - A_k) \quad (1)$$

Если заменить стопку слайдов одним цветом C_x с непрозрачностью A_x, то для него можно написать:

$$C_{res} = C_x \bullet A_x + C_{back} \bullet (1 - A_x) \quad (2)$$

Из сравнения (1) и (2) можно найти непрозрачность и цвет стопки слайдов:

$$A_x = 1 - \prod_{k=1}^{n} (1 - A_k) \quad (3)$$

$$C_x = \frac{C_1 \bullet A_1 + C_2 \bullet A_2 \bullet (1 - A_1) + \ldots + C_n \bullet A_n \bullet \prod_{k=1}^{n-1} (1 - A_k)}{1 - \prod_{k=1}^{n} (1 - A_k)} \quad (4)$$

Рассмотрим пару величин: цвет, умноженный на непрозрачность (premultiplied color) $P = C \bullet A$ и величину, обратную прозрачности, т.е. прозрачность $Tr = 1 - A$. Для этой пары величин формулы (3) и (4) приобретут следующий вид:

$$Tr_x = \prod_{k=1}^{n} Tr_k \quad (5)$$

$$P_x = P_1 + P_2 \bullet Tr_1 + \ldots + P_n \bullet \prod_{k=1}^{n-1} Tr_k \quad (6)$$

Видно, что пара формул (5) и (6) может быть легко реализована итеративно при последовательном микшировании произвольного числа слайдов, начиная с самого верхнего.

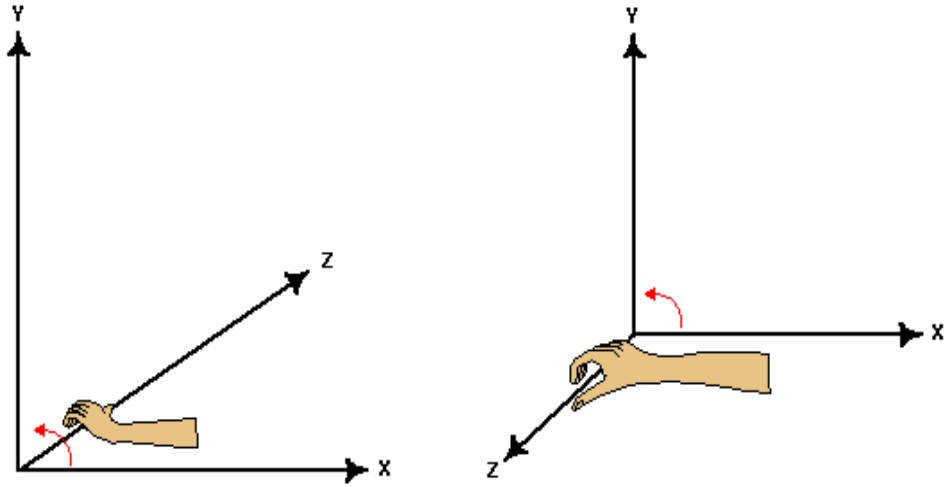
Обратите внимание, что при этом получается результирующая прозрачность стопки слайдов и ее цвет, умноженный на вес (непрозрачность). Результирующие значения цвета и непрозрачности (традиционно используемые в компьютерной графике) могут быть получены из прозрачности $A = 1 - Tr$ и premultiplied color $C = P/A$

2 Трехмерная графика

2.1 Геометрия

Вектор в пространстве. Системы координат. Проекция. Обобщённые координаты.

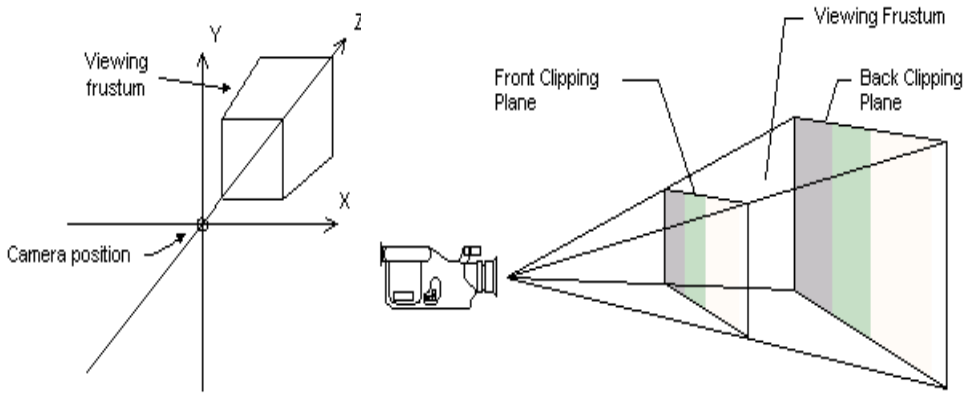
Поскольку трехмерная графика имеет дело с изображением трехмерных сцен и объектов, то и при расчетах этих изображений используется соответствующий мат-аппарат. Одним из основных терминов является трёхмерный вектор, представляемый тремя координатами – x, y, z. Говоря о векторе в пространстве, разумеется, необходимо определить и само пространство. Введём понятие системы координат. Система координат определяется одной точкой – началом системы координат и тремя ортогональными, нормированными векторами - ортами. Мы будем использовать такие системы координат, в которых при взгляде по орте z – орта x направлена вправо, а орта y вверх. Такие соглашения приняты в пакетах DirectX и OpenGL



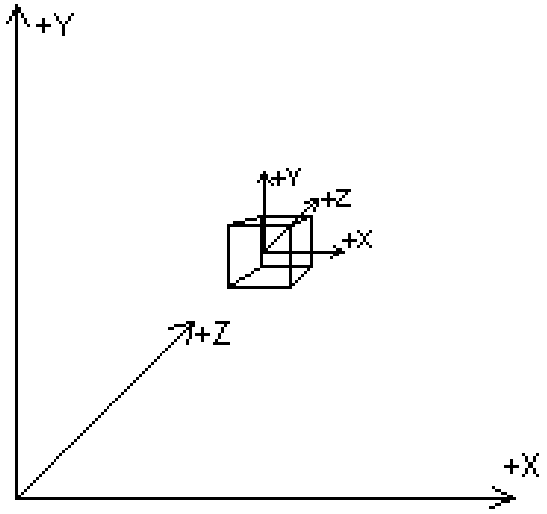
Каждая система координат расположена тем или иным образом относительно другой системы координат. Для определённости используют некую «мировую» систему координат, которая является как - бы абсолютной. Все остальные системы координат определяются относительно её, либо относительно тех которые определяются относительно её.

При построении изображений трёхмерного мира очень часто нужно уметь переводить вектора из одной системы координат в другую. Для осуществления этой операции используют векторно-матричный аппарат. Т.е. для каждой системы координат существует такая матрица, которая переводит все вектора в родительскую систему координат (система координат, в которой определено начало и орты). Соглашение при перемножении вектора на матрицу – строка вектор слева умножается на столбцы матрицы стоящей справа.

Изображение трёхмерной сцены, по сути, является двумерным, поскольку представляется на экране монитора. Для того чтобы понять где трёхмерная точка будет отображена на двумерной плоскости (плоскости экрана) используется перспективная проекция. Трехмерные пакеты как правило позволяют строить и ортографические проекции, но сейчас мы будем обсуждать именно перспективную.



Вернёмся к вопросу о векторно-матричном аппарате. Если использовать трёхмерные вектора, то матрицы на которые они умножаются, должны быть 3x3. При помощи таких матриц можно задавать повороты (масштабирование и пр) , но нельзя задать сдвиг всех векторов на заданный вектор. Для того чтобы задать одну систему координат относительно другой, сдвиги нужны обязательно.



Кроме этого нужно уметь осуществлять перспективное преобразование. Для этого было предложено использовать обобщенные векторно-матричные преобразования. Обобщенный вектор

(x, y, z, w) – считается, что вектор всегда можно нормировать так чтобы w была равна 1.

Матрица в этом случае должна быть четыре на четыре:

$$(x', y', z', w') = (x, y, z, w) \times \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$,т.е.

$$x' = x \times M_{11} + y \times M_{21} + z \times M_{31} + w \times M_{41}$$
$$y' = x \times M_{12} + y \times M_{22} + z \times M_{32} + w \times M_{42}$$
$$z' = x \times M_{13} + y \times M_{23} + z \times M_{33} + w \times M_{43}$$
$$w' = x \times M_{14} + y \times M_{24} + z \times M_{34} + w \times M_{44}$$

Переноса:

$$(x', y', z', 1) = (x, y, z, 1) \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

Вращение:

Вокруг оси X:

$$(x', y', z', 1) = (x, y, z, 1) \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вокруг оси Y:

$$(x', y', z', 1) = (x, y, z, 1) \times \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Вокруг оси Z:

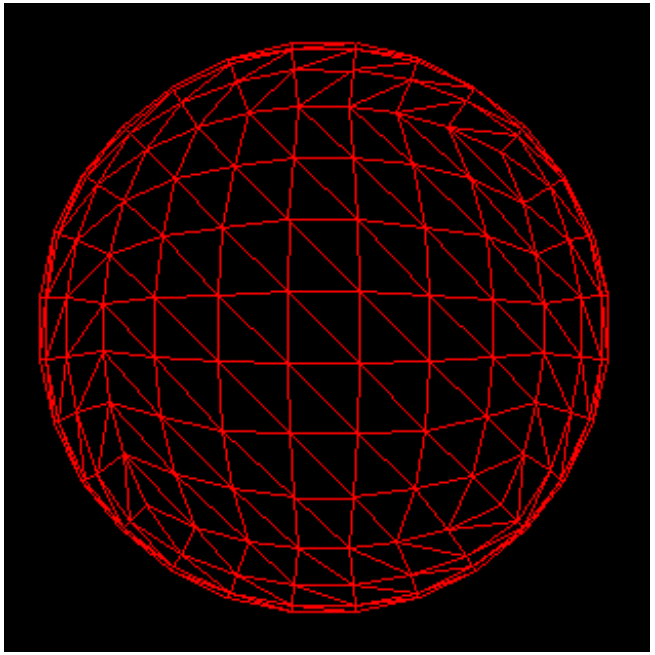
$$(x', y', z', 1) = (x, y, z, 1) \times \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Масштабирование:

$$(x', y', z', 1) = (x, y, z, 1) \times \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Грань. Проекция грани. Клипирование.

Все трехмерные поверхности, отображаемые на экране, имеют описание в виде набора граней (мы будем обсуждать только треугольные грани).



Таким образом, чтобы получить изображение нужно спроецировать каждую грань на проективную плоскость, после чего задача становиться двумерной. Для того чтобы спроецировать грань нужно спроецировать его ребра, а для этого нужно спроецировать его вершины.

В ряде ситуаций возникают проблемы, например, что делать, если проекцию вершины на плоскость построить нельзя.

Кроме этого на экране монитора будет отображаться только часть поверхности, ограниченная краями монитора. Соответственно часть граней ненужно проецировать вовсе, а часть нужно обрезать линиями, соответствующими краям монитора. Часть объектов может находится далеко, поэтому их тоже не нужно отображать. Для разрешения этих вопросов предложена операция клипирования т.е. обрезания не нужных для отображения частей. Обрезание происходит по шести

плоскостям – передней (соответствующей плоскости проектирования): задней (также параллельной ортам xy) и четырём плоскостям проходящим через начало системы координат наблюдателя и через стороны прямоугольника соответствующего экрану монитора. Операция клипирования предшествует операции проецирования, тем самым решаются выше описанные проблемы.

Односторонние поверхности. Отбраковка. Заливка. Понятие первичного и вторичного буфера

В большинстве случаев поверхность имеет одну сторону и нет необходимости отображать поверхность если она ориентировано обратной стороной. Для отбраковки треугольников повёрнутых к наблюдателю тыльной частью существует как правило настройка механизма отбраковки. Отбраковка осуществляется сообразно тому в каком порядке происходит обход грани со стороны наблюдателя – по часовой стрелке или против.

После клипирования и проецирования мы получим на экране двумерный многоугольник (дополнительные углы могут образоваться в результате клипирования). Этот многоугольник можно залить цветом соответствующим данной поверхности. Поступив подобным образом со всеми треугольниками сцены, мы получим изображение сцены на экране.

Построение сцены – процесс не бесконечно быстрый. Поэтому если грани отображать непосредственно на экран монитора, то мы увидим этот процесс. Выглядеть это будет мигающе и крайне неестественно. Для этого используют два буфера с изображением сцены. Первый содержит уже полностью построенное изображение и именно его предьявляют зрителю. А построение сцены осуществляют во второй буфер. В этот момент не видимый. После того как во втором буфере сцена снова готова его меняют с первым (например во время обратного хода луча монитора).

Способы удаления невидимых частей поверхностей (Художник и Z).

В реальной жизни часть объектов закрывает другие и в результате некоторые объекты видны частично. При отображении отображении трехмерной сцены на экран монитора также нужно учесть этот эффект. Алгоритмы, обеспечивающие корректное изображение в этом аспекте называются алгоритмами удаления невидимых поверхностей. Мы рассмотрим два из них.

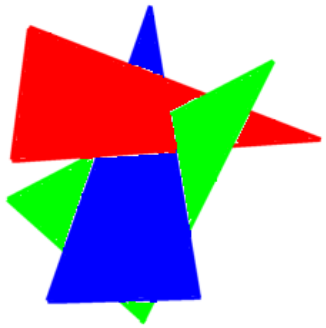
Первый называется алгоритмом «художника». Его смысл состоит в том что поверхности рисуются в таком порядке, что последним рисуется та поверхность, которая ближе. Этот алгоритм имеет несколько недостатков. О них мы поговорим чуть позже.

Второй использует понятия Z буфера. Помимо первичного и вторичного буферов, которые хранят изображение сцены, заводится дополнительный буфер, который хранит расстояние до точки в поверхности. При заливке треугольника для каждого пиксела расчитывается растояние до него и сравнивается с тем значением, которое хранится в буфере. Соответственно всегда можно понять данный пиксел ближе или дальше нарисованного ранее. Таким образом принимается решение о том нужно ли реально рисовать данный пиксел.

Сортировка художника.

Для алгоритма «художника» можно выделить два разных подхода.

В первом грани сортируются на каждом кадре. Этот вариант имеет следующие недостатки: сортировка всех граней может занимать довольно много времени. Возможны ситуации, когда отсортировать



невозможно. Т.е. первая грань закрывает вторую, вторая третью, третья первую. В этом случае необходимо произвести разрез одной из граней.

Во втором случае сортировка производится заранее. Часть граней можно отсортировать статически. Например, есть грани, в полуплоскости нормали которых нет других граней – эти грани не зависимо от положения наблюдателя нужно рисовать последними. Наоборот могут быть грани которые находятся в противоположном пространстве остальных граней – их нужно рисовать первыми, так как они не закрывают ни кого.

Есть и динамические методы сортировки – например метод «разделяющей плоскости». Предположим рядом стоят два больших объекта. Предположим, что между ними можно провести плоскость. Тогда если наблюдатель находится по туже сторону от плоскости что и один из объектов, то его можно рисовать позже, так как его закрывать другой объект не может.

В случае когда расстояния между несколькими объектами больше их линейных размеров, то их можно рисовать сообразно расстоянию до их центров – это тоже даст правильный результат. Этот вариант имеет те недостатки, что опять же есть ситуации, когда нет возможности отсортировать и приходится резать. Кроме того, в случае когда анимация меняет расположение объектов относительно друг друга, результат предварительной сортировки теряется.

2.2 Цвет

Цвет. Нормаль в точке. Источники освещения. Модели освещённости.

• Цвет задаётся тройкой или четвёркой чисел R, G, B, [A].

• Пусть C1 и C2 – два цвета. Определим операции:

• Сложение: C1 + C2 == (r1 + r2, g1 + g2, b1 + b2, a1 + a2).

• Умножение: C1 * C2 == (r1 *r2, g1 * g2, b1 * b2, a1 * a2).

4-й компонент цвета (альфа) используется для:

1. Альфа -теста: рисовать пиксели, альфа которых удовлетворяет некоторому условию (см. IDirect3DDevice9::SetRenderState).
2. Альфа – смешивания: цвет пикселя смешивается в некоторой пропорции с цветом, находящемся в back буфере (см. IDirect3DDevice9::SetRenderState).

Для лучшего понимания мы рассмотрим несколько упращенных формул расчета освещенности точки поверхности.

I = N*L>0 ? N*L:0 – характерна для космоса, так как отвернутая сторона абсолютно черна

I = N*L>0 ? N*L + E : E – характерна для атмосферы,в этой формуле не учитывается нормировка на 1

Можно придумывать и другие. Сложность модели зависит от вычислительных возможностей. В DirectX встроенна более сложная формула, но есть возможность разрабатывать свои с учетом тех визуальных характеристик которые требует конкретное приложение. Обычно учитывают и цвет источника. Например:

P = Cm*I*Cl + B

Глобальная модель освещения

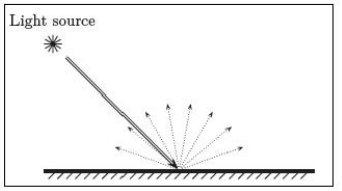
$$L(\mathbf{x},\bar{\omega}_o) = L_e(\mathbf{x},\bar{\omega}_o) + \int_S f_r(\mathbf{x},\bar{\omega}_i \to \bar{\omega}_o) L(\mathbf{x}',\bar{\omega}_i) G(\mathbf{x},\mathbf{x}') V(\mathbf{x},\mathbf{x}') d\omega_i$$

$L(\mathbf{x},\bar{\omega}_o)$	Интенсивность света, отражённого в точке x в направлении w_0.
$L_e(\mathbf{x},\bar{\omega}_o)$	Интенсивность света, излучаемого объектом из точки x.
$L(\mathbf{x}',\bar{\omega}_i)$	Интенсивность света, приходящего от точки x'с направления w_i.
$G(\mathbf{x},\mathbf{x}')$	Функция ослабления интенсивности.
$f_r(\mathbf{x},\bar{\omega}_i \to \bar{\omega}_o)$	BRDF.
$V(\mathbf{x},\mathbf{x}')$	Функция видимости точки x' из точки x (1 или 0).

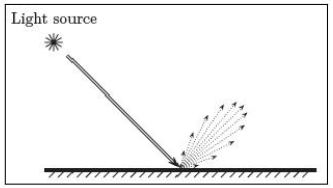
Локальная модель освещения

- Локальная, потому что нет учёта расположения на сцене других объектов.
- Illumination = ambient + diffuse + specular + emissive.

- Ambient – свет, приходящий в точку со всех направлений.
- Emissive – свет, излучаемый объектом.
- Diffuse – свет, равномерно отражаемый объектом во всех направлениях.
- Specular – «зеркально» отражаемый поверхностью свет.



Diffusely reflected light



Secularly reflected light

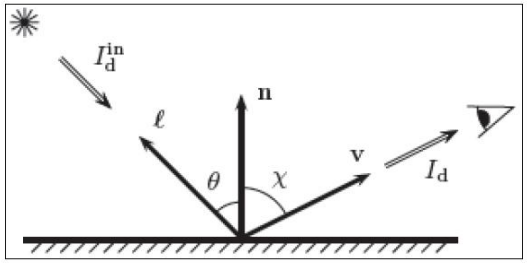
Ambient = Ia*Pa .

• Ia– интенсивность приходящего рассеяного освещения.

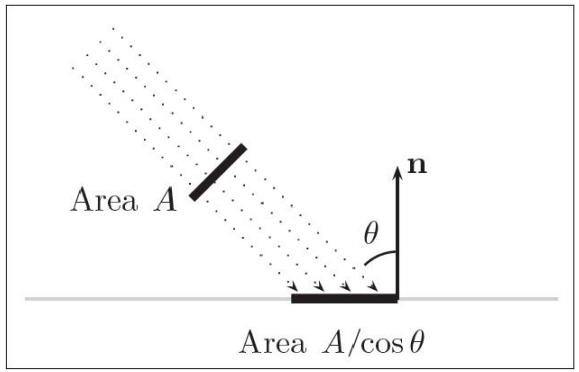
• Pa – коэффициент отражения рассеяного освещения.

• Emissive = I

• ILD= Id* Pd * cos(teta) = Id * Pd * (n, l).
• ||n|| = ||l|| = 1.
• Id – интенсивность приходящего диффузного света.
• Pd – коэффициент диффузного отражения.



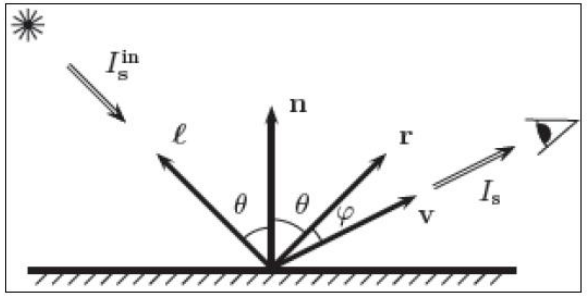
Diffuse = SIGMA(ILD)



Блик

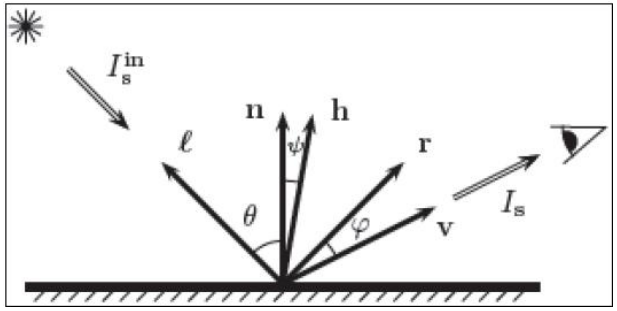
Блик расчитывается обычно в зависимости от косинуса в степини n между отраженным лучом от источника и направлением на глаз от точки поверхности для которой ведется расчет. При этом n определяет остроту блика.

Specular по Фонгу
• ILS= Is * Ps * (cos(teta))^f = Is* Ps * ((r, v))^f.
• ||r|| = ||v|| = 1.
• r = 2(l, n) * n – l.



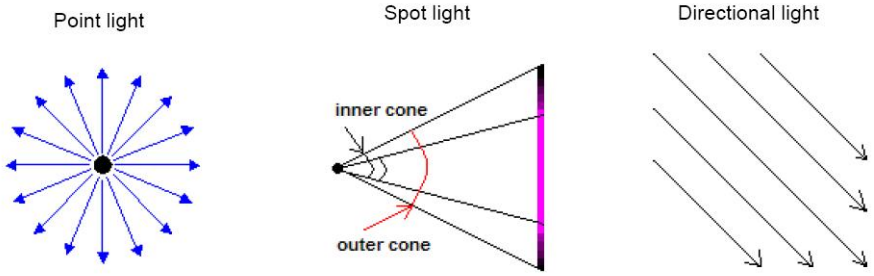
Specular по Блинну

• ILS= Is * Ps * (cos(teta))^f = Is * Ps * ((h, n))^f.
• ||h|| = ||n|| = 1.
• h = (l + v) / ||l + v||.



Specular = SIGMA (ILS)

Illumination = ambient + diffuse + specular + emissive =
= S(ILD) + S(ILS) + Ia * Pa + Ie=
[SIGMAi(Idi * Pd * (n, li))] + [SIGMAi (Isi * Ps * ((ri, v))^f)] + Ia * Pa + Ie



С увеличением расстояния от точки до источника света интенсивность ослабевает.
attenuation = 1 / (a + b * d + c * d^2).

d – расстояние.
a, b, c – одновременно не равные 0 коэффициенты;
• Face normals:
• Vertex normals:
Пусть вершина принадлежит n треугольникам, имеющим нормали Ni i=[0..n-1].
Гладкие нормали: «наивный» подход
N = (1/n) * SUMM(Ni)
Гладкие нормали: «продвинутый» подход
N = SUMM (di * Ni), где 1 = SUMM(di)
Например di= si/ S, где S – сумма площадей треугольников, которым принадлежит вершина

Преобразование нормали из одной систем координат в другую

При переводе нормали из одной системы координат в другую достаточно считать что четвертая компонента равна 0.

Гуро интерполяция

Это упрощение позволяет рассчитывать освещенность только в вершинах треугольника. При заливке всей площади треугольника используется интерполяция вдоль экрана.

Фонг

В этом методе расчет ведется для каждого пиксела на экране заливаемого треугольника.

Туман

Наличие тумана илит дымки существенно улучшает реалистичность картинки, а расчет сравнительно не сложен. Нужно просто взвесить цвет тумана и цвет освещенной поверхности

P = (1-f)*Cp + f*Cf

F – вычисляется из растояния от глаза до поверхности. Это расстояние и так известно при Z методе удаления невидимых поверхностей.

Полупрозрачность. (Полупрозрачность и Z) Альфа-Блендинг

Еще один эффект очень полезен при построении реалистичного изображения – полупрозрачные поверхности. Для этого при при отрисовке полупрозрачной поверхности нужно использовать ранее нарисованный (он должен быть дальше по z)

P = (1-a)*S + a*Cp

Для реализации полупрозрачности и других эффектов в системах компьютерной графики обычно используют технологию альфаблендинга которая основана на задании функции:

```
Color = TexelColor x SourceBlend +
CurrentPixelColor x DestBlend
```

Где:
Color – результирующее значение
TexelColor - расчитанное значение
CurrentPixelColor – значение, хранящееся во вторичном буфере
SourceBlend DestBlend - значение которое может например быть:

D3DBLEND_ZERO
Blend factor is (0, 0, 0, 0).
D3DBLEND_ONE
Blend factor is (1, 1, 1, 1).
D3DBLEND_SRCCOLOR
Blend factor is (R_s, G_s, B_s, A_s).
D3DBLEND_INVSRCCOLOR
Blend factor is (1 - R_s, 1 - G_s, 1 - B_s, 1 - A_s).

Так для организации полупрозрачности применяют D3DBLEND_SRCALPHA
и D3DBLEND_INVSRCALPHA соответственно а для отрисовки луны на небе D3DBLEND_ONE, поскольку свет от луны просто добавляется к свету который идет от неба

2.3 Текстура

Текстура, текстурные координаты.

Как правило, все поверхности, которые можно моделировать в трехмерной графике имеют некоторый рисунок. Например изделие из древесины, поверхность луга, кожа человека.

Текстура представляется как правило прямоугольным изображением. Для того чтобы «намазать» изображение картинки на поверхность желаемым образом, в каждой вершине проставляют текстурные координаты – пара чисел, которые задают точку на изображении которой эта вершина соответствует.

Фильтрация. Методы расчёта текстурных координа

Поскольку при растеризации может возникнуть ситуации когда одному элементу изображения текстуры (текселу) соответствует большая область на экране, то чтобы избежать эффекта «мазаики» конкретное значение берут соответственно из четырех соседних взвешивая их сообразно расстоянию. В DirectX каждого направления по текстуре можно указать брать ли тексел просто выборкой или интерполировть тем или иным способом.

Иногда вызывают наоборот ситуации когда большая текстура отображается в небольшое количество пикселов экрана. В этом случае для текстур вычисляют набор текстурных уровней каждый из которых в два раза меньше по разрешению, и используют тот уровень (или интерполяцию уровней) который адекватен масштабу мэпирования.

Мульти - текстурирование Bump Тени Отражения

На моделируемую поверхность можно «натягивать» не одно изображение, а несколько с применением тех или иных операций над их текселами. Например, на кирпичную стену можно наложить сложное распределение освещенности. Это

может дать реалистический эффект и сэкономить текстурную память.

Зачастую поверхности бывают не ровными, причем неровности небольшие и с точки зрения расположения участка поверхности можно считать поверхность плоской, а вот степень освещенности из-за небольших неровностях может существенно отличаться. Для решения таких задач используют текстурирование нормалей, которые используют для освещенности.

В моделях освещенности которые основаны на заливке спроецированного на экран примитива (в отличии от систем с трассиоркой лучей). Тени организовать существенно сложнее. Один из методов – построить изображение сцены со стороны источника и использовать z буффер результата как еще одну текстуру на объекты с перемножением текселов. Затененные части будут темными.

Еще один эффект где важно уметь на одной поверхности иметь несколько текстур, это отражение. Моделей построения отражений несколько, но все они сводятся к тому что строится еще одно изображение сцены а выборка происходит сообразно нормали отрожающей поверхности.

2.4 Анимация и эффекты

Анимация предметов и камеры

Для того чтобы от кадра к кадру перемещать предмет достаточно менять матрицу объекта. Для перемещения камеры – матрицу камеры. Несложно также менять расположение части объекта (вращение башни танка), для этого можно организовать иерархию матриц и в качетве объектной матрици для башни выставлять перемноженные матрици танка и башни. Гораздо более сложной задачей является анимация живых существ.

Веса вершин, Skeleton

Одним из популярных способов анимации живых существ является привязка вершин с теми или иными весовыми коэффициентами к сразу нескольким матрицам, которые в свою очередь могут соотвествовать тем или иным костям «скелета» живого существа

Lipsync

Еще одной из характерных задач компьютерной графики это синхронизация движения губ и воспроизводимой речи. Решение делиться на два этапа:

1 Распознавание характерных фонем с привезкой к тайм лайну. Это распознование звука возможно с использованием текста. Так как например в играх тексты реплик известны. Эта задача решается один раз. Не в процессе использования приложения. Результатом является матрици для всех лицевых «костей». Кости в данном случае вируальные и соответствуют мышцам: левая бровь, правый угол рта.
2 В процессе выполнения приложения липсинк становиться обычной скелетонной анимацией

Эффекты. Системы частиц. Огонь, Дымы. Пары.

При симуляции реального мира, много важных явлений не описываются устойчивой поверхностью или даже поверхностью вообще. Например: огонь, дымы, пары, брызги. Еще на заре компьюерной графики был разработан метод под названием «система частиц». Его суть заключается в том, что отображается большое колтичество мелких обьектов с характерным поведением. Т.е много небольших граней двигаются по схожим, но разным траекториям и при этом меняют свой цвет и прозрачность.

С повышением производительности современных видеокарт варианты алгоритма системы частиц могут быть применены и для задач реального времени. Задавая характерное поведение частиц и их свойств можно получать разные эффекты – те же дымы, брызги, пар, снег, дождь.

Трава, деревья.

Растительный мир также важен для построения реалистичного изображения.

Ландшафт, Водная Поверхность.

Реализация ландшафта имеет свои сложности. Так если мы стоим на земле, то можем видеть до нескольких километров в даль или даже дальше, если ландшафт имеет дальные особенности (далекие горы). Детальность ландшафта вокруг нас должна быть давольно высокой. Если весь ландшафт отображать с одинаковой подробностью то количество примитивов будет огромным даже для современных карт. Существует большое количество алгоритмов которые позволяют отображать ближние участки ландшафта подробнее дальних. Основные проблемы связанные с этим вызваны возможностью наблюдателя перемещаться, тем самы заставляе менять триангуляцию поверхности на ходу не допуская разрывов, скачков и пр.

Вода чем то схожий объект с ландшафтом. У нее есть то что делает задачу проще – например волны сравнительно одной высоты и в дали ее легче сводить к плоскости. Так же есть и очевидная сложность – ее динамика. Реализация воды разбивается как правило на два этапа:

1 Расчет карты высот или ее аналогов (например на основе спектра и обратного преобразования Фурье)
2 Триангуляция. Неплохие результаты дает project grid но он давольно старый и есть более сложные алгоритмы

3 Самостоятельный практикум

Для решения предлагаются следующие задачи:

1. Статический уровень с bump/stencil shadows а-ля Doom3. FPS-like камера с collision.
2. Патиклы. Статическое окружение, в которое эмитируются патиклы с различными параметрами. Столкновения с геометрией, spawn on collision. Огонь, дым, взрывы. Пост эффекты.
3. Софтверный рендер. Текстурирование, перспективная коррекция, вертексное освещение, смешивание текстур, альфа-блендинг.
4. Визуализация объемных текстур. Возможность динамики (примеры - game of life, решение разностных диффуров типа Навье-Стокса).
5. Прототип трехмерного интерфейса. Форматы описания интерфейса, события, трехмерные эффекты (возможности - патиклы, текстурирование, тени, пост эффекты).
6. Биллипард. От шаров тень, шары имеют гладкий силуэт. Регулируется сила и направление удара. Биллиардный стол находится в некотором environment.
7. Модель солнечной системы. Солнце должно иметь корону. Звезды на “бесконечном” расстоянии. У планет должна быть атмосфера. По космосу летает космический кораблик управляемый игроком. Анимируемый выхлоп корабля.
8. Террэйн. Наблюдатель должен “летать” над террэйном. Террэйн должен быть текстурирован несколькими текстурами. Террэйн должен обладать уровнями детализации. Должно быть небо. На террэйне присутствуют озера в которых отражается небо.
9. Лабиринт. Генерация “случайных” лабиринтов. Наблюдатель сталкивается со стенами. Пол отражает стены и потолок.

10. Море. Небо. Море анимируется, отражает небо. По морю плавают кораблики. Кораблик должен оставлять пенный след. Кораблики могут обстреливать друг друга из пушек. Взрывы на воде. Взрыв когда ядро попадает в кораблик. Тонущие кораблики. Одним корабликом управляет игрок а другим ai.
11. Генератор деревьев. Дерево и листья шевелятся на ветру. Дерево растет на лугу на котором растет трава, которая тоже анимируется от ветра.
12. Игра “Cannons”. На террэйне из п. 3 расставляются пушки, которые по очереди обстреливают друг друга. Анимация взрывов. На террэйне остаются следы от взрывов.
13. Игра “догонялки”. По лабиринту из п. 4 бегают монстры и пытаются догнать игрока, игрок убегает. 2.5 View на лабиринт.
14. Гонки на машинках по террэйну из п. 3. ездят машинки. Побеждает та машинка, которая придет первой из точки 1 в точку 2. На террэйне растут кустики. Машинки оставляет следы на террэйне. Одной из машинок управляет игрок остальными ai.

СПИСОК ЛИТЕРАТУРЫ

1. Д. Хьюбел «Глаз, мозг, зрение», Перевод с английского канд. биол. наук О. В. Левашова, канд. биол. наук Г. А. Шараева, под редакцией чл.-корр. АН СССР А. Л. Вызова, Издательство Москва «Мир», 1990.
2. "Психофизиология цветового зрения" Ч.А.Измайлов, Е.Н.Соколов, А.М.Черноризов
3. "Основы цветного телевидения" В.Ф.Самойлов, Б.П.Хромой
4. Певзнер Б. М. Качество цветных телевизионных изображений. М.: Радио и связь, 1988. 224 с.
5. "Encyclopedia of Graphics File Formats" James D.Murray & William vanRyper
6. И.М. Арсенин, Б.Б. Морозов, В.Г. Садов, И.Г. Таранцев «Метод построения изображений в системах генерации титров», Автометрия №2, 1997.
7. И.Г. Таранцев «Оптимизированный алгоритм сжатия изображений “Феникс”», Автометрия №2, 1997.
8. Орлов А. Компьютерная анимация: возвращение на землю. Мир ПК № 9, 1993, стр. 95-104.
9. Роджерс Д.Ф., Адамс Дж. “Математические основы машинной графики” М. Машиностроение, 1980
10. Эйнджел Э. “Интерактивная компьютерная графика. Вводный курс на базе OpenGL”, 2 изд.: М: Издательский дом “Вильямс”, 2001
11. Marc Stamminger George Drettakis “Perspective Shadow Maps” Proceedings of ACM SIGGRAPH 2002 July 2002
12. Tom Malzbender, Dan Gelb, Hans Wolters “Polynomial Texture Mapping (PTM)” Hewlett-Packard Laboratories http://www.hpl.hp.com/ptm
13. Cass Everitt and Mark J. Kilgard “Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering” NVIDIA Corporation, Copyright 2002 http://developer.nvidia.com

14. D. Sim Dietrich Jr. “Per-Pixel Lighting”
15. NVIDIA Corporation, Copyright 2002 http://developer.nvidia.com