Направление подготовки: «Информатика и вычислительная техника»
Профиль образовательной программы: «Системотехника и автоматизация проектирования в строительстве»

Наименование дисциплины: Компьютерная графика

КОМПЬЮТЕРНАЯ ГРАФИКА

Кафедра «Информационных систем, технологий и автоматизации в строительстве» (ИСТАС)
Постнов Константин Владимирович,
Рыбакова Ангелина Олеговна
УЛК, 310 каб.

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ
Москва 2020

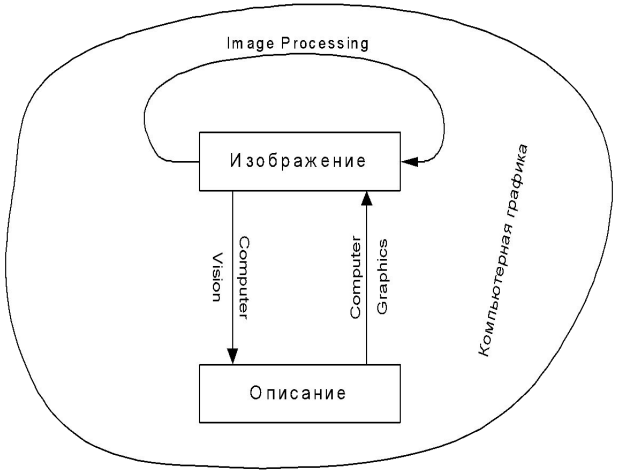
Глава 1. ВВЕДЕНИЕ, БАЗОВЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Важнейшая функция компьютера - обработка информации. Особо можно выделить обработку информации, связанную с изображениями. Она разделяется на три основные направления: компьютерная графика (КГ), обработка и распознавание изображений.

Задача компьютерной графики (Computer Graphics) - визуализация, то есть создание изображения. Визуализация выполняется, исходя из описания (модели) того, что нужно отображать. Существует много методов и алгоритмов визуализации, которые различаются между собою в зависимости от того что и как отображать.

Обработка изображений (Computer Vision) - это преобразования изображений. Входными данными является изображение, и результат обработки - тоже изображение. Примерами обработки изображений могут служить: повышение контраста, чёткости, коррекция цветов, редукция цветов, сглаживание, уменьшение шумов и так далее. В качестве материала для обработки могут использоваться космические снимки, сканированные изображения, радиолокационные, инфракрасные изображения и т. п.

Для **распознавания изображений (Image Processing)** основная задача - получение описания объектов, представленных изображением. Методы и алгоритмы распознавания разрабатывались прежде всего для обеспечения зрения роботов и для систем специального назначения.



Давая обобщенное определение можно сказать, что ***компьютерная графика*** – это процесс создания, хранения и обработки изображений и моделей объектов с помощью ЭВМ.

Основные области применения компьютерной графики:

- **Конструкторская графика (автоматизация проектно-конструкторских работ)**
- **Иллюстративная графика**
- **Деловая графика**
- **Научная графика**
- **Художественная и рекламная графика**
- **Компьютерная анимация и игры**
- **Мультимедиа**
- **Геоинформационные системы**

Классификация применений КГ:

1. По интерактивности – интерактивная и пассивная компьютерная графика.
2. По типу объекта и типу выводимого изображения: линейные двумерные рисованные объекты, линейные трехмерные объекты (каркасные), линейные трехмерные объекты с удалением скрытых линий, двумерные черно-белые тоновые изображения, двумерные цветные изображения, трехмерные изображения с удалением скрытых линий и граней, и т.д.
3. По уровню возможностей и интерактивности при управлении изображением: автономное вычерчивание из готовой графической БД, создание нового объекта с помощью имеющихся графических примитивов, интерактивное вычерчивание объекта с заданием динамика, текстур и освещенности.
4. По роли изображения – цель и средство достижения цели
5. По логическим и временным соотношениям между объектами: 1 независимый объект, несколько независимых объектов, структура взаимосвязанных объектов, матрица структур объектов, и т.д.

Базовые понятия КГ

- Графический примитив
- Мировая система координат
- Пользовательская система координат
- Нормированная система координат
- Графический блок
- Сегмент
- Объектное окно
- Отсечение
- Свертывание
- Поле вывода
- Слой

Системные принципы создания графических пакетов

1. Управление видимостью объекта
2. Обратное (инверсное) отображение
3. Мягкое стирание
4. Двойная буферизация
5. Зуммирование
6. Панорамирование
7. Буксировка
8. Масштабирование
9. Поддержка сеансового протокола
10. Наличие цветовых моделей

Глава 2. ОСНОВНЫЕ СПОСОБЫ ОРГАНИЗАЦИИ ГРАФИЧЕСКИХ ДАННЫХ

2.1. Векторная графика

Изображение, созданное **в векторных программах**, основывается на математических формулах.

Средства создания векторных изображений

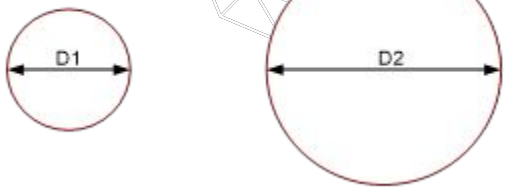
Векторные изображения могут быть созданы несколькими видами программ.

1. Программами векторной графики.

2. Программами САПР, типичным представителем которых является программа AutoCAD. Ее векторный формат - DXF (Dynamic Exchange Format) понимается многими современными программами.

3. *Специализированными программами конвертирования растровых изображений в векторные.* (CorelTrace 9 и Adobe Streamline).

Основа любой векторной графики – **примитив.**



Структура векторной иллюстрации

Структуру любой векторной иллюстрации можно представить в виде иерархического дерева. В такой схеме сама иллюстрация занимает верхний уровень, а ее составные части занимают более низкие уровни иерархии. Иллюстрация объединяет в своем составе *объекты + узлы + линии + заливки*

1. Первый уровень иерархии - ***объекты,*** представляющие собой разнообразные векторные формы.
2. *Объекты* иллюстрации состоят из одного или нескольких ***контуров.*** *Контуры могут быть замкнутыми* и *открытыми*. Обычно все объекты в иллюстрации сгруппированы, поэтому для получения доступа к редактированию отдельных объектов иллюстрации их нужно сначала разгруппировать. Вообще к*онтуром* называется любая геометрическая фигура, созданная с помощью рисующих инструментов векторной программы и представляющая собой очертания того или иного графического объекта.

Замкнутый контур - это замкнутая кривая, у которой начальная и конечная точки совпадают. Примером замкнутого контура является окружность. В некоторых редакторах замкнутый контур называют *фигурой.*

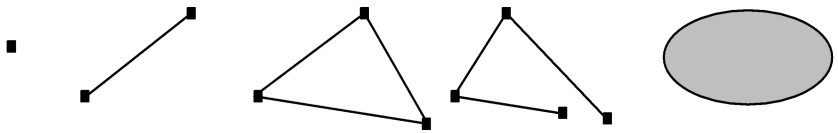
Открытый контур имеет четко обозначенные концевые точки. Синусоидальная линия, например, является открытым контуром.

3. Следующий уровень иерархии составляют ***сегменты,*** которые выполняют функции кирпичиков, используемых для построения контуров (каждый контур может состоять из одного или нескольких *сегментов).* Начало и конец каждого сегмента называют *узлами,* или *опорными точками,* поскольку они фиксируют положение сегмента, «привязывая» его к определенной позиции в контуре. Перемещение узловых точек приводит к модификации сегментов контура и к изменению его формы. Наряду с узлами в состав сегмента входят также соединяющие узлы линии (прямые или кривые).

4. На следующем уровне иерархии расположены ***отрезки линий или сплайнов,*** соединяющих между собой соседние узлы. Линии выполняют функции основных элементов векторного изображения. Простейшая незамкнутая линия имеет две вершины, называемые *узлами* (или *концевыми точками).* В двухмерной графике узел (точка) задается двумя числами (x, y).

5. ***Узлы, вершины, точки.***

В широком смысле любой из перечисленных здесь элементов иллюстрации, начиная от самой иллюстрации и кончая узлами и линиями, можно трактовать как объект.



Отдельными элементами векторной графики являются:

Заливка (штриховка) – это цвет или узор, выводимый в замкнутой области, ограниченной кривой.

Текст

Математические способы представления основных элементов векторной графики: *точки, прямые линии, отрезки прямой, кривые второго порядка, кривые третьего порядка, кривые Безье.*

На плоскости ***точка*** представляется двумя числами (*X, Y*), задающими его положение относительно начала координат, при описании в пространстве добавляетяс третья координата Z.

Для описания ***прямой линии*** используется уравнение $Y = aX + b$. Поэтому для построения данного объекта требуется задание всего двух параметров: *a* и *b*. Результатом будет построение бесконечной прямой в декартовых координатах. В отличие от прямой, ***отрезок прямой*** требует для своего описания двух дополнительных параметров, соответствующих началу и концу отрезка (например, X_1 и X_2).

К классу ***кривых второго порядка*** относятся параболы, гиперболы, эллипсы и окружности, то есть все линии, уравнения которых содержат переменные в степени не выше второй. В векторной графике эти кривые используется для построения базовых форм (примитивов) в виде эллипсов и окружностей. Кривые второго порядка не имеют точек перегиба. Используемое для описания этих кривых каноническое уравнение требует для своего задания пяти параметров:

$$x^2 + a_1y^2 + a_2xy + a_3x + a_4y + a_5 = 0.$$

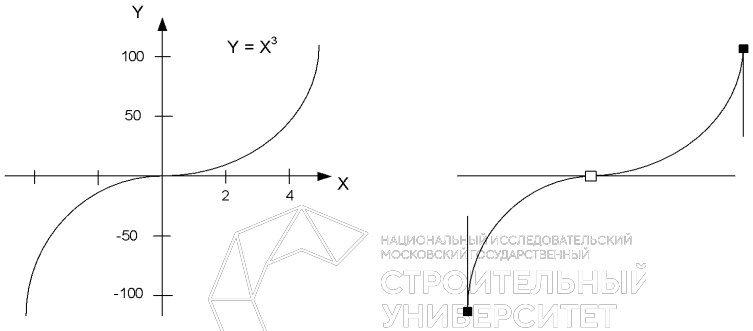
Для построения отрезка кривой требуется задать два дополнительных параметра.

Кривые третьего порядка могут иметь точку перегиба. Например, график функции $Y = X^3$ имеет точку перегиба в начале координат (0,0). Именно эта особенность данного класса функций позволяет использовать их в качестве основных кривых для моделирования различных природных объектов в векторной графике.

Каноническое уравнение, используемое для описания уравнения третьего порядка, требует для своего задания девяти параметров:

$$x^3 + a_1y^3 + a_2x^2y + a_3xy^2 + a_4x^2 + a_5y^2 + a_6xy + a_7x + a_8y + a_9 = 0.$$

Кривые Безье — это частный вид кривых третьего порядка, требующий для своего описания меньшего количества параметров - восьми вместо одиннадцати. В основе построения кривых Безье лежит использование двух касательных, проведенных к крайним точкам отрезка линии. Будут рассмотрены позже.



Достоинства векторной графики

1. Возможность неограниченного масштабирования/редактирования изображения без потери качества и практически без увеличения размеров исходного файла. Это связано с тем, что векторная графика содержит только описания объектов, формирующих изображения, а компьютер или устройство печати интерпретирует их необходимым образом.
2. Векторным программам свойственна высокая точность рисования (до сотой доли микрона).
3. Векторная графика экономна в плане объемов дискового пространства, необходимого для хранения изображений. Это связано с тем, что сохраняется не само изображение, а только некоторые основные данные (математическая формула объекта), используя которые программа всякий раз воссоздает изображение заново. Описание цветовых характеристик почти не увеличивает размер векторного файла.
4. Для векторных редакторов характерно прекрасное качество печати рисунков и отсутствие проблем с экспортом векторного изображения в растровое.

Недостатки

1. Практически невозможно осуществить экспорт изображения из растрового формата в векторный. И наоборот, обратное преобразование (то есть превращение векторного изображения в растровое) выполняется практически автоматически не только с помощью графических редакторов, но и буфера обмена Windows.
2. Векторная графика ограничена в чисто живописных средствах и не позволяет получать фотореалистичные изображения с тем же качеством, что и растровая.
3. Векторный принцип описания изображения не позволяет автоматизировать ввод графической информации, как это делает сканер для растровой графики. К сожалению, не существует, например, векторных мониторов или векторных сканеров.
4. В векторной графике невозможно применение обширной библиотеки эффектов (фильтров), используемых при работе с растровыми изображениями.

2.2. Растровая графика

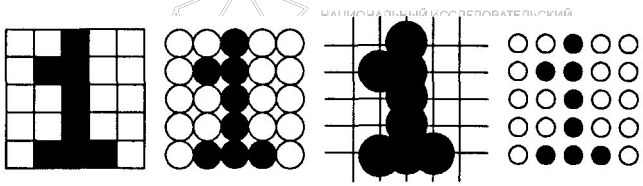
Растр - это матрица ячеек (пикселов). Любой **пиксел** (*pixel - Picture Element)* имеет свой цвет. *Совокупность пикселов различного цвета образует изображение*. В зависимости от расположения пикселов в пространстве различают квадратный, прямоугольный, гексагональный или иные типы растра.

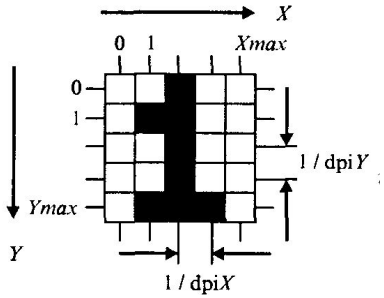
Разрешающая способность. Она характеризует расстояние между соседними пикселами - шаг дискретной сетки растра. Разрешающую способность измеряют количеством пикселов на единицу длины. Наиболее популярная единица измерения - *dpi (dots per inch)* - количество пикселов в одном дюйме длины (2.54 см). Не следует отождествлять шаг с размерами пикселов - размер пикселов может равняться шагу, а может быть как меньше, так и больше шага.

Количество цветов (глубина цвета) - важная характеристика любого изображения, не только растрового. В соответствии с психофизиологическими исследованиями, глаз человека способен различать 350 000 цветов.

Классифицируют изображения следующим образом.

- Двухцветные (бинарные) - 1 бит на пиксел. Среди двухцветных наиболее часто встречаются черно-белые изображения.
- Полутоновые - градации серого или другого цвета. Например, 256 градаций (1 байт на пиксел).
- Цветные изображения (2 бита на пиксел и больше). Глубина цвета 16 битов на пиксел (65536 цветов) получила название *High Color,* 24 бита на пиксел (16.7 млн. цветов) - *True Color.* В компьютерных графических системах используют и большую глубину цвета - 32, 48 и более битов на пиксел.





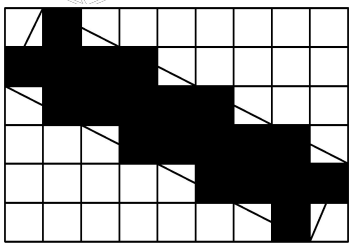
Методы улучшения растровых изображений

Устранение ступенчатого эффекта

В растровых системах при невысокой разрешающей способности (меньше 300 dpi) существует проблема ступенчатого эффекта *(aliasing)* - при большом шаге сетки растра пикселы линий образуют как бы ступени лестницы.

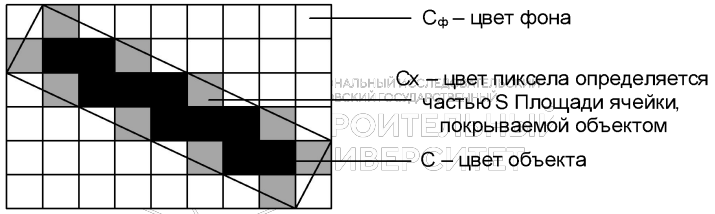
Рассмотрим это на примере отрезка прямой линии. Вообще говоря, растровое изображение объекта определяется алгоритмом закрашивания пикселов, соответствующих телу изображаемого объекта. Разные алгоритмы могут дать существенно отличающиеся варианты растрового изображения одного и того же объекта. Можно сформулировать условие корректного закрашивания следующим образом - если в контур изображаемого объекта попадает больше половины площади ячейки сетки растра, то соответствующий пиксел закрашивается цветом объекта (*С*), иначе - пиксел сохраняет цвет фона (*Сф*).

На рисунке показано растровое изображение толстой прямой линии, на которую для сравнения наложен идеальный контур исходной линии.



Устранение ступенчатого эффекта называется на английском языке *antialiasing.* Для того чтобы растровое изображение линии выглядело более ровным, можно цвет угловых пикселов "ступенек лестницы" заменить определенным оттенком, промежуточным между цветом объекта и цветом фона. Будем вычислять цвет пропорционально части площади ячейки растра, покрываемой идеальным контуром объекта. Если площадь всей ячейки обозначить как *S,* а часть площади, покрываемой контуром, - *Sx,* то искомый цвет равняется

$$C_x = \frac{C_x \cdot S_x + C_\phi \cdot (S - S_x)}{S}$$

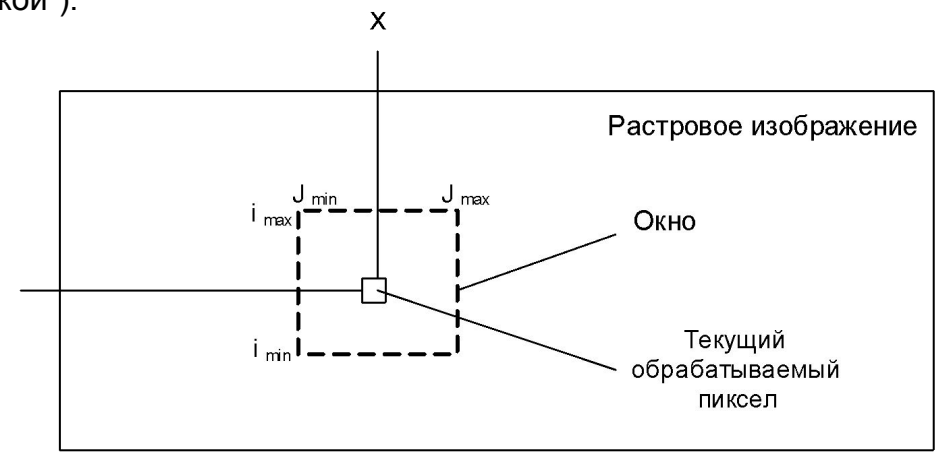


Другую группу методов сглаживания составляют методы обработки уже существующего изображения. Для сглаживания растровых изображений часто используют алгоритмы цифровой фильтрации. Один из таких алгоритмов — локальная фильтрация. Она осуществляется путем взвешенного суммирования яркости пикселов, расположенных вокруг текущего обрабатываемого пиксела. Можно представить себе, что в ходе обработки изображения по растру скользит прямоугольное окно, которое выхватывает пикселы.

Для определения цвета текущего пиксела вычисляется некоторая функция, учитывающая значение цветов пикселов этого окна. Базовую операцию такого фильтра можно представить так:

$$F_{x,y} = \frac{1}{K} \sum_{i=i_{\min}}^{i\max} \sum_{j=j_{\min}}^{j\max} P_{x+j,y+i} \cdot M_{i-i_{\min},j-j_{\min}}$$

где *P* - значение цвета текущего пиксела, *F* - новое значение цвета пиксела, *K* - нормирующий коэффициент, *M* - двумерный массив коэффициентов, который определяет свойства фильтра (обычно этот массив называют "маской").



Размеры окна фильтра: ($j_{max} - j_{min} + 1$) - по горизонтали и ($i_{max} - i_{min} + 1$) - по вертикали. При

$$i_{\min}, j_{\min} = -1$$

$$i_{\max}, j_{\max} = +1$$

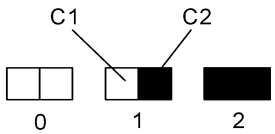
имеем фильтр с окном 3x3, часто используемый на практике.

Если графическое устройство (устройство вывода) не способно воссоздавать достаточное количество цветов, тогда используют ***растрирование*** - независимо от того, растровое это устройство или нерастровое. В полиграфии растрирование известно давно. Оно использовалось несколько столетий тому назад для печати гравюр. В гравюрах изображение создается многими штрихами, причем полутоновые градации представляются или штрихами разной толщины на одинаковом расстоянии, или штрихами одинаковой толщины с переменной густотой расположения. Такие способы используют особенности человеческого зрения и в первую очередь - пространственную интеграцию. Если достаточно близко расположить маленькие точки разных цветов, то они будут восприниматься как одна точка с некоторым усредненным цветом. Если на плоскости густо расположить много маленьких разноцветных точек, то будет создана визуальная иллюзия закрашивания плоскости определенным усредненным цветом. Однако, если увеличивать размеры точек и (или) расстояние между ними, то иллюзия сплошного закрашивания исчезает - включается другая система человеческого зрения, которая обеспечивает способность различать объекты, подчеркивать контуры.

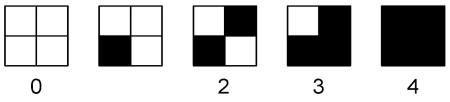
В компьютерных графических системах часто используют эти методы. *Они позволяют увеличить количество оттенков цветов за счет снижения пространственного разрешения растрового изображения. Иначе говоря - это обмен разрешающей способности на количество цветов.* В литературе по КГ такие методы растрирования получили название ***dithering*** (разрежение, дрожание).

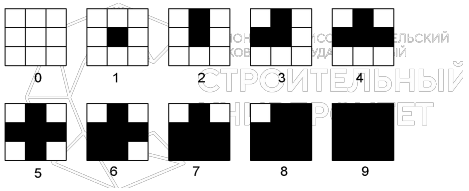
Простейшим вариантом дизеринга можно считать создание оттенка цвета парами соседних пикселов. Если рассмотреть ячейки из двух пикселов, то ячейка номер 1 дает оттенок цвета С:

$$C = \frac{C_1 + C_2}{2}$$



где C_1 и C_2 - цвета, которые графическое устройство непосредственно способно воссоздать для любого пиксела.
Наиболее часто используют квадратные ячейки больших размеров. Приведем пример ячеек размером 2x2. Такие ячейки дают 5 градаций, из них три комбинации (1, 2, 3) образуют новые оттенки.





Базовые растровые алгоритмы

1. Прямое вычисление координат

Пусть заданы координаты конечных точек отрезка прямой. Найдем координаты точки внутри отрезка. Запишем соотношения катетов для подобных прямоугольных треугольников:

$$\frac{x - x_1}{y - y_1} = \frac{x_2 - x_1}{y_2 - y_1}$$

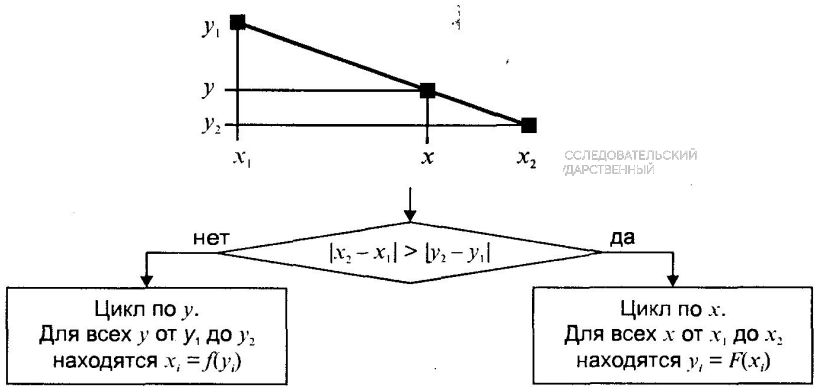
Перепишем это соотношение как *x =f(y):*

$$x = x_1 + (y - y_1)\frac{x_2 - x_1}{y_2 - y_1}$$

а также как *y =F(x):*

$$y = y_1 + (x - x_1)\frac{y_2 - y_1}{x_2 - x_1}$$

В зависимости от угла наклона прямой выполняется цикл по оси *x* или по *y*



Положительные черты прямого вычисления координат.

1. Простота, ясность построения алгоритма.
2. Возможность работы с нецелыми значениями координат отрезка.

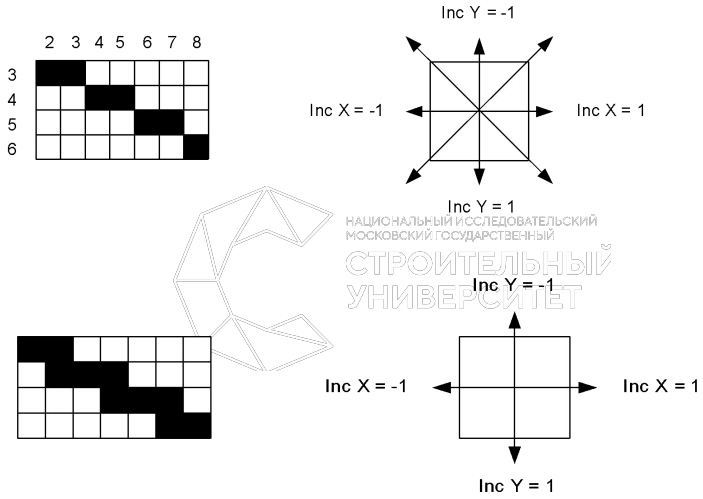
Недостатки.

1. Использование операций с плавающей точкой или целочисленных операций умножения и деления обуславливает маленькую скорость.
2. При вычислении координат добавлением приращений может накапливаться ошибка вычислений координат.

2. Инкрементные алгоритмы

Брезенхэм предложил подход, позволяющий разрабатывать так называемые *инкрементные алгоритмы растеризации.* Основной целью при разработке таких алгоритмов было построение циклов вычисления координат на основе только целочисленных операций сложения/вычитания без использования умножения и деления. Были разработаны инкрементные алгоритмы не только для прямых, но и для кривых линий.

Инкрементные алгоритмы выполняются как последовательное вычисление координат соседних пикселов путем добавления приращений координат. Приращения рассчитываются на основе анализа функции погрешности. В цикле выполняются только целочисленные операции сравнения и сложения/вычитания. Достигается повышение быстродействия для вычислений каждого пиксела по сравнению с прямым способом.



Инструменты растровых графических пакетов

К фундаментальным инструментам растровой графики относятся такие инструменты обработки изображений, как:

- инструменты выделения;
- каналы и маски;
- инструменты ретуширования;
- гистограммы;
- кривые;
- инструменты для цветовой (цветовой баланс) и тоновой коррекции (уровни);
- фильтры (спецэффекты);
- слои

Инструменты выделения. Каналы и маски

Растровое изображение в отличие от векторного не содержит объектов, которые можно легко «расцепить для выполнения их индивидуального редактирования. Поэтому для создания, например, коллажей (фотомонтажей) из отдельных фрагментов нескольких изображений каждый из них предварительно необходимо выделить. Такая работа, напоминающая вырезание кусков изображений из бумаги ножницами, называется процессом выделения (или обтравки) изображений. *Выделение* (Selection) – это область, ограниченная замкнутой рамкой выделения в виде движущейся пунктирной линии (контура), которая отмечает часть изображения, доступную для копирования, редактирования и выполнения различных типов преобразований. На жаргоне программистов эта пунктирная линия получила название марширующие муравьи». Она отделяет выделенную область от защищенной области.

Маски – это один из базовых инструментов профессиональных растровых редакторов. Хотя концепции *маски* и *выделения* тесно связаны, понятие маски шире. Всякая маска включает в себя два типа областей: непрозрачные и прозрачные. Первые используются для защиты закрываемых ими частей изображений или объектов от нежелательных изменений. Они, собственно, и выполняют функцию *маскирования.* Прозрачные области можно рассматривать как отверстия в маске. Их используют для выделения фрагментов изображения или объекта, которые собираются модифицировать. Эти области называются *выделенной областью,* или *выделением (обтравкой).*

Таким образом, маска не есть нечто противоположное выделению. Противоположными свойствами обладают части маски, а именно *защищенные* и *выбранные* (выделенные) области. Соотношение между этими частями не является постоянным. В процессе работы над изображением оно может изменяться за счет увеличения доли одной из них и соответственно уменьшения доли другой. Для этой цели в растровых

Таким образом, под термином *выделение* (или выделенная область) будем понимать области изображений и объектов, доступные для перемещения, копирования, редактирования и выполнения любых других преобразований. И наоборот, термин *маска* используется для обозначения областей изображений и объектов, защищенных от применения перечисленных операций.

Понятие маски возникло не на пустом месте. По смыслу и назначению оно близко к понятию трафарета. Представьте себе художника, вырезающего из ватмана трафарет какого-либо слова. Затем он набивает по этому трафарету текст поролоновой губкой, смоченной в краске. При этом часть краски попадает в прорезанные отверстия, а часть остается на трафарете, который и выполняет в данном случае роль защитной маски.

Создание маски приводит к одновременному созданию альфа-канала, в который помещается «серое» изображение маски. Чтобы более четко понять связь этих двух понятий, давайте остановимся на физической природе маски. Если говорить техническим языком, то маска сама является изображением. Это изображение помещается поверх другого изображения, над фрагментами которого мы собираемся выполнить определенные операции. Для любого пикксела маски значение оттенка серого цвета можно изменять в пределах 256 градаций серого (от 0 до 255). Область маски со значением цвета пикселов, равного 0 (черный), полностью защищает изображение от изменений (собственно, и служит маской). Область, пикселы которой имеют значение 255 (белый), полностью открыта для проведения изменений. Такая область называется выбранной (выделенной).

Инструменты выделения бывают:

Обычные (геометрические), использующие для построения выделений разнообразные геометрические формы: прямоугольную, квадратную, круглую и эллиптическую.

Инструменты выделения от «руки». Типичным примером таких инструментов являются: Лассо в Photoshop и Freehand Mask в Corel PHOTO-PAINT. Они используются для выделения объектов сложной формы путем их обводки.

Инструменты выделения контуров (path tools) похожи на инструменты предыдущей группы. Однако в данном случае выделенные области представляют собой векторные объекты.

Цветочувствительные, в которых выделенная область изображения определяется цветом изображения. В основе работы этих инструментов лежит назначение двух параметров:

- базового цвета, выбираемого щелчком мыши на соответствующей точке изображения;
- диапазона цветов, близких к базовому.

Ретушь

Внструменты ретуширования изображений предназначены для восстановления поврежденных изображений, например, для ретуши фотографий.

Ретушь (retouch) - коррекция изображения с целью устранения мелких дефектов, исправления тонального и цветового балансов.

Основные группы операций:

• устранить детали, мешающие созданию нужного эффекта. Обычно это морщины на лице, блики и мелкие посторонние предметы;

• добавить некоторые детали, чтобы подчеркнуть (усилить) нужный эффект.

Осноынве инструменты ретуширования:

Инструменты клонирования (Cloning Tools) предназначены для копирования деталей из одного места изображения (неповрежденного) в другое (поврежденное). Клонирование рекомендуется применять для удаления дефектов сканирования, следов пыли, царапин пятен путем замены на тона и детали того же или другого изображения, сходного по цвету или более совершенного.

Инструменты размытия (Blur) и *повышения резкости* (Sharpen) позволяют соответственно локально снижать или усиливать контраст между пикселами изображения. Так, локальное ослабление нежелательных подробностей (морщин, нездорового цвета кожи и т. д.) позволит акцентировать внимание на главных деталях изображения, маскируя второстепенные детали. В то же время локальное увеличение резкости может привлечь внимание к каким-то особенностям изображения (например, блеск драгоценностей), что составляет основу рекламы производимых изделий или имиджа человека, использующего эти изделия.

Инструменты сглаживания минимизируют различия между соседними оттенками в тех местах, где проходит кисть. Они применяются для удаления морщин, складок на одежде, случайного шума, наложенного на изображение при сканировании, а также для сглаживания границ между исходными и клонированными с помощью соответствующего инструмента участками изображения.

Инструменты Осветления (Dodge) и *Затемнения* (Burn) делают объекты более светлыми или тусклыми. Эти средства предназначены для коррекции освещенности или изменения значения яркости, чтобы выделить или скрыть отдельные детали.

Гистограммы

Инструмент *Гистограмма* (Histogram) позволяет оценить разброс между минимальной и максимальной яркостью изображения (динамический диапазон). С его помощью можно получить также наглядное представление о распределении всех тонов в изображении. Гистограммой называется график, отображающий распределение пикселов изображения по яркости.

При построении этого графика по оси X откладываются значения яркостей в диапазоне от 0 (черный) до 255 (белый), а по оси Y – количество пикселов, имеющих соответствующее значение яркости.

Параметры раздела *Входные уровни (Input Levels)* используются для установки новых значений черной и белой точек изображения, что позволяет сократить диапазон яркостей изображения и повысить его контрастность. Для этих целей можно воспользоваться перемещением находящихся над гистограммой треугольников либо ввести численные значения в соответствующие поля ввода. Например, установка в левом поле значения 30 приведет к тому, что все цвета, имеющие значение яркости меньше этой величины, станут черными, и соответственно ввод значения 220 в правом поле приведет к обращению в максимум всех яркостей в диапазоне 220-255. В результате диапазон яркостей исходного изображения понизится с 255 до 195, а контрастность возрастет. Следует помнить, что при выполнении такого преобразования информация, содержащаяся в тоновых диапазонах 0 - 30 (светах) и 220 - 255 (тенях), будет потеряна. Однако при неудовлетворительном результате всегда можно воспользоваться кнопкой Отмена (Cancel) или экспериментировать с копией исходного изображения.

Между крайними треугольниками, характеризующими значение светов и теней изображения, расположен третий треугольник, который предназначен для управления яркостью в области средних тонов изображения. Этот элемент управления в растровой графике имеет специальное название – *коэффициент гамма (гамма* - коэффициент контраста в средних тонах изображения), а действия, выполняемые путем перемещения среднего треугольника, называют настройкой гаммы. Установка значения этого параметра меньше 1 (это значение задается по умолчанию) приводит к затемнению изображения, и наоборот, больше 1 – к осветлению изображения в области средних тонов. В обоих случаях происходит изменение контрастности изображения.

Параметрами раздела Выходные уровни (Output Levels) можно управлять точно так же, как и входными параметрами. Однако в отличие от них, здесь перемещение левого треугольника приводит к осветлению более темных пикселов (теней), и наоборот, перемещение правого треугольника затемняет более светлые пикселы (света). Например, задан в левом поле значение, равное 40, вы настраиваете на эту величину яркость самого темного пиксела, что приводит к повышению уровня освещенности изображения. Аналогичным образом с помощью правого поля ввода можно установить новое, более низкое, значение самого светлого пиксела. В итоге это приводит к снижению контрастности изображения.

Кривые

По принципу действия команда Кривые близка к команде Уровни. Только здесь для настройки яркости изображения в окне диалога Кривые (Curves) вместо гистограммы используется инструментальное средство, известное под именем *кривая* (в локализованных версиях растровых редакторов встречаются и другие термины - *настроечная кривая* и *градационная кривая).*

В момент открытия окна диалога Кривые (Curves) его основное инструментальное средство - настроечная кривая - предстает в виде прямой линии с наклоном 45. Это говорит о том, что все входные и выходные пикселы имеют идентичные значения яркости.

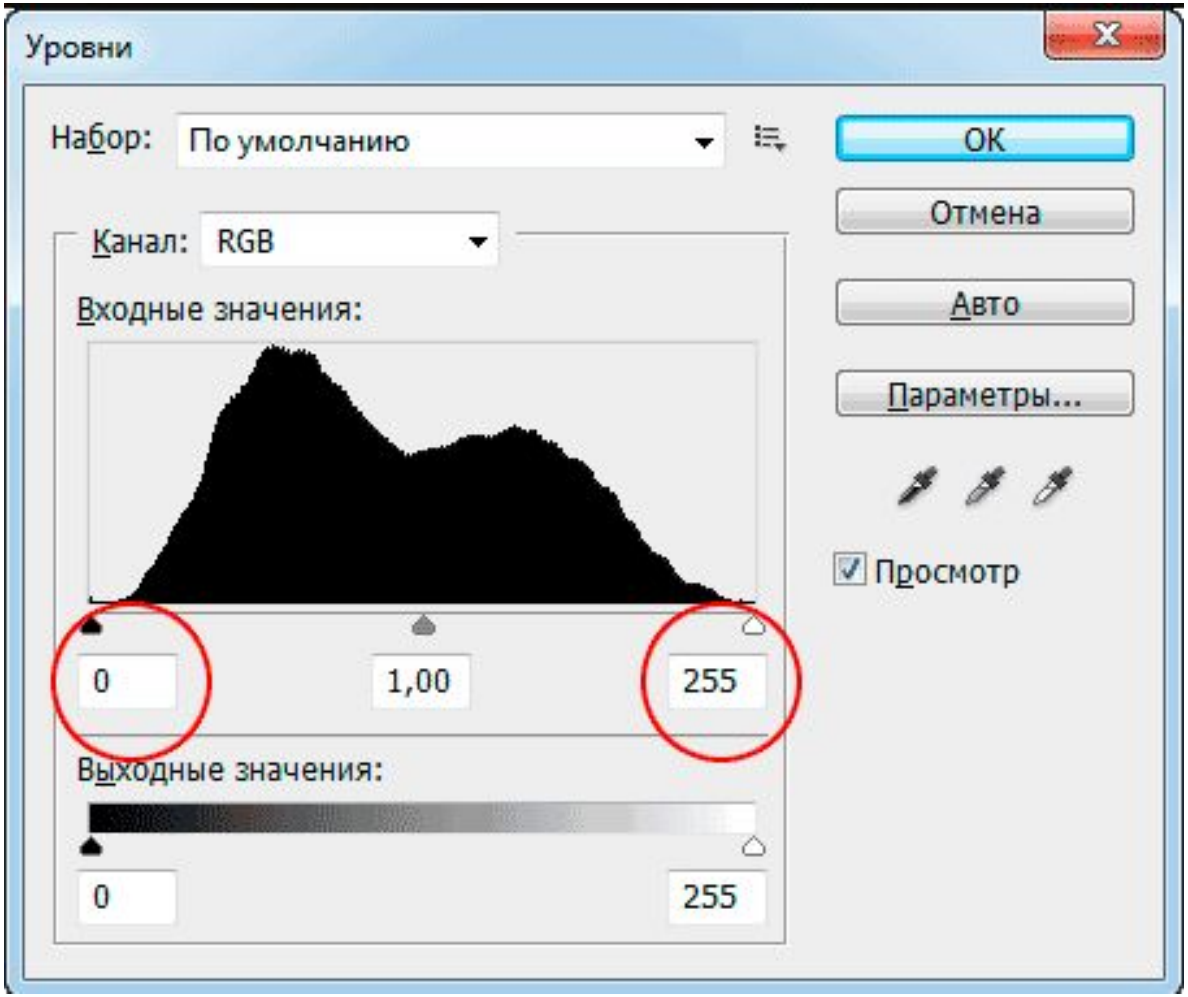
Кривая (curves) – это график, с помощью которого осуществляется преобразование спектрального диапазона исходного изображения (входные данные) к спектральному диапазону скорректированного изображения (выходные данные). В некоторых источниках это инструментальное средство называют также *яркостная кривая, настроечная кривая и градационная кривая.*

Иными словами, *кривая* – это инструмент для одновременного изменения контраста во многих яркостных диапазонах изображения.

Сущность белой и черной точек

Белой точкой (White point) называется то место изображения, где оно выглядит очень светлым, но при этом в нем еще можно различить какие-то детали изображения. Белую точку в изображении можно задать искусственно более темной. В этом случае все элементы изображения, более светлые, чем указанные данным инструментом, будут полностью белыми без видимых деталей.

Черной точкой (Black point) называется то место изображения, где оно выглядит очень темным, но при этом в нем еще можно различить какие-то детали изображения. Черную точку в изображении можно задать искусственно более светлой, указан в изображении на более светлое место. В этом случае все элементы изображения, более темные, чем указанные данным инструментом, будут полностью черными без видимых деталей.


Уровни
Набор:
По умолчанию
OK
Отмена
Канал:
RGB
Авто
Входные значения:
Параметры...
Просмотр
0
1,00
255
Выходные значения:
0
255

Цветовая коррекция и цветовой баланс

См. подраздел «Цветовые модели в КГ

Фильтры (Plug-ins) и спецэффекты (Effects)

Большинство фильтров (filters или plug-ins) предназначено для создания специальных эффектов, например имитации мозаики или живописного стиля Ван-Гога. С помощью трехмерных спецэффектов двухмерные графические программы способны трансформировать плоское двухмерное изображение в объемное. При этом можно имитировать самые разные виды живописи (масло, акварель и т. п.) и стили любых художников.

Фильтры и спецэффекты представляют собой небольшие программы, выполняющие заранее установленную последовательность команд. Они автоматически вычисляют значения и характеристики каждого пиксела изображения и затем модифицируют их в соответствии с новыми значениями.

С алгоритмической точки зрения получение специальных эффектов не представляет особого труда. Секрет каждого из них кроется в крошечной матрице чисел, которую называют *ядром свертки.* Матрица размером 3 x 3 содержит три строки по три числа в каждой. Для преобразования каждого пиксела изображения необходимо выполнить следующие действия:

- Шаг 1. Значение цвета пиксела умножается на число в центре ядра (a22).
- Шаг 2. На следующем шаге выполняется умножение восьми значений цветов пикселов, окружающих центральный пиксел, на соответствующие им коэффициенты ядра с последующим суммированием всех девяти значений. В результате получается новое значение цвета преобразуемого пиксела.
- Шаг 3. Для каждого пиксела изображения повторяется процесс, включающий выполнение шагов 1 и 2. Данную процедуру принято называть *фильтрацией изображения.*

Коэффициенты ядра свертки определяют результат процесса фильтрации. Их формирование зависит от типа эффекта. Например, *ядро размывания* состоит из совокупности коэффициентов, каждый из которых меньше 1, а их сумма составляет 1. Это означает, что каждый пиксел поглощает что-то из цветов соседей, но полная яркость изображения остается неизменной (если сумма коэффициентов больше 1, яркость увеличится; если меньше 1, яркость уменьшится). В *ядре резкости* центральный коэффициент больше 1, а окружающие его значения являются отрицательными числами, сумма которых на единицу меньше центрального коэффициента. Таким образом достигается увеличение существующего контраста между цветом пиксела и цветами его соседей. Это звучит немного мистически, но цифровое изображение, в конце концов, всего лишь связка чисел. Изменяя эти числа правильным способом, можно прийти к удивительным и, возможно, очень полезным спецэффектам.

Пример 1. Алгоритм работы фильтра Размывание

1. При подготовке к размыванию цифровое изображение считывается в память компьютера в виде красного, зеленого и синего компонентов цвета каждого пиксела.

2. Ядро размывания размером 3 x 3 применяется к красному, зеленому и синему компонентам цвета каждого пиксела изображения. Значение цвета пиксела (который, собственно, находится в центре ядра) вычисляется умножением соответствующего весового коэффициента на соответствующее ему значение цвета в изображении с последующим суммированием результатов. Итоговое изображение получается размытым по сравнению с оригиналом, потому что цвет каждого пиксела выровнялся (усреднился) благодаря влиянию соседей.

3. Степень размывания можно увеличить одним из трех способов:

• использованием большего размера ядра для распределения цвета среди большего числа соседей (в окне диалога этого фильтра значение размера ядра задается установкой параметра Радиус (Radius) в пикселах);

• подбором коэффициентов ядра и уменьшением влияния центрального коэффициента,

• повторной фильтрацией изображения с тем же ядром размывания.

Преимущества и недостатки растровой графики

Достоинства

1.Простота и, как следствие, техническая реализуемость (автоматизация) ввода (оцифровки) изобразительной информации. Существует развитая система внешних устройств ввода изображений (к ним относятся сканеры, видеокамеры, цифровые фотокамеры, графические планшеты).

Наш мир создан как растровый. И его объекты трудно представить в векторном, то есть математическом, представлении. Фотореалистичность подразумевает, что в растровой программе можно получать живописные эффекты, например туман или дымку, добиваться тончайшей нюансировки цвета, создавать перспективную глубину и нерезкость, размытость и т. д.

2. Форматы файлов, предназначенные для сохранения точечных изображений, являются стандартными, поэтому не имеет решающего значения, в каком графическом редакторе создано то или иное изображение.

Недостатки

1. Объем файла точечной графики однозначно определяется произведением площади изображения на разрешение и на глубину цвета (если они приведены к единой размерности). Этот объем значительно превосходит объемы хранения векторных изображений.
2. Любые трансформации (повороты, масштабирование, наклоны) в точечной графике не бывают без искажений.
3. Простота оцифровки изображений.

2.3. Фрактальная графика

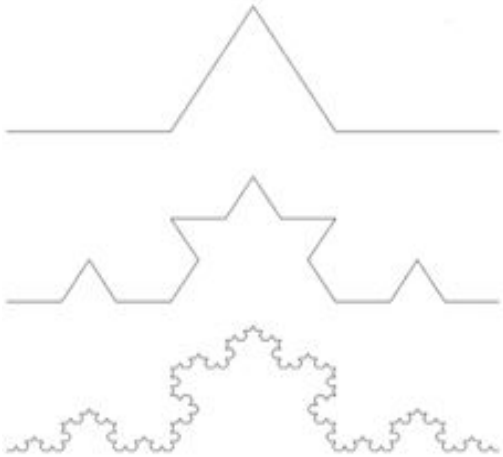
Понятия фракталы, фрактальная геометрия и фрактальная графика, появившиеся в конце 70-х, сегодня прочно вошли в обиход математиков и компьютерных художников. Слово фрактал образовано от латинского *fractus* и в переводе означает “состояние из фрагментов”. Оно было предложено математиком Бенуа Мандельбромом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался.

Одним из основных свойств фракталов является самоподобие. Объект называют самоподобным, когда увеличенные части объекта походят на сам объект и друг на друга.

Типы фракталов

1. Геометрические

Их форма может быть описана как последовательность простых геометрических операций. Например, *кривая Кох* становится фракталом в результате бесконечного количества итераций, в ходе которых выполняется деление каждого отрезка прямой на три части

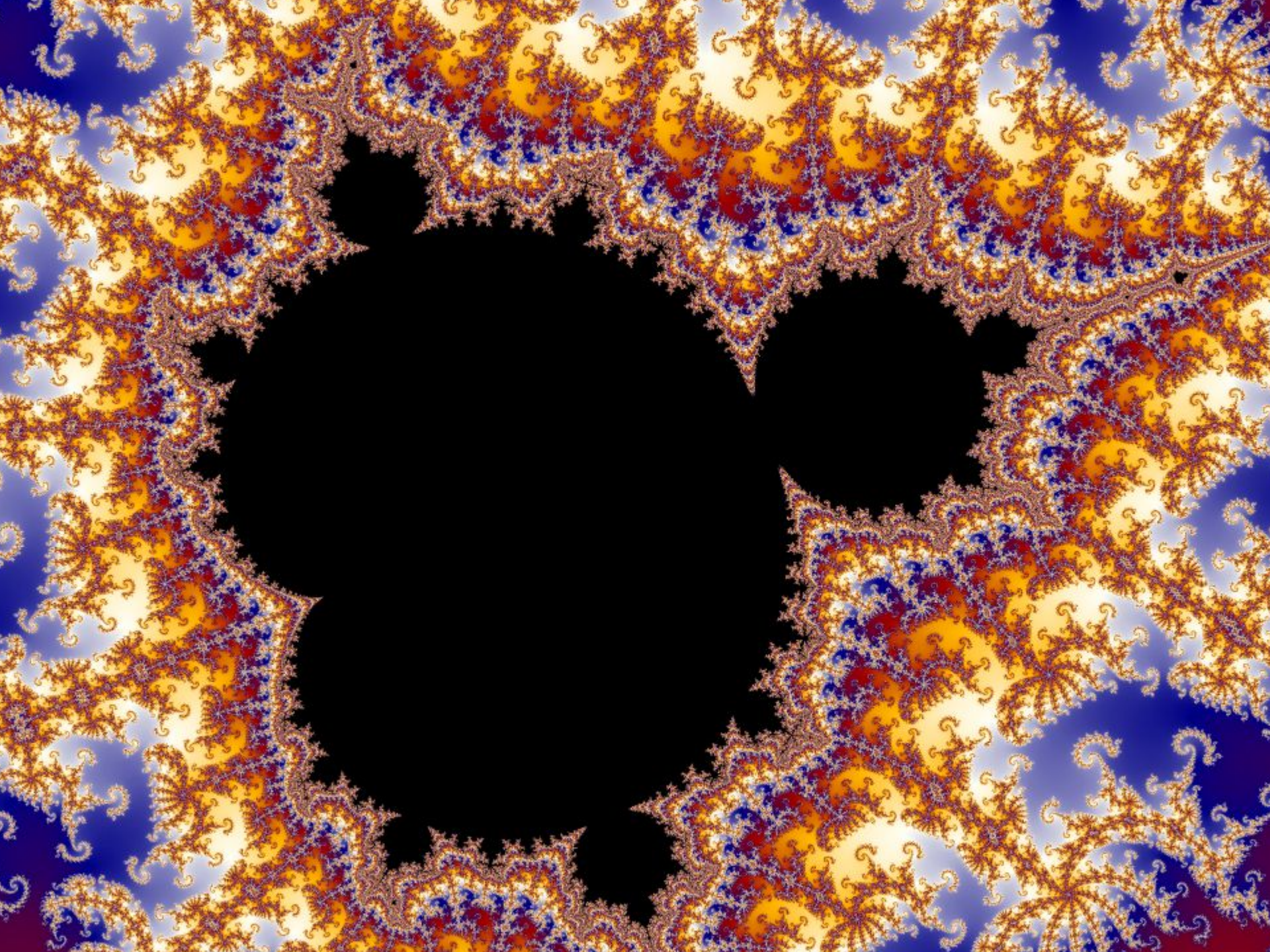


2. Математические фракталы

Фракталом Мандельброта названа фигура, которая порождается очень простым циклом. Для создания этого фрактала необходимо для каждой точки изображения выполнить цикл итераций в соответствии с формулой:

$$z_{k+1} = z_k^2 + z_0$$

где $k = 0, 1, ..., n$. Величины z_k - это комплексные числа, $z_k = x_k + iy_k$, причем стартовые значения x_0 и y_0 - это координаты точки изображения. Для каждой точки изображения итерации выполняются ограниченное количество раз *(п)* или до тех пор, пока модуль числа z_k не превышает 2. Модуль комплексного числа равняется корню квадратному из $x^2 + y^2$. Для вычисления квадрата величины z_k можно воспользоваться формулой $z^2 = (x + iy)(x + iy) = x^2 - y^2 + i2xy$, поскольку $i^2 = -1$. Цикл итераций для фрактала Мандельброта можно выполнять в диапазоне x = (от -2. 2, до 1), y = (от -1. 2 до 1. 2). Для того чтобы получить изображение в растре, необходимо пересчитывать координаты этого диапазона в пиксельные



Фрактал Жулиа совсем не похож на фрактал Мандельброта, однако, он определяется итерационным циклом, почти полностью тождественным циклу генерации Мандельброта. Формула итераций для фрактала Жулиа такова:

$$z_{k+1} = z_k^2 + c$$

где *c* – комплексная константа.

Условием завершения итераций является $| z_k | > 2$ - так же, как для фрактала Мандельброта.

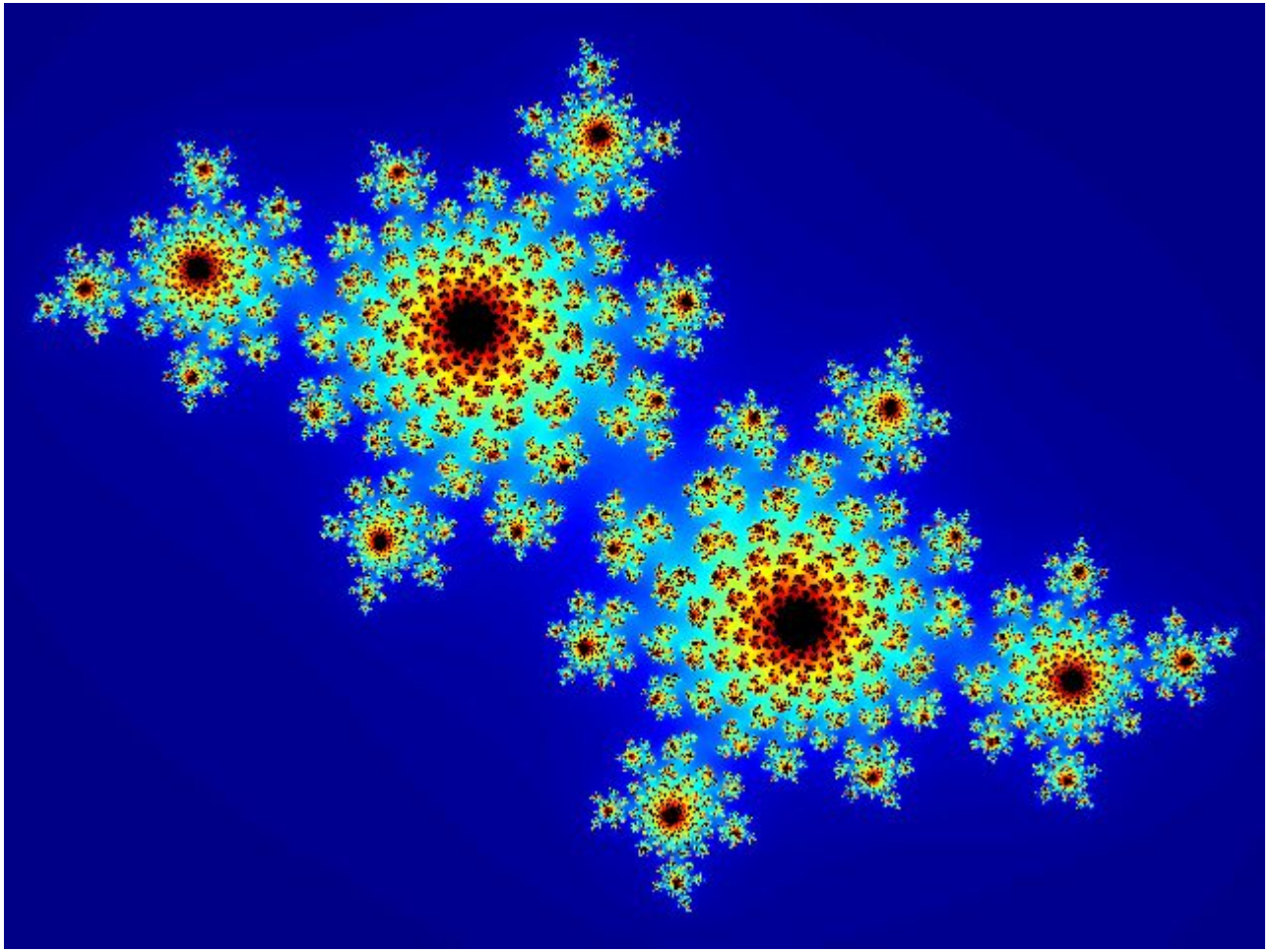
Как видим, фрактал самоподобный – при любом увеличении отдельные части напоминают формы целого. Самоподобие считается важным свойством фракталов. Это отличает их от других типов объектов сложной формы.

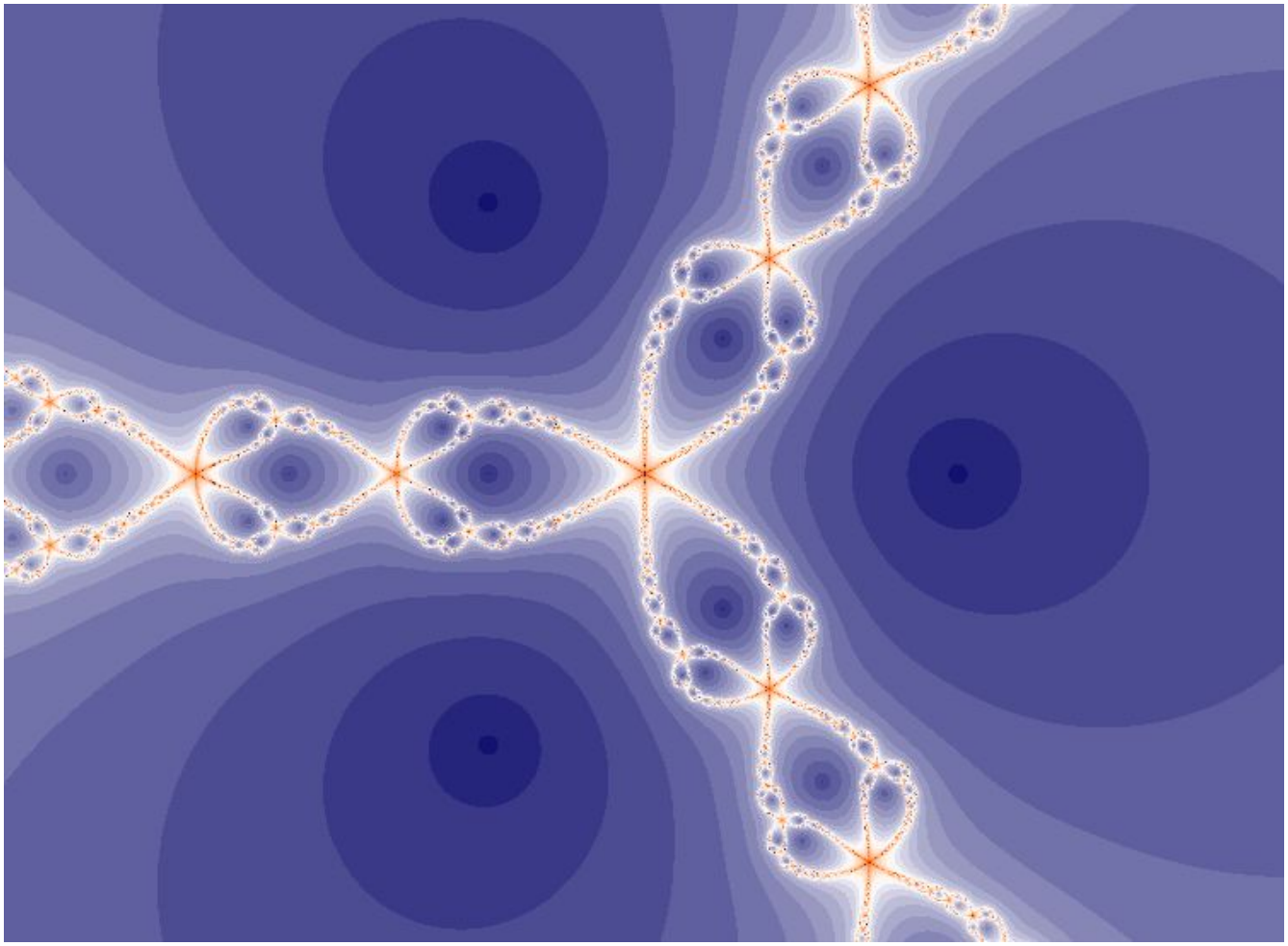
Фрактал Ньютон. Для него итерационная формула имеет такой вид:

$$z_{k+1} = \frac{3z_k^4 + 1}{4z_k^3}$$

где *z* – также комплексные числа, причем $z_0 = x + iy$ соответствует координатам точки изображения.

Условием прекращения цикла итераций для фрактала Ньютон есть приближение значений $|x^4-1|$ к нулю.

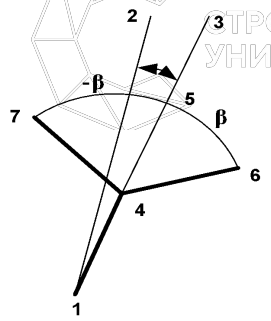




3. Фракталы, которые генерируются согласно методу "систем итеративных функций" - IFS (Iterated Functions Systems). Этот метод может быть описан, как последовательный итеративный расчет координат новых точек в пространстве:

$$x_{k+1} = F_x(x_k, y_k),$$
$$y_{k+1} = F_y(x_k, y_k),$$

где F_x и F_y - функции преобразования координат, например, аффинного преобразования. Эти функции и обуславливают форму фрактала. В случае аффинного преобразования необходимо найти соответствующие числовые значения коэффициентов.



Попробуем разработать фрактал, который выглядел бы, как растение. Вообразим ствол, на котором много веточек. На каждой веточке много меньших веточек и так далее. Наименьшие ветви можно считать листвой или колючками. Все элементы будем рисовать отрезками прямой. Каждый отрезок будет определяться двумя конечными точками.

Для начала итераций необходимо задать стартовые координаты концов отрезка. Это будут точки 1, 2. На каждом шаге итераций будем рассчитывать координаты других точек.

Сначала находим точку 3. Это повернутая на угол α точка 2, центр поворота - в точке 1

$$x_3 = (x_2 - x_1)\cos\alpha - (y_2 - y_1)\sin\alpha + x_1,$$
$$y_3 = (x_2 - x_1)\sin\alpha + (y_2 - y_1)\cos\alpha + y_1.$$

Если α = 0 , то ствол и все ветви прямые. Потом находим точку 4. От нее будут распространяться ветви. Пусть соотношение длин отрезков 1- 4 и 1 - 3 равняется *k,* причем 0 < *k* < 1. Тогда для вычисления координат точки 4 можно воспользоваться такими формулами:

$$x_4 = x_1 (1-k) + x_3 k,$$
$$y_4 = y_1 (1-k) + y_3 k.$$

Теперь зададим длину и угол наклона ветвей, которые растут из точки 4. Сначала найдем координаты точки 5. Введем еще один параметр - *k1,* который будет определять соотношение длин отрезков 4-5 и 4-3, причем 0 < *kl* < 1. Координаты точки 5 равняются

$$x_5 = x_4 (1-k1)/+ x_3 k1,$$
$$y_5 = y_4 (1-k1) + y_3 k1.$$

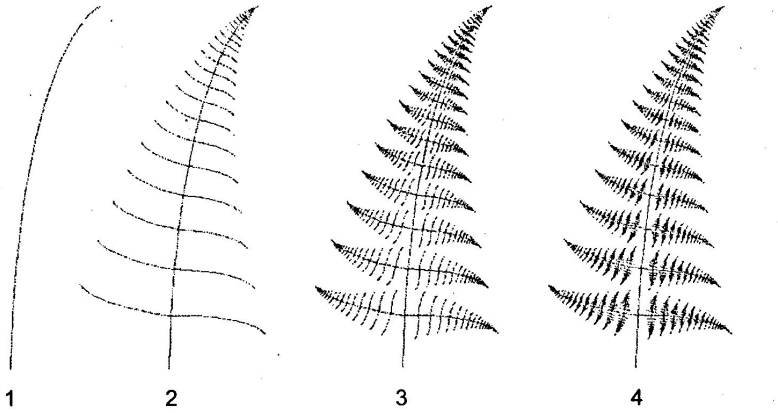
Точки 6 и 7 - это точка 5, но повернутая относительно точки 4 на углы β и - β соответственно:

$$x_6 = (x_5 - x_4)\cos\beta - (y_5 - y_4)\sin\beta + x_4,$$
$$y_6 = (x_5 - x_4)\sin\beta + (y_5 - y_4)cojs\,\beta + y_4,$$
$$x_7 = (x_5 - x_4)\cos\beta + (y_5 - y_4)\sin\beta + y_4,$$
$$y_7 = (x_5 - x_4)\sin\beta + (y_5 - y_4)\cos\beta + y_4.$$

Кроме расчета опорных точек, на каждом шаге будем рисовать один отрезок 1 - 4. В зависимости от номера итераций можно изменять цвет отрезка. Также можно устанавливать его толщину, например, пропорционально длине.

Величины а, β, *к, k1* - это параметры, которые описывают вид фрактала в целом. Они являются константами на протяжении всего итеративного процесса. Это дает возможность в итерациях использовать только операции сложения, вычитания и умножения, если вычислить значения *sin(), cos(), (1 - к) и (1 - k1)* только один раз перед началом итераций как коэффициенты-константы.

Фрактал при α = 2°, β = 86°, *к* = 0. 14, k1 = 0. 3 похож на папоротник



Сферы применения фрактальной графики.

1.Создание алгоритма фрактального сжатия графической информации. Так, метод IFS используется не только для создания изображений. Основная идея такая: поскольку фракталы могут представлять очень сложные изображения с помощью простых итераций, то описание этих итераций требует значительно меньшего объема информации, чем соответствующие растровые изображения. Для кодирования изображений необходимо решать обратную задачу - для изображения (или его фрагмента) подобрать соответствующие коэффициенты аффинного преобразования. Этот метод используется для записи цветных фотографий в файлы со сжатием в десятки и сотни раз без заметного ухудшения изображения. Формат таких графических файлов был назван FIF (Fractal Image Format) и запатентован фирмой Iterated Systems.

2. Искусство и реклама.

Преимущества и недостатки фрактальной графики

Достоинства

Качественный, легко реализуемый математический аппарат.

Недостатки

1. Отсутствие «родных» форматов файлов.
2. Узкая область применения

Глава 3. Методы, модели и алгоритмы создания сложных реалистических объектов и сцен

Основными проблемами, которые возникают при создании, обработки и представлении сложных реалистических объектов и сцен, являются:

1. **Сложность геометрических форм**
2. **Сложность описания взаимного расположения объектов**
3. **Сложность упорядочения объектов и их отдельных составляющих (вершин, ребер, граней) при выводе (визуализации)**
4. **Цветовая гамма объектов**
5. **Освещенность объектов**
6. **Динамика объектов**
7. **Текстурирование**

3.1. Модели построения и представления сложных геометрических форм

Метод проекций

В общем случае проекции преобразуют точки, заданные в системе координат размерностью *n*, в системы координат размерностью меньше чем *n (n-1)*.

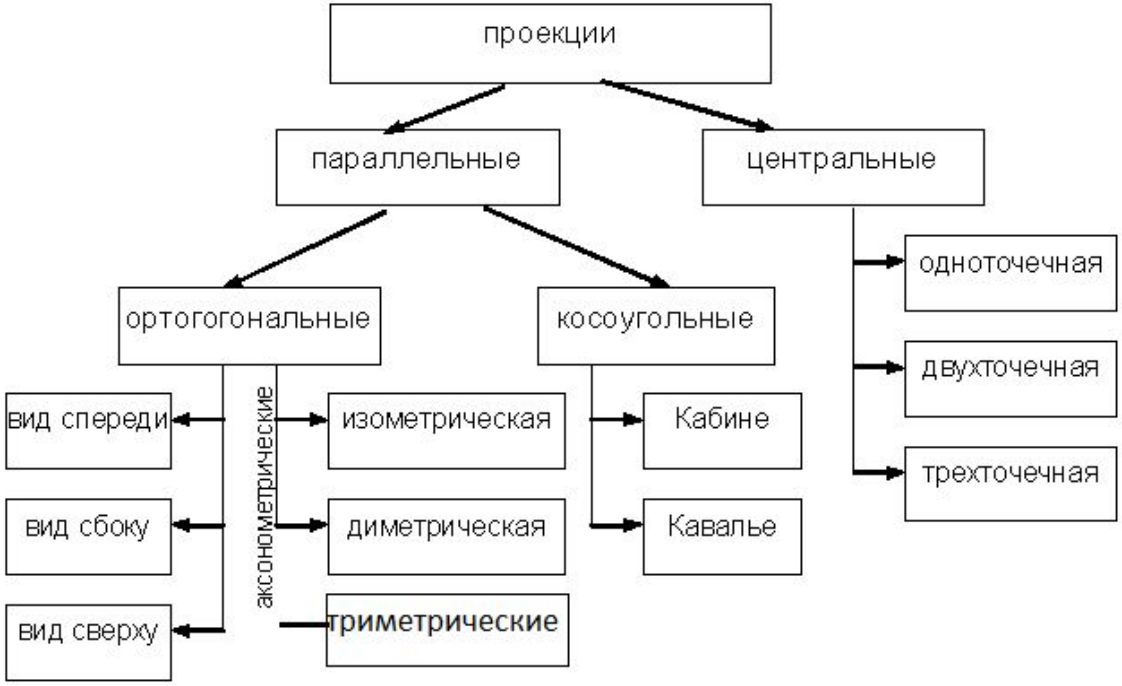
В КГ чаще всего рассматривается случай проецирования трех измерений в два. Проекция трехмерного объекта (представленного в виде совокупности точек) строится при помощи прямых проекционных лучей, которые называются ***проекторами*** и которые проходят через каждую точку объекта и, пересекая картинную плоскость, образуют ***проекцию***.

Для КГ принципиальным, и чаще всего программно-реализуемым является класс ***плоских геометрических проекций***, так как проецирование производится на плоскость, а не на искривленную поверхность и в качестве проекторов используются прямые, а не кривые линии.

Многие картографические проекции являются либо не плоскими, либо не геометрическими.

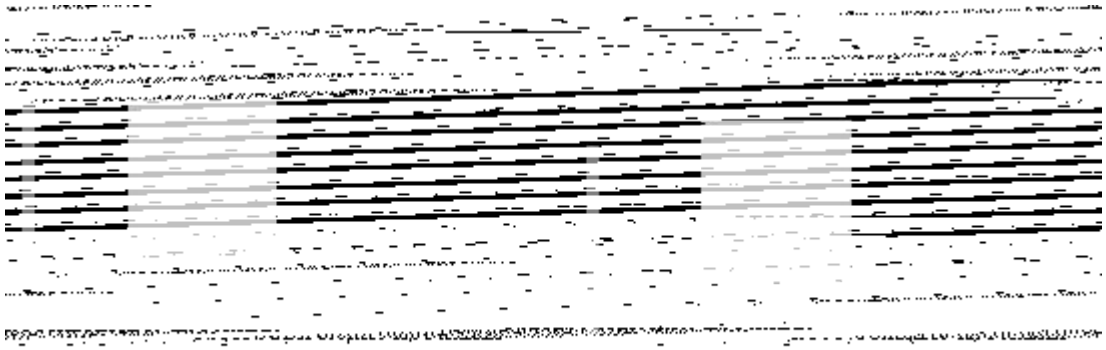
Проекции делятся на два основных класса:

- параллельные;
- центральные (перспективные).



Параллельные проекции делятся на два типа в зависимости от соотношения между направлением проецирования и нормалью к проекционной плоскости:

1) ***ортографические (ортогональные)*** – направления совпадают, т. е. направление проецирования является нормалью к проекционной плоскости;
2) ***косоугольные*** – направление проецирования и нормаль к проекционной плоскости не совпадают.

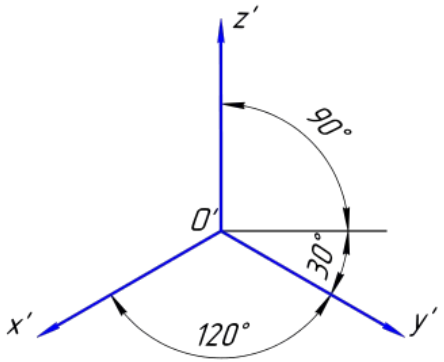


Наиболее широко используемыми видами ортографических проекций является вид спереди, вид сверху (план) и вид сбоку, в которых картинная плоскость перпендикулярна главным координатным осям. Если проекционные плоскости не перпендикулярны главным координатным осям, то такие проекции называются ***аксонометрическими***.

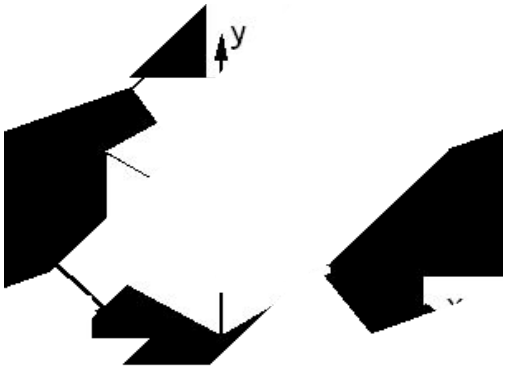
При аксонометрическом проецировании сохраняется параллельность прямых, а углы изменяются; расстояние можно измерить вдоль каждой из главных координатных осей (в общем случае с различными масштабными коэффициентами).

Изометрическая проекция – нормаль к проекционной плоскости, (а следовательно и направление проецирования) составляет равные углы с каждой из главных координатных осей. Если нормаль к проекционной плоскости имеет координаты (a,b,c), то потребуем, чтобы $|a| = |b| = |c|$, или $\pm a = \pm b = \pm c$, т. е. имеется 8 направлений (по одному в каждом из октантов), которые удовлетворяют этому условию. Однако существует лишь 4 различных изометрических проекции (если не рассматривать удаление скрытых линий), так как векторы (a, a, a) и $(-a,-a,-a)$ определяют нормали к одной и той же проекционной плоскости.

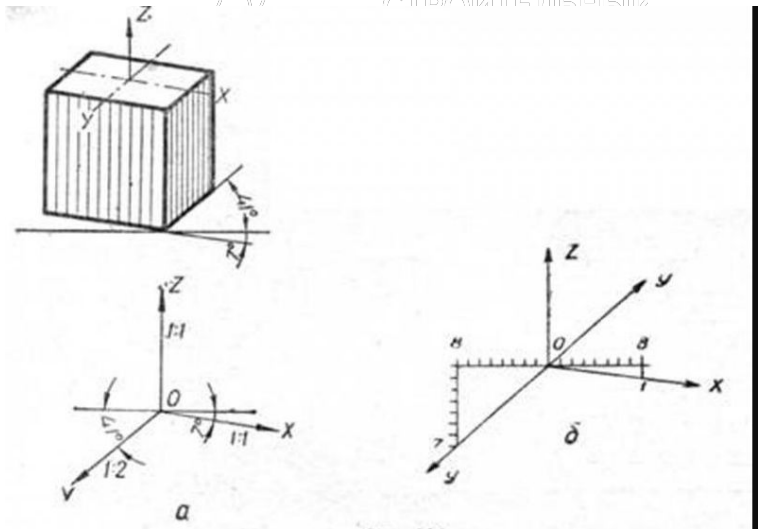
Изометрическая проекция обладает следующим свойством: все 3 главные координатные оси одинаково укорачиваются. Поэтому можно проводить измерения вдоль направления осей с одним и тем же масштабом. Кроме того, главные координатные оси проецируются так, что их проекции составляют равные углы друг с другом (120°).



Изометрическая проекция — это разновидность аксонометрической проекции, при которой в отображении трёхмерного объекта на плоскость коэффициент искажения (отношение длины спроектированного на плоскость отрезка, параллельного координатной оси, к действительной длине отрезка) по всем трём осям один и тот же. Слово «изометрическая» в названии проекции пришло из греческого языка и означает «равный размер», отражая тот факт, что в этой проекции масштабы по всем осям равны. По западным стандартам изометрическая проекция, помимо равенства масштабов по осям, включает условие равенства 120° углов между проекциями любой пары осей.



Диметрические проекции получаются на плоскости аксонометрических проекций в том случае, если она наклонена под одинаковыми углами не к трём главным направлениям, а только к двум. Обычно принимают такое положение плоскости проекций, при котором одинаковые искажения получатся по направлениям длины и высоты проектируемого предмета. Искажение по направлению глубины в этом случае получается вдвое большим, чем по направлению длины и высоты, и равно 0,47 натуральной величины. Коэффициент искажения по направлению длины и высоты равен 0,94. Практически коэффициенты искажения для прямоугольной диметрии принимают 1:0,5:1 и соответствующий им масштаб изображения 1,06:1.
Проекции координатных осей X и Y будут наклонены к горизонтальной прямой, первая на 7° и вторая на 41° (фиг. а). Построение этих же осей можно выполнить упрощённо (фиг. б). Они определяются построением уклонов 1:8 для оси X и 7:8 для оси Y, соответствующим углам 7 и 41°.



Триметрические проекции - нормальный вектор плоскости картинки образует с координатными осями различные углы.

Косоугольные (наклонные) проекции сочетают в себе свойства ортографических проекций (видов спереди, сверху и сбоку) со свойствами аксонометрии. В этом случае проекционная плоскость перпендикулярна главной координатной оси, поэтому сторона объекта, параллельная этой плоскости, проецируется так, что можно измерить углы и расстояния. Проецирование других сторон объекта также допускает проведение линейных измерений (но не угловых) вдоль главных осей. Отметим, что нормаль к проекционной плоскости и направление проецирования не совпадают.

Двумя важными видами косоугольных проекций являются проекции:

- ***Кавалье*** (cavalier) – горизонтальная косоугольная изометрия (военная перспектива);
- ***Кабине*** (cabinet) – фронтальная косоугольная диметрия.

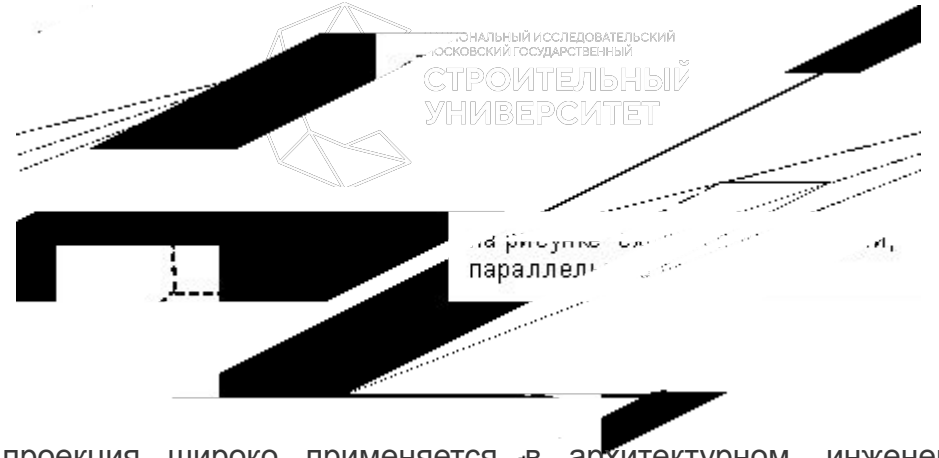
В проекции ***Кавалье*** направление проецирования составляет с плоскостью угол 45°. В результате проекция отрезка, перпендикулярного проекционной плоскости, имеет ту же длину, что и сам отрезок, т. е. укорачивание отсутствует.

Проекция ***Кабине*** имеет направление проецирования, которое составляет с проекционной плоскостью угол α = arctg(½) (≈26,5°). При этом отрезки, перпендикулярные проекционной плоскости, после проецирования составляют ½ их действительной длины. Проекции Кабине являются более реалистическими, чем проекции Кавалье, так как укорачивание с коэффициентом ½ больше согласуется с нашим визуальным опытом.

Центральная проекция любой совокупности параллельных прямых, которые не параллельны проекционной плоскости, будет сходиться в точке схода. Точек схода бесконечно много. Если совокупность прямых параллельна одной из главных координатных осей, то их точка схода называется ***главной точкой схода***. Имеются только три такие точки, соответствующие пересечениям главных координатных осей с проекционной плоскостью. Центральные проекции классифицируются в зависимости от числа главных точек схода, которыми они обладают, а следовательно и от числа координатных осей, которые пересекают проекционную плоскость.

Центральная проекция любой совокупности параллельных прямых, которые не параллельны проекционной плоскости, будет сходиться в точке схода. Точек схода бесконечно много. Если совокупность прямых параллельна одной из главных координатных осей, то их точка схода называется ***главной точкой схода***. Имеются только три такие точки, соответствующие пересечениям главных координатных осей с проекционной плоскостью. Центральные проекции классифицируются в зависимости от числа главных точек схода, которыми они обладают, а следовательно и от числа координатных осей, которые пересекают проекционную плоскость.

1. ***Одноточечная*** проекция



2. Двухточечная проекция широко применяется в архитектурном, инженерном и промышленном проектировании.

3. *Трехточечные* центральные проекции почти совсем не используются, во-первых, потому, что их трудно конструировать, а во-вторых, из-за того, что они добавляют мало нового с точки зрения реалистичности по сравнению с двухточечной проекцией.

Сеточное описание поверхностей

Полигональная сетка — это совокупность вершин, рёбер и граней, которые определяют форму объекта в компьютерной графике и объёмном моделировании. Гранями обычно являются треугольники, четырехугольники или другие простые выпуклые многоугольники (полигоны), так как это упрощает визуализацию, но сетки могут также состоять и из наиболее общих вогнутых многоугольников, или многоугольников с дырками.

Учение о полигональных сетках — это большой подраздел компьютерной графики и геометрического моделирования. Множество операций, проводимых над сетками, может включать булеву алгебру, сглаживание, упрощение и многие другие. Разные представления полигональных сеток используются для разных целей и приложений. Объемные сетки отличаются от полигональных тем, что они явно представляют и поверхность и объём структуры, тогда как полигональные сетки явно представляют лишь поверхность, а не объём.

Объекты созданные с помощью полигональных сеток должны хранить разные типы элементов, такие как вершины, ребра, грани, полигоны и поверхности. Во многих случаях хранятся лишь вершины, ребра и либо грани, либо полигоны.

Вершина — это главная единица описания ПС.

Ребро — это соединение между двумя вершинами.

Грань — это замкнутое множество ребер, в котором треугольная грань имеет три ребра, а четырехугольная - четыре.

Полигон — это множество граней. В системах, которые поддерживают многосторонние грани, полигоны и грани равнозначны. Однако, большинство аппаратного обеспечения для рвизуализации поддерживает лишь грани с тремя или четырьмя сторонам, так что полигоны представлены как множество граней.

Полигональные сетки могут быть представлены множеством способов, используя разные способы хранения вершин, ребер и граней. В них входят:

1. *Список граней:* описание граней происходит с помощью указателей в список вершин.
2. "Крылатое" представление: в нём каждая точка ребра указывает на две вершины, две грани и четыре (по часовой стрелке и против часовой) ребра, которые её касаются. Крылатое представление позволяет обойти поверхность за постоянное время, но у него бóльшие требования по памяти хранения.
3. Полуреберные сетки: способ похож на "крылатое" представление, за исключением того, что используется информация обхода лишь половины грани.
4. Четырехреберные сетки которые хранят ребра, полуребра и вершины без какого-либо указания полигонов. Полигоны прямо не выражены в представлении, и могут быть найдены обходом структуры. Требования по памяти аналогичны полуреберным сеткам.
5. Таблица углов, которые хранят вершины в предопределенной таблице, такой что обход таблицы неявно задает полигоны.

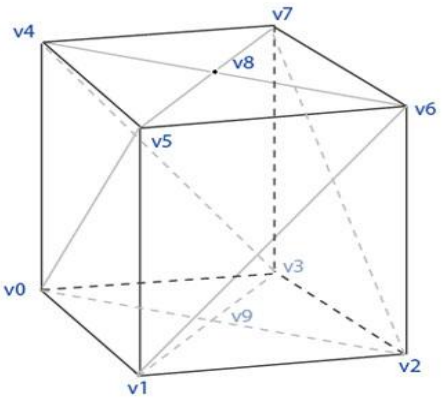
Вершинное (вертексное) представление описывает объект как множество вершин, соединенных с другими вершинами. Это простейшее представление, но оно не широко используемое, так как информация о гранях и ребрах не выражена явно. Поэтому нужно обойти все данные чтобы сгенерировать список граней для визуализации. Кроме того, нелегко выполняются операции на ребрах и гранях.

Однако, сетки ВП извлекают выгоду из малого использования памяти и эффективной трансформации.

Вершинное представление

Список вершин

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3



Сетка с использованием **списка граней** представляет объект как множество граней и множество вершин. Это самое широко используемое представление, будучи входными данными типично принимаемыми современным графическим оборудованием. Список граней лучше для моделирования, чем вершинное представление тем, что он позволяет явный поиск вершин грани, и граней окружающих вершину.

Моделирование требует легкого обхода всех структур. С сеткой использующей список граней очень легко найти вершины грани. Также, список вершин содержит список всех граней связанных с каждой вершиной. В отличие от вершинного представления, и грани и вершины явно представлены, так что нахождение соседних граней и вершин постоянно по времени. Однако, ребра не заданы явно, так что поиск все ещё нужен, чтобы найти все грани, окружающие заданную грань. Другие динамические операции, такие как разрыв или объединение грани, также сложны со списком граней.

Список граней

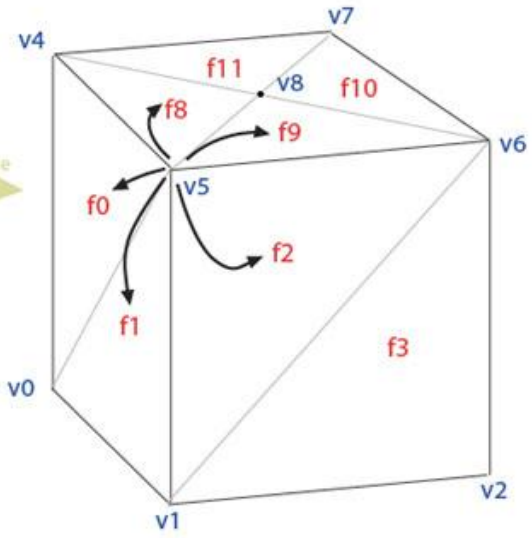
Список граней

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

Список вершин

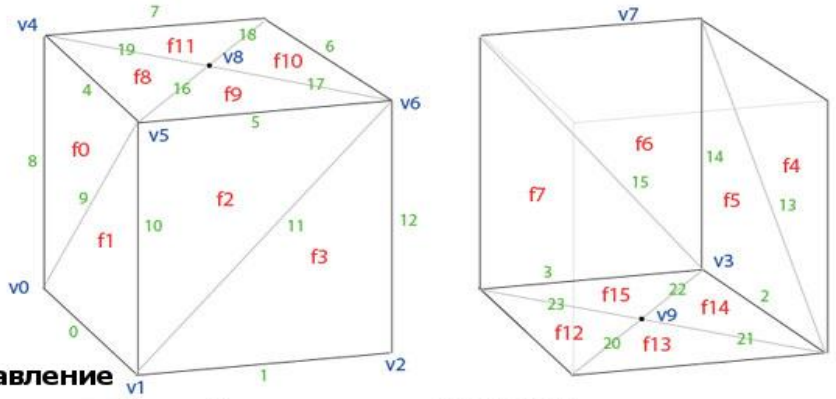
v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 13 14 15

example



"Крылатое" (каркасное) представление явно представляет вершины, грани и ребра сетки. Это представление широко используется в программах для моделирования для предоставления высочайшей гибкости в динамическом изменении геометрии сетки, потому что могут быть быстро выполнены операции разрыва и объединения. Их основной недостаток - высокие требования памяти и увеличенная сложность из-за содержания множества индексов.

"Крылатое" представление решает проблему обхода от ребра к ребру и обеспечивает упорядоченное множество граней вокруг ребра. Для любого заданного ребра число исходящих ребер может быть произвольным. Чтобы упростить это, "крылатое" представление предоставляет лишь четыре, ближайшие ребра по часовой и против часовой стрелки на каждом конце ребра. Другие ребра можно обойти постепенно. Поэтому информация о каждом ребре напоминает бабочку, поэтому представление называется "крылатым".



"Крылатое" представление

Список граней

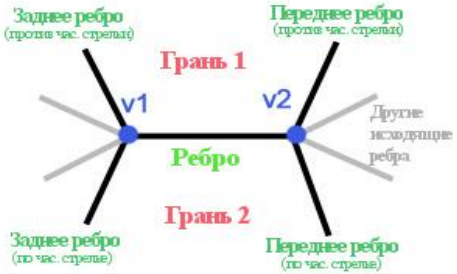
f0	4 8 9
f1	0 10 9
f2	5 10 11
f3	1 12 11
f4	6 12 13
f5	2 14 13
f6	7 14 15
f7	3 8 15
f8	4 16 19
f9	5 17 16
f10	6 18 17
f11	7 19 18
f12	0 23 20
f13	1 20 21
f14	2 21 22
f15	3 22 23

Список ребер

e0	v0 v1	f1 f12	9 23 10 20
e1	v1 v2	f3 f13	11 20 12 21
e2	v2 v3	f5 f14	13 21 14 22
e3	v3 v0	f7 f15	15 22 8 23
e4	v4 v5	f0 f8	19 8 16 9
e5	v5 v6	f2 f9	16 10 17 11
e6	v6 v7	f4 f10	17 12 18 13
e7	v7 v4	f6 f11	18 14 19 15
e8	v0 v4	f7 f0	3 9 7 4
e9	v0 v5	f0 f1	8 0 4 10
e10	v1 v5	f1 f2	0 11 9 5
e11	v1 v6	f2 f3	10 1 5 12
e12	v2 v6	f3 f4	1 13 11 6
e13	v2 v7	f4 f5	12 2 6 14
e14	v3 v7	f5 f6	2 15 13 7
e15	v3 v4	f6 f7	14 3 7 15
e16	v5 v8	f8 f9	4 5 19 17
e17	v6 v8	f9 f10	5 6 16 18
e18	v7 v8	f10 f11	6 7 17 19
e19	v4 v8	f11 f8	7 4 18 16
e20	v1 v9	f12 f13	0 1 23 21
e21	v2 v9	f13 f14	1 2 20 22
e22	v3 v9	f14 f15	2 3 21 23
e23	v0 v9	f15 f12	3 0 22 20

Список вершин

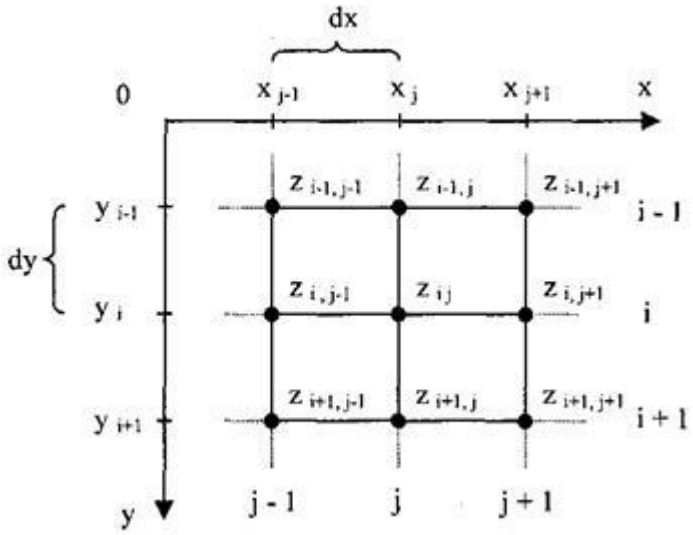
v0	0,0,0	8 9 0 23 3
v1	1,0,0	10 11 1 20 0
v2	1,1,0	12 13 2 21 1
v3	0,1,0	14 15 3 22 2
v4	0,0,1	8 15 7 19 4
v5	1,0,1	10 9 4 16 5
v6	1,1,1	12 11 5 17 6
v7	0,1,1	14 13 6 18 7
v8	.5,.5,0	16 17 18 19
v9	.5,.5,1	20 21 22 23



Структура "крылатого" представления

Равномерная сетка. Эта модель описывает координаты отдельных точек поверхности следующим способом - каждому узлу сетки с индексами (*i*, *f*) приписывается значение высоты *zy*. Индексам (*i*, *y*) отвечают определенные значения координат *(x, y)*. Расстояние между узлами одинаковое – *dx* по оси *x* и *dy* по *оси y*. Фактически, такая модель— двумерный массив, растр, матрица, каждый элемент которой сохраняет значение высоты.

Не каждая поверхность может быть представлена этой моделью. Если в каждом узле записывается только одно значение высоты, то это означает, что поверхность описывается однозначной функцией *z* = *f (x, y)*. Иначе говоря, это такая поверхность, которую любая вертикаль пересекает только один раз. Не могут моделироваться также вертикальные грани. Необходимо заметить, что для сетки не обязательно использовать только декартовые координаты. Например, для того чтобы описать поверхность шара однозначной функцией, можно использовать полярные координаты. Равномерная сетка часто используется для описания рельефа земной поверхности.



Неравномерной сеткой называется модель описания поверхности в виде множества отдельных точек $\{(x_0, y_0, z_0), (x_1, y_1, z_1),\ldots,(x_{n-1}, y_{n-1}, z_{n-1})\}$, принадлежащих поверхности. Эти точки могут быть получены, например, в результате измерений поверхности какого-нибудь объекта с помощью определенного оборудования. Такую модель можно считать обобщением для некоторых рассмотренных выше моделей. Например, векторная полигональная модель и равномерная сетка могут считаться разновидностями неравномерной сетки. Рассмотрим модель поверхности в виде множества точечных значений, логически никак не связанных между собой. Неравномерность задания опорных точек усложняет определение координат для других точек поверхности, которые не совпадают с опорными точками. Требуются специальные методы пространственной интерполяции.

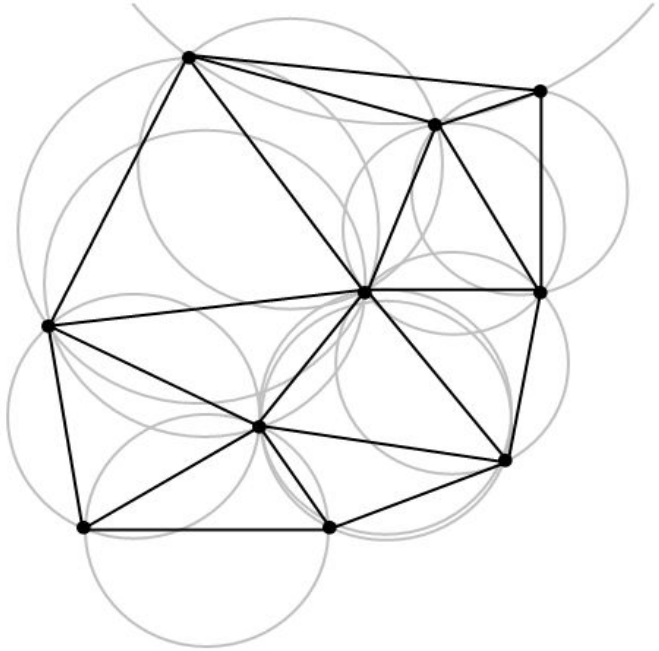
Пусть задача заключается в вычислении значения координаты z по известным координатам (x, y). Для этого необходимо найти несколько самых близких точек, а затем вычислить искомое значение z, исходя из взаимного расположения этих точек в проекции (x, y). Для равномерной сетки эта задача решается достаточно просто – поиска фактически нет, сразу рассчитываются индексы самых близких опорных точек.

Вторая задача заключается в отображении (визуализации) поверхности. Эту задачу можно решать несколькими способами. Один из наиболее распространенных – *триангуляция*. Процесс триангуляции может быть представлен следующим образом:

- находим первые три самые близкие друг к другу точки - получаем одну плоскую треугольную грань;
- находим точку, ближайшую к этой грани, и образовываем смежную грань, и т.д., пока не останется ни одной отдельной точки.

Это – *общая схема триангуляции*. В литературе можно встретить множество алгоритмов триангуляции, сводящихся к описаному выше. Один из наиболее распространенных – триангуляция Делоне.

Описание поверхности треугольными гранями можно уже считать разновидностью векторной полигональной модели. В англоязычной литературе для ее названия используется аббревиатура TIN (Triangulated Irregular Network). После триангуляции получаем полигональную поверхность, отображение которой выполнить уже достаточно просто.



Триангуляцией Делоне для множества точек S на плоскости называют триангуляцию DT(S), такую ч то никакая точка A из S не содержится внутри окружности, описанной вокруг любого треугольника из DT(S), такого, что ни одной из вершин его не является точка A.

Параметрические кубические кривые

Перейдем к более сложному случаю – заданию кривых в трехмерном пространстве. В случае функционального задания кривой возможны многозначности в случае самопересечений и неудобства при значениях производных равных ∞

Ввиду этого будем искать функцию в параметрическом виде. Пусть *t* - независимый параметр, такой что $0 \le t \le 1$

Кубическим параметрическим сплайном назовем следующую систему уравнений:

$$\begin{cases} x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \\ y(t) = a_y t^3 + b_y t^2 + c_y t + d_y \\ z(t) = a_z t^3 + b_z t^2 + c_z t + d_z \end{cases}$$

Координаты точек на кривой описываются вектором $(x(t), y(t), z(t))$

а три производные задают координаты соответствующего касательного вектора в точке. Например, для координаты *x*

$$\frac{dx}{dt} = 3a_x t^2 + 2b_x t + c_x$$

Одним из способов задания параметрического кубического сплайна является указание координат начальной и конечной точек, а также векторов касательных в них. Такой способ задания называется ***кривой Эрмита.*** Обозначим концевые точки P1 и P4, а касательные векторы в них R1 и R4. Индексы выбраны таким образом с учетом дальнейшего изложения. Будем решать задачу нахождения четверки коэффициентов

$$a_x, b_x, c_x, d_x$$

так как для оставшихся двух уравнений коэффициенты находятся аналогично. Запишем условие для построения сплайна:

$$x(0) = P_{1x} \quad x(1) = P_{4x} \quad x'(0) = R_{1x} \quad x'(1) = R_{4x}$$

Перепишем выражение для *x* в векторном виде

$$x(t) = \begin{bmatrix} t^3, t^2, t, 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x$$

Обозначим вектор строку $T = \begin{bmatrix} t^3, t^2, t, 1 \end{bmatrix}$ и вектор столбец коэффициентов

$$C_x = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}_x$$

тогда $x(t) = TC_x$

$$x(0) = P_{1x} = \begin{bmatrix} 0,0,0,1 \end{bmatrix} C_x \qquad x(1) = P_{4x} = \begin{bmatrix} 1,1,1,1 \end{bmatrix} C_x$$

Для касательных $x'(t) = \begin{bmatrix} 3t^2, 2t, 1, 0 \end{bmatrix} C_x \Rightarrow$

$$x'(0) = R_{1x} = \begin{bmatrix} 0,0,1,0 \end{bmatrix} C_x \qquad x'(1) = R_{4x} = \begin{bmatrix} 3,2,1,0 \end{bmatrix} C_x$$

Отсюда получаем векторно-матричное уравнение:

$$\begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} C_x$$

Эта система решается относительно *Cx* нахождением обратной матрицы размером 4x4

$$C_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = M_h G_{hx}$$

M_h - эрмитова матрица

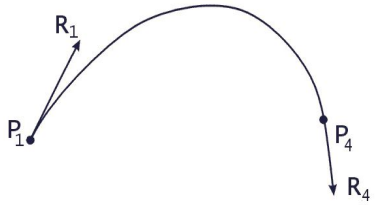
G_h - геометрический вектор Эрмита. Подставим выражение *Cx* для нахождения $x(t)$

$$x(t) = TM_h G_{hx}$$

Аналогично для остальных координат:

$$y(t) = TM_h G_{hy} \qquad z(t) = TM_h G_{hz}$$

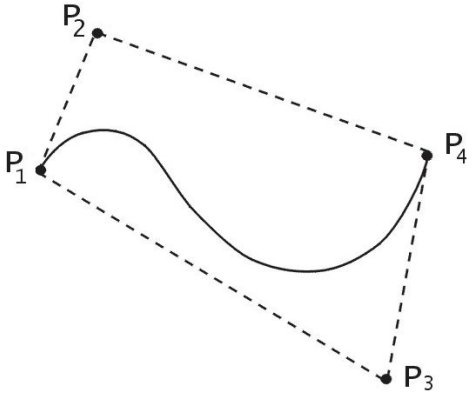
Форму кривой, заданной в форме Эрмита, легко изменять если учитывать, что направление вектора касательной задает начальное направление, а модуль вектора касательной задает степень вытянутости кривой в направлении этого вектора.



Кривая Безье

В начале 70-х годов профессор Пьер Безье, проектируя на компьютере корпуса автомобилей «Рено», впервые применил для этой цели особый вид кривых, описываемых уравнением третьего порядка, которые впоследствии стали известными под названием *кривые Безье* (функция Bezier).

Кривые Безье описываются в параметрической форме: $x = P_x(t)$, $y = P_y(t)$,



Многочлены Безье для P_x и P_y имеют такой вид:

$$P_x(t) = \sum_{i=0}^{m} c_m^i t^i (1-t)^{m-i} x_i, \text{ и } P_y(t) = \sum_{i=0}^{m} c_m^i t^i (1-t)^{m-i} y_i$$

где x_i и y_i - координаты точек-ориентиров P_i, а величины

c_m^i - это известные из *комбинаторики,* так называемые *сочетания* (они также известны как коэффициенты *бинома Ньютона):*

$$c_m^i = \frac{m!}{i!(m-i)!}$$

Значение *m* можно рассматривать и как степень полинома, и как значение, которое на единицу меньше количества точек-ориентиров.

Рассмотрим кривые Безье, классифицируя их по значениям *m.*

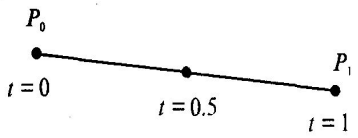
m = 1 (по двум точкам)

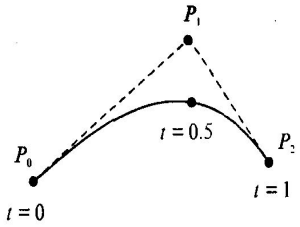
Кривая вырождается в отрезок прямой линии, которая определяется конечными точками P_o и P_1,

$$P(t) = (1-t)\, P_0 + t_1$$

m=2 (по трем точкам):

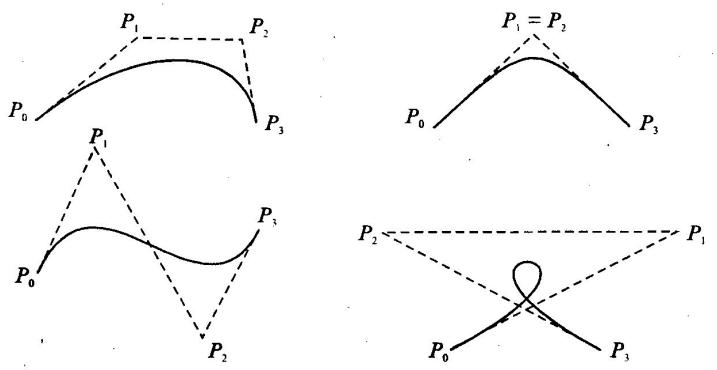
$$P(t) = (1-t)^2\, P_0 + 2t\,(1-t)\, P_1 + t^2 P_2$$





m = 3 (по четырем точкам, кубическая). Используется довольно часто, в особенности в сплайновых кривых:

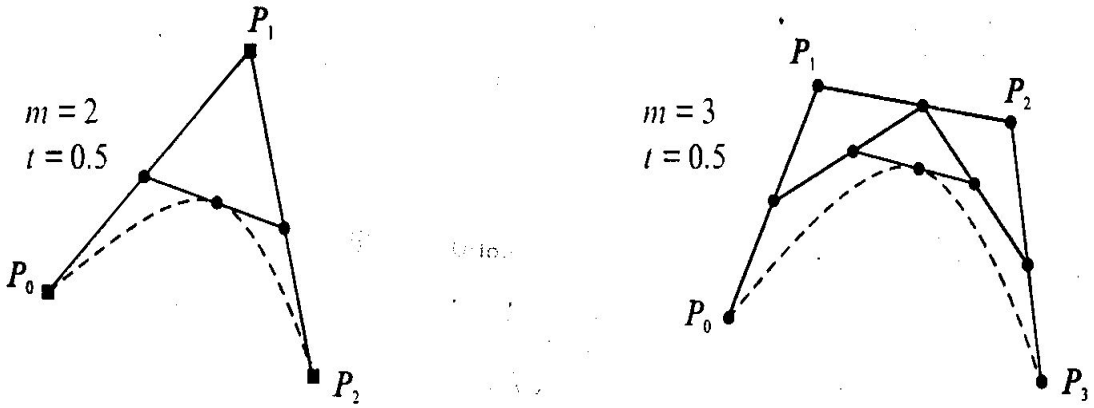
$$P(t) = (1-t)^3 P_0 + 3t\,(1-t)^2\,P_1 + 3\,t^2(1-t)P_2 + t^3 P_3$$



Геометрический алгоритм для кривой Безье

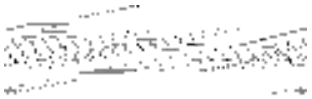
Этот алгоритм позволяет вычислить координаты (*x*, *y*) точки кривой Безье по значению параметра *t*.

1. Каждая сторона контура многоугольника, который проходит по точкам -ориентирам, делится пропорционально значению *t*.
2. Точки деления соединяются отрезками прямых и образуют новый многоугольник. Количество узлов нового контура на единицу меньше, чем количество узлов предшествующего контура.
3. Стороны нового контура снова делятся пропорционально значению *t*. И так далее. Это продолжается до тех пор, пока не будет получена единственная точка деления. Эта точка и будет точкой кривой Безье.



B-сплайны

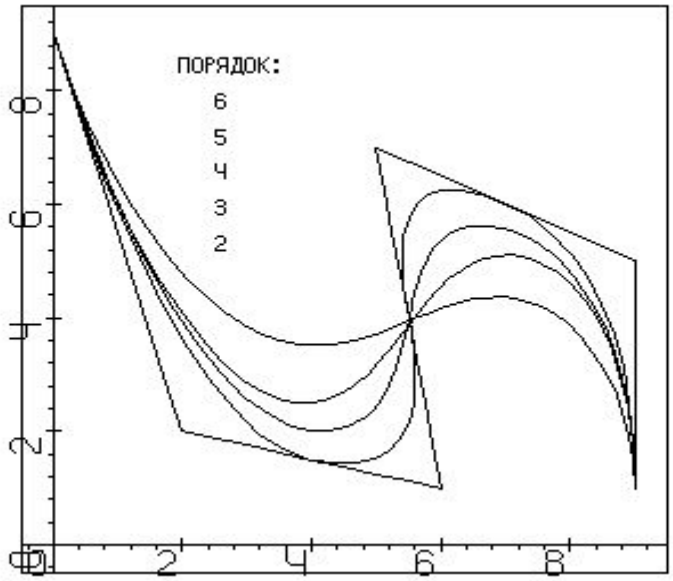
Кривая, построенная на основе *B-сплайн*-базиса, описывается следующим образом



где  - радиус-вектор точек на кривой,

 - вершины аппроксимируемой ломаной (всего вершин $n+1$), а $N_{ik}(t)$ - весовая функция i-й нормализованной *B-сплайн* базисной кривой порядка k (т. е. степени $k-1$), задаваемая рекуррентными соотношениями:

Здесь x_i – элементы *узлового вектора*, а t – параметр, изменяющийся в диапазоне от 0 до $t_{max}=(n-k+2)$.



Узловой вектор, длина которого $(n+k+1)$, вводится для учета собственной кривизны *B-сплайн*-кривых и представляет собой неубывающую последовательность целых чисел - *параметрических узлов*. Узловой вектор определяется числом точек в аппроксимируемой ломаной, порядком кривой, а также наличием сложных (кратных) узлов.

B-сплайн-кривая является полиномом степени (*k*–1) на каждом интервале (x_i, x_i+1) и что все ее производные до (*k*–2)-го порядка включительно непрерывны вдоль всей кривой. То есть эта кривая представляет собой сплайн-функцию порядка *k* (степени *k*–1).

Параметрические кубические поверхности

Существуют три широко используемых способа представления поверхностей:

- 1) с помощью функций;
- 2) в параметрическом виде;
- 3) в полигональном виде.

Из класса параметрических поверхностей рассмотрим бикубические поверхности в форме Эрмита, Безье и 5-сплайнов.

Бикубические поверхности задаются кубическими уравнениями от двух переменных *s u t.* Изменяя оба параметра от 0 до 1, можно определить все точки на куске поверхности. Если одному из параметров присвоить постоянное значение, а другой - изменять в диапазоне от 0 до 1, то в результате получим кубическую кривую. Для удобства мы будем рассматривать только уравнение для *x:*

$$x(s,t) = A_{11}s^3t^3 + A_{12}s^3t^2 + A_{13}s^3t + A_{14}s^3 + A_{21}s^2t^3 + A_{22}s^2t^2 + A_{23}s^2t + A_{24}s^2 + \\ + A_{31}st^3 + A_{32}st^2 + A_{33}st + A_{34}s + A_{41}t^3 + A_{42}t^2 + A_{43}t + A_{44}.$$

Запишем в более удобной форме

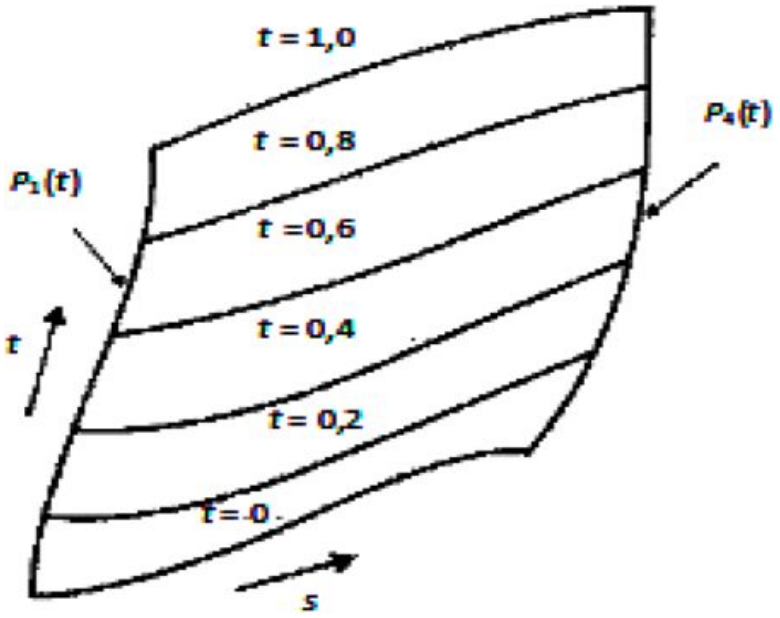
$x(s, t) = SC_XT^T$,

где $S = [s^3\ s^2\ s\ 1]$; $T=[t^3t^2t\ 1]$, а T^m - транспонированная матрица *T.*

Такая запись поверхности называется алгебраической формой представления, так как C_x задаёт коэффициенты бикубического многочлена. Существуют также и C_y и Cz, которые определяют коэффициенты *y(s, t)* и *z(s, t).*

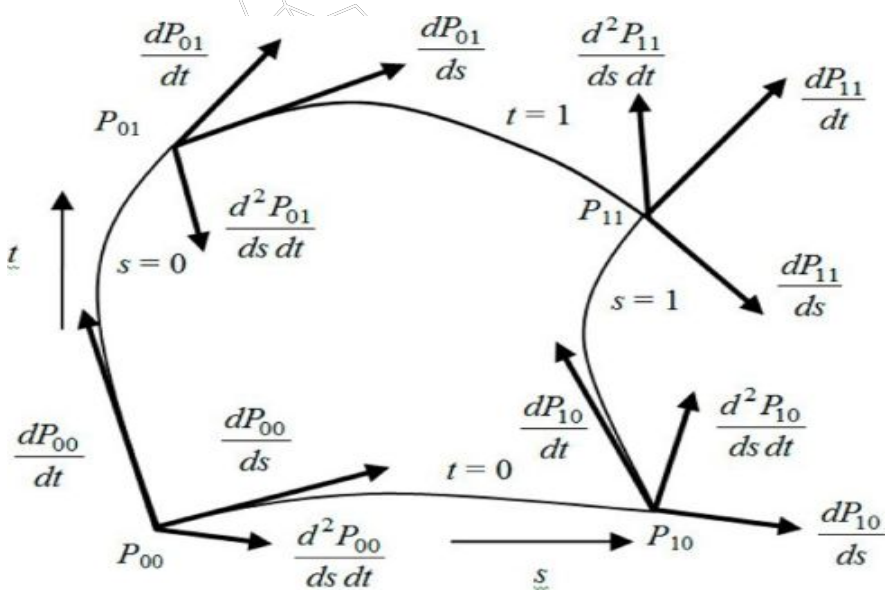
Форма Эрмита

Поверхности в форме Эрмита используют в качестве исходных данных управляющие точки и касательные векторы.



$$\begin{vmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{vmatrix}$$

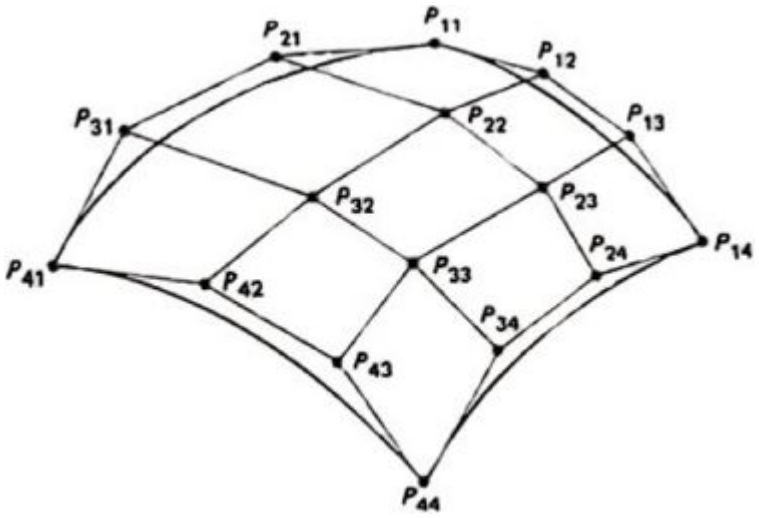
В верхнем левом углу матрицы размером 2x2 находятся четыре координаты углов фрагмента поверхности, в верхней правой и нижней левой частях матрицы размещены тангенсы углов наклона касательных векторов в угловых точках для каждой из граничных параметрических кривых. В нижнем углу матрицы расположены частные производные по обоим параметрам в угловых точках. Это определяет кривизну, так как чем больше их значения, тем сильнее изгиб в угловой точке фрагмента поверхности.

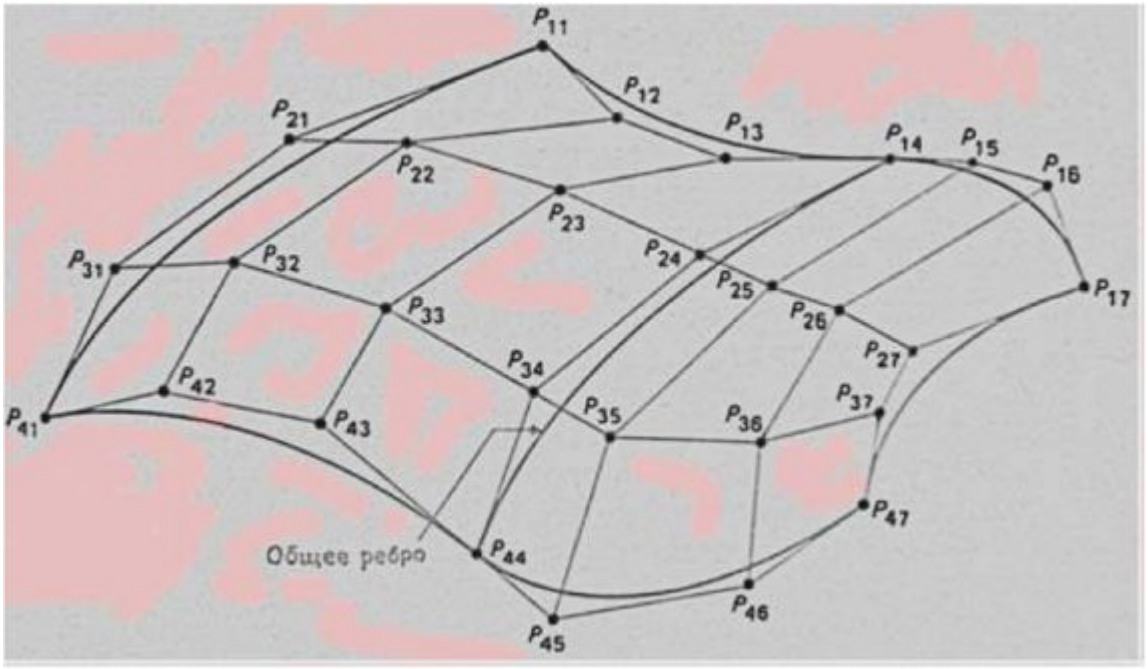


Поверхность Безье

Поверхности Безье используются часто при интерактивном проектировании по тем же причинам, что и кривые Безье: управляющие точки позволяют легко изменять форму куска поверхности. Поверхности Безье, так же как и кривые Безье, обладают свойством выпуклой оболочки.

Для достижения непрерывности в поперечном направлении относительно рёбер кусков необходимо равенство четырёх управляющих точек, принадлежащих общим рёбрам соседних кусков. Для непрерывности касательного вектора требуется, чтобы две четвёрки управляющих точек по обеим сторонам общего ребра были коллинеарные другим точкам ребра.

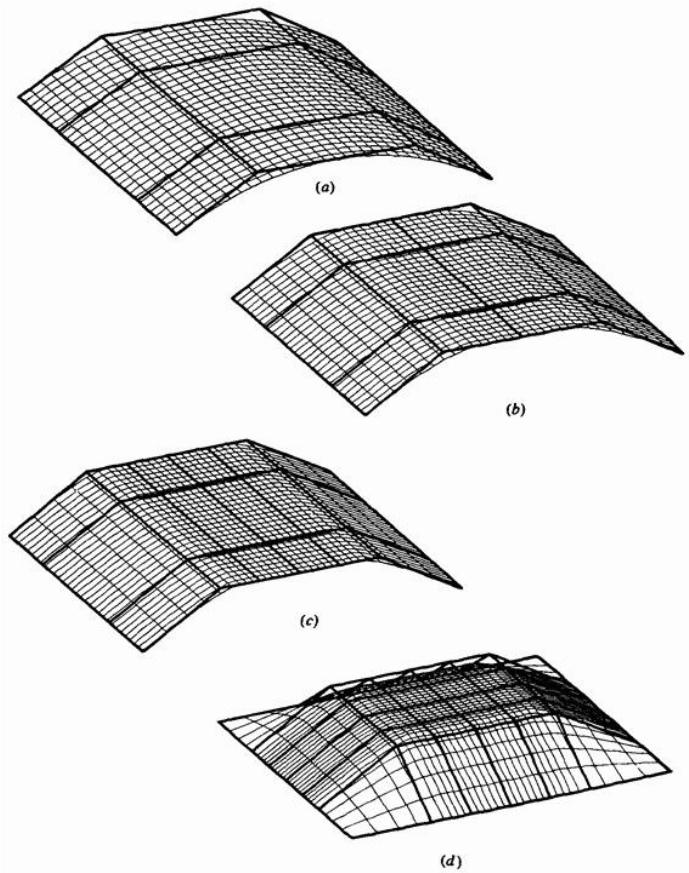




Поверхности в форме B-сплайнов

Как и для кривых в форме B-сплайнов, должна достигаться непрерывность куска. Матрица, состоящая из 16 управляющих точек, описывает кусок, а также в общем случае и точки, не лежащие на самом куске. Куски в форме B-сплайнов на практике «мягко» сшиваются между собой при моделировании сложных 3D-объектов.

B-сплайн поверхность может содержать плоские области и линии резкого нарушения гладкости. Это свойство очень полезно во многих ситуациях, возникающих при конструировании. На рисунке изображена серия незамкнутых B-сплайн поверхностей и их характеристических многогранников третьего порядка в каждом характеристическом направлении. Каждая из линий задающей полигональной сетки в направлении *i* является прямой линией с четырьмя вершинами

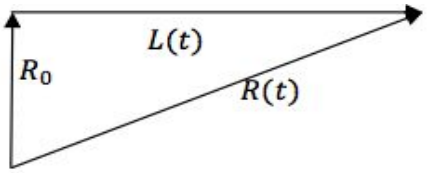


3.2. Алгоритмы пересечения

Алгоритмы определения взаимных пересечений графических объектов.

Алгоритм создания пересечения простого луча и сферы.

При создании любых алгебраических пересечений описываемые объекты обязательно переводятся в параметрическую форму, удобную для формирования геометрических выводов по геометрическим пересечениям объектов.



$$R(t) = R_0 + L(t), t > 0$$

$$\begin{cases} x = x_0 + lt \\ y = y_0 + mt \\ z = z_0 + nt \end{cases}$$

Сфера: $r, \{x_c; y_c; z_c\}$

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = r^2$$

$$At^2 + Bt + C = 0$$

$$A = x_c^2 + y_c^2 + z_c^2$$

$$B = l(x_0 - x_c) + m(y_0 - y_c) + n(z_0 - z_c)$$

$$C = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2$$

$t_{1,2}^* < 0$ не имеет точек

$t_i^* > 0$ имеет одну точку

$t_{1,2}^* > 0$ прошивает сферу

Алгоритм создания пересечения простого луча и сферы.

$$Ax + By + Cz + D = 0$$

$$t^* = \frac{-(Ax_0+By_0+Cz_0+D)}{Al+Bm+Cn} = \frac{-(NR_0+D)}{NL}$$

$$N = (A; B; C)$$

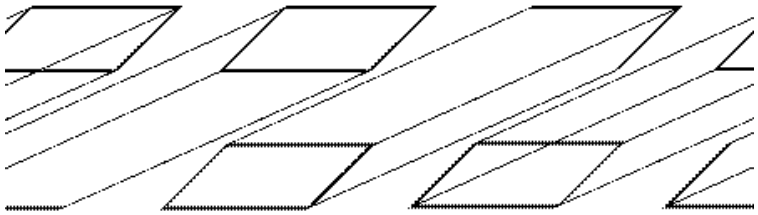
1) N*L=0 (луч параллелен)

2) N*L≠0; $t^* < 0$ (луч не пересекает)

3) N*L≠0; $t^* > 0$ (точка пересечения)

3.3. Алгоритмы упорядочения. Удаление скрытых ребер и поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства



Сложность задачи удаления невидимых линий и поверхностей привела к появлению большого числа различных способов ее решения. Многие из них ориентированы на специализированные приложения. Наилучшего решения обшей задачи удаления невидимых линий и поверхностей не существует. Для моделирования процессов в реальном времени, например для авиатренажеров, требуются быстрые алгоритмы, которые могут порождать результаты с частотой видеогенерации 30 кадр/с. Для машинной мультипликации, например, требуются алгоритмы, которые могут генерировать сложные реалистические изображения, в которых представлены тени, прозрачность и фактура, учитывающие эффекты отражения и преломления цвета в мельчайших оттенках. Подобные алгоритмы работают медленно, и зачастую на вычисления требуется несколько минут или даже часов. Строго говоря, учет эффектов прозрачности, фактуры, отражения и т. п. не входит в задачу удаления невидимых линий или поверхностей. Естественнее считать их частью процесса визуализации изображения. Однако многие из этих эффектов встроены в алгоритмы удаления невидимых поверхностей. Существует тесная взаимосвязь между скоростью работы алгоритма и детальностью его результата. Ни один из алгоритмов не может достигнуть хороших оценок для этих двух показателей одновременно. По мере создания все более быстрых алгоритмов можно строить все более детальные изображения. Реальные задачи, однако, всегда будут требовать учета еще большего количества деталей.

Все алгоритмы удаления невидимых линий (поверхностей) включают в себя сортировку. Порядок, в котором производится сортировка координат объектов, вообще говоря, не влияет на эффективность этих алгоритмов. Главная сортировка ведется по геометрическому расстоянию от тела, поверхности, ребра или точки до точки наблюдения. Основная идея, положенная в основу сортировки по расстоянию, заключается в том, что чем дальше расположен объект от точки наблюдения, тем больше вероятность, что он будет полностью или частично заслонен одним из объектов, более близких к точке наблюдения.

Алгоритмы удаления невидимых линий или поверхностей можно классифицировать по способу выбора системы координат или пространства, в котором они работают. Выделяют три класса алгоритмов удаления невидимых линий или поверхностей:

- Алгоритмы, работающие в объектном пространстве.
- Алгоритмы, работающие в пространстве изображения (экрана).
- Алгоритмы, формирующие список приоритетов.

Алгоритмы, работающие в объектном пространстве, имеют дело с физической системой координат, в которой описаны эти объекты. При этом получаются весьма точные результаты, ограниченные лишь точностью вычислений. Полученные изображения можно свободно увеличивать во много раз. Алгоритмы, работающие в объектном пространстве, особенно полезны в тех приложениях, где необходима высокая точность.

Алгоритмы же, работающие в пространстве изображения, имеют дело с системой координат того экрана, на котором объекты визуализируются. При этом точность вычислений ограничена разрешающей способностью экрана. Обычно разрешение экрана бывает довольно низким, типичный пример: 512´512 точек. Результаты, полученные в пространстве изображения, а затем увеличенные во много раз, не будут соответствовать исходной сцене. Например, могут не совпасть концы отрезков. Алгоритмы, формирующие список приоритетов, работают попеременно в обеих упомянутых системах координат.

Объем вычислений для любого алгоритма, работающего в объектном пространстве и сравнивающего каждый объект сцены со всеми остальными объектами этой сцены, растет теоретически, как квадрат числа объектов (n^2). Аналогично, объем вычислений любого алгоритма, работающего в пространстве изображения и сравнивающего каждый объект сцены с позициями всех пикселов в системе координат экрана, растет теоретически, как nN. Здесь n обозначает количество объектов (тел, плоскостей или ребер) в сцене, а N — число пикселов. Теоретически трудоемкость алгоритмов, работающих в объектном пространстве, меньше трудоемкости алгоритмов, работающих в пространстве изображения, при $n < N$. Однако на практике это не так. Дело в том, что алгоритмы, работающие в пространстве изображения, более эффективны потому, что для них легче воспользоваться преимуществом когерентности при растровой реализации

Алгоритм плавающего горизонта

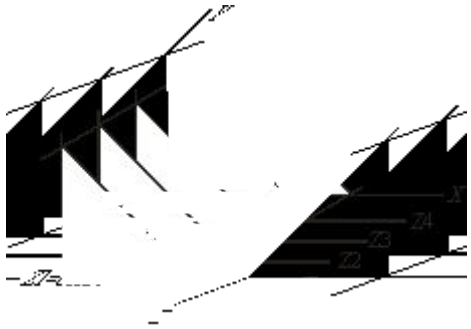
Алгоритм плавающего горизонта можно отнести к классу алгоритмов, работающих в пространстве изображения. Алгоритм плавающего горизонта чаше всего используется для удаления невидимых линий трехмерного представления функций, описывающих поверхность в виде

$$F(x, y, z) = 0.$$

Подобные функции возникают во многих приложениях в математике, технике, естественных науках и других дисциплинах.

Главная идея данного метода заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат *x*, *y* или *z*.

На рисунке приведен пример, где указанные параллельные плоскости определяются постоянными значениями *z*. Функция $F(x,y,z) = 0$ сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, например к последовательности $y=f(x,z)$ или $x=g(y,z)$, где *z* постоянно на каждой из заданных параллельных плоскостей.



Поверхность складывается из последовательности кривых, лежащих в каждой из этих плоскостей

Предполагается, что полученные кривые являются однозначными функциями независимых переменных. Если спроецировать полученные кривые на плоскость $z = 0$, как показано на рисунке, то сразу становится ясна идея алгоритма удаления невидимых участков исходной поверхности

Алгоритм сначала упорядочивает плоскости $z = \text{const}$ по возрастанию расстояния до них от точки наблюдения. Затем для каждой плоскости, начиная с ближайшей к точке наблюдения, строится кривая, лежащая на ней, т. е. для каждого значения координаты x в пространстве изображения определяется соответствующее значение y. Алгоритм удаления невидимой линии заключается в следующем.

Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше значения y для всех предыдущих кривых при этом значении x, то текущая кривая видима в этой точке; в противном случае она невидима.

Невидимые участки показаны пунктиром. Реализация данного алгоритма достаточно проста. Для хранения максимальных значений y при каждом значении x используется массив, длина которого равна числу различимых точек (разрешению) по оси x в пространстве изображения. Значения, хранящиеся в этом массиве, представляют собой текущие значения "горизонта". Поэтому по мере рисования каждой очередной кривой этот горизонт "всплывает". Фактически этот алгоритм удаления невидимых линий работает каждый раз с одной линией.

Алгоритм Робертса

Алгоритм Робертса представляет собой первое известное решение задачи об удалении невидимых линий. Алгоритм прежде всего удаляет из каждого тела те ребра или грани, которые экранируются самим телом. Затем каждое из видимых ребер каждого тела сравнивается с каждым из оставшихся тел для определения того, какая его часть или части, если таковые есть, экранируются этими телами. Поэтому вычислительная трудоемкость алгоритма Робертса растет теоретически, как квадрат числа объектов. Это в сочетании с ростом интереса к растровым дисплеям, работающим в пространстве изображения, привело к снижению интереса к алгоритму Робертса. Однако математические методы, используемые в этом алгоритме, просты, мощны и точны. Кроме того, этот алгоритм можно использовать для иллюстрации некоторых важных концепций.

Работа Алгоритм Робертса проходит в два этапа:

1. Определение нелицевых граней для каждого тела отдельно.
2. Определение и удаление невидимых ребер.

Определение нелицевых граней

Пусть F — некоторая грань многогранника. Плоскость, несущая эту грань, разделяет пространство на два подпространства. Назовем положительным то из них, в которое смотрит внешняя нормаль к грани. Если точка наблюдения – в положительном подпространстве, то грань – ***лицевая***, в противном случае – ***нелицевая***. Если многогранник выпуклый, то удаление всех нелицевых граней полностью решает задачу визуализации с удалением невидимых граней.

Для определения, лежит ли точка в положительном подпространстве, используют проверку знака скалярного произведения (l, n), где l – вектор, направленный к наблюдателю, фактически определяет точку наблюдения; n – вектор внешней нормали грани. Если $(l, n) > 0$, т. е. угол между векторами острый, то грань является лицевой. Если $(l, n) < 0$, т. е. угол между векторами тупой, то грань является нелицевой.

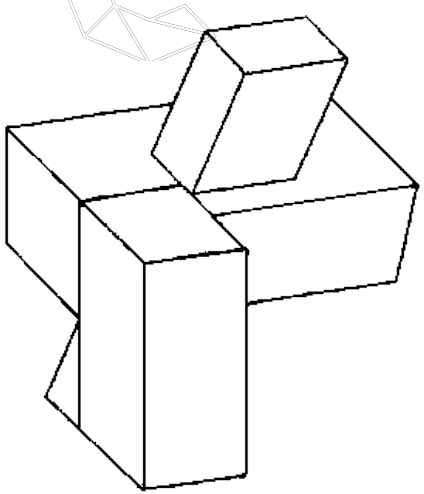
В алгоритме Робертса требуется, чтобы все изображаемые тела или объекты были выпуклыми. Невыпуклые тела должны быть разбиты на выпуклые части. В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представиться набором пересекающихся плоскостей.

Удаление невидимых ребер

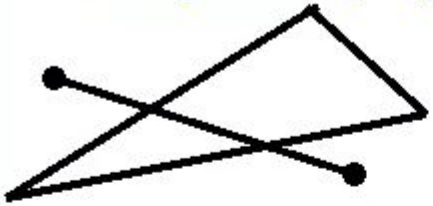
После первого этапа удаления нелицевых отрезков необходимо выяснить, существуют ли такие отрезки, которые экранируются другими телами в картинке или в сцене. Для этого каждый оставшийся отрезок или ребро нужно сравнить с другими телами сцены или картинки.

Возможны следующие случаи:

- Грань ребра не закрывает. Ребро остается в списке ребер.
- Грань полностью закрывает ребро. Ребро удаляется из списка рассматриваемых ребер.
- Грань частично закрывает ребро. В этом случае ребро разбивается на несколько частей, видимыми из которых являются не более двух. Само ребро удаляется из списка рассматриваемых ребер, но в список проверяемых ребер добавляются те его части, которые данной гранью не закрываются.



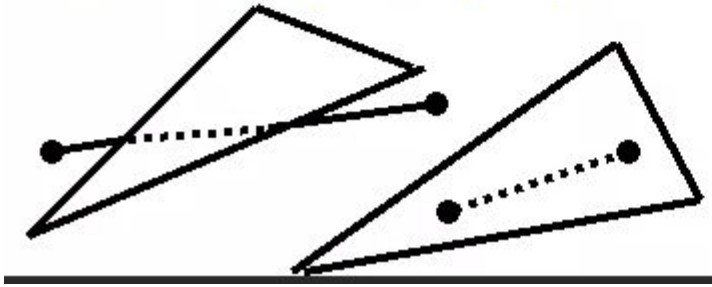
Грань не закрывает ребро:



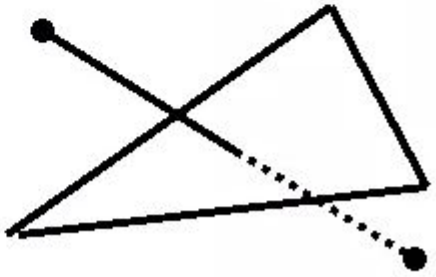
ребро находится в том же полупространстве относительно грани, что и наблюдатель

ребро лежит в полупространстве, не содержащем наблюдателя, проекции грани и ребра не пересекаются, проекция ребра не лежит внутри проекции грани

Грань полностью или частично закрывает ребро:



Грань и ребро пересекаются

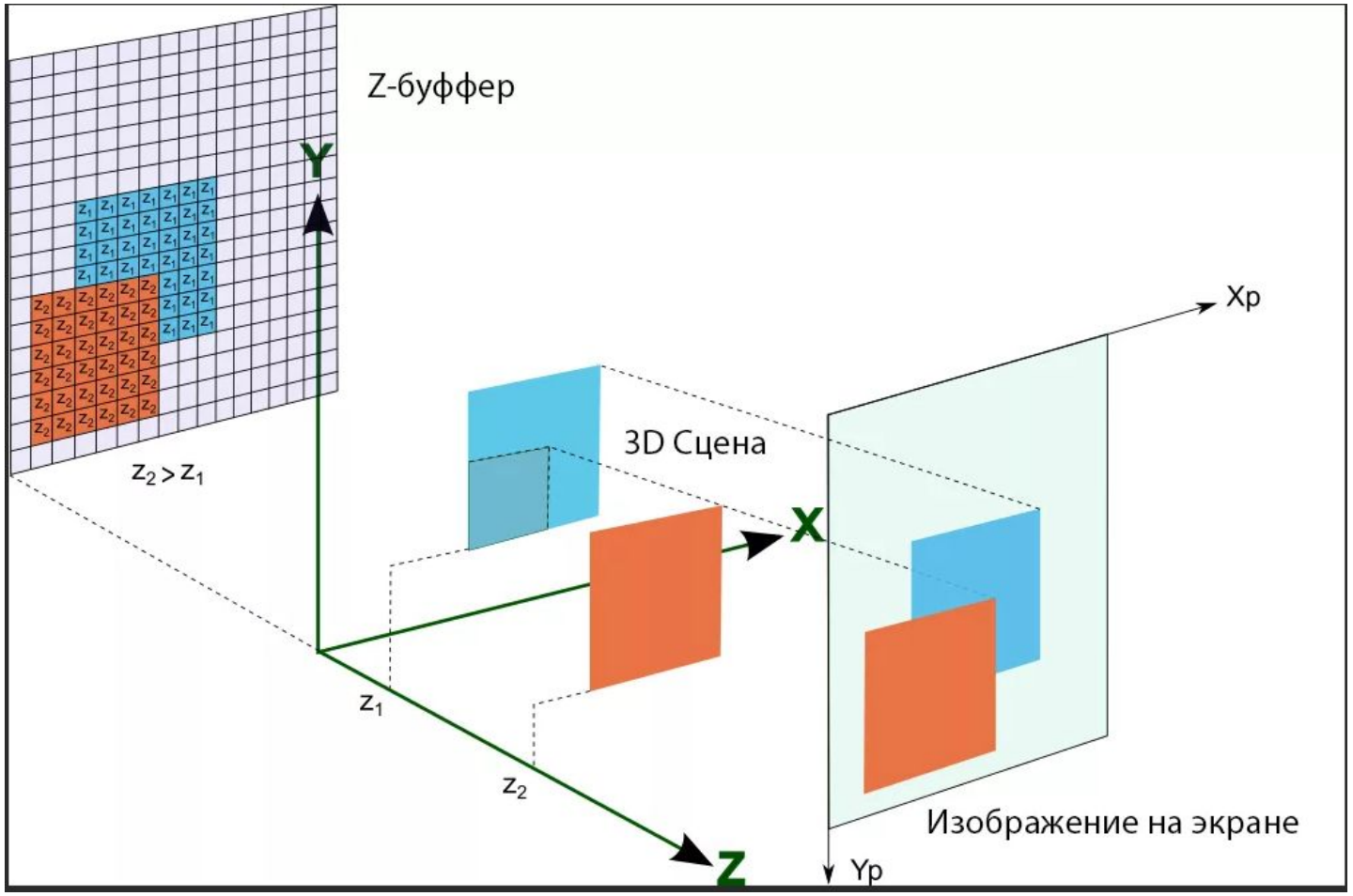


Алгоритм z–буфера

Алгоритм, использующий *z-буфер* - это один из простейших алгоритмов удаления невидимых поверхностей. Работает этот алгоритм в пространстве изображения. Идея *z*-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов (интенсивности) каждого пиксела в пространстве изображения. z-буфер - это отдельный буфер глубины, используемый для запоминания координаты *z* или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение *z* каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесен в *z*-буфер. Если это сравнение показывает, что новый пиксел расположен впереди пиксела, находящегося в буфере кадра, то новый пиксел заносится в этот буфер и, кроме того, производится корректировка *z*-буфера новым значением *z*. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по *x* и *y* наибольшего значения функции *z (x, y).*

Главное преимущество алгоритма – его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в *z*-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. Если сцена подвергается видовому преобразованию и отсекается до фиксированного диапазона значений координат *z*, то можно использовать *z*-буфер с фиксированной точностью.


Z-буффер
Y
$z_2 > z_1$
3D Сцена
X
Xp
z_1
z_2
Z
Изображение на экране
Yp

Другой недостаток алгоритма *z*-буфера состоит в трудоемкости и высокой стоимости устранения лестничного эффекта, а также реализации эффектов прозрачности и просвечивания. Поскольку алгоритм заносит пикселы в буфер кадра в произвольном порядке, то нелегко получить информацию, необходимую для методов устранения лестничного эффекта, основывающихся на предварительной фильтрации. При реализации эффектов прозрачности и просвечивания пикселы могут заноситься в буфер кадра в некорректном порядке, что ведет к локальным ошибкам.

Алгоритм Аппеля.

В алгоритме вводится понятие *количественной невидимости* (quontative invisibility) точки как число лицевых граней, ее закрывающих.

Контурная линия полигонального объекта состоит из тех ребер, для которых одна из проходящих граней является лицевой, а другая - нелицевой.

Так, для многогранника на рисунке контурной линией является ломаная ABCIJDEKLGA.

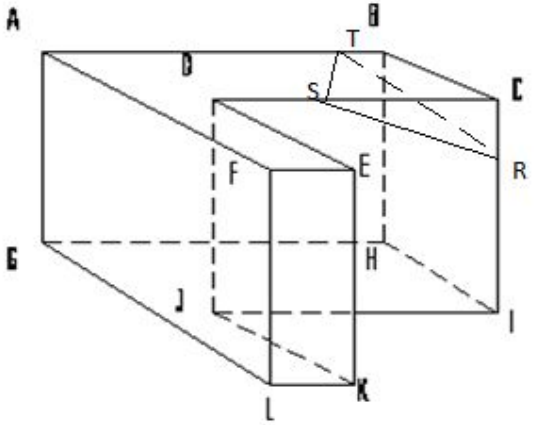
Рассмотрим, как меняется количественная невидимость вдоль ребра.

Для определения видимости ребер произвольного многогранника сначала берется какая-либо его вершина и ее количественная невидимость определяется непосредственно.

Далее прослеживается изменение количественной невидимости вдоль каждого из ребер, выходящих из этой вершины.

Эти ребра проверяются на прохождение позади контурной линии, и их количественная невидимость в соответствующих точках изменяется. При прохождении ребра позади контурной линии количественная невидимость точек ребра изменяется на единицу. Те части отрезка, для которых количественная невидимость равна нулю, сразу же рисуются.

Следующим шагом является определение количественной невидимости для ребер, выходящих из новой вершины, и т. д.

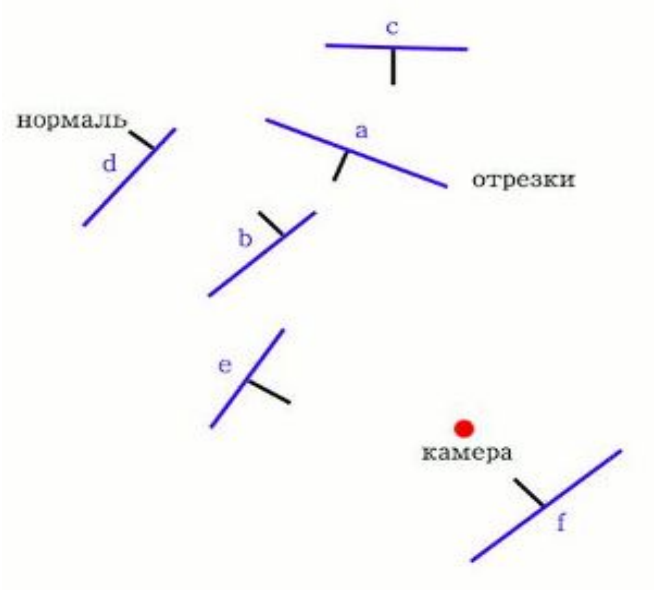


В результате определяется количественная невидимость всех ребер связной компоненты сцены, содержащей исходную вершину (и при этом видимые части ребер этой компоненты сразу же рисуются).
В случае, когда рассматривается изменение количественной невидимости вдоль ребра, выходящего из вершины, принадлежащей контурной линии, необходимо проверить, не закрывается ли это ребро одной из граней, выходящей из этой вершины (как, например, грань DEKJ закрывает ребро DJ, и это является аналогом точки сборки).

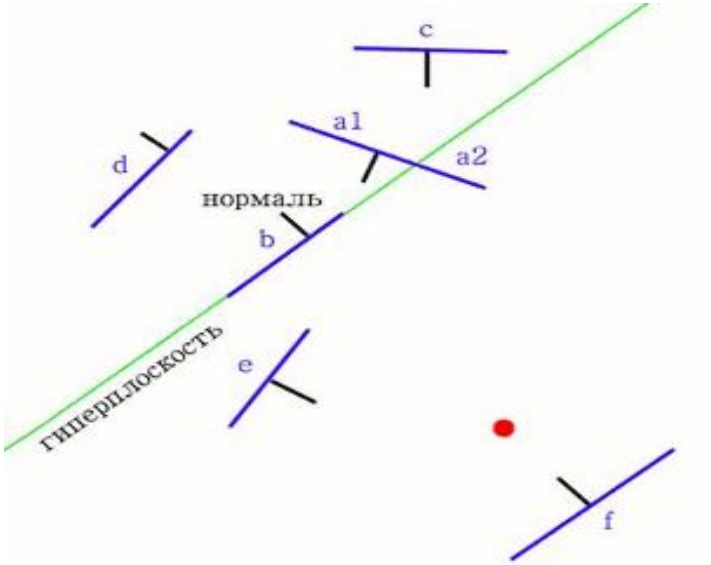
Алгоритм двоичного разбиения пространства

Скорость работы метода двоичного разбиения пространства достигается за счёт разбиения исходного пространства и проведения предварительных вычислений. На вход метода поступает набор некоторых объектов (в случае сортировки полигонов в сцене этими объектами являются полигоны), а потом с помощью рекурсивного алгоритма создаётся двоичное дерево таким образом, что можно совершать обход дерева в порядке от более удалённых к менее удалённым или от менее удалённых к более удалённым многоугольникам.

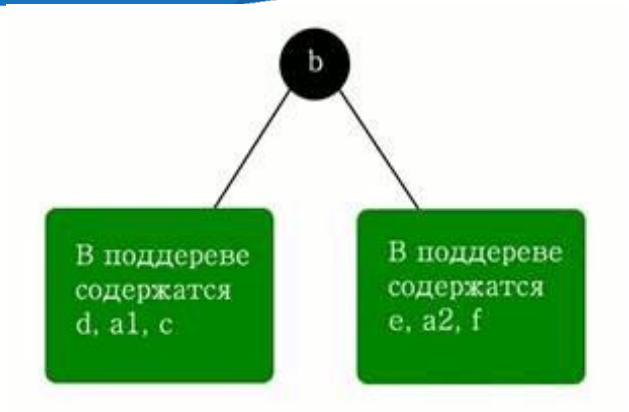
Принцип работы алгоритма заключается в том, что все полигоны пространства (в общем случае n-мерного) разбиваются на группы, лежащие в разных выпуклых подпространствах относительно некоторой гиперплоскости (гиперплоскость - это пространство размерности n-1). Не нарушая общности рассуждений будем рассматривать 2-хмерное пространство. В этом случае гиперплоскость будет представлять собой прямую линию. Для наглядности рассмотрим пример



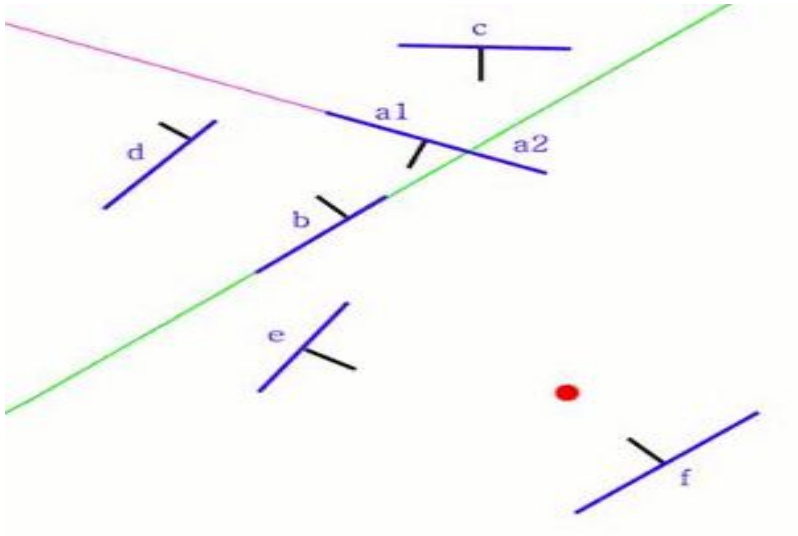
У нас есть плоскость (пространство), на которой расположены отрезки (участки гиперплоскостей) с заданными нормалями. Первым этапом алгоритма является определение разделяющей плоскость прямой.

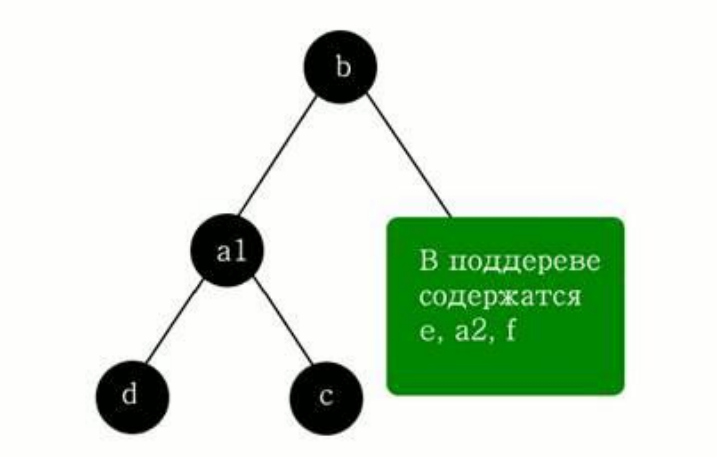


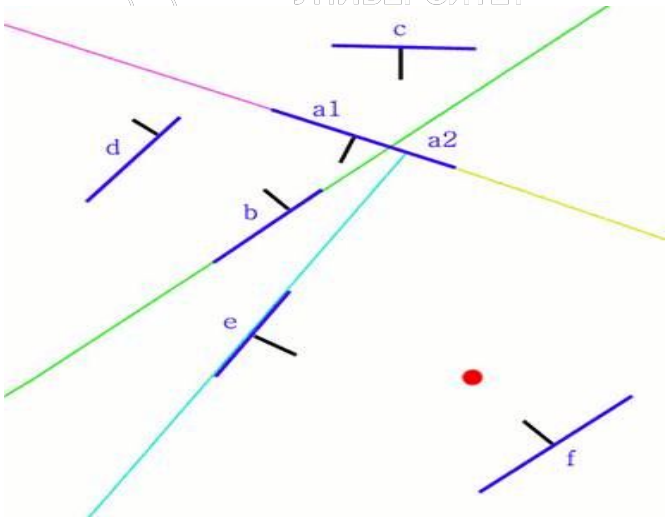
Благодаря прямой, мы получили плоскость, разбитую на две полуплоскости. Нормаль к прямой совпадает с нормалью отрезка, через который она проведена. По направлению нормали мы будем определять переднюю и заднюю полуплоскости: если нормаль находится в полуплоскости, то данная полуплоскость - передняя; иначе - задняя. Теперь нужно определить, каким полуплоскостям принадлежат отрезки. Таким образом все отрезки разбиваются на три группы: отрезки, лежащие в передней полуплоскости ("c" и "d"), отрезки, лежащие в задней полуплоскости ("e" и "f"), и отрезки, лежащие на прямой (только "b"). Если отрезок принадлежит обоим полуплоскостям, то он делится на два (так "a" делится на "a1" и "a2"). Если узлу двоичного дерева приписать прямую и все отрезки, лежащие на ней, а две оставшиеся группы приписать его дочерним поддеревьям, то получим образование следующей структуры



Теперь нужно рекурсивно повторить алгоритм для каждого поддерева. То есть, мы имеем в качестве пространства переднее полуподпространство (в него входят отрезки "d", "a1", "c"). Выбираем любой из этих отрезков, например, "a1" и проводим через него гиперплоскость: получаем в качестве переднего поддерева отрезок "d", а заднего - "c"





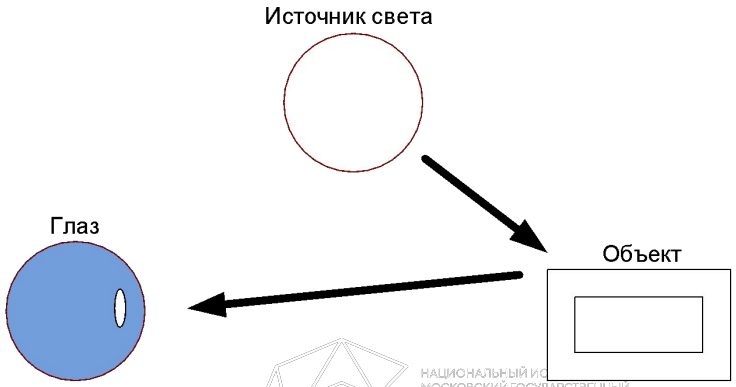


Зная исходное положение камеры (наблюдателя), нужно обойти все дерево по полигонам от самого дальнего до самого ближнего к камере. Начинается обход дерева, естественно, с его корня. Порядок обхода каждого узла задается положением камеры относительно прямой, соответствующей данному узлу, следующими правилами:

- Если камера находится в передней полуплоскости относительно прямой, соответствующей данному узлу, то обходим сначала заднее поддерево, потом все полигоны, которые находятся в данном узле, и в последнюю очередь переднее поддерево.
- Наоборот, если камера в задней полуплоскости, то обходим узел в порядке от переднего поддерева к заднему.
- Если же камера находится на данной прямой, то сначала обходятся поддеревья в любом порядке, а полигоны самого узла не обходятся вовсе (т.к. они, фактически, не видны наблюдателю) или обходятся в последнюю очередь, упорядоченные некоторым образом (например, по расстоянию от дальнего ближнему).

3.4. Цветовые модели в компьютерной графике

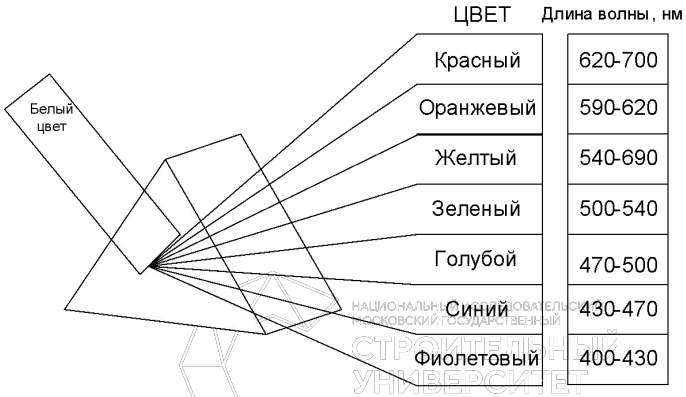
Для того чтобы «увидеть» цвет, необходимо иметь:
источник света;
объект;
ваш глаз (приемник излучения).



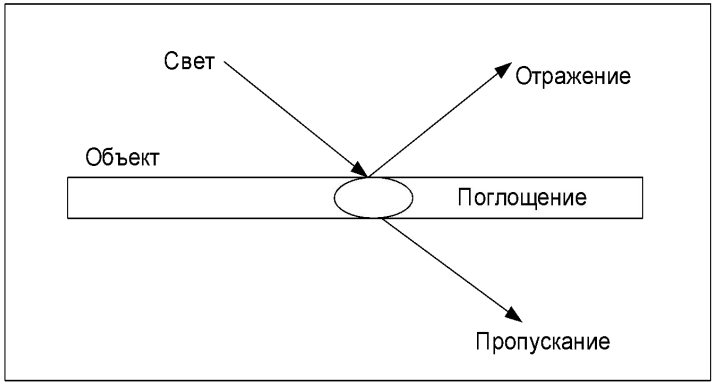
Свет имеет **двойственную природу**, обладая свойствами волны и частицы. Корпускулы света, называемые фотонами, излучаются источником света в виде волн, распространяющихся с постоянной скоростью порядка 300000 км/с. Аналогично морским волнам световые волны имеют гребни и впадины. Поэтому в качестве характеристики световых волн используют длину волны, представляющую собой рас стояние между двумя гребнями (единица измерения - метры или ангстремы, равные 10^8 м), и амплитуду, определяемую как расстояние между гребнем и впадиной.

Разные длины волны воспринимаются нами как разные цвета: свет с большой длиной волны будет красным, а с маленькой - синим или фиолетовым. В случае если свет состоит из волн разной длины (например, белый цвет содержит все длины волн, то наш глаз смешивает разные длины воли в одну, получаем таким образом один результирующий цвет.

Если пропустить луч белого света через простую призму, он разложится на цветной спектр (опыты И. Ньютона).



Цвета этого спектра, называемого видимым спектром света, условно классифицируют как красный, оранжевый, желтый, зеленый, голубой, синий и фиолетовый. Любой из них, в свою очередь, представляет собой электромагнитное излучение, перекрывающее достаточно широкий диапазон длин волн видимого спектра. Для нашего глаза каждый кусочек этого видимого спектра обладает своими уникальными характеристиками, которые и называются цветом. Поскольку в видимом спектре содержатся миллионы цветов, то различие между двумя соседними цветами практически неощутимо.



Все, что мы видим в окружающем нас пространстве, либо излучает свет, либо его отражает.

Излученный цвет - это свет, испускаемый активным источником. Примерами таких источников могут служить солнце, лампочка или экран монитора. В основе их действия обычно лежит нагревание металлических тел либо химические или термоядерные реакции. Цвет любого излучателя зависит от спектрального состава излучения. Если источник излучает световые волны во всем видимом диапазоне, то его цвет будет восприниматься нашим глазом как белый. Преобладание в его спектральном составе длин волн определенного диапазона (например, 400 - 450 нм) даст нам ощущение доминирующего в нем цвета (в данном случае сине-фиолетового). И наконец, присутствие в излучаемом свете световых компонент из разных областей видимого спектра (например, красной и зеленой) дает восприятие нами результирующего цвета (в данном случае желтого). Но при этом в любом случае попадающий в наш глаз излучаемый цвет сохраняет в себе все цвета, из которых он был создан.

Отраженный свет возникает при отражении некоторым предметом (вернее, его поверхностью) световых волн, падающих на него от источника света. Механизм отражения цвета зависит от цветового типа поверхности, которые можно условно разделить на две группы: ахроматические; хроматические.

Первую группу составляют ахроматические (иначе бесцветные) цвета: черный, белый и все серые (от самого темного до самого светлого). Их часто называют нейтральными. В предельном случае такие поверхности либо отражают все падающие на них лучи, ничего не поглощая (идеально белая поверхность), либо полностью лучи поглощают, ничего не отражая (идеальная черная поверхность). Все остальные варианты (серые поверхности) равномерно поглощают световые волны разной длины. Отраженный от них цвет не меняет своего спектрального состава, изменяется только его интенсивность. Вторую группу образуют поверхности, окрашенные в хроматические цвета, которые по-разному отражают свет с разной длиной волны.

Световые волны, излучаемые или отражаемые объектом, собираются хрусталиком и через стекловидное тело проецируются на сетчатку. Там они возбуждают определенные нервные клетки, физиологическое назначение которых состоит в распознавании световых волн. В результате возбуждения в нервных клетках возникает электрический сигнал, который по зрительному нерву поступает в зрительный центр мозга, где с помощью пока еще до конца не понятных механизмов и возникает зрительное восприятие цвета.

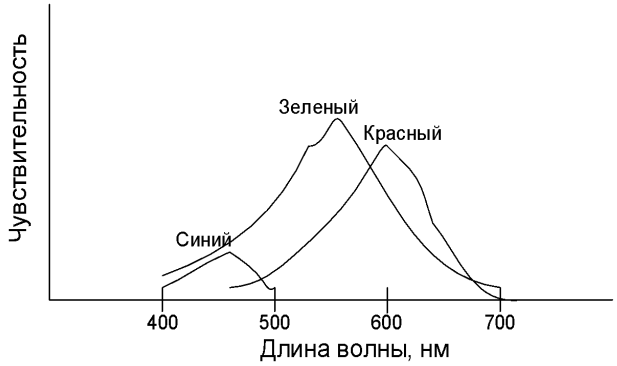
На самой сетчатке можно выделить две области, которые называют желтым пятном и слепым пятном. На слепом пятне нервные пути сетчатки переходят в зрительный нерв. Поскольку в атом месте нервных клеток нет, то свет, попадающий на слепое пятно, не обнаруживается. На желтом пятне имеет место обратная картина. Оно расположено по центру зрительной оси и содержит много зрительных клеток, чувствительных к цвету (колбочек; см. ниже). При хорошем освещении глаз обычно фокусирует падающий свет на желтом пятне. Наоборот, ночью сильной фокусировки приходится избегать, поскольку из-за низкой чувствительности колбочек зрительное восприятие значительно ослабляется.

За цветовое и яркостное восприятие человеческого глаза отвечают два различных типа нервных клеток (рецепторов), называемых соответственно колбочками и палочками.

Процесс функционирования палочек и колбочек не имеет принципиальных отличий. В обоих случаях происходит поглощение световых волн и по достижении определенного порога вырабатывается нервный импульс. При этом оба вида нервных клеток реагируют на интенсивность падающего света.

Палочки «отвечают» за черно-белое зрение, поскольку способны регистрировать только суммарную энергию света. Этот тип рецепторов равномерно распределен по сетчатке глаза и обладает очень высокой чувствительностью, примерно в 1000 раз превышающей чувствительность колбочек. Именно благодаря им обеспечивается возможность распознавания предметов в условиях плохой освещенности («ночью все кошки серы»).

Колбочки предназначены для распознавания цветовой информации. В отличие от палочек имеются три сорта колбочек, каждая из которых реагирует на определенный диапазон длин волн. Из экспериментальных данных видно, что первый тип воспринимает световые волны с длинами волн в диапазоне 400-500 нм («синяя» составляющая спектра), второй - от 500 до 600 нм («зеленая» составляющая спектра) и третий - от 600 до 700 («красная» составляющая спектра). В зависимости от того, световые волны какой длины и интенсивности присутствуют в спектре, те или иные группы колбочек возбуждаются сильнее или слабее. Полученная с помощью зрительных рецепторов информация поступает в виде сигналов в мозг, который определяет, в каких соотношения возбуждены три вида колбочек, создавая на базе этого цветовое восприятие.



Для характеристики цвета используются следующие атрибуты.

1. *Цветовой тон.* Его можно определить преобладающей длиной волны в спектре излучения. Цветовой тон позволяет отличать один цвет от другого - например, зеленый от красного, желтого и других.

2. *Яркость.* Определяется энергией, интенсивностью светового излучения. Выражает количество воспринимаемого света.

3. *Насыщенность* или чистота тона. Выражается долей присутствия белого цвета. В идеально чистом цвете примесь белого отсутствует. Если, например, к чистому красному цвету добавить в определенной пропорции белый цвет (у художников это называется "разбелом"), то получится светлый бледно-красный цвет.

Наука, которая изучает цвет и его измерения, называется колориметрией**.** Она описывает общие закономерности цветового восприятия света человеком. Одними из основных законов колориметрии являются законы смешивания цветов. Эти законы в наиболее полном виде были сформулированы в 1853 году немецким математиком *Германом Грассманом.*

1. *Цвет - трехмерен, для его описания необходимы три компонента. Любые четыре цвета находятся в линейной зависимости, хотя существует неограниченное число линейно-независимых совокупностей из трех цветов.* Другими словами, для любого заданного цвета ***(Ц)*** можно записать такое цветовое уравнение, которое выражает линейную зависимость цветов:

$$Ц = k_1 Ц_1 + k_2 Ц_2 + k_3 Ц_3,$$

где $Ц_1$, $Ц_2$,$Ц_3$ - некоторые базисные, линейно-независимые цвета, коэффициенты k_1, k_2 и k_3 указывают количество соответствующего смешиваемого цвета, Линейная независимость цветоа $Ц_1$, $Ц_2$, $Ц_3$ означает, что ни один из них не может быть выражен взвешен ной суммой (линейной комбинацией) двух других. Первый закон можно трактовать и в более широком смысле, а именно, в смысле трехмерности цвета. Необязательно для описания цвета использовать смесь других цветом можно применять и другие компоненты, но их обязательно должно быть три.

2. Если в смеси *трех цветовых компонентов один меняется непрерывно, в то время как* два *других* остаются *постоянными, цвет смеси также изменяете непрерывно.*

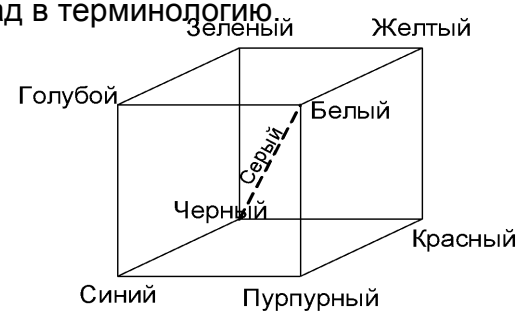
3. Цвет *смеси зависит только от цветов смешиваемых компонентов и не зависит от их спектральных составов.* Смысл третьего закона становится более понятным, если учесть, что один и тот же цвет (в том числе и цвет смешиваемых компонентов) может быть получен разными способами. Например, смешиваемый компонент может быть получен, в свою очередь, смешиванием других компонентов.

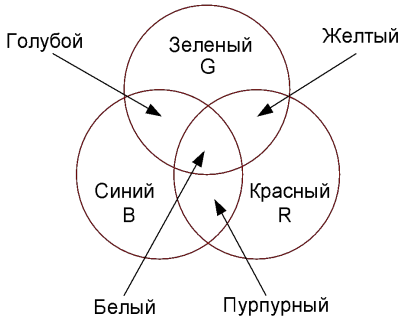
Аддитивные цветовые модели

Аддитивный цвет получается на основе законов Грассмана путем соединения лучей света разных цветов. В основе этого явления лежит тот факт, что большинство цветов видимого спектра могут быть получены путем смешивания в различных пропорциях трех основных цветовых компонент. Этими компонентами, которые в теории цвета иногда называются первичными цветами, являются красный (Red), зеленый (Green) и синий (Blue) цвета. При попарном смешивании первичных цветов образуются вторичные цвета: голубой (Cyan), пурпурный (Magenta) и желтый (Yellow). Следует отметить, что первичные и вторичные цвета относятся к базовым цветам.

Базовыми цветами называют цвета, с помощью которых можно получить практически весь спектр видимых цветов.

Используемые для построения RGB-модели первичные, или аддитивные, цвета имеют еще одно название. Иногда, чтобы подчеркнуть тот факт, что при добавлении света интенсивность цвета увеличивается, эту модель называют добавляющей. Такое обилие терминов, используемых для описания RGB-модели, связано с тем, что она возникла задолго до появления компьютера и каждая область ее применения внесла свой вклад в терминологию.





К настоящему времени система RGB - это официальный стандарт. Решением Международной Комиссии по Освещению - *МКО (CIE - Commision International de VEclairage)* в 1931 году были стандартизированы основные цвета, которые было рекомендовано использовать в качестве *R, G* и *B.* Это монохроматические цвета светового излучения с длинами волн соответственно:

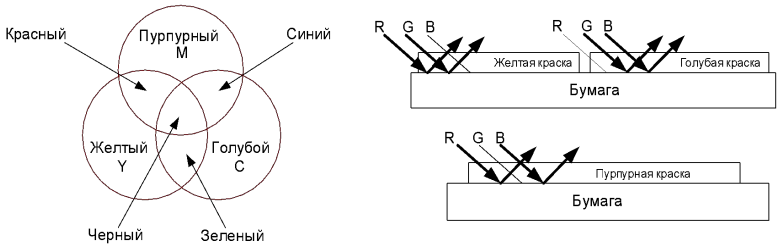
R - 700 нм, *G* - 546.1 нм, B - 435.8 нм.

Субтрактивные цветовые модели

Субтрактивные цвета в отличие от аддитивных цветов (той же RGB-модели) получаются вычитанием вторичных цветов из общего луча света. В этой системе белый цвет появляется как результат отсутствия всех цветов, тогда как их присутствие дает черный цвет.

В качестве синонима термина «субтрактивная» иногда используют термин «исключающая». Происхождение этого названия связано с явлением отражения света от покрытой красителем поверхности, а также с тем фактом, что при добавлении красителей интенсивность света уменьшается, поскольку свет поглощается тем больше, чем больше красителя нанесено на поверхность.

Нанесение на бумагу трех базовых цветов: голубого (Cyan), пурпурного (Magenta) и желтого (Yellow) позволяет создать множество субтрактивных цветов.



Нанесение желтой краски на белую бумагу означает, что поглощается отраженный синий цвет. Голубая краска поглощает красный цвет. Пурпурная краска - зеленый. Комбинированне красок позволяет получить цвета, которые остались - зеленый, красный, синий и черный. Черный соответствует поглощению всех цветов при отражении. На практике добиться черного смешиванием сложно из-за неидеальности красок, поэтому в принтерах используют еще и краску черного цвета (black). Тогда модель называется CMYK, Необходимо также отметить, что не всякие краски обеспечивают указанное выше вычитание цветов CMY.

Цвет	Модель RGB			Модель CMY		
	R	G	B	C	M	Y
Красный	1	0	0	0	1	1
Желтый	1	1	0	0	0	1
Ярко-Зеленый	0	1	0	1	0	1
Голубой	0	1	1	1	0	0
Синий	0	0	1	1	1	0
Пурпурный	1	0	1	0	1	0
Черный	0	0	0	1	1	1
Белый	1	1	1	0	0	0

Существуют две наиболее распространенные версии субтрактивной модели: CMY и CMYK. Первая из них используется в том случае, если изображение или рисунок будут выводиться на черно-белом принтере, позволяющем заменять черный картридж на цветной (color upgrade). В ее основе лежит использование трех субтрактивных (вторичных) цветов: голубого (Cyan), пурпурного (Magenta) и желтого (Yellow). Теоретически при смешивании этих цветов на белой бумаге в равной пропорции получается черный цвет.

В аббревиатуре модели CMYK используется буква «К» (последняя буква слова Black) для того, чтобы избежать путаницы, поскольку в английском языке с буквы «В» начинается не только слово Black (черный), но и слово Blue (синий). Встречается еще один вариант трактовки использования этой буквы как аббревиатуры термина Key color (ключевой цвет).

Перцепционные цветовые модели

Модель HSB (Hue - цветовой тон, Saturation - насыщенность, Brightness - яркость) или ее ближайший аналог HSL представлены в большинстве современных графических пакетов. Из всех используемых в настоящее время моделей эта модель наиболее точно соответствует способу восприятия цветов человеческим глазом. Она позволяет описывать цвета интуитивно ясным способом.

В HSB-модели все цвета определяются с помощью комбинации трех базовых параметров:

- цветовой тон (H);
- насыщенность(S);
- яркость (B).

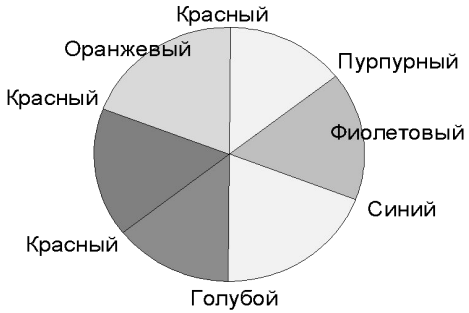
Цветовой тон

Как уже отмечалось, каждый реальный источник света воспроизводит его в виде смеси волн, имеющих разные длины. Под *цветовым тоном* (hue) понимается свет с доминирующей длиной волны. Обычно для описания цветового тона (в некоторых источниках применяется термин оттенок) используется название цвета, например красный, оранжевый или зеленый. В традиционной интерпретации этой модели каждый цветовой тон занимает определенное положение на периферии цветового круга и характеризуется величиной угла в диапазоне от 0 до 360° . Обычно для красного цвета берется угол 0°, для чисто зеленого - 120° и для чисто синего - 240°.

Насыщенность

Цветовой тон не единственный атрибут цвета, различаемый людьми. Другой компонент - *насыщенность* - характеризует чистоту цвета. Он определяет соотношение между основной, доминирующей компонентой цвета и всеми остальными длинами волн (количеством серого), участвующими в формировании цвета. Ко личественное значение этого параметра выражается в процентах от *0%* (серый) до *100%* (полностью насыщенный).

По другому определению, насыщенность отражает, насколько далеко отстоит данный цвет от равного с ним по яркости белого цвета. *В* этом случае насыщенность можно измерять числом едва заметных переходов (градаций), лежащих между данным цветом и белым.



Яркость

характеризует интенсивность, с которой энергия света воздействует на рецепторы нашего глаза. Ее можно интерпретировать также как относительную освещенность или затемненность цвета (светлоту цвета). Солнечный зайчик - пример высокой интенсивности освещения (яркого). В то же время тлеющие угли - низкой. Любые цвета и оттенки независимо от их цветового тона можно сравнить по яркости, то есть определить, какой из них темнее, а какой светлее.

Яркость никоим образом не влияет на цветность, но от нее зависит, насколько сильно цвет будет восприниматься нашим глазом. При нулевой яркости мы не видим ничего, поэтому любой цвет будет восприниматься как черный. Исходя из этого яркость иногда трактуют подобно насыщенности, то есть как величину, обратную степени разбавленности цвета черным. В этом случае при отсутствии черного мы получаем чистый спектральный цвет, а максимальная яркость вызывает ощущение ослепительно белого цвета.

Когда говорят о яркости как атрибуте цвета, под белым цветом понимают абсолютную яркость, а под черным цветом - полное отсутствие яркости. Серый цвет характеризует промежуточное значение яркости.

Системы соответствия цветов

Для упрощения процедуры идентификации цвета ведущими фирмами, специализирующимися в области полиграфии и производстве красителей, были созданы системы соответствия цветов.

Система соответствия цветов включает в себя набор следующих основных компонентов:

- ✔ Эталонные таблицы (атласы или каталоги) цветов, содержащихся в одноименных палитрах
- ✔ Электронные палитры (или просто палитры)
- ✔ Специальные программные и аппаратные средства для калибровки устройств вывода.

Рассмотрим подробнее эталонные таблицы и электронные палитры.

Назначение эталона

Эталонные таблицы предоставляют собой набор цветов (образцов), которые могут быть адекватным образом отображены в процессе печати на соответствующей им бумаге.

Изготовление эталона тщательно контролируется с целью минимизации вариаций цветов. Каждому цвету присваивается свое уникальное имя и указывается тип пигмента или состав смеси из различных пигментов, необходимых для его реализации. Указывается также идентифицированный с данным пигментом тип бумаги. В дополнение к этой таблице, используемой как справочник, пользователь получает образцы цветов, которые можно вырезать и прикрепить к изображению. Благодаря этим образцам система обеспечивает точный визуальный контроль соответствия того, что мы видим на экране, с тем, что мы получим на печати.

Кодирование цвета. Палитра

Для того чтобы компьютер имел возможность работать с цветными изображениями, необходимо представлять цвета в виде чисел - кодировать цвет. Способ кодирования зависит от цветовой модели и формата числовых данных в компьютере.

Для модели RGB любой из компонентов может быть представлен числами, ограниченными определенным диапазоном - например, дробными числами от 0 до 1, или целыми числами от 0 до какого-либо максимального значения. В настоящее время довольно распространен формат True Color, в котором каждый компонент представлен в виде банта, что дает 256 градаций для любого компонента: R - 0 ... 255, G = 0 ... 255, B = 0 ... 255. Количество цветов составляет 256x256x256 = 16.7 млн. (2^{24}).

Такой способ кодирования цветов можно назвать компонентным. В компьютере коды изображений True Color представлены в виде троек байтов или упаковываются в длинное целое (четырехбайтовое) - 32 бита (так, например, сделано в API Windows):

$$C = 00000000\ bbbbbbbb\ gggggggg\ rrrrrrrr.$$

Триадные и плашечные цвета

Для печатания результатов работы, выполненной вами в графической программе на полиграфическом оборудовании, возможно использование одной из двух схем печати: плашечной или многослойной.

Плашечными (или простыми) цветами называются цвета, которые воспроизводятся на бумаге готовыми смесовыми красками.

Каждый плашечный цвет репродуцируется с помощью отдельной печатной формы (плашки).

Многослойная печать основана на использовании триадных (иначе составных) цветов и включает в себя как минимум четыре процесса.

Триадные цвета воспроизводятся путем смешивания в разных пропорциях триадных красок (голубой, пурпурной, желтой), применяемых в стандартной четырехкрасочной печати.

В графических программах все цветовые модели работают именно с триадными цветами. Поэтому воспроизведение плашечного цвета на экране монитора с помощью, например, цветовой модели RGB приводит к аппроксимации плашечного цвета триадным цветом.

Плашечная схема печати применяется тогда, когда количество цветов в рисунке меньше четырех или когда отдельные цвета не могут быть получены путем смешивания красок (например, неоновые или имитирующие металлизированную поверхность).

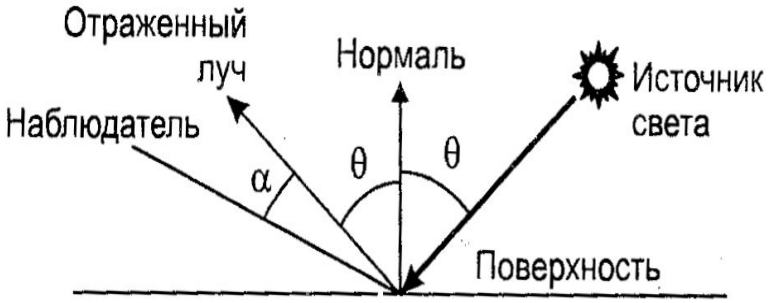
В случае необходимости прецизионного воспроизведения цвета или создания специальных цветовых эффектов возможны реализация плашечной печати с большим количеством цветов или совмещение плашечной и многослойной печати.

3.5. Модели расчета освещенности в компьютерной графике. Закрашивание поверхностей

Немного о свете

Зеркальное отражение света. Угол между нормалью и падающим лучом равняется углу между нормалью и отраженным лучом. Падающий луч, отраженный луч и нормаль располагаются в одной плоскости.

Поверхность считается *идеально зеркальной,* если на ней отсутствуют какие-либо неровности, шероховатости. Собственный цвет у такой поверхности не наблюдается. Световая энергия падающего луча отражается только по линии отраженного луча. Любое рассеивание в стороны от этой линии отсутствует. В природе, вероятно, нет идеально гладких поверхностей, поэтому полагают следующее: если глубина шероховатостей существенно меньше длины волны излучения, то рассеивание не наблюдается. Для видимого спектра можно принять, что глубина шероховатости поверхности зеркала должна быть меньше 0.5 мкм.



Диффузное отражение. Этот вид отражения присущ *матовым* поверхностям. Матовой можно считать такую поверхность, размер шероховатостей которой уже настолько большой, что падающий луч рассеивается равномерно во все стороны. Такой тип отражения характерен, например, для гипса, песка, бумаги. Диффузное отражение описывается *законом Ламберта,* согласно которому интенсивность отраженного света пропорциональна косинусу угла между направлением на точечный источник света и нормалью к поверхности:

$$Id = I \cos\theta,$$

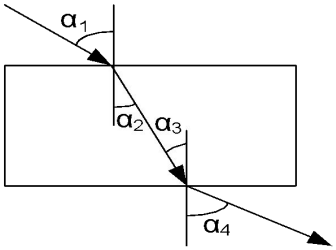
где I - интенсивность источника света.

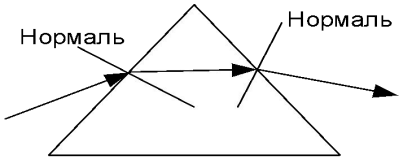
Идеально преломление. Согласно этой модели луч отклоняется на границе двух сред, причем падающий луч, преломленный луч и нормаль лежат в одной плоскости (в этой же плоскости лежит и зеркально отраженный луч).

Обозначим угол между падающим лучом и нормалью как α_1, а угол между нормалью и преломленным лучом как α_2. Для этих углов известен закон *Синеллиуса,* согласно которому

$$\eta_1 \sin \alpha_1 = \eta_2 \sin \alpha_2$$

где η_1 *и* η_2 - *абсолютные показатели преломления* соответствующих сред.





Диффузное преломление.
Согласно этой модели луч отклоняется на границе двух сред, причем преломленные лучи уходят внутрь второй среды под разными углами без единого преволирующего направления преломления.

Метод Гуро

Этот метод предназначен для создания иллюзии гладкой криволинейной поверхности, которая описана в виде многогранников или полигональной сетки с плоскими гранями. Если каждая плоская грань имеет один постоянный цвет, который определен в соответствии с учетом отражения, то разные цвета соседних граней очень заметны, и поверхность выглядит именно как многогранник. Казалось, этот дефект можно замаскировать за счет увеличения количества граней при аппроксимации поверхности. Но зрение человека имеет способность подчеркивать перепады яркости на границах смежных граней - такой эффект называется эффектом *полос Маха.* Вследствие этого, для создания иллюзии гладкости нужно намного увеличить количество граней, что приводит к существенному замедлению визуализации - чем больше граней, тем меньше скорость рисования объектов.

Метод Гуро основан на идее закрашивания каждой плоской грани не одним цветом, а плавно изменяющимися оттенками, которые вычисляются путем интерполяции цветов прилегающих граней. Закрашивание граней по методу Гуро осуществляется в четыре этапа.

- Вычисляются нормали к каждой грани.
- Определяются нормали в вершинах. Нормаль в вершине определяется усреднением нормалей прилегающих граней.

$$I_{рез} = I_a K_a + K_d \sum_j I_d(j) + Ks \sum_j I_s(j)$$

$$Is = I \cos^p \alpha,$$

$$Id = I \cos\theta,$$

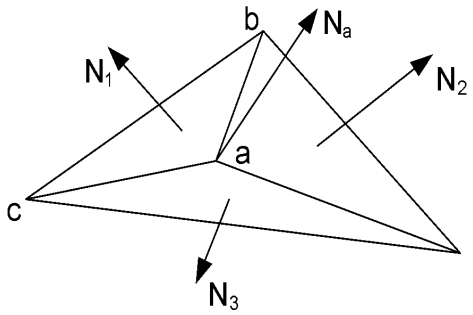
где *I* - интенсивность точечного источника света,

Is - интенсивность зеркально отраженного излучения,

Id - интенсивность отраженного света,

Ia - интенсивность рассеянного света,

Ka, Ks, Kd – экспериментальные константы

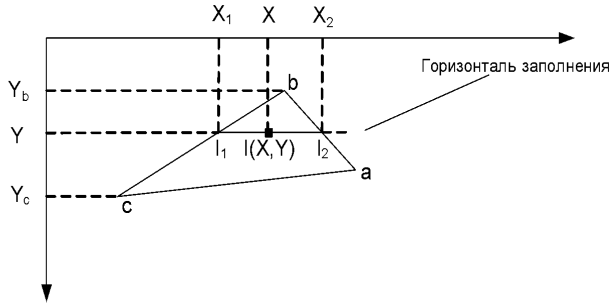


На основе нормалей в вершинах вычисляются значения интенсивностей в вершинах в соответствии с вы бранной моделью отражения света. Закрашиваются полигоны граней цветом, который соответствует линейной интерполяции значений интенсивности в вершинах.

Вектор нормали в вершине *(a)* равняется

$$N_a = \frac{N_1 + N_2 + N_3}{3}$$

Определение интерполированных значений интенсивности отраженного света в каждой точке грани (и, следовательно, цвет каждого пиксела) удобно выполнять во время цикла заполнения полигона. Рассмотрим заполнение контура грани горизонталями в экранных координатах



Интерполированная интенсивность *I* в точке *(X, У)* определяется, исходя из пропорции

$$\frac{I - I_1}{X - X_1} = \frac{I_2 - I_1}{X_2 - X_1}$$

откуда

$$I = I_1 + \frac{(I_2 - I_1)(X_2 - X_1)}{X_2 - X_1}$$

Значение интенсивности I_1 и I_2 на концах горизонтального отрезка вычислим путем интерполяции интенсивности в вершинах:

$$\frac{I_1 - I_b}{Y - Y_b} = \frac{I_c - I_b}{Y_c - Y_b}$$

$$\frac{I_2 - I_b}{Y - Y_b} = \frac{I_a - I_b}{Y_a - Y_b}$$

$$I_1 = I_b + \frac{(Ic - I_b)(Y - Y_b)}{Y_c - Y_b}$$

$$I_2 = I_b + \frac{(Ia - I_b)(Y - Y_b)}{Ya - Y_b}$$

Метод Фонга

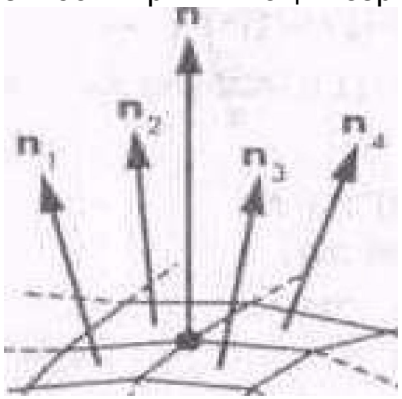
Фонг предложил вместо интерполяции интенсивностей произвести интерполяцию вектора нормали к поверхности на сканирующей строке. Этот метод требует больших вычислительных затрат, поскольку формулы
интерполяции применяются к трем компонентам вектора нормали, но зато дает лучшую аппроксимацию кривизны поверхности.

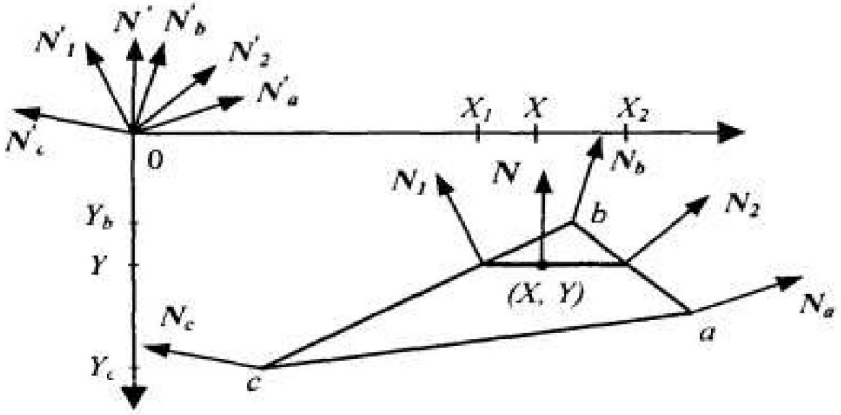
Этот метод позволяет устранить ряд недостатков метода Гуро

Аналогичен методу Гуро, но при использовании метода Фонга для определения цвета в каждой точке интерполируются не интенсивности отраженного света, а векторы нормалей.

* Определяются нормали к граням.
* По нормалям к граням определяются нормали в вершинах. В каждой точке закрашиваемой грани определяется интерполированный вектор нормали.
* Цвет каждой точки грани вычисляется в соответствии с направлением интерполированного вектора нормали и согласно выбранной модели отражения света.

Метод Фонга сложнее метода Гуро. Для каждой точки (пиксела) поверхности необходимо выполнять намного больше вычислительных операций. Тем не менее, он дает значительно лучшие результаты, в особенности при имитации зеркальных поверхностей.





Трассировка лучей

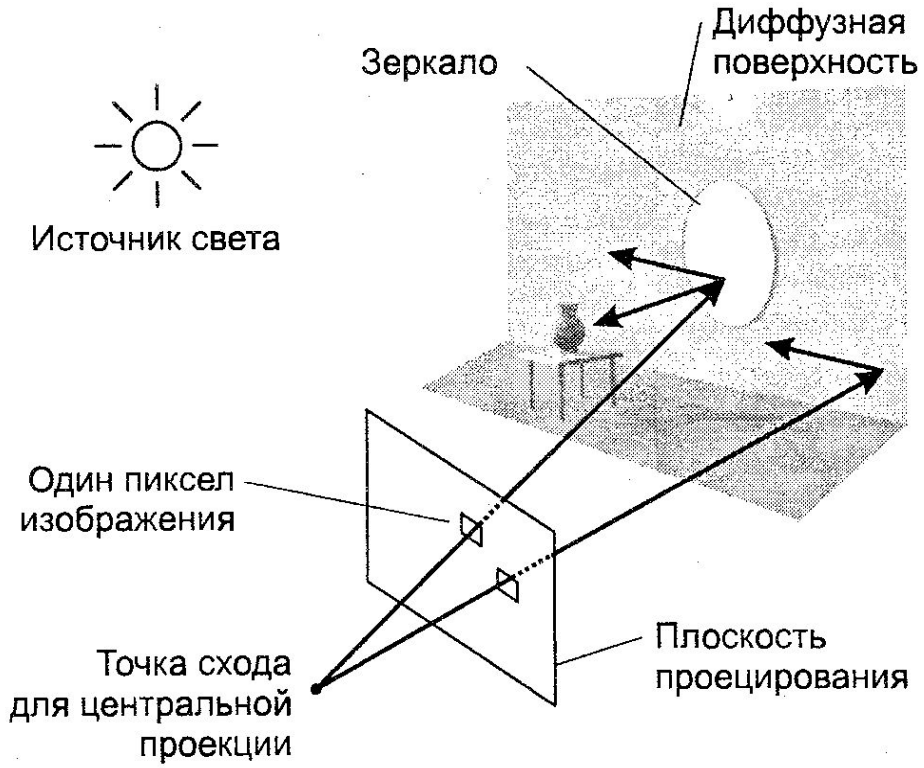
Методы трассировки лучей *(Ray Tracing)* на сегодняшний день считаются наиболее мощными и универсальными методами создания реалистичных изображений. Известно много примеров реализации алгоритмов трассировки для качественного отображения самых сложных трехмерных сцен. Можно отметить, что универсальность методов трассировки в значительной мере обусловлена тем, что в их основе лежат простые и ясные понятия, которые отражают опыт восприятия окружающего мира.

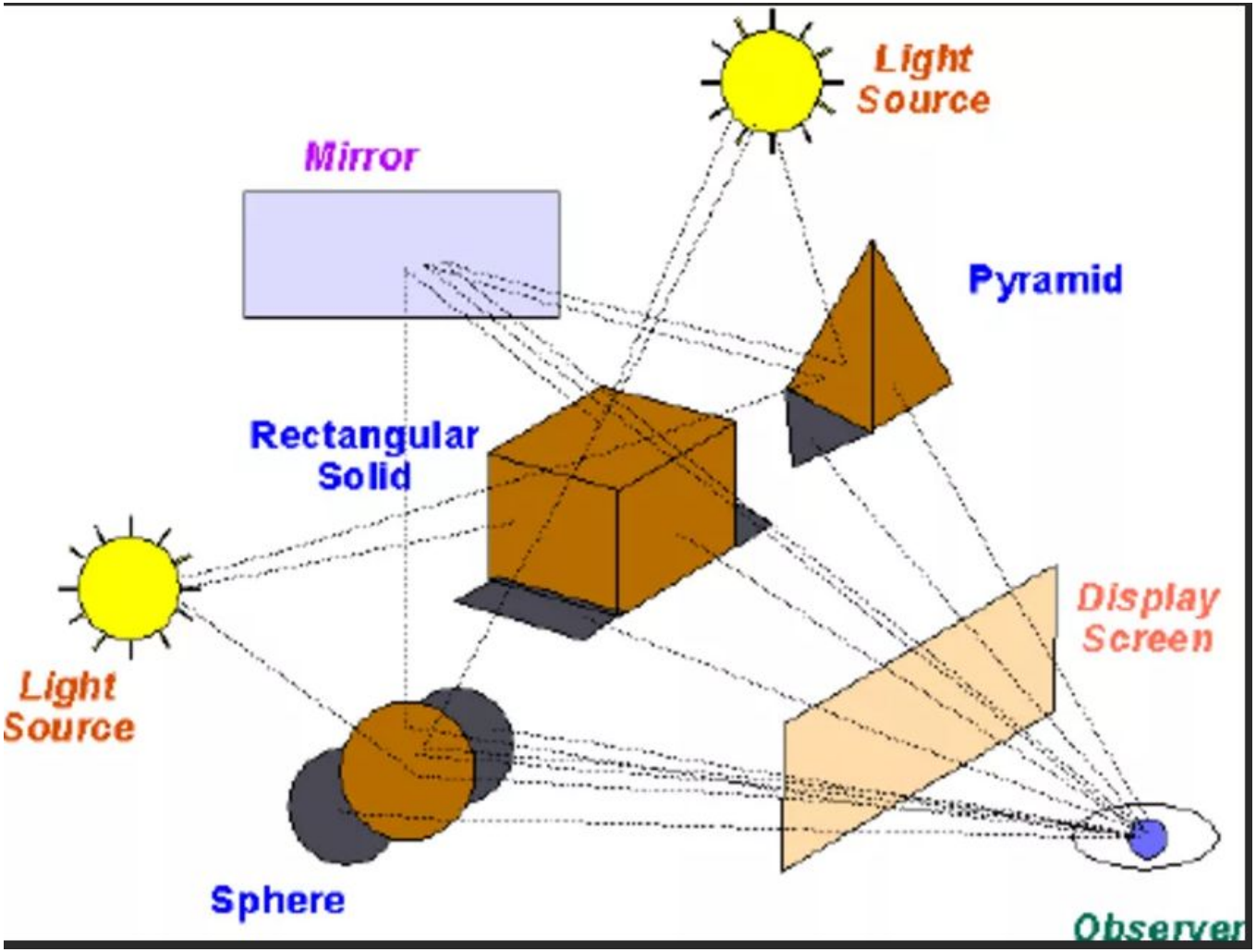
Как в модели формируется изображение некоторой сцены, которая содержит несколько пространственных объектов? Считается, что из точек поверхности (объема) излучаемых объектов выходят лучи света. Можно назвать такие лучи первичными - они освещают все другое.

Важным моментом является предположение, что световой луч в свободном пространстве распространяется *вдоль прямой линии* (хотя в специальных разделах физики изучаются также и причины возможного искривления). Но в *геометрической оптике* принято, что луч света распространяется прямолинейно до тех пор, пока не встретится отражающая поверхность или граница среды преломления. От источников излучения исходит по разным направлениям бесчисленное множество первичных лучей Некоторые лучи уходят в свободное пространство, а некоторые (их также бесчисленное множество) попадают на другие объекты. Если луч попадет в прозрачный объект, то, преломляясь, он идет дальше, при этом некоторая часть световой энергии поглощается. Подобно этому, если на пути луча встречается зеркально отражающая поверхность, то он также изменяет направление, а часть световой энергии поглощается. Если объект зеркальный и одновременно прозрачный (например, обычное стекло), то будут уже два луча - в этом случае говорят, что луч расщепляется.

Можно сказать, что в результате воздействия на объекты первичных лучей возникают вторичные лучи. Бесчисленное множество вторичных лучей уходит в свободное пространство, но некоторые из них попадают на другие объекты. Так, многократно отражаясь и преломляясь, отдельные световые лучи приходят в точку наблюдения - глаз человека или оптическую систему камеры. Очевидно, что в точку наблюдения может попасть и часть первичных лучей непосредственно от источников излучения. Таким образом, изображение сцены формируется некоторым множеством световых лучей.

Цвет отдельных точек изображения определяется спектром и интенсивностью первичных лучей источников излучения, а также поглощением световой энергии в объектах, встретившихся на пути соответствующих лучей.


Зеркало
Диффузная поверхность
Источник света
Один пиксел изображения
Точка схода для центральной проекции
Плоскость проецирования


Light
Source
Mirror
Pyramid
Rectangular
Solid
Display
Screen
Light
Source
Sphere
Observer

Непосредственная реализация данной лучевой модели формирования изображения представляется затруднительной. Можно попробовать разработать алгоритм построения изображения указанным способом. В таком алгоритме необходимо предусмотреть перебор всех первичных лучей и определить, какие из них попадают в объекты и в камеру. Потом выполнить перебор всех вторичных лучей, и также учесть только те, которые попадают в объекты и в камеру. И так далее. Можно назвать такой метод ***прямой*** **трассировкой** лучей. Очевидно, что полный перебор бесконечного числа лучей в принципе невозможен. Даже если каким-то образом свести это к конечному числу операций (например, разделить всю сферу направлений на угловые секторы и оперировать уже не бесконечно тонкими линиями, а секторами), все равно остается главный недостаток метода — много лишних операций, связанных с расчетом лучей, которые потом не используются. Так, во всяком случае, это представляется в настоящее время.

Метод ***обратной трассировки*** лучей позволяет значительно сократить перебор световых лучей. Метод разработан в 80-х годах, основополагающими считаются работы *Уиттеда* и *Кэя*. Согласно этому методу отслеживание лучей осуществляется не от источников света, а в обратном направлении - от точки наблюдения. Так учитываются только те лучи, которые вносят вклад в формирование изображения.

Для определения освещенности некоей точки P сначала рассчитывается непосредственная освещенность этой точки от источников света (выпустив из нее лучи ко всем источникам). Для определения вторичной освещенности из точки P выпускается один луч для отраженного направления и один луч для преломленного. Тем самым для определения освещенности точки необходимо будет отслеживать лишь небольшое количество лучей. *При этом неидеально зеркальное отражение лучей, идущих от других объектов, игнорируется*.

Для расчета освещенности точки вводятся некоторые ограничения на рассматриваемую сцену:

- используются только точечные источники света;
- при трассировании преломленного луча игнорируется зависимость его направления от длины волны;
- полагается, что освещенность объекта состоит из диффузной и зеркальной составляющих (с заданными весами).

Обычно для компенсации неучитываемых составляющих освещенности вводится так называемое фоновое освещение — равномерное освещение со всех сторон, которое ни от чего не зависит и не затеняется. Тогда энергия, покидающая точку P в заданном направлении, задается следующей формулой:

$$I(\lambda) = K_a I_a(\lambda) C(\lambda) + K_d C(\lambda) \sum_i I_{l_i}(\lambda)(n, l_i) + K_s \sum_i I_{l_i}(\lambda) \frac{F_r(\lambda, \theta) D(\alpha) G(n, v, l_i)}{(n, l_i)(n, v)} +$$
$$+ K_s I_r(\lambda) F_r(\lambda, \theta_r) e^{-\beta_r d_r} + K_t I_t(\lambda)(1 - F_r(\lambda, \theta_t)) e^{-\beta_t d_t},$$

где

$I_a(l)$ — интенсивность фонового освещения;

$I_{li}(l)$ — интенсивность i-го источника света;

$I_r(l)$ — интенсивность, приходящая по отраженному лучу;

$I_t(l)$ — освещенность, приносимая преломленным лучом;

$C(l)$ — цвет в точке P;

K_a — коэффициент фонового освещения;

K_d — коэффициент диффузного освещения;

K_s — коэффициент зеркального освещения;

K_t — вклад преломленного луча;

n — вектор внешней нормали в точке P;

l_i — единичный вектор направления из точки P на i-йисточник света;

q_r — угол отражения (для отраженного луча);

q_t — угол преломления;

d_r- расстояние, пройденное отраженным лучом;

d_t — расстояние, пройденное преломленным лучом;

b_r — коэффициент ослабления для отраженного луча;

b_t — коэффициент ослабления для преломленного луча.

Алгоритмы трассировки носят характер рекурсивной процедуры, которая вызывает саму себя при появлении вторичного луча (анализируемый луч отражается или преломляется). Большая часть вычислений при реализации методов трассировки приходится на расчет пересечений лучей с поверхностями, в связи с чем они применяются для изображения оптических эффектов в сценах с небольшим числом объектов.

При практической реализации метода обратной трассировки вводят нижеприведенные ограничения. Некоторые из них необходимы, чтобы можно было в принципе решить задачу синтеза изображения, а некоторые ограничения позволяют значительно повысить быстродействие трассировки.

Ограничения метода обратной трассировки:

1. Среди всех типов объектов выделим источники света. Они могут только излучать свет, но не могут его отражать или преломлять. Обычно рассматриваются точечные источники.
2. Свойства отражающих поверхностей описываются суммой двух компонентов: диффузного и зеркального.
3. Зеркальность, в свою очередь, также описывается двумя составляющими. Первая (reflection) учитывает отражение от других объектов, не являющихся источниками света. Строится только один зеркально отраженный луч *r* для дальнейшей трассировки. Вторая компонента (specular) означает световые блики от источников света. Для этого направляются лучи на все источники определяются углы, образуемые этими лучами с зеркально отраженным лучом обратной трассировки (*r*). При зеркальном отражении цвет точки поверхности определяется цветом того, что отражается.
4. При диффузном отражении учитываются только лучи от источников света. Лучи от зеркально отражающих поверхностей игнорируются. Если луч, направленный на данный источник света, закрывается другим объектом, значит, данная точка объекта находится в тени. При диффузном отражении цвет освещенной точки поверхности определяется собственным цветом поверхности и цветом источников света.
5. Для прозрачных (transparent) объектов обычно не учитывается зависимость коэффициента преломления от длины волны.
6. Для учета освещенности объектов светом, рассеиваемым другими объектами, вводится фоновая составляющая (ambient).
7. Для завершения трассировки вводят некоторое пороговое значение освещенности, которое уже не должно вносить вклад в результирующий цвет, либо ограничивают число итераций.

Положительные черты метода обратной трассировки:

● универсальность, применимость для синтеза изображений достаточно сложных пространственных сцен. Воплощает многие законы оптики. Просто реализуются разнообразные проекции;

● даже усеченные варианты данного метода позволяют получить достаточно реалистичные изображения. Например, если ограничиться только первичными лучами (из точки проецирования), то это дает удаление невидимых точек. Трассировка уже одного-двух вторичных лучей дает тени, зеркальность, прозрачность;

● все преобразования координат (если таковые есть) линейны, поэтому достаточно просто работать с текстурами;

● для одного пиксела растрового изображения можно трассировать несколько близко расположенных лучей, а потом усреднять их цвет для устранения эффекта ступенчатости;

● поскольку расчет отдельной точки изображения выполняется независимо от других точек, то это может быть эффективно использовано при реализации данного метода в параллельных вычислительных системах, в которых лучи могут трассироваться одновременно.

Недостатки метода обратной трассировки:

● проблемы с моделированием диффузного отражения и преломления;

● для каждой точки изображения необходимо выполнять много вычислительных операций. Трассировка лучей относится к числу самых медленных алгоритмов синтеза изображений.

Метод излучательности

Алгоритм трассировки лучей плохо работает с диффузным отражением, влияние света, отраженного от одной грани, который попадает на другую практически не учитывается. К тому же, при изменении положения наблюдателя необходимо пересчитывать все освещение.

Для более качественного освещения применяют метод излучательности. Для этого все грани в сцене разбивают на небольшие фрагменты и составляют для них уравнения баланса энергии.

$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$, где

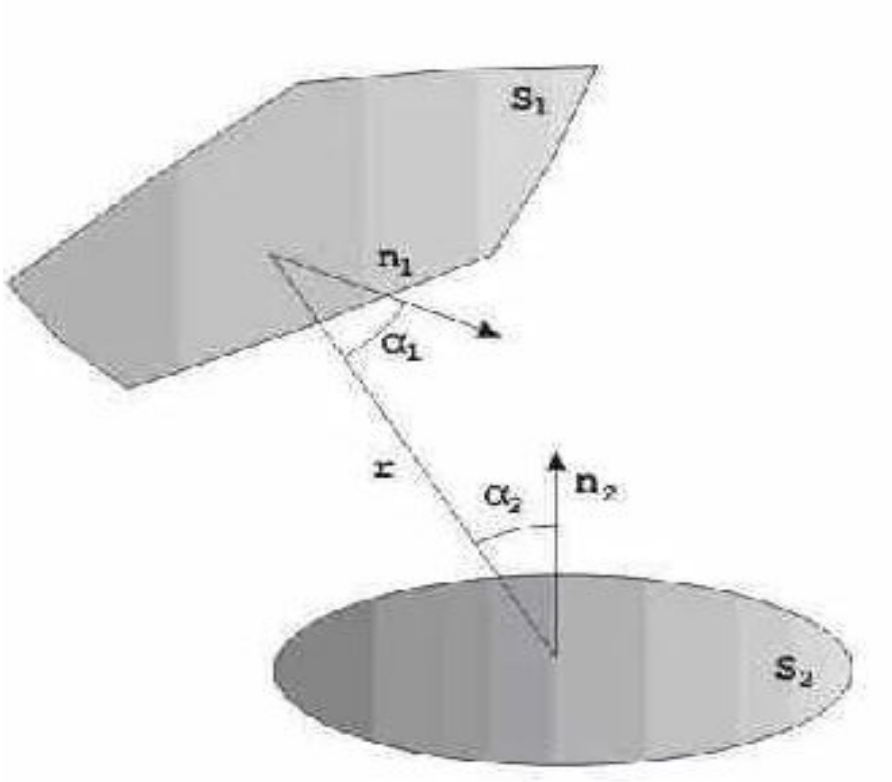
B_i – энергия, отбрасываемая i-м фрагментом;

E_i – энергия излучаемая i-м фрагментом (если это источник света);

F_{ij} – доля энергии j-ого фрагмента, попадающая на i-м фрагментом (коэффициент формы);

ρ_i – коэффициент отражения i-ого фрагмента;

Если учитывать закон сохранения энергии $\sum_{j=1}^{n} F_{ij} < 1$, то эта система хорошо решается итеративным методом.



$$F_{i,j} \approx (cos\alpha_1 * cos\alpha_1)/r^2$$

использованы при создании игрушек тауматроп (1825 г.) и зоотроп (1834 г.). Тауматроп представлял собой плоский диск с рисунками, нанесенными на обе его стороны, а зоотроп - бумажную ленту с рисунками. При вращении этих игрушек возникала иллюзия движения.

3.6. Анимация в компьютерной графике

Анимация - это создание зрительной иллюзии движения, изменения чего-то во времени.

Эффект анимации основан на некоторых особенностях зрения человека, а именно: след изображения сохраняется некоторое время на сетчатке глаза свойственна способность объединять быстро сменяющие друг друга изображения в единый зрительный ряд, который даёт иллюзию непрерывности.

Эти особенности зрения человека были использованы при создании игрушек тауматроп (1825 г.) и зоотроп (1834 г.). Тауматроп представлял собой плоский диск с рисунками, нанесенными на обе его стороны, а зоотроп - бумажную ленту с рисунками. При вращении этих игрушек возникала иллюзия движения.

Методы анимации

Известные к настоящему времени технологии компьютерной анимации можно разделить на два такие класса: 2D - и 3D - анимация. Несмотря на то, что результатом в обоих случаях является создание серии изображений в плоской проекции, методы 2D- и 3D-анимация существенно различаются. ***Под 2D-анимацией*** обычно подразумевается перемещение, наложение в определенном порядке отдельных спрайтов. Например, на фон накладываются изображения фигурок людей и животных. Для каждой движущейся фигурки заготавливается несколько картинок, изображающих персонаж в различных фазах движения. Картинка может быть небольшим растром. Тогда для правильного наложения такого спрайта на фон в прямоугольнике растра пикселы за границами контура фигурки делаются прозрачными.

Современные программы трехмерной анимации позволяют построить первоначальную сцену (ключевой кадр), передвинуть вперед указатель на временной шкале, изменить первоначальную сцену (следующий ***ключевой кадр***) и получить построенные самим компьютером промежуточные кадры. Таким образом реализуется так называемый ***метод анимации по ключевым кадрам.*** Но идеи метода анимации по ключевым кадрам возникли и использовались еще при создании рисованных мультфильмов, когда ведущий аниматор определял, и сам делал ключевые кадры, а другие аниматоры рисовали остальные кадры. Аналогичный по идеям ***метод расчета промежуточных изображений & 3D-моделей - tweening*** - используется для уменьшения количество хранимых кадров (тогда при воспроизведении анимации на экране осуществляется интерполяция «на лету» положения вершин полигональной модели). Таким образом, одни и те же идеи и методы могут использовать и при традиционной, и при компьютерной 2D - или 3D - анимации.

Для создания ***иллюзии механического движения*** достаточно перемещать (поворачивать) одни модели относительно других моделей или относительно неподвижного фона (кстати, можно передвигать фон, оставляя модель неподвижной). Это методы, так сказать, простой анимации.

Разработана также группа методов, связанных либо с ***деформацией двумерной сетки,*** на которую помещен объект, либо ***с глобальной деформацией пространства,*** в котором задан объект. Часто используются различные модификации ***метода свободной деформации (FFD - free-form deformation),*** являющегося трехмерным расширением метода деформации двумерной сетки. Общая идея этих методов основана на том, что пользователю проще оперировать системой локальных координат, в которую помещен искажаемый объект, чем вершинами этого объекта. Поэтому, после определенной художником-аниматором деформации локальной системы координат, производится пересчет координат вершин искажаемого объекта в глобальное пространство.

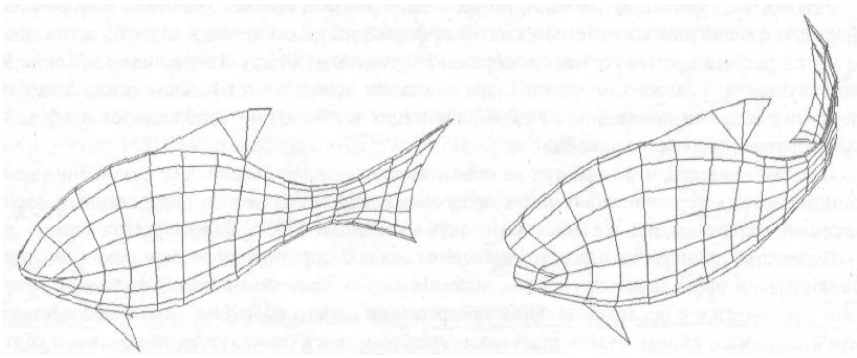
Метод анимации на основе событий. При этом методе "событиям и считаются изменения в состоянии того или иного параметра. В качестве параметров выступают предусмотренные конкретной анимационной программой элементы сцены (формы объектов, текстуры, параметры источников света, координаты камеры и т. д.). Для каждого параметра на временной шкале выделяется отдельная дорожка, что позволяет перемещать события вдоль временной шкалы или подвергать иным преобразованиям независимо друг от друга. При этом траектория объектов может быть нелинейной.

Метод вершинной (вертексной) анимации объектов связан с представлением объекта как цельной полигональной модели (еще говорят, что он должен представлять собой одну сетку). Тогда, например, лицевая анимация выполняется путем передвижения по заданной траектории определенных вершин полигональной модели, в то время как положение остальных вершин не меняется, тем самым осуществляется деформация полигональной модели (деформация сетки объекта). Таким образом создается ряд трехмерных моделей, отображающих последовательность движения "живого" объекта в различные моменты времени.

В качестве примера этого метода можно привести ***морфинг*** (термин происходит от слова ***metamorphosing*** - проведение преобразования). Морфинг заключается в последовательном превращении одного объекта в другой посредством перемещении по определенной траектории заданных точек (линий) одного объекта в соответствующие точки (линии) другого объекта в сочетании с наложением этих двух объектов.

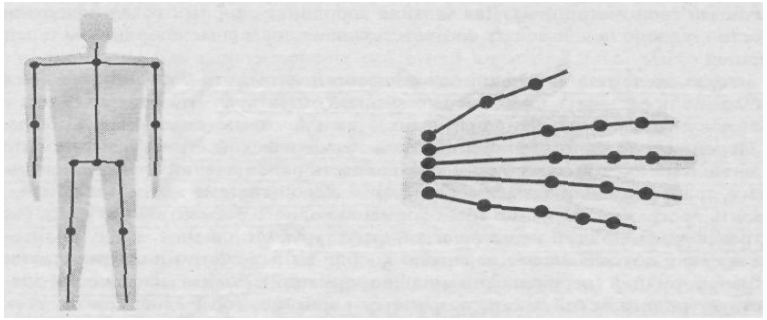
Метод взвешенно-целевого морфинга достаточно успешно применяется для реалистичной анимации лица. Такое название метод получил вследствие особенности обеспечения требуемого выражения лица. В соответствии с этим методом вначале подготавливаются ключевые состояния лица: улыбка, широко раскрытые глаза, насупленные брови и т. д. Затем указываются "весовые" доли для каждого из этих ключевых состояний, тем самым создается требуемое выражение лица.

Метод скелетной анимации - перемещение вершин полигональной модели осуществляется с помощью невидимых анимированных "костей" ***(bones),*** составляющих иерархическую структуру - "скелет" ***(skeleton).*** Для каждой кости задаются длина и некоторые параметры, характеризующие ее положение.



Сложность скелета определяется требуемым уровнем детализации изображаемого объекта. Например, при изображении человека, шагающего где-то вдали, достаточно показать основные движения рук и ног, в то время как для крупных планов, возможно, потребуется показать движения отдельных пальцев.

Скелет состоит из костей (звеньев) и сочленений. Каждая *i*-я кость описывается такими параметрами: длиной (l_i) и поворотами относительно родительской кости. Если поворот возможен только в одной плоскости, то говорят, что такое сочленение имеет одну вращательную степень свободы. Если повороты могут осуществляться в двух или трёх плоскостях, то это называют двумя или тремя степенями свободы.



Скелет имеет древовидную иерархическую структуру - с родительской костью соединяется одна или несколько костей, которые, в свою очередь могут являться родительскими для других соединенных с ними костями. Рассмотрим конструкцию из двух костей.

Зафиксируем систему трехмерных координат (x_0, y_0, z_0) в начале родительской кости (ось y_0 смотрит на нас). Угол поворота родительской кости (α_1) здесь отсчитывается от вертикали (хотя это не принципиально).

Найдем координаты произвольной точки ***P,*** связанной с концом второй кости:

$$\boldsymbol{P = R_1 \times T_1 \times R_2 \times T_2 \times P_2,}$$

где $\boldsymbol{P_2}$ - это координаты искомой точки, заданные в локальной системе координат (x_2, y_2, z_2), центр которой располагается в конце второй кости, $\boldsymbol{R_1}$ и $\boldsymbol{R_2}$ - матрицы поворотов на углы $\boldsymbol{\alpha_1}$ и $\boldsymbol{\alpha_2}$, $\boldsymbol{T_1}$ и $\boldsymbol{T_2}$ - матрицы сдвига вдоль оси *z* на длину костей.

Обобщим эту формулу для шарнирного соединения ***n*** костей.

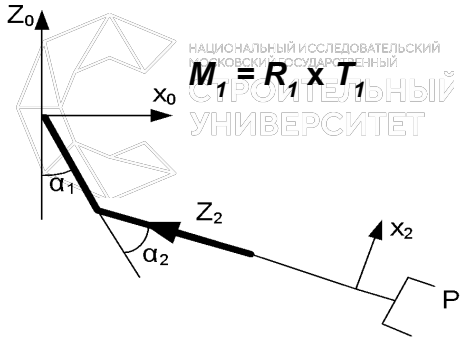
$$P = M_n * P_n,$$

где P_n — это координаты искомой точки в локальной системе координат, связанной с концом последней кости шарнира, M_n — матрица преобразований координат. Эту матрицу удобно вычислять рекурсивно:

$$M_i = M_{i-1} \times R_i \times T_i,$$

где $i = 1,2, \ldots, n$, причем

$$M_1 = R_1 \times T_1$$



Следует заметить, что в целях экономии времени расчета и объема памяти при анимации иногда используют простейший вариант использования скелета - каждая вершина полигональной сетки поверхности связывается только с одной костью. Однако лучшее качество, большая реалистичность достигаются при учете влияния на одну вершину нескольких костей. В этом случае для вершины задают степень влияния каждой кости с помощью коэффициента веса, сумма которых для одной вершины, как правило, должна быть равна единице. В соответствии со значениями этих коэффициентов выполняется интерполяция координат вершин полигонов (метод интерполяции вершин - ***vertex blending).***

Как подмножество метода интерполяции вершин может рассматриваться ***vertex skinning*** - метод трансформации вершин геометрической сетки в местах сгиба модели. Применяется метод ***vertex skinning*** для того, чтобы на стыках текстур (особенно на сгибах и сочленениях - это, например, все суставы модели человека или животного) были плавные, естественные переходы, особенно во время движения. Таким образом, при помощи этого метода осуществляется правильное расположение вершин. Для этого используются матрицы, называемые ***skinning matrices.*** Текстуры же на корректно трансформированную геометрию натягиваются правильно автоматически. Основная идея этого метода - интерполировать результаты матричного преобразования, используя веса, основанные на начальном местоположении каждой вершины. Это позволяет отдельному треугольнику сетки деформироваться (натягиваться подобно коже) и сохранять связь, например, с суставами, поскольку каждой вершине приписывают различный вес.

Таким образом, вычисление координат некоторой вершины ***V*** интерполяцией в соответствии с методом ***vertex blending*** можно описать так:

$$V = V_1 \times k + V_2 \times (1-k),$$

где V_1, V_2 - используемые вершины, а коэффициент ***k*** принимает значение в диапазоне от **0** до ***1.***

Метод vertex skinning - это vertex blending для вершин, обработанных разными матри**цами** (M_1,M_2):

$$V = V_1 \times M_1 \times k + V_2 \times M_2 \times (1-k).$$

В соответствии с ***методом прямой кинематики (Forward Kinematics)*** управляя углами поворота костей скелета, добиваются требуемых поз. Процесс подбора углов можно представить следующим образом. Движение родительского звена (например, ноги) автоматически приводит в движение всю цепь дочерних звеньев (в данном случае ступню), причем дочерние звенья будут перемещаться, не изменяя своего положения относительно объекта-предка. Если родительское, звено поворачивается, то дочернее соответственно и перемещается, и поворачивается, чтобы его ориентация по отношению к родительскому звену осталось прежней. Этот метод несколько утомителен для художника-аниматора, так как требует указывать множество углов в сочленениях.

При использовании ***метода обратной (инверсной) кинематики (Inverse Kinematics)*** исходными являются позы, а точнее, координаты концевых точек звеньев скелета. Исходя из этих координат, находятся соответствующие углы поворота всех костей. Движение задается перемещением самого младшего дочернего звена (в нашем случае ступни), что заставляет всю остальную цепочку (ногу, туловище, таз и т.д.) перемещаться. Как правило, расчет перемещений осуществляется с учетом ограничений на работу сочленений звеньев: например, вводятся приоритеты сочленений, их фиксация, угловые ограничения и трение в узлах сочленений и т.п. При этом метод обратной кинематики, в отличии от метода прямой кинематики, может дать несколько вариантов решения (или странные и непредвиденные решения) - это зависит от количества звеньев и ограничений.

Метод процедурной анимации применяется в тех случаях, когда моделирования движений трудно (неэффективно) воспроизводить с помощью ключевых кадров. При процедурной анимации текущие значения параметров анимации рассчитываются на основе заданных начальных значений и математических выражений, описывающих изменение параметров во времени. Процедурная анимация часто используется для качественной анимации разнообразных физических эффектов.

Аналогично применяется ***параметрическая анимация.*** *В* роли параметра может выступать любой объект - кривая, поверхность, точка, систем координат и т.д. Например, частоту или скорость движения объекта можно задать графиком, а затем анимировать этот график, изменяя тем самым параметры движения объекта.

При использовании **метода моделирования частиц (particles)** создается набор частиц (часто в качестве частиц используются точки, то есть объекты, не имеющие размеров). Для частиц могут быть заданы законы их существования, например, основанные на каких-то реальных физических законах, а именно, законах притяжения под действием силы тяжести, электростатических, магнитных сил, и т.п.

Метод канальной анимации (channel animation) основан на снятии информации о каком-либо параметру объекта с датчика (канала). Например, для снятия информации о движении актера датчики крепятся по всему его телу в тех местах, которые будут приведены в соответствие с контрольными точками компьютерной модели для ввода и оцифровки движения, а приемники информации подключены к компьютеру. Датчики могут быть разных видов, например, электромеханическими, электромагнитными (беспроводными или соединяющимися с компьютером проводами) или оптико-электронными, информацию с которых считывают специальные оптические устройства, подключенные к компьютеру. Следует заметить, что в настоящее время беспроводные датчики используются реже, так как для снабжения их энергией актеру приходится носить на себе аккумулятор. Оцифрованные движения реального человека служат для создания моделей, изображающих компьютерный персонаж.

На этом методе основано отдельное направление в анимации - ***технология real time performance animation,*** основанная ***на захвате (видеозахвате) движения (Motion Capture),*** который дает возможность передавать естественные, реалистичные движения в реальном времени. Для захвата движений часто используют пассивные оптические метки и видеотехнологию для записи движений объекта. В этом случае актеру приходится носить только отражающие свет метки, закрепленные на одежде. Естественно, что качество синтезированного движения напрямую зависит от количества и расположения датчиков.

3.7. Текстурирование в компьютерной графике

Текстура представляет собой двумерное растровое изображение, которое накладывается (натягивается) на поверхность объекта, например на плоский треугольник. Текстуры, как правило, хранятся в графических файлах форматов bmp, jpeg,tiff, tga, gif.

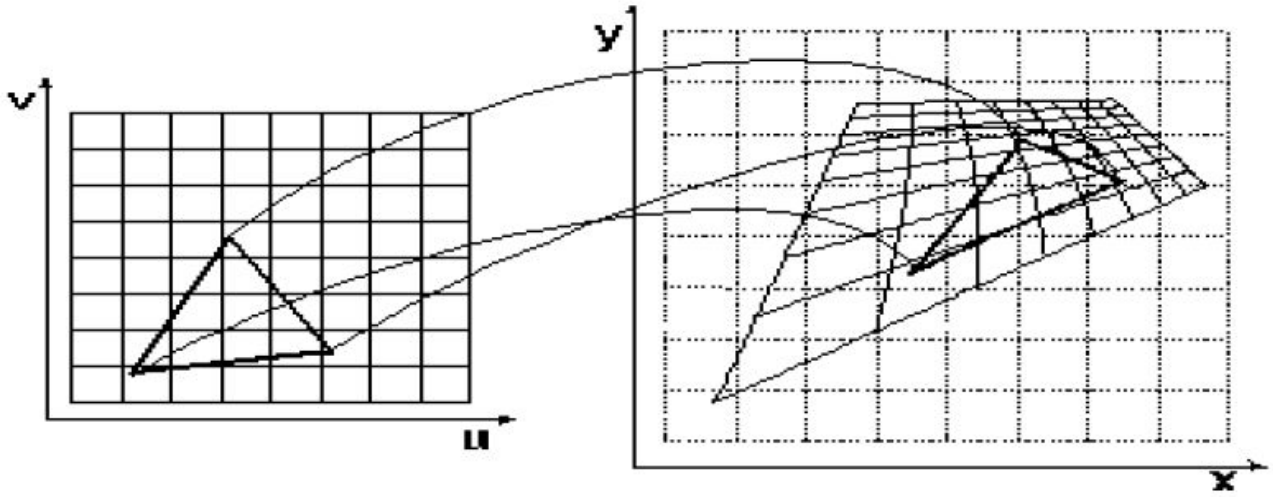
Наложение текстуры или ***текстурирование*** (texture mapping) – это метод, посредством которого на поверхность объекта накладывается некоторое изображение, называемое изображением текстуры.

В общем контексте графического конвейера этот метод открывает огромные возможности, но простота идеи метода наложения текстуры весьма обманчива.

Технология текстурирования заключается в проецировании изображения (текстуры) на трехмерную поверхность. Таким образом обеспечивается дополнительная детализация 3D объекта без усложнения его геометрии.

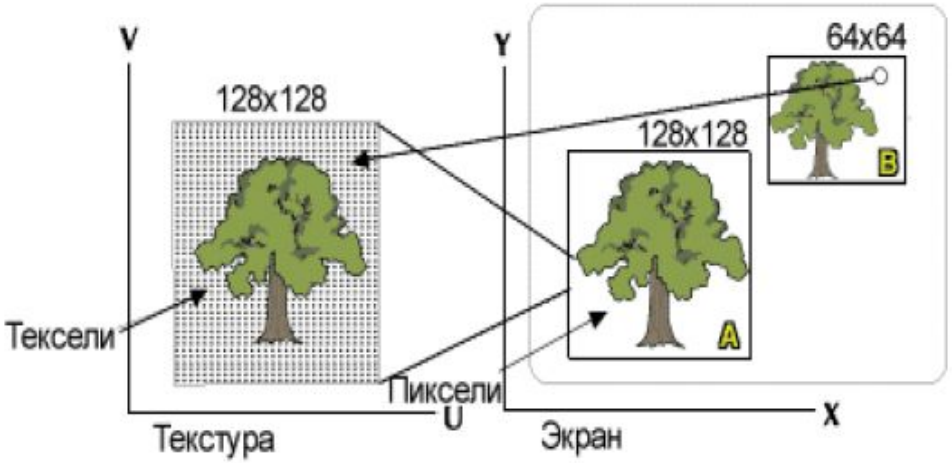
Фильтрация текстур – это механизм, с помощью которого происходит наложение текстуры на полигоны отличающегося размера. Наиболее распространенными по использованию являются следующие типы фильтрации текстур:

- точечная фильтрация (используется по умолчанию) – самая быстрая по скорости, но самая низкая по качеству;
- линейная фильтрация – приемлемое качество и скорость;
- анизотропная – самая медленная, но самая качественная.



Когда изображение используется в качестве текстуры, накладываемой на 3D примитив, проявляется множество разнообразных ошибок визуализации, называемых ***артефактами.***

Сэмплинг (point-sampling) – простейший метод текстурирования, в котором **текселы** непосредственно переносятся в **пикселы** изображения с учетом масштаба. Методу присущ серьезный артефакт: когда наблюдатель приближается вплотную к текстурированной поверхности, происходит пикселизация. Для избежания этого артефакта используют методы текстурирования, основанные на фильтрации текстур.

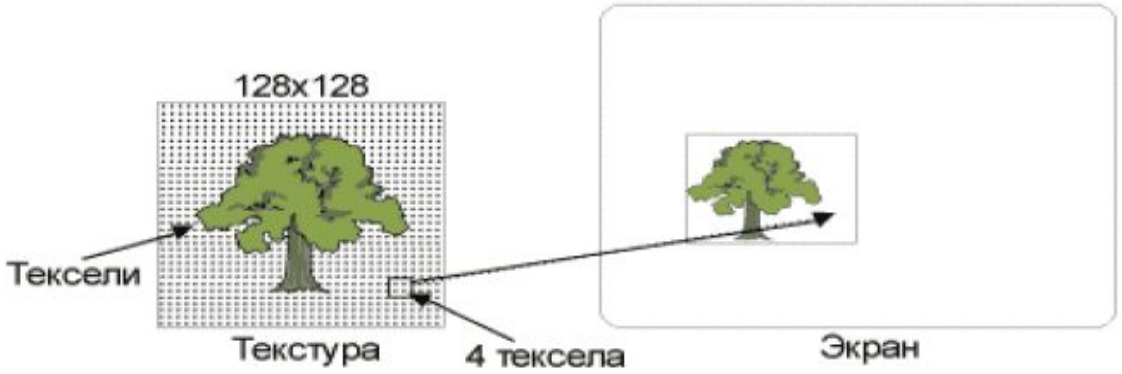


A: однозначное соответствие между текселами и пикселами
B: pixel 0:texel 0; pixel 1:texel 2; pixel 2:texel 4...

Bi-linear filtering - это техника устранения искажений изображения (фильтрация), таких, как "блочность" текстур при их увеличении.

При билинейной фильтрации, в качестве цвета каждого пиксела берется взвешенное среднее значение (линейная интерполяция) цвета четырех смежных текселов. **Результирующий цвет пиксела определяется в результате операций смешивания:**

1. сначала смешиваются цвета двух пар текселов по x,
2. а потом смешиваются два полученных цвета по y.



Существует класс артефактов наложения текстур известный под названием "***depth aliasing***« (ошибки глубины сцены или Z-aliasing), от которых билинейная фильтрация не избавляет и не может избавить. Ошибки "depth aliasing" возникают из-за того, что объекты более отдаленные от наблюдателя, выглядят более маленькими на экране.

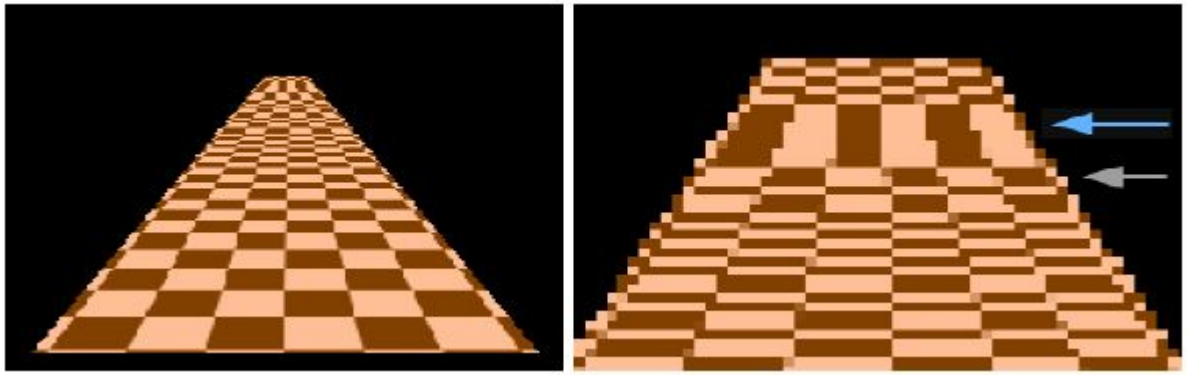
Эти ошибки визуализации особенно нежелательны в анимации, где такие артефакты становятся причиной мерцания и эффекта медленного движения в той части изображения, которая должна быть неподвижной. Как только вертикальная сторона квадрата (высота) сокращается до двух пикселей, появляются артефакты "depth-aliasing" - несколько квадратов сливаются в один.

Одним из способов устранения depth aliasing, имеющим и самостоятельное значение, является ***перспективная коррекция.*** Перспективная коррекция – ресурсоемкая процедура (одна операция деления на каждый пиксел), поэтому 3D-ускорители реализуют ее аппаратно. Но разные ускорители достигают разного качества перспективной коррекции.

Для избежания ошибок "depth aliasing" используется техника, известная как ***mip-mapping.*** Если говорить кратко, то ***mip-mapping*** — наложение текстур, имеющих разную степень или уровень детализации, когда в зависимости от расстояния до точки наблюдения выбирается текстура с необходимой детализацией.

Mip-текстура (mip-map) состоит из набора заранее отфильтрованных и масштабированных изображений. В изображении, связанном с уровнем mip-map, пиксель представляется в виде среднего четырех пикселей из предыдущего уровня с более высоким разрешением. Отсюда, изображение связанное с каждым уровнем mip-текстуры в четыре раза меньше по размеру предыдущего mip-map уровня.

Степень или уровень детализации — *Level of Detail или просто LOD*, используются для определения, какой mip-map уровень (или какую степень детализации) следует выбрать для наложения текстуры на объект. LOD должен соответствовать числу текселей накладываемых на пиксель (т.е. какое количество элементов текстуры должно быть наложено на элемент выводимого на экран изображения). Например, если текстурирование происходит с соотношением близким к 1:1, то LOD будет 0, а значит и будет использоваться mip-map уровень с самым высоким разрешением. Если 4 текселя накладываются на один пиксель, то LOD будет 1 и будет использоваться следующий mip уровень с меньшим разрешением. Обычно, при удалении от точки наблюдения, объект, заслуживающий наибольшего внимания имеет более высокое значение LOD.



Объем MIP-текстуры:

$$1+\frac{1}{4}+\frac{1}{4^2}+\frac{1}{4^3}+\ldots\leq\frac{1}{1-1/4}=\frac{4}{3}$$

Слева направо мы имеем MIP-map уровни детализации 0, 1, 2 и т.д. Чем меньше становится изображение, тем больше теряется деталей, вплоть до самого маленького, когда не видно ничего, кроме расплывающегося пятна из серых пикселей.

Проблемы Mip-текстурирования

В то время, как mip-текстурирование решает проблему ошибок "depth-aliasing", его использование может стать причиной появления других артефактов. Для борьбы с этими артефактами используются различные техники фильтрации. При удалении объекта все дальше от точки наблюдения, происходит переход от низкого mip-map уровня (соответствующего изображению с высокой детализацией) к высокому mip-map уровню (соответствующего изображению с высокой степенью фильтрации и низкой детализацией). В момент нахождения объекта в переходном состоянии от одного mip-map уровня к другому, появляется особый тип ошибок визуализации, известных под названием "***mip-banding" (мип-бендинг)*** — полосатость или слоеность, т.е. явно различимые границы перехода от одного mip-map уровня к другому.

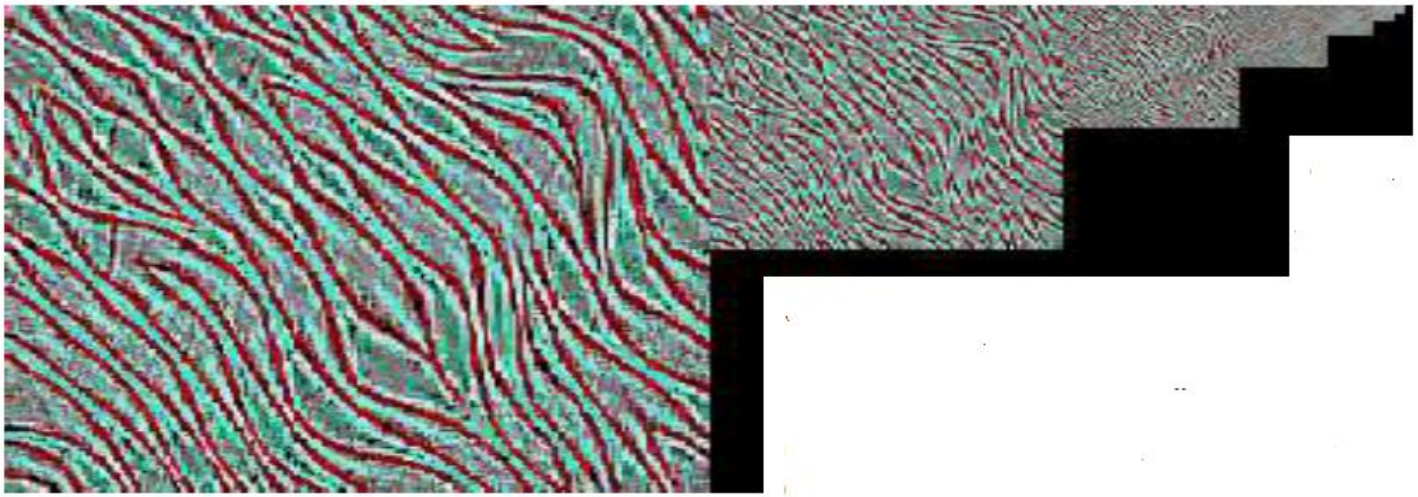
"Mip-banding" имеет место, когда значение LOD округляется до целого и только один соответствующий уровень mip-текстуры используется для генерации пикселя. После перехода LOD к более высокому уровню состоящему из треугольников, соседние пиксели будут генериться с различными mip уровнями и будут иметь совершенно различное количество примененных к ним фильтров. Результатом является появление линии или нескольких линий проходящих через треугольник в местах, где происходит внезапный переход между mip уровнями. Это один из недостатков использования только билинейного или поточечного mip-текстурирования.

Особенно остро проблема наличия ошибок "mip-banding" стоит в анимации, за счет того, что человеческий глаз очень чувствителен к смещениям и может легко заметить место резкого перехода между уровнями фильтрации при движении вокруг объекта.

Трилинейная фильтрация

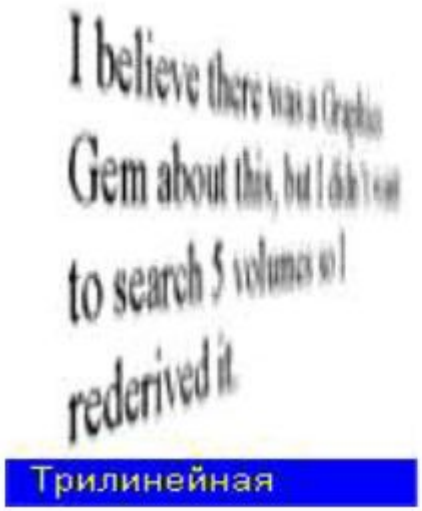
Трилинейная фильтрация (trilinear filtering) представляет собой технику, которая удаляет артефакты "mip-banding", возникающие при использовании mip-текстурирования. При трилинейной фильтрации для определения цвета пикселя берется среднее значение цвета восьми текселей, по четыре из двух соседних текстур и в результате семи операций смешивания определяется цвет пикселя. При использовании трилинейной фильтрации возможен вывод на экран текстурированного объекта с плавно выполненными переходами от одного mip уровня к следующему, что достигается за счет определения LOD путем интерполяции двух соседних mip-map уровней. Таким образом решается большинство проблем, связанных с mip-текстурированием и ошибками из-за неправильного расчета глубины сцены ("depth aliasing").

Пример использования трилинейной фильтрации приведен ниже. Здесь опять используется все тот же прямоугольник, текстурированный волнообразным изображением, но с плавными переходами от одного mip уровня к следующему за счет использования трилинейной фильтрации. Обратите внимание на отсутствие каких-либо заметных ошибок визуализации.



Анизотропная фильтрация обычно оперирует не менее чем 8 текселями, во все стороны mip-map уровней, при этом используется модель неопределенной заранее формы. В результате убираются шумы и искажения объектов, а изображение в целом получается более качественным. Анизотропная фильтрация работает с текселами как с эллипсами и для получения одного пиксела обрабатывает до 32 (2x16) текселов. Качество текстуры при анизотропной фильтрации даже на дальних дистанциях остается схожей с оригинальным; при изотропной фильтрации же видна тенденция в "сглаживанию" изображения, в результате теряется качество.

Анизотропная фильтрация, как и трилинейная, уменьшает неровность текстур. Но при использовании анизотропной фильтрации качество получается лучшим.



Трилинейная

Анизотропная

Глава 4. ФОРМАТЫ ГРАФИЧЕСКИХ ФАЙЛОВ

Формат графического файла - способ представления и расположения графических данных на внешнем носителе.

4.1. Векторные форматы

Эти графические форматы служат для хранения изображений в виде совокупности геометрических примитивов - линий, дуг, прямоугольников, эллипсов и т. п. Графические форматы этого типа либо состоят из списка примитивов, либо содержат в себе набор инструкций, команд для построения примитивов. Не исключена и комбинация этих способов. В векторном виде хранят информацию системы автоматизированного проектирования, например, AutoCAD, программы, создающие иллюстративную графику, такие как CorelDraw. Векторные плоттеры обрабатывают изображения только в векторных форматах. Векторные форматы могут содержать также либо введенные в файл растровые объекты, либо ссылки на растровые файлы (технология OPI).

Название формата	Программы, которые могут открывать файлы
WMF Windows MetaFile	Большинство приложений WINDOWS
EPS Encapsulated PostScript	Большинство настольных издательских систем и векторных программ, некоторые растровые программы
DXF Drawing Interchange Format	Все программы САПР, многие векторные редакторы, некоторые настольные издательские системы
CGM Computer Graphics Metafile	Большинство программ редактирования векторных рисунков, САПР и издательские системы

DXF (Drawing Exchange Format)

Формат *DXF* разработан фирмой AutoDesk в 1982 году для обмена чертежами и другими графическими документами в среде AutoCAD. Несмотря на возраст этого формата и его недостатки, DXF сейчас поддерживается многими программами как формат обмена данными.

Файл DXF состоит из пар:

<код группы>
<команда>

Код группы служит для пояснения того, что помешается в следующей строке. Например, последовательность пар:

0 (означает начало нового элемента) LINE
10 (далее будет значение первой координаты X)
12.354
20 (первая координата Y)
-34
11 (следующая координата X
23.08
21 (следующая координата Y)
5.7
определяет линию.

Коды группы:

Код группы	Что означает
0	Начало элемента (LINE, CIRCLE, BLOCK, TABLE и т.п.)
1	Текстовые значения
2	Имя элемента
3-5	Другие текстовые значения или имена
6	Тип линии
7	Имя шрифта
8	Имя слоя
9	Имя переменной
10	Первая координата X
11-18	Другие координаты X
20	Первая координата Y
21-28	Другие координаты Y
30	Первая координата Z
31-37	Другие координаты Z
38	Уровень объекта
4048	Размеры
…	…
999	Комментарий

Общая структура файла DXF такая:

Заголовок (секция HEADER)

Таблицы (секция TABLES)

Примитивы (секция ENTITIES)

Блоки (секция BLOCKS)

Завершитель (EOF)

Каждая секция описывается так:
0
SECTION
……..(далее идут элементы секции)
0
ENDSEC

4.2. Растровые форматы

Растровые форматы служат для описания растровой графической информации. Каждый отдельный пиксел изображения представляет самого себя, вне зависимости от его расположения и роли, которую он играет в рисунке. Наиболее распространенные из них: TIFF, BMP, PCX, GIF, JPEG, PNG.

Формат	Фирма-Разработчик	Максимальное количество цветов (бит на пиксел)	Максимальный размер изображения	Метод сжатия	Запись нескольких изображений
BMP	Microsoft	16'777'216 (24 бита)	65535x65535	RLE (ис пользуется редко)	-
GIF	CompuServe	256 (8 бит)	65535x65535	LZW	+
JPEG	Joint Photographic Experts Group	16'777'216 (24 бита)	65535x65535	JPEG	-
PCX	Z-Soft	16'777'216 (24-бита)	65535x65535	RLE	-
PNG	WZC	281'474'976'710'656 (48 бит)	2147483647* 2147483647	Deflate	-
TGA	Truevision	4'294'967'296 (32 бита)	65535x65535	RLE	-
TIFF	Aldus Corporation	16'777'216 (24-бита)	Всего 4'294'967'295	LZW, RLE, JPEG и прочие	+

В файлах растровых форматов запоминаются:

• размер изображения - количество видеопикселей в рисунке по горизонтали и вертикали

• битовая глубина - число битов, используемых для хранения цвета одного видеопикселя

• данные, описывающие рисунок (цвет каждого видеопикселя рисунка), а также некоторая дополнительная информация.

BMP (Windows Device Independent Bitmap)

Родной формат Windows. Он поддерживается всеми графическими редакторами, работающими под управлением этой операционной системы. Применяется для хранения растровых изображений, предназначенных для использования в Windows и, по сути, больше ни на что не пригоден. Способен хранить как индексированный (до 256 цветов), так и RGB-цвет (более 16 млн. оттенков). Возможно применение сжатия по принципу RLE, но делать это не рекомендуется, так как очень многие программы таких файлов (они могут иметь расширение .rle) не понимают. Существует разновидность формата BMP для операционной системы OS/2.

Общая структура BMP-файла такая:

Заголовок файла BMP называется **BITMAPFILEHEADER.** В нем помешается общее описание файла. Еще один заголовок - **BITMAPINFOHEADER,** в котором хранится описание размеров растра и цветового формата пикселов . Далее в файле помешается палитра в виде нескольких записей . Далее – непосредственно битовый массив.

OPI (Open Prepress Interface) технология, разработанная фирмой Aldus, позволяющая импортировать не оригинальные файлы, а их образы, создавая в программе лишь копию низкого разрешения (эскиз) и ссылку на оригинал. В процессе печати на принтер, эскизы подменяются на оригинальные файлы. Применение OPI, вместо простого внедрения, (embedding) дает возможность экономить ресурсы компьютера (прежде всего, память), заметно повышая его производительность. OPI является основной работы с импортированными графическими файлами в таких программах, как FreeHand и QuarkXPress, широко применяется в других продуктах.

Методы сжатия графических данных

При ***сжатии методом RLE (Run - Length Encoding)*** последовательность повторяющихся величин (в нашем случае - набор бит для представления видеопикселя) заменяется парой - повторяющейся величиной и числом её повторений.

RLE - один из самых старых и простых алгоритмов компрессии графики. Основная его идея такова: если в строках растра встречаются цепочки одинаковых пикселов, например

2 2 2 2 2 2 2 135 11 11 11 11,

их можно заменить цепочкой из пары чисел - <счетчик повторений, значение>. Для отдельных пикселов, не входящих в цепочки, счетчик не нужен:

<7 2> 135 <4 11>.

Чем больше цепочек в растре и чем они длиннее, тем больше эффект сжатия.

Метод сжатия LZW основан на поиске повторяющихся узоров в изображении.

Алгоритм LZW (Lempel-Ziv-Welch) разработан в 1984 г. Уелшем (Terry A. Welch). Этот алгоритм при

Приведем пример кодирования. Строка байтовых данных

241 16 72 10 10 10 10 10 241 16 72 13 5

преобразуется в

241 16 72 10 259 259 256 72 13 5,

В таблице приведено содержимое словаря для этой строки.

Код (индекс)	Содержимое (S + c)	Компактная запись	
		код S	c
0	0		0
1	1		1
…	…	…	…
255	255		255
256	241 16	241	16
257	16 72	16	72
258	72 10	72	10
259	1010	10	10
260	10 1010	259	10
261	10 10 241	259	241
262	241 16 72	256	72
263	72 13	72	13
264	135	13	5

Метод сжатия JPEG обеспечивает высокий коэффициент сжатия для рисунков фотографического качества. Формат файла **JPEG,** использующий этот метод сжатия, разработан объединенной группой экспертов по фотографии (**J**oint **P**hotographic **E**xperts **G**roup). Сжатие по методу **JPEG** сильно уменьшает размер файла с растровым рисунком (возможен коэффициент сжатия 100:1). Высокий коэффициент сжатия достигается за счет сжатия с потерями, при котором в результирующем файле теряется часть исходной информации. Метод **JPEG** использует тот факт, что человеческий глаз очень чувствителен к изменению яркости, но изменения цвета он замечает хуже.

Поэтому при сжатии этим методом запоминается больше информации о разнице между яркостями видеопикселей и меньше - о разнице между их цветами. Так как вероятность заметить минимальные различия в цвете соседних пикселей мала, изображение после восстановления выглядит почти неизменным. Пользователю предоставляется возможность контролировать уровень потерь, указывая степень сжатия.

Кодирование осуществляется за несколько шагов.

Шаг 1. 24-битное изображение из RGB преобразуется в цветовую модель *YCbCr.* Известно, что канал яркости *Y (luminance)* содержит значительно больший объем информации сравнительно с двумя цветовыми каналами *Cb* и *Cr (chrominance blue* and *chrominance red),* поэтому в телевизионных системах каналу *Y* предоставляется больший диапазон частот. В JPEG наибольшему сжатию подвергаются компоненты *Cb* и *Cr.*

Шаг 2. *Изображение разделяется на блоки размером 8x8 пикселов, и каждый такой блок подвергается двумерному дискретному косинусному преобразованию (ДКП) - Discrete Cosine Transform (DCT).*

Преобразование выполняется в отдельности для компонентов *Cb* и *Cr,* причем для *Y* обрабатывается блок пикселов размерностью 8x8, а для *Cb* и *Cr* делается прореживание - берется каждый второй пиксел. Что дает ДКП? Оно преобразует пространственное распределение в частотное (подобно преобразованию Фурье). В результате получаем блоки также размерами 8*8, причем коэффициенты для нижних частот располагаются в левом верхнем углу, а для высоких частот - в правом нижнем углу. Поскольку основная энергия в спектре - у нижних частот, то максимальное числовое значение будет в левом верхнем углу.

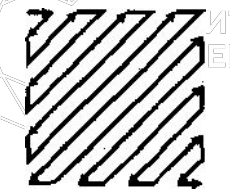
Шаг 3. Квантование. Каждый элемент блока 8*8 после ДКП будет делиться на соответствующий элемент матрицы квантования

После деления выполняется округление до ближайшего целого числа. В стандарте JPEG имеются рекомендованные таблицы квантования отдельно для *Y*, *Cb* и *Cr.* Значения коэффициентов квантования меньше в левой верхней части таблиц по сравнению с коэффициентами в правой нижней части. Фактически таблица квантования определяет цифровой фильтр, который ослабляет верхние частоты.

Указанные в стандарте таблицы не являются обязательными - любая программа файлов JPEG может использовать собственные таблицы (понятно, что тогда эти таблицы должны быть записаны в файл, иначе невозможно декодирование).

Квантование - это основной фактор сжатия в соответствии с методом JPEG. В результате деления и округления большинство элементов блока 8*8 будет равняться нулю, что способствует компактности записи блока. Чем больше значение таблицы квантования, тем вероятнее преобразование результата деления в нуль. Однако это приводит к потере информации - чем больше значащих элементов преобразуется в нуль, тем меньше результат декодирования будет похож на исходное изображение.

Шаг 4. Элементы блока 8x8 записываются в виде одномерного вектора байтов. Элементы ты выбираются "зигзагом", как показано на рис.



Шаг 5. Кодирование вектора. Сначала вектор кодируется методом RLE. Получает код в виде таких пар: <счетчик, значение>. Здесь счетчик задает количество нулей, которые записываются перед байтом значения. Потом выполняется кодирование пар методом Хаффмана. В файл JPEG, кроме сжатых кодов, записывается также таблица Хаффмана для обеспечения дальнейшего декодирования.

В соответствии с методом Хаффмана сначала для каждого символа вычисляется вероятность его появления. Потом символу приписывается битовый код, длина которого зависит от этой вероятности. Символы, которые чаще встречаются, получают более короткий битовый код, а символы, которые встречаются реже - длинный код

Название формата	Программы, которые могут открывать файлы	Метод сжатия
BMP Windows Device Independent Bitmap	Все программы WINDOWS, которые используют растровую графику	RLE для 16- и 256- цветных изображений (по желанию)
PCX Z - Soft PaintBrush	Почти все графические приложения для PC	RLE (всегда)
GIF Graphic Interchange Format	Почти все растровые редакторы; большинство издательских пакетов; векторные редакторы, поддерживающие растровые объекты	LZW (всегда)
TIFF Tagged Image File Format	Большинство растровых редакторов и настольных издательских систем; векторные редакторы, поддерживающие растровые объекты	LZW (по желанию) и др.
TGA TrueVision Targa	Программы редактирования растровой графики	RLE (по желанию)
IMG Digital Research GEM Bitmap	Некоторые настольные издательские системы и редакторы изображений WINDOWS	RLE (всегда)
JPEG Joint Photographic Experts Group	Последние версии программ редактирования растровой графики; векторные редакторы, поддерживающие растровые объекты	JPEG (можно выбрать степень сжатия)

4.3. 3D-форматы

Предназначены для хранения данных 3d-геометрии. К форматам относятся *VRML, 3DS, MDL, MD2 и MD3, SMD, X, MAX* и др.

Формат VRML

VRML (язык моделирования виртуальной реальности - *Virtual Reality Modeling Language)* - графический формат, который базируется на подмножестве Open Inventor фирмы Silicon Graphics. Он предназначен для описания трехмерных изображений и обмена ими в сети World Wide Web.

Язык VRML стал первым языком трехмерного моделирования для Web. VRML-файл имеет расширение .WRL. Он использует формат ASCII и представляет собой обычный текстовый файл со списком объектов, которые названы узлами (nodes). К узлам VRML 2.0, в частности, относятся 3D-геометрия, свойства света, который создается с помощью VRML, файлы изображений формата JPEG, видеофайлы формата MPEG, звуковые файлы формата MIDI, текстовые документы формата HTML.

Перечень некоторых узлов VRML 2.0 приведен в таблице.

Узлы VRML 2.0	Состав
Grouping Nodes - группируемые узлы (узлы, которые могут иметь в себе несколько узлов)	Anchor, Billboard, Collision, Transform
Special Groups - узлы специальной группы	Inline, LOD, Switch
Common Nodes - общие узлы (узлы, которые могут входить в группируемые узлы)	AudioClip, DirectionalLight, PointLight, Script, Shape, Sound, SpotLighl, WorldInfo
Sensors - узлы сенсоров	CylinderSensor, PlaneSensor, Proximity-Sensor, SphereSensor, TimeSensor, Touch-Sensor, VisibilitySensor
Geometry - узлы геометрии	Box, Cone, Cylinder, Extrusion, ElevationGrid, IndexedFaceSet,

4.4. Форматы мультимедиа

Один из приемов, который используется при работе с видеоданными, состоит в ***вычислении последовательных различий,*** а именно - в простом вычитании данных следующего кадра из данных предшествующего и сжатии полученных разностей. Как правило, кадры, идущие друг за другом, во многом похожи, поэтому большие фрагменты изображений будут давать нулевую разность. Однако в ряде случаев этот прием оказывается неэффективным, например, при медленном панорамировании сцены с множеством мелких объектов.

Более эффективным приемом является ***предсказание движения****,* а именно - программа кодирования ищет блоки пикселов, которые перемещаются, и кодирует только координаты этого блока и его перемещение. Естественно, сравнительно с декодированием процесс самого кодирования (сжатия) при предсказании движения значительно сложнее реализовать. В результате некоторые наилучшие алгоритмы сжатия видеоданных *асимметричны,* то есть процесс сжатия требует намного больше усилий (времени), чем процесс декодирования. Следует отметить, что, в данном случае, асимметричность алгоритмов не является важным недостатком из-за того, что сжатие видеоданных осуществляется, как правило, один раз в конце работы над ними. Главное, чтобы при просмотре видеоданных обеспечивалась достаточная скорость декодирования.

Формат AVI (Audio Video Interleaved), являющийся специальным форматом представления видеофайлов в операционных системах семейства Windows и поэтому широко использующийся в персональных компьютерах. Этот формат создает отдельные моментальные фотографии, фреймы, и потом связывает их в единое целое, причем аудиосигнал может быть частью фреймов. В отличие от других форматов, формат AVI служит как бы оболочкой для программ сжатия видеоданных. Видов сжатия, которые подходят для файлов AV1, очень много. Для создания файла AVI необходимо Использовать соответствующее оборудование, причем возможно использование традиционной аналоговой видеокамеры с последующей оцифровкой аналогового видеосигнала.

Формат видеофайлов MPEG (Motion Picture Experts Group), сжимающий видеофайлы по алгоритмам, удаляющим избыточную информацию, в частности использующим вычисление последовательных различий и предсказание движения. Так, например, если на протяжении нескольких секунд демонстрируется один и тот же неподвижный объект, то в MPEG файле за это время будут сохранены только один фрейм и информация о том, на протяжении какого времени в этом фрейме не происходили изменения. Именно поэтому размеры MPEG файлов меньше в среднем в три раза, чем размеры других видеофайлов. Обычно для их создания и воспроизведения нужно особое аппаратное и программное обеспечение.

Для удобства кодирования видеоданных весь видеопоток разбивается на группы, которые названы ***GOP*** *(Group of Pictures- группа изображений).* GOP строится, например, следующим способом:

I B B P B B P B B I,

где *I - Independent* (независимые, опорные) кадры. MPEG-последовательности без этих кадров не может быть. Опорные кадры содержат всю информацию об изображении и не требуют для декодирования никакой дополнительной информации. При компрессии I-ых кадров с помощью метода типа JPEG происходит удаление только пространственной избыточности. Именно с этого кадра начинается декодирование изображения в последовательности.

P - Predictive кадры. "Предсказанные" кадры, при формировании которых используется разность между предшествующим I - или P-кадром и текущим кадром. Эта разность кодируется и вместе с вектором движения прибавляется к сжатым данным, P-кадр создается с помощью межкадровой компрессии, которая уменьшает как пространственную, так и временную чрезмерность. То есть, при создании P-кадра происходит сжатие, которое учитывает избыточность последовательных кадров. Этот кадр также служит для дальнейшего предсказания изображения.

B - Bi-directional. "Двунаправленные" кадры. Сохраняют только наиболее важную информацию про соседние I- или P-кадры. То есть B-кадры являются промежуточными интерполированными кадрами и имеют высочайшую степень компрессии. Наличие 3-кадров в видеофильме - фактор, благодаря которому MPEG-1 имеет высокий коэффициент сжатия (но не очень высокое качество). Формирование разных 5-кадров в рамках одной GOP может происходить по разным правилам. Разные B-кадры могут использовать информацию (воз можно, используя сжатие разности) или об одном, или о двух кадрах (предшествующем и последующем),

Формат QuickTime, разработанный фирмой Apple для операционных систем семейств Macintosh и Windows. QuickTime - расширение программного обеспечения System Macintosh. Программы, поддерживающие QuickTime, отображают анимационные или видеоряды точно синхронизированными с высококачественным цифровым звуком. В операционной системе Windows видеофайлы формата QuickTime имеют расширение .MOV. С 1993 по 1998 г. этот формат был доминирующим. Его версии с номером выше 4.1 позволяет передавать данные в потоковом режиме. Это значит, что нет необходимости целиком загружать файл, чтобы начать просмотр видеофильма. Однако с появлением спецификаций MPEG данный формат постепенно теряет популярность. Одна из проблем состоит в закрытости стандарта QuickTime. Способы, с помощью которых кодируется видео, Apple держит в секрете.

О некотоых других форматах

Adobe PostScript

PostScript - язык описания страниц (язык управления лазерными принтерами) фирмы Adobe. Был создан в 80-х годах для реализации принципа WYSIWYG (What You See is What You Get). Файлы этого формата представляют из себя программу с командами на выполнение для выводного устройства. Они имеют расширение .ps или, реже, .prn и получаются с помощью функции Print to File графических программ при использовании драйвера PostScript-принтера. Такие файлы содержат в себе сам документ (только то, что располагалось на страницах), все связанные файлы (как растровые, так и векторные), использованные шрифты, а так же другую информацию: платы цветоделения, дополнительные платы, линиатуру растра и форму растровой точки для каждой платы и другие данные для выводного устройства.

PDF (Portable Document Format)

PDF предложен фирмой Adobe как независимый от платформы формат для создания электронной документации, презентаций, передачи верстки и графики через сети.

PDF-файлы создаются путем конвертации из PostScript-файлов или функцией экспорта ряда программ. Для конвертации используется программа Adobe Acrobat Distiller, это лучший способ создания PDF. Создание PDF методом экспорта из программ дает, как правило худший результат - файлы получаются более тяжелыми, часто имеют проблемы со встраиванием шрифтов.

Для создания PDF так же существует программа PDFWriter, работающая как виртуальный принтер. PDFWriter не основан на PostScript и не может корректно обрабатывать графику. Он предназначен для быстрого изготовления простых текстовых документов. У него наблюдается та же проблема со встраиванием шрифтов, что и многих программ, умеющих экспортировать PDF. Самые надежные и максимально близкие к оригиналу PDF создает из PostScript и EPS-файлов программа Acrobat Distiller, поставляемая в пакете Adobe Acrobat.

RTF (Microsoft Rich Text Format)

Текстовый формат RTF имеет неординарные способности к переносу текстов из одной программы в другую. Он позволяет передавать форматированный текст из программ оптического распознавания символов или текстовых редакторов в графические программы или в любых других направлениях. RTF может оказаться хорошим решением (а, иногда, и единственным выходом) при переброске из программы в программу нелатинского, например, ивритского текста или русского в разных версиях ОС Windows.

Секрет совместимости заключается в использовании специальных тегов форматирования RTF и Unicode. Именно Unicode позволяет легко переносить русские тексты с PC на Мак и обратно в файлах MS Word. RTF используется как основной в поставляемом вместе с Mac OS X редакторе TextEdit и в прилагаемом к Windows программе WordPad.

Глава 5. СТАНДАРТИЗАЦИЯ В КОМПЬЮТЕРНОЙ ГРАФИКЕ

Основная цель стандартизации - переносимость графических систем, которая достигается стандартизацией интерфейса между графическим ядром системы (базовой графической системой), реализующим собственно графические функции, и моделирующей системой - проблемно-ориентированной прикладной программой, использующей функции графического ядра. Базовая система должна обладать: независимостью от вычислительных систем; независимостью от языков программирования; независимостью от области применения; независимостью от графических устройств.

Структура прикладной графической системы, удовлетворяющей сформулированным требованиям

Процесс преобразования графической информации при выполнении вывода может быть представлен состоящим из следующих стандартизованных этапов:

1. Модельные преобразования. Проблемно-ориентированный уровень из геометрических моделей отдельных объектов, задаваемых в собственных локальных системах координат, формирует описание совокупного объекта в некоторой единой (мировой) системе координат.

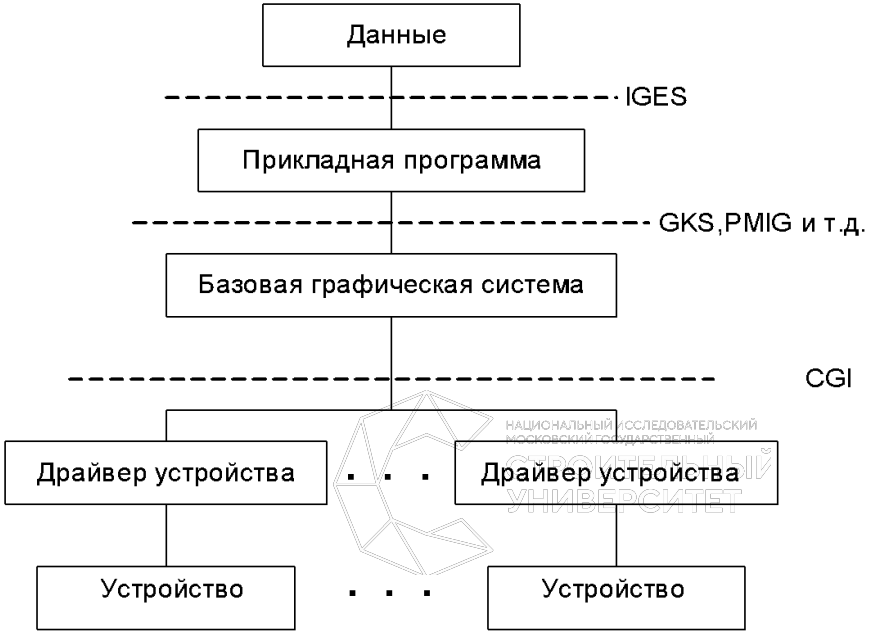
2. Нормализующие преобразования. Графическая система переводит описание из мировой, вообще говоря произвольной, системы координат в т.н. нормализованные координаты устройства, имеющие фиксированные пределы изменения координат, например, от 0.0 до 1.0.

3. Преобразования сегментов. Если графическая система предоставляет средства манипулирования отдельными подкартинами изображения (часто именуемыми сегментами), например, для независимого размещения отдельных самостоятельных частей изображения, то могут потребоваться такие преобразования.

4. Видовые преобразования. В случае 3D описания изображения и 2D устройства вывода необходимо выполнить проецирование изображения на заданную картинную плоскость. Наоборот, при 2D сцене и 3D устройстве вывода необходимо выполнить преобразование, связанное с размещением изображения.

5. Преобразование рабочей станции. Для выполнения вывода на конкретное устройство необходимо преобразование данных из аппаратно-независимой формы в координаты устройства.

Схема преобразований координатной информации в графической системе



Архитектура переносимой графической системы

Главными организациями формирующими международные стандарты в области информационной технологии являются ISO (International Organization for Standartization) и IEC (International Electrotechnical Commission). В конце 1987 г. был сформирован первый совместный технический комитет (JTC1) ISO/IEC с целью стандартизации в области информационной технологии. Стандартизацией в машинной графике занимается 24-й подкомитет (ISO/IEC JTC1/SC24). В 1988 г. была создана постоянная советская (российская) часть этого подкомитета. Основными стандартами являются :

Основными стандартами являются :

1. GKS (Graphical Kernel System) - набор базовых функций для 2D аппаратно-независимой машинной графики.
2. GKS-3D (Graphical Kernel System for 3 Dimensions) - расширение GKS для поддержки базовых функций в 3D.
3. PHIGS (Programmer's Hierarchical Interactive Graphics System) - набор базовых функций 3D графики аналогичный GKS-3D, но в отличие от GKS-3D, ориентированной на непосредственный вывод графических примитивов, группируемых в сегменты, графическая информация накапливается в иерархической структуре данных. В целом PHIGS ориентирован на приложения, требующие быстрой модификации графических данных, описывающих геометрию объектов.
4. Языковые интерфейсы (Language bindings) - представление функций и типов данных функциональных графических стандартов в стандартизованных языках программирования.
5. CGM (Computer Graphics Metafile) - аппаратно-независимый формат обмена графической информацией. Используется для передачи и запоминания информации, описывающей изображения.
6. CGI (Computer Graphics Interafce) - набор базовых элементов для управления и обмена данными между аппаратно-независимым и аппаратно-зависимым уровнями графической системы.
7. CGRM (Computer Graphics Reference Model) - модель стандартов в машинной графике, которая определяет концепции и взаимосоотношения применительно к будущим стандартам в машинной графике.
8. Регистрация - механизм регистрации стандартизуемых аспектов примитивов вывода, обобщенных примитивов, escape-функций (для доступа к аппаратным возможностям устройств) и других графических элементов.
9. Тестирование реализаций на соответствие графическим стандартам - основные цели этого проекта: специфицирование характеристик стандартизованных тестов, используемых для определения соответствия реализаций графическим стандартам, и выработка предписаний разработчикам функциональных стандартов относительно правил соответствия.

Стандарты компьютерной графики

Уровневая модель стандартизации в компьютерной графике

БАЗОВЫЕ СТАНДАРТЫ (GKS, CORE, PHIGS, CGI и др.)

ГРАФИЧЕКИЕ ПРОТОКОЛЫ : аппаратно-зависимые (TEKTRONIX. REGIS, HP-GL и др.),
аппаратно-независимые (NAPLPS , GKSM, CGM и др.), **проблемно-ориентированные** (STEP, MAP, VDAFS, EDIF и др.)

ЯЗЫКИ ОПИСАНИЯ СТРАНИЦ (PostScript, PCL и др.)

СТАНДАРТЫ РЕГИСТРАЦИИ

СТАНДАРТЫ ТЕСТИРОВАНИЯ

ФОРМАТЫ РАСТРОВЫХ ФАЙЛОВ – СТАНДАРТЫ ДЕ-ФАКТО

Core-System

Существенным этапом в области стандартизации машинной графики явилась публикация проекта стандарта CORE-SYSTEM (GSPC-77). Главные идеи, положенные в основу системы CORE : разделение функций ввода и вывода; минимизация отличий между выводом на графопостроитель и интерактивный дисплей; концепция двух координатных систем - мировой системы координат, в которой конструируется выдаваемое изображение, и приборной системы координат, в которой представляются данные для отображения; концепция дисплейного файла, содержащего приборную координатную информацию; понятие дисплейного файла сегментов, каждый из которых может независимо модифицироваться как элемент; обеспечение функций преобразования данных из мировой системы координат в приборную путем вызова видового преобразования.

В системе выделены следующие группы функций: вывода; сегментирования дисплейного файла; установления и опроса атрибутов примитивов (цвет, яркость, ширина линии и т.д.) и атрибутов сегментов (тип, видимость, указуемость и т.д.); визуализации; выполнения ввода с виртуальных устройств ввода типа указка, клавиатура, кнопка, локатор, датчик; управления и доступа к специальным аппаратным возможностям.

GKS

Результатом работ в ФРГ было создание системы В 1979 г. GKS была принята в качестве отправной точки международного стандарта. В процессе разработки международного стандарта в исходную версию GKS был внесен целый ряд изменений, приблизивших ее к CORE-SYSTEM, но сохранивших ряд отличительных фундаментальных концепций. Основной из таких отличительных черт является введение понятия рабочей станции, представляющей собой абстракцию совокупности виртуальных устройств ввода/вывода, более полно соответствующей современной тенденции использования интеллектуальных терминалов.

Набор примитивов GKS подобен набору примитивов CORE, хотя меньше и несколько более приспособлен для целей растровой графики (ломаная; текст; многоугольник, который может быть "залит" одним цветом, заштрихован, покрыт узором, либо может быть не закрашен за исключением границ; массив прямоугольных ячеек, закрашенных одним цветом; обобщенный примитив черчения, дающий доступ к специальным геометрическим и графическим возможностям устройств).

В отличие от CORE в GKS нет понятия текущей позиции, так что построение примитива не зависит от предыстории вычерчивания. В GKS имеется 2 способа задания атрибутов примитивов. Первый - индивидуальный способ аналогичен используемому в CORE, не зависит от рабочей станции и основан на индивидуальном задании атрибутов, которые сохраняют свое значение пока не будут изменены прикладной программой. Второй - групповой способ зависит от рабочей станции и основан на задании независимых групп значений атрибутов.

GKS-3D (Graphical Kernel System for Three Dimensions)

Отличия GKS-3D от GKS заключаются в добавлении 3D функций:

- примитивов 3D вывода;
- установки атрибутов вывода (2 функции);
- поддержки 3D преобразований (9 функций);
- работы с 3D сегментами и преобразований 2D сегментов в 3D и наоборот (4 функции);
- ввода с 3D координатных устройств (10 функций);
- утилит работы с матрицами 3D преобразований (2 функции).

PHIGS (Programmer's Hierarchical Interactive Graphics System)

Использование GKS или GKS-3D для отображения результатов моделирования предполагает, что моделирование целиком должна выполнять прикладная программа, так как эти системы ориентированы на прямой ввод/вывод и в них не предусмотрено иного манипулирования графическими данными кроме накопления в сегментах.

PHIGS же комбинирует графику с техникой моделирования и представляет собой набор функций программирования графики с поддержкой быстрой модификации графических данных, описывающих геометрические соотношения объектов.

Принципиальное отличие PHIGS от GKS состоит в том, что в PHIGS создание и отображение изображения разделены на две независимых фазы - занесения в централизованную структурную память (CSS - Centralized Structure Store) и отображения заданной структуры на требуемую рабочую станцию.

PHIGS+

PHIGS+ - расширение PHIGS, имеющее дополнительные функциональные возможности для приложений, требующих учета освещенности, раскраски (интерполяции цветов по поверхности), а также дополнительные возможности по управлению отображением и новые примитивы для поддержки эффективного описания сложных поверхностей. Эти расширения были сформулированы как поправка к существующим частям 1-3 стандарта PHIGS и введением новой части 4 стандарта.

Поддержка освещенности и раскраски основана на предоставлении средств задания позиции источника света и наличии примитивов "с данными", задающими вектора нормалей и цвета вершин.

CGI (Computer Graphics Interface)

Это стандарт ISO на интерфейс между аппаратно-независимой частью графического программного обеспечения (базисной графической системой) и аппаратно-зависимой (драйверами). Этот интерфейс ранее (в рамках ANSI) назывался интерфейсом виртуального устройства.

Для эффективного использования аппаратных возможностей современных графических устройств набор функций CGI перекрывает аппаратно-реализуемые возможности и включает в себя следующие функции:

- управление устройством,
- вывод графических примитивов,
- изменение графических атрибутов,
- сегментация изображений,
- графический ввод,
- растровые операции.

Отличительными особенностями CGI являются следующие: расширенный набор графических примитивов, одноступенчатое преобразование координат, увеличенное количество логических устройств ввода, наличие растровых операций. В целом набор функций CGI достаточно удобен для создания надстроенного над ним графического программного обеспечения. Последнее позволяет эффективно создавать на основе CGI различные базисные графические системы.

Графические протоколы

Анализ применяемых в настоящее время графических протоколов и проектов по их стандартизации позволяет выделить протоколы следующих типов:

- аппаратно-зависимые графические протоколы или команды графических устройств,
- аппаратно-независимые графические протоколы или метафайлы,
- прикладные графические протоколы (проблемно-ориентированные протоколы),
- растровые графические файлы.

1. Аппаратно-зависимые графические протоколы

Аппаратно-зависимые графические протоколы разрабатываются фирмами, производящими графическое оборудование. Они представляют собой последовательность команд для построения изображений на устройствах выпускаемых данной фирмой. Для интерпретации таких протоколов не требуется дополнительных ресурсов если используется соответствующее устройство. Поэтому, такие протоколы могут успешно применяться в распределенных системах при отсутствии локальной ЭВМ.

Вопрос о поддержке тех или иных аппаратно-зависимых графических протоколов определяется составом используемого оборудования. Целесообразно, чтобы центральная ЭВМ обеспечивала возможность генерации команд для наиболее распространенных графических устройств. В настоящее время значительная часть производящейся в мире графической аппаратуры работает с протоколами TEKTRONIX, REGIS и HPGL. Поддержка этих протоколов обеспечивается также в наиболее распространенных зарубежных программных продуктах.

2. Языки описания страниц

Любая страница может быть описана как просто пиксельный массив, но это практически неприемлемо. Язык описания страниц должен описывать любой текст и графику на высоком уровне в терминах абстрактных графических элементов.

Выполнение вывода с использованием языка описания страниц идет в две стадии:

1. Приложение генерирует аппаратно-независимое описание на языке описания страниц.
2. Программа, управляющая некоторым растровым устройством вывода, интерпретирует описание и отображение его на устройство.

Эти две стадии могут быть выполнены в разное время и в разных местах.

Примитивы вывода выдаются на растровое устройство вывода процессом, называемым преобразованием сканирования (растеризация).

Наиболее распространенные представители – языки PostScript и PCL.

3. Аппаратно-независимые графические протоколы

Аппаратно-независимый графический протокол или метафайл представляют собой процедурное описание изображения в функциях виртуального графического устройства. Он обеспечивает возможность запоминать графическую информацию единым образом, передавать ее между различными графическими системами (в том числе работающими на различных ЭВМ) и интерпретировать информацию для вывода на различные графические устройства. Для интерпретации метафайла требуется локальная ЭВМ, выполняющая эмуляцию не реализованных в аппаратуре функций и кодирование в команды конкретных устройств.

Наиболее активно поддерживаются стандартизованные аппаратно-независимые протоколы NAPLPS, GKSM, CGM и WMF - стандарт де-факто фирмы Microsoft на метафайл.

4.Проблемно-ориентированные протоколы

Прикладные графические протоколы это объектно - ориентированные протоколы передачи данных между прикладными системами. Они наиболее компактны (вследствие высокой семантической насыщенности), допускают свободу в выборе различных способов графического представления, но требуют большей мощности локальной ЭВМ для интерпретации. Прикладные протоколы стандартизованы пока только для САПР машиностроения и электроники.

В качестве примеров можно привести следующие стандарты:

- STEP (STandard for the Exchange Product Model Data),
- MAP (Manufacturing Automation Protocol),
- VDAFS (Sculptured Surface Interface),
- EDIF (Electronic Design Interchange Format).

Основные трудности, связанные с разработкой протоколов этого уровня, состоят в том, что во многих областях применения до сих пор не унифицированы основные объекты (в том числе графические) и операции над ними. Для работы в этом направлении потребуются объединенные усилия проблемных специалистов, математиков и системных программистов в области баз данных, машинной графики, телекоммуникаций и т.д.

Технические средства компьютерной графики: устройства ввода и устройства вывода

Устройства ввода информации

Дайте название каждому из представленных устройств

Сканер

Звуковая карта и микрофон

Клавиатура

Цифровая камера

Координатные устройства ввода информации

Графические планшеты

Сенсорные панели

Манипуляторы типа мыши

Устройства вывода информации

- Монитор (является универсальным устройством вывода текстовой, числовой и графической информации

- Принтер (печатающее устройство, предназначен для вывода информации на бумагу

- Колонки (для прослушивания звука)
- Плоттер (используется для вывода сложных и широкоформатных графических объектов (плакатов,чертежей,электрических и электронных схем и пр.)

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Каминский В.П., Иващенко Е.И. Высшее образование. Инженерная и компьютерная графика для строителей. – М.: Феникс, 2008 – 288 с.
2. Компьютерная графика. Пантюхин П. Я., Быков А. В., Репинская А. В. Профессиональное образование – М.: Инфра-М, 2007 – 88 с.
3. Постнов К. В. Компьютерная графика: учебное пособие. Московский государственный строительный университет. - Москва : МГСУ, 2012. - 289 с
4. Аббасов И.Б. Все о графике. Профессиональное образование – М.: Инфра-М, 2007 – 88 с.
Основы трехмерного моделирования в 3DS MAX 2018 – М.: ДМК Пресс, 2017 – 186 с.
5. Трошина Г.В. Моделирование сложных поверхностей [Электронный ресурс]: учебное пособие - Электрон. текстовые данные - Новосибирск: Новосибирский государственный технический университет, 2015 - 91 с.
6. Никулин Е.А..Компьютерная графика. Модели и алгоритмы [Электронный ресурс]: учеб. пособие - Электрон. дан. - Санкт-Петербург : Лань, 2018. - 708 с.
7. Лейкова М.В. Инженерная компьютерная графика. Методика решения проекционных задач с применением 3D-моделирования [Электронный ресурс]: учебное пособие - Электрон. текстовые данные. - М. : Издательский Дом МИСиС, 2016. - 92 с.