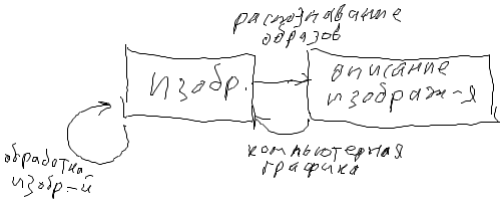
Лк №1
07.09.03

Владимир Алексеевич

Алгоритмы и программное обеспечение синтеза изображений



пассивная графика - формирование изображения без диалога

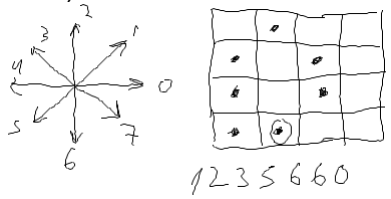
интерактивная графика - форма взаимодействия человека и компьютера с целью формирования изображения. графический диалог обеспечивает интерактивная подсистема, она видоизменяет изображение в соответствии с командами пользователя.

началом современной компьютерной интерактивной графики считают диссертацию Сазерленда в 1963 году. далее фирма General Motors начала применять САПР (системы автоматизированного проектирования) для проектирования автомобилей.

Лк №2
07.09.06

разновидности изображений:

1. полутоновые и цветные изображения. обычно 1 или 3 байта на пиксел.
2. двухуровневые (бинарные) изображения. 1 бит на пиксел.
3. непрерывные кривые и линии. контуры областей, графики, диаграммы. - это последовательности точек, записанные в координатах x,y. эффективный аппарат для их хранения - цепные коды (автор - Фримен).



для хранения необходимо около 2 бит на пиксел.

4. множество отдельных точек, отстоящих далеко друг от друга.

переход от класса 1 к классу 2 - задача сегментации. из 2 в 3 - задача прослеживания контуров.

принципы построения технических средств компьютерной графики

два технологических принципа построения мониторов - ЭЛТ и с использованием жидких кристаллов.

дисплейный процессор - специальное устройство, встроенное в монитор, выполняющее графические операции.

векторные дисплеи - используют принцип произвольного сканирования, когда луч переходит и рисует отдельные точки на экране. обычные дисплеи - с растровым сканированием.

каждое графическое устройство имеет свой алгоритм развертки.

барабанные графопостроители - принцип заключается в том, что бумага накручена на два барабана, и движется в обе стороны по вертикали, а головка движется по горизонтали.

электростатические графопостроители - "темные" участки бумаги заряжаются, и к ним позже прилипает порошок.

ксерографический принцип рисования - используется селеновый барабан, на нем создается заряд, и краска осаждается на барабане, а затем переносится на бумагу.

графические акселераторы - специальные устройства, которые повышают скорость графических операций. 3 вида:

1. для графических оболочек (рисование окон, геометрических фигур)
2. 3D-акселераторы. формирутю проекции из трехмерного в двумерное пространство, отсекают

невидимые части.

3. мультимедиа-акселераторы - для видеофильмов и графических изображений.

Лк №3
07.09.10

стандарты в компьютерной графике

существуют специальные организации, которые занимаются стандартизацией программного обеспечения в компьютерной графике. в США - это ANSI и NBS, в ФРГ - DIN. также существует международная организация ISO.

двумерные геометрические преобразования

основные преобразования - это перенос, масштабирование и поворот.

пусть есть некоторая точка P(x,y). тогда ее перенос $P(x,y) \to P'(x,y)$ можно осуществить по следующим формулам:

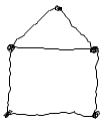
$$\begin{cases} x' = x + dx \\ y' = y + dy \end{cases}$$

$P = [x, y] \qquad P' = [x', y']$

$T = [dx, dy] \qquad P' = P + T$

преобразозвание осуществляется по всем точкам.

тем не менее, простые изображения можно преобразовывать не по всем точкам, а только по ключевым:



масштабирование в s_x и s_y раз соответственно по осям x и y производится следующим образом:

$$\begin{cases} x' = x \cdot s_x \\ y' = y \cdot s_y \end{cases}$$

- это масштабирование относительно начала координат.

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

$P' = P \cdot S$

если $s_x < 1$, то масштаб по оси x уменьшается, если $s_x > 1$ - увеличивается.

если $s_x = s_y$, то масштабирование называется *однородным*.

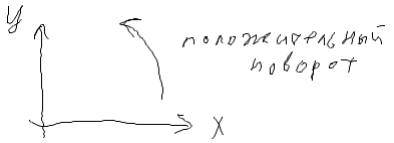
поворот:

$$\begin{cases} x' = x\cos\Theta - y\sin\Theta \\ y' = x\sin\Theta + y\cos\Theta \end{cases}$$

$|a_{ij}| = 1$

$P' = P \cdot R$

$$R = \begin{bmatrix} \cos\Theta & \sin\Theta \\ -\sin\Theta & \cos\Theta \end{bmatrix}$$



однородные координаты и матричные представления двумерных преобразований

практически *во всех* системах компьютерной графики применяются *однородные* координаты. однородные координаты дают возможность свести все геометрические преобразования к умножению матриц. это экономит аппаратные ресурсы и унифицирует запись любого преобразования в виде матрицы.

в однородной системе координат точка P(x,y) записывается как P(Wx,Wy,W), $W \neq 0$.

$$\begin{cases} x = X/w \\ y = Y/w \end{cases}$$

обычно принимается W=1, поэтому деление не требуется.

матрица смещения в однородных координатах записывается как

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{bmatrix}$$

два подряд переноса выполняются через матричное перемножение, и результирующая матрица имеет вид

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx_1 + dx_2 & dy_1 + dy_2 & 1 \end{bmatrix}$$

матрица масштабирования:

$$S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

последовательные два масштабирования мультипликативны:

$$S_2 = \begin{bmatrix} s_{x_1} s_{x_2} & 0 & 0 \\ 0 & s_{y_1} s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

матрица поворотов:

$$R = \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

два последовательных поворота аддитивны:

$$R = \begin{bmatrix} \cos(\Theta_1+\Theta_2) & \sin(\Theta_1+\Theta_2) & 0 \\ -\sin(\Theta_1+\Theta_2) & \cos(\Theta_1+\Theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Лк №4
07.09.13

композиция трехмерных преобразований

к точкам более эффективно применять одно результирующее преобразование, чем ряд преобразований друг за другом. часто нужно одновременно применять смещения, повороты и масштабирование. при этом из-за некоммутативности матричного произведения, последовательность преобразований влияет на конечный результат.

пример 1:

преобразование поворота вокруг произвольной точки P_1 представляется как 3 шага:

а) перенос, при котором точка P_1 помещается в начало координат.

б) собственно поворот

в) перенос, при котором точка начала координат переходит в первоначальное положение.

$P_1(x_1, y_1)$

$$T(-x_1,-y_1)\cdot R(\Theta)\cdot T(x_1,y_1) = \begin{bmatrix} cos\Theta & sin\Theta & 0 \\ -sin\Theta & cos\Theta & 0 \\ x_1(1-cos\Theta)+y_1 sin\Theta & y_1(1-cos\Theta)-x_1 sin\Theta & 1 \end{bmatrix}$$

применение однородных координат упрощает задачу, т.к. преобразование делается сразу.

пример 2:

преобразование масштаба вокруг произвольной точки P_1 осуществляется через перенос, масштабирование и обратный перенос. матрица имеет вид:

$$T(-x_1,-y_1)\cdot S_1 \cdot T(x_1,y_1) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ x_1(1-S_x) & y_1(1-S_y) & 1 \end{bmatrix}$$

пример 3:

масштаб и поворот вокруг произвольной точки P_1 и смещение - задает новое положение объекта в виде произведения матриц:

$$T(-x_1,-y_1)\cdot S(S_x,S_y)\cdot R(\Theta)\cdot T(x_2,y_2)\cdot T(x_1,y_1)$$

надо иметь в виду, что поворот и однородное масштабирование коммутативны, а все остальные пары (масштаб,смещение и поворот,смещение) - нет.

пример 4:

композиция самого общего вида имеет вид матрицы

$$M = \begin{bmatrix} r_{11} & r_{12} & 0 \\ r_{21} & r_{22} & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

при этом матрица r_{ij} объединяет поворот и масштаб, а (t_x, t_y) описывает перенос.

для вычисления произведения вектора на матрицу $P\cdot M$, требуется 9 умножений и 6 сложений.

учитывая вид последнего столбца матрицы M, более эффективно записать:

$$\begin{cases} x' = xr_{11} + yr_{21} + t_x \\ y' = xr_{12} + yr_{22} + t_y \end{cases}$$

есть и другие способы экономии. например, если осуществляются малые (плавные) повороты, то

можно воспользоваться соотношением $cos\Theta \approx 1$:

$$\begin{cases} x' = x - ysin\Theta \\ y' = xsin\Theta + y \end{cases}$$

если таких поворотов много, то ошибка становится большой.

трехмерные геометрические преобразования

точка в однородных координатах представляется как четверка P=(X,Y,Z,W). преобразования в трехмерном пространстве описываются в виде суперпозиции вращений, масштабирований и переносов. имеются особенности - три матрицы вращения вокруг каждой из осей координат.

для примера, матрица R_x вращения вокруг оси X имеет следующий вид:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\varphi & sin\varphi & 0 \\ 0 & -sin\varphi & cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

пример:

построить матрицу вращения на угол φ вокруг прямой L, проходящей через точку A(a,b,c) и имеющей направляющие косинуса l,m,n.

получим произведение шести матриц (некоммутативное):

$R_{ez} = T \cdot R_x \cdot R_y \cdot R_z \cdot R_y^{-1} \cdot R_x^{-1} \cdot T^{-1}$ (перемножение справа налево)

а) осуществляется перенос прямой L, чтобы она проходила через начало координат

б) совмещение прямой с осью Z двумя поворотами R_x и R_y вокруг осей X и Y.

в) собственно вращение на угол φ

г) обратные повороты и смещения

все возможные преобразования подобного рода дают матрицу

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ l_x & l_y & l_z & 1 \end{bmatrix}$$

обычно в трехмерной графике геометрическим преобразованиям подвергаются многогранники, которые однозначно задаются набором своих вершин $V_i(x_i, y_i, z_i), \quad i = \overline{1,n}$. подвергая каждую точку этого многогранника преобразованию, получаем новый многогранник, в котором строим новые грани.

особенности практического применения преобразований:

1. в целях исключения "пропадания" точек выполняем *обратное* преобразование.
2. поле изображения ограничено, т.е. для некоторых точек образа не существует.
3. преобразование происходит точка-в-точку, без интерполяции.

копирование точки изображения:

```
Image2->Canvas->Pixels[x][y]=Image1->Canvas->Pixels[x][y];
```

Лк №5
07.09.17

Интерактивное графическое программирование

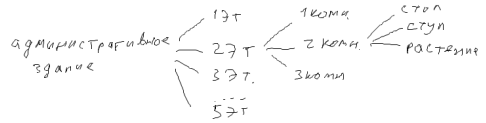
в интерактивной графике выполняются три основных действия:

1. построение прикладной модели (т.е. структуры данных)
2. описание объектов для графической системы

3. интерактивная работа

1. построение прикладной модели (т.е. структуры данных)

в качестве примера построения прикладной модели рассмотрим графическую программу для размещения мебели в комнате.



в распоряжении дизайнера должны быть геометрия и размеры каждой комнаты (и предметов), данные об изготовителях, материалы, цвет, цена, сроки поставки. часть из этих данных должна располагаться сразу на экране.

вся эта информация хранится либо в специализированных структурах данных, либо в БД общего назначения.

существуют специальные методы моделирования иерархических систем.

2. описание объектов для графической системы

элементы структуры данных должны быть описаны для графической системы в виде выходных графических примитивов (точки, линии, надписи...).

обычно в целях разделения взаимодействия структуры данных и технических устройств, описывают *зависимую* от приложения модель в виде *независимых* универсальных графических примитивов.

графическую систему рассматривают как воображаемую фотокамеру.

3. интерактивная работа

пользователь управляет "фотокамерой" и влияет на визуальные атрибуты объекта через устройства ввода. вводимая информация расшифровывается и передается графической системе. графическая система выводит изображение на экран.

суть интерактивной графики состоит в диалоге между пользователем и системой.

системы координат в компьютерной графике

в задачах синтеза изображений используется 5 систем координат:

1. мировая система координат - реальная физическая система
2. система координат камеры
3. система координат объекта
4. действительная система координат изображения (координатное пространство). (это еще не пикселы)

пространственная точка проектируется на действительную плоскость изображения в точку с координатами x,y,f (f - фокусное расстояние). здесь формируется функция интенсивности (яркости), которая дискретизируется в процессе формирования цифрового изображения.

5. пиксельная система координат

системы координат 1-4 задаются числами с плавающей точкой и изменяются в пределах отрезка [0,1].

пиксельная система - целые числа в виде координат матрицы.

в компьютерной графике изображение формируется путем движения по каркасу объекта, в мировых координатах, и проецирования каждой точки на действительную систему координат изображения.

некоторые понятия графического программирования

графическая программа преобразует мировые координаты в координаты устройства.

в мировой системе координат можно задать часть объекта (окно) и проецировать ее на некоторую область экрана - поле вывода. при этом выводится только та часть объекта, которая видна из окна.

часть изображения, не попадающая в окно, делается невидимой с помощью процедуры отсечения. процесс отсечения и отображения окна на поле вывода называется видовая операция.

при обновлении изображения в интерактивном режиме обычно изменяется только его часть. для ускорения обновления, изображение разбивается на отдельные сегменты, которые перерисовываются независимо. это называется сегментация.

Лк №6
07.09.20

Алгоритмы растровой графики

растровое изображение формируется на основе графических примитивов, простейшим из которых есть пиксел. для ускорения работы, также используются более сложные примитивы - линии и фигуры.

Алгоритмы проведения отрезков

процесс проведения отрезков непростой, каждый раз надо принимать решение об отнесении точки к отрезку.

общие требования к алгоритмам проведения отрезков:

1. отрезки должны выглядеть прямыми, начинаться и заканчиваться в заданных точках.
2. яркость вдоль отрезка должна быть постоянной и не зависеть от длины и наклона.
3. рисовать нужно быстро.

обычный компромисс: нахождение приближенной длины отрезка, сведение вычислений к минимуму, предпочтительно использование целой арифметики и реализация алгоритма на аппаратном или микропрограммном уровнях.

алгоритм с прямым вычислением координат

точки отрезка находятся через уравнение прямой $\frac{x-x_1}{y-y_1}=\frac{x_2-x_1}{y_2-y_1}$

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

```
int x,x1,y1,x2,y2;
float y,k;
k=(float)(y2-y1)/(x2-x1);
y=y1;
for (x=x1; x<=x2; x++)
{
  PutPixel(x,y);
  y+=k;
}
```

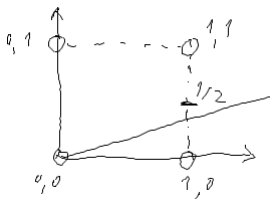
в этом алгоритме возможно несовпадение конечной точки с нужным значением.

простой пошаговый алгоритм

```
поз, шаг, нач, кон, точн
поз=нач
1: пиксел(поз)
если поз-кон<точн то 4
если поз>кон то 2
если поз<кон то 3
2: поз=поз-шаг
идти к 1
3: поз=поз+шаг
4: конец
```

алгоритм Брезенхема

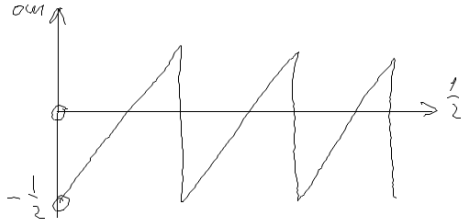
этот алгоритм выбирает оптимальные растровые координаты для представления отрезков. одна из координат (x или y), в зависимости от углового коэффициента, изменяется на 1. изменение другой координаты зависит от расстояния между действительным положением отрезка и ближайшими координатами сетки. это расстояние называется *ошибкой*. алгоритм построен на проверке знака этой ошибки.



решение принимается в зависимости от того, проходит ли прямая выше середины точки (1/2). в этом и заключается оптимальность алгоритма.

пример:

тангенс угла наклона равен m=tg=3/8.
первоначально ошибка устанавливается равной ош=1/2.
ош=ош+m=-1/2+3/8=-1/8
ош<0, => отрезок проходит ниже середины пикселя, у не увеличиваем.
-1/8+3/8=1/4
ош>0, => y=y+1;
как только ошибка поменяла знак, из нее вычитается 1:
1/4-1=-3/4.
-3/4+3/8=-3/8



схематично алгоритм Брезенхема сводится к следующему:

$\Delta x = x_2 - x_1$
$\Delta y = y_2 - y_1$
$e = \frac{\Delta y}{\Delta x} - \frac{1}{2}$
for i=1 to Δx
  PutPixel(x,y);
  while (e>=0)
    y=y+1;
    e=e-1;
  end while;
  x=x+1;
  e=e+ $\Delta y/\Delta x$
next i

переход к целочисленному алгоритму осуществляется за счет преобразования $\bar{e} = 2e\Delta x$:

$\Delta x = x_2 - x_1$
$\Delta y = y_2 - y_1$
$\bar{e} = 2\Delta y - \Delta x$
for i=1 to Δx
  PutPixel(x,y);
  while (e>=0)
    y=y+1;
    $\bar{e} = \bar{e} + 2\Delta x$
  end while;
  x=x+1;
  $\bar{e} = \bar{e} + 2\Delta y$
next i

есть алгоритмы Брезенхема для проведения прямой в любом направлении, а также для генерации дуг, окружностей и т.д.

Лк №7
07.09.27

Алгоритмы заполнения замкнутых областей

Генерация сплошных областей из описания ребер или вершин называется *растровой разверткой сплошных областей*, *заполнением многоугольников* или *заполнением контуров*.

существующие методы делятся на две группы:
1. растровая развертка
2. затравочное заполнение

алгоритмы растровой развертки

смысл алгоритмов растровой развертки - они построчно сканируют весь экран и определяют, лежит ли точка внутри области. если да, то закрашивают ее.
они отличаются быстродействием и возможностью аппаратной реализации.

простой алгоритм с упорядоченным списком ребер

1. определить для каждого ребра многоугольника точки пересечений со сканирующими строками, проведенными через середины интервалов, для чего применяется алгоритм Брезенхема. горизонтальные ребра игнорируются. каждое пересечение в виде $\left(x, y+\frac{1}{2}\right)$ заносится в список.

2. отсортировать список по строкам и по возрастанию x в строке.
$(x_1, y_1) < (x_2, y_2)$, если $y_1 > y_2$ или $y_1 = y_2$ и $x_1 \leqslant x_2$

3. выделить из отсортированного списка пары элементов $(x_1, y_1), (x_2, y_2)$. структура списка гарантирует, что эти пары находятся рядом. активировать на сканирующей строке *y* пикселы для целых значений *x*, таких что $x_1 \leqslant x + \frac{1}{2} \leqslant x_2$.

недостатки этого алгоритма - генерируется большой список, который нужно сортировать. повысить эффективность сортировки можно применением методов групповой сортировки.

другие алгоритмы, не использующие сортировку:

алгоритм заполнения по ребрам

для каждой сканирующей строки, пересекающей ребро многоугольника в точке (x_1, y_1), заполнить все пикселы, у которых центры лежат справа от этой точки, до следующего пересечения. этот алгоритм закрашивает область по-другому.

алгоритм со списком ребер и флагом

этот алгоритм является двухшаговым:
1. рисуется контур
2. заполняются пикселы внутри контура

достоинства: здесь каждый пиксел обрабатывается один раз. также этот алгоритм несложно реализовать аппаратно.

методы затравочного заполнения

известна некоторая точка внутри замкнутого контура. алгоритм ищет точку, соседнюю с затравочной и расположенную внутри.

простой алгоритм заполнения с затравкой

использует стек.
1. поместить затравочный пиксел в стек.
2. цикл, пока стек не пуст:

- извлечь пиксел из стека
- закрасить пиксел
- для каждого из соседних к текущему четырехсвязных пикселов, проверить, является ли он граничным или не присвоено ли ему уже требуемое значение. если да, то проигнорировать пиксел.
- в противном случае, поместить пиксел в стек.

недостатки: стек может оказаться очень большим.

построчный алгоритм заполнения с затравкой

размер стека минимизируется за счет хранения только одного затравочного пиксела для непрерывного интервала на сканирующей строке.

запоминаются начало и конец каждой строки. далее проверяются строки, лежащие сверху и снизу от нее.

работа прекращается при опустошении стека. этот алгоритм успешно справляется с дырами и зубцами на границе. максимальная глубина стека - 5-10 точек.

Лк №8
07.11.12

рекурсия в компьютерной графике. фракталы

автор фрактальной геометрии - Мандельброт, придумал ее в 1975 году.

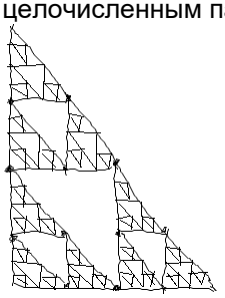
литература:

1. Аммерал Л. 1992 г.
2. Основы компьютерной графики в Visual C++ (автор книги есть в методичке)

программа "Треугольник"

1. рисуется треугольник
2. определяются средние точки его сторон и полученные точки опять образуют треугольник
3. процедура рисования треугольника опять вызывается трижды с координатами вершин маленьких треугольников

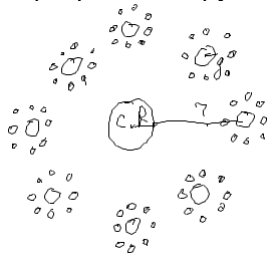
целочисленным параметром n задается глубина рекурсии.



парадокс брадобрея:

пусть у брадобрея есть такое правило, что он бреет только тех, кто не бреется сам. пока это правило применяется к другим людям, все нормально. но если задаться вопросом "бреет ли брадобрей сам себя?" - возникает парадокс. если брадобрей бреет себя, значит по правилу, он не может себя брить. но если он сам себя не бреет, то он должен себя брить. /* бгг :)) */

программа "окружности"

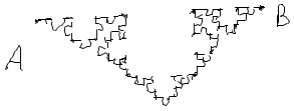


$r = f_1(R)$

$g = f_2(R)$

фрактал по Аммералу - это замкнутая кривая, граница которой имеет очень большую величину по

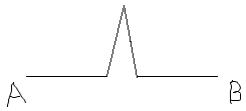
сравнению с самой областью, которую охватывает кривая (как береговая линия острова).



общая схема рисования фрактала базируется на заданном множестве модельных точек. эта модель применяется рекурсивно для каждой отдельной части фрактала.

пример множества модельных точек и соответствующей ему фигуры:

x	0.45	0.5	0.55
y	0	0.45	0



кроме модельных точек еще задается базовая фигура, к которой они применяются.

результат:

принцип симметрии во фракталах имеет важное значение, так же как он встречается и в природе.

Лк №9
07.11.19

основные принципы трехмерной компьютерной графики

построение модели

наиболее популярные модели - *полигональные*, т.е. состоящие из граней. они далее разбиваются на треугольники, этот процесс разбиения называется триангуляцией.

один из недостатков этой модели - для сложных объектов требуется большое количество треугольников.

для объектов с плавными очертаниями более удобным является *сплайновое моделирование*.

следующая группа моделей - *процедурное описание*. это создание объектов с помощью математической модели (т.е. можно описать формулами).

твердотельное моделирование - объект моделируется как твердое тело разными способами:

- с помощью сочетания кубов, шаров и т.д.
- другой способ - двумерные сечения. например, формирование модели змеи путем задания срезов вдоль нее.

экструзия или выдавливание - формирует поверхность объекта по его срезам.

для моделирования живых существ существует также технология *metaballs*. она реализуется как поверхность, натянутая на сферы.

создание материалов

материал характеризуется несколькими компонентами:

1. заливающая компонента (цвет, не зависящий от источников света)
2. диффузная компонента (зависит от источника света, но не от наблюдателя)
3. отражаемая компонента (зависит от источника света и от наблюдателя)
4. также параметрами матераила являются прозрачность и коэффициент преломления света

для имитации поверхностей, имеющих углубления и шероховатости (кожура апельсина, рябь на поверхности воды..), используется карта неровностей.

один из этапов создания материала - задание текстуры. текстура в компьютерной графике - это плоское изображение, привязанное к поверхности.

существуют разные способы наложения текстуры.

бывают процедурные текстуры - такие, которые задаются формулами.

источники света и наблюдатель

источники света бывают разных типов: рассеянный свет, точечные источники, распределенные источники (излучает свет от целой поверхности).

характеристики источника: координаты, цвет, насыщенность, дистанция действия, степень ослабления света с расстоянием.