Компьютерная графика

Лекция 7

24/25 октября 2013

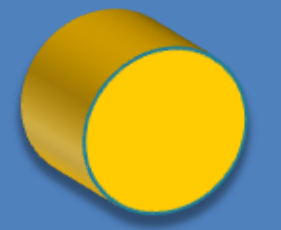
Графический процесс
Геометрическое моделирование

Алексей Викторович Игнатенко
Лаборатория компьютерной графики и мультимедиа
ВМК МГУ

Три части: две вводных + более подробно о геометрическом моделировании

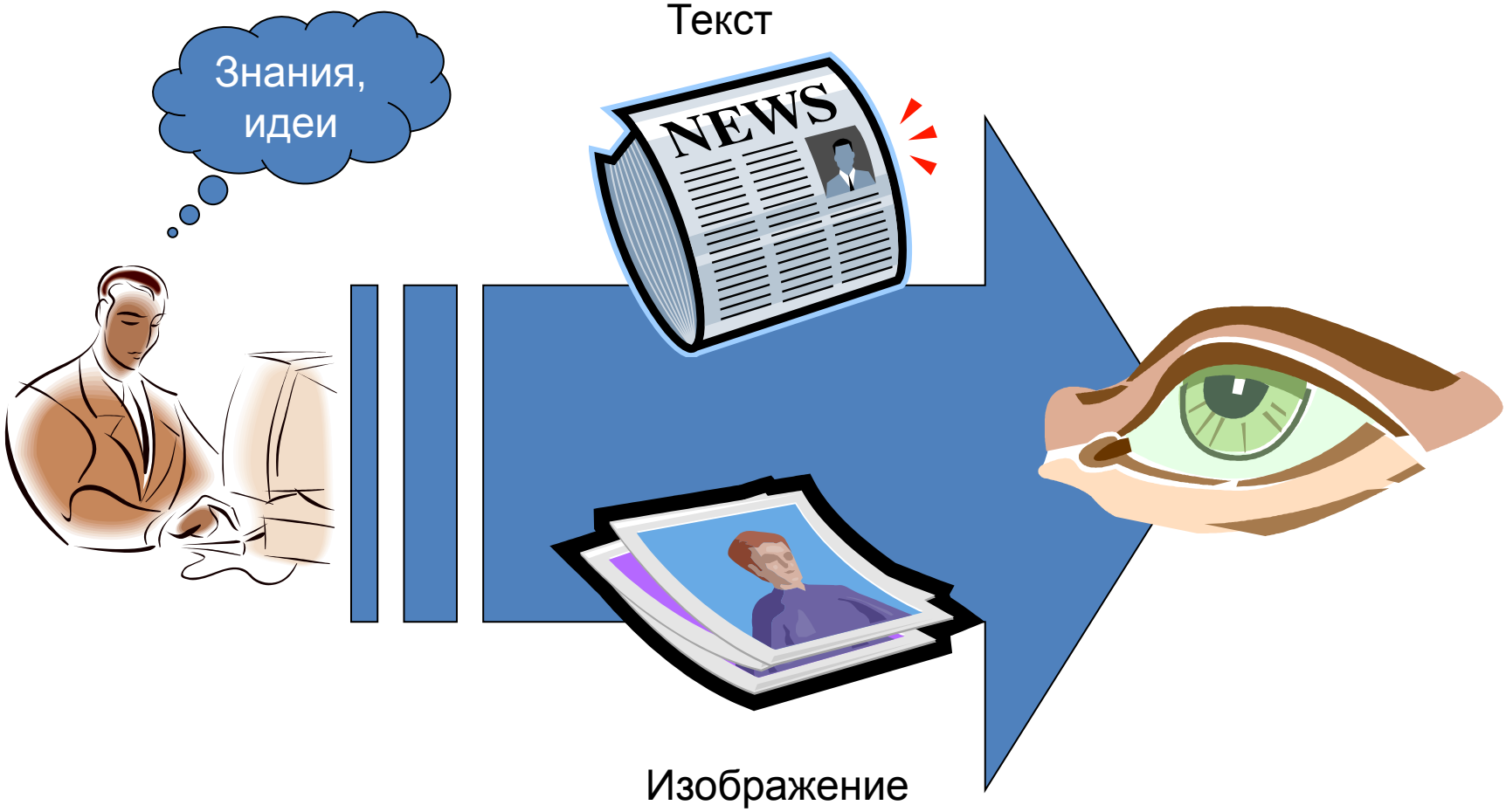
Задача синтеза изображений

Графический процесс

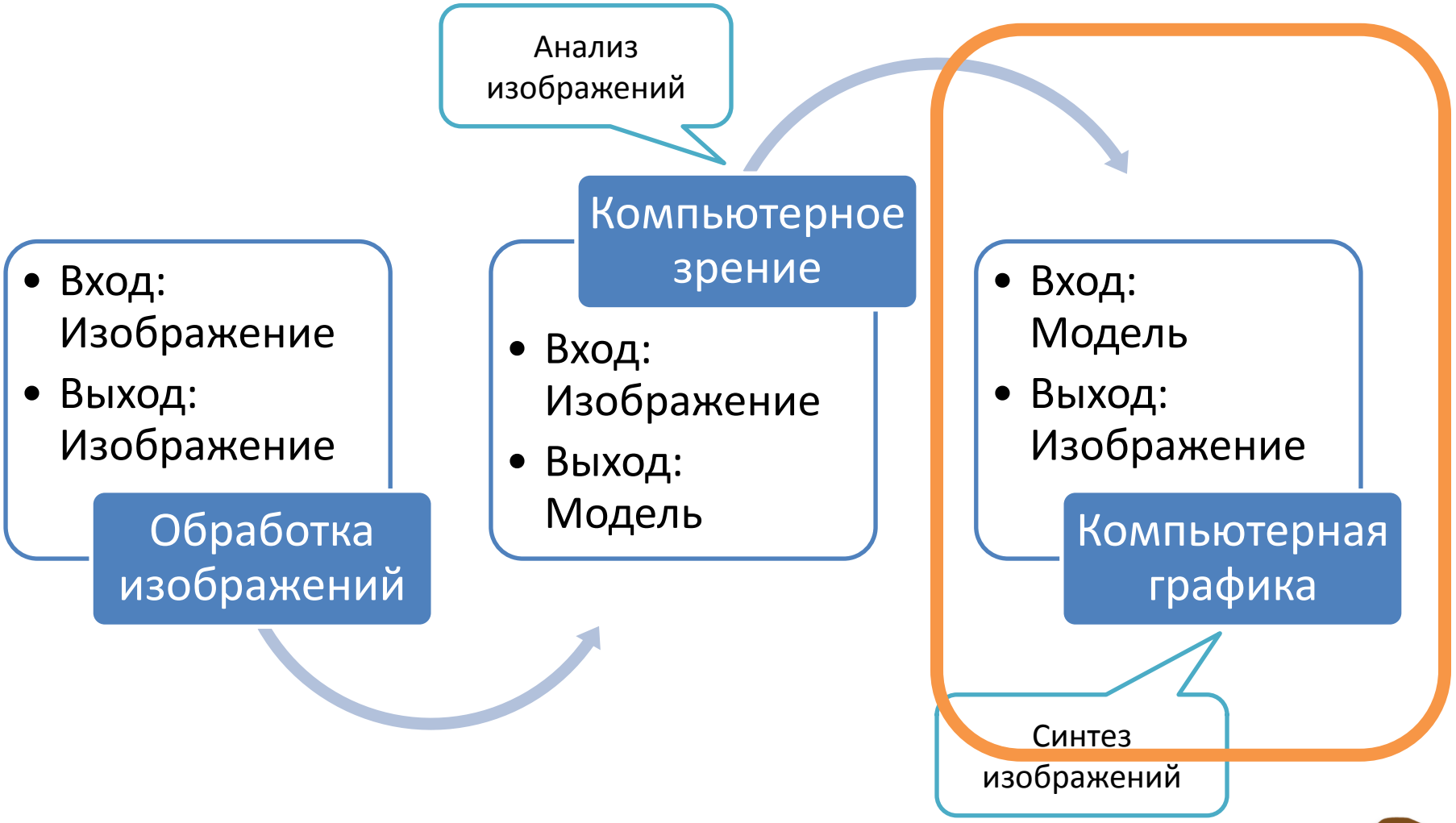


Геометрическое моделирование

Задача синтеза изображений – визуальное представление информации



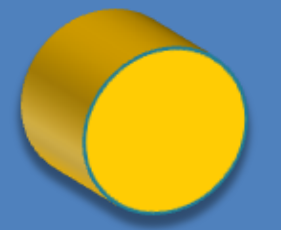
Обработка изображений, зрение и графика связаны по данным и алгоритмам



Три части: две вводных + более подробно о геометрическом моделировании

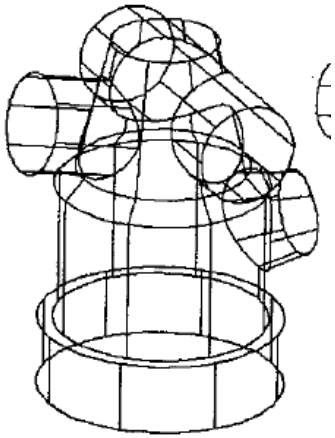
Задача синтеза изображений

Графический процесс



Геометрическое моделирование

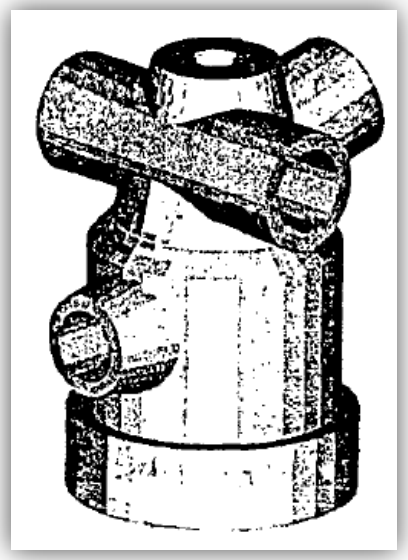
Компьютерная графика изучает модели и алгоритмы синтеза



Модель

Алгоритм синтеза

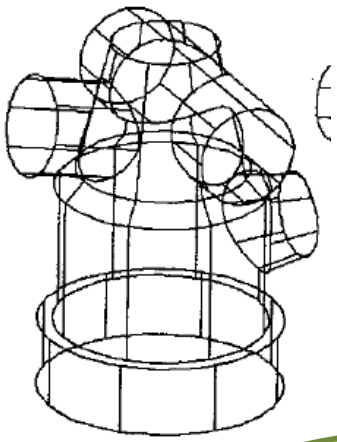
Изображение



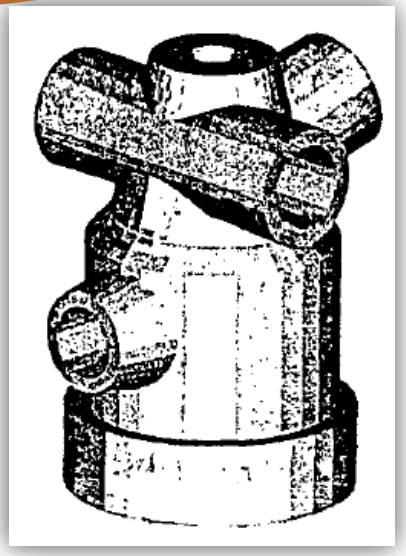
Графический процесс: типовая последовательность применения алгоритмов

Геометрическая модель
Модель освещения
Модель анимации

Цветокоррекция
Гамма
Полутонирование

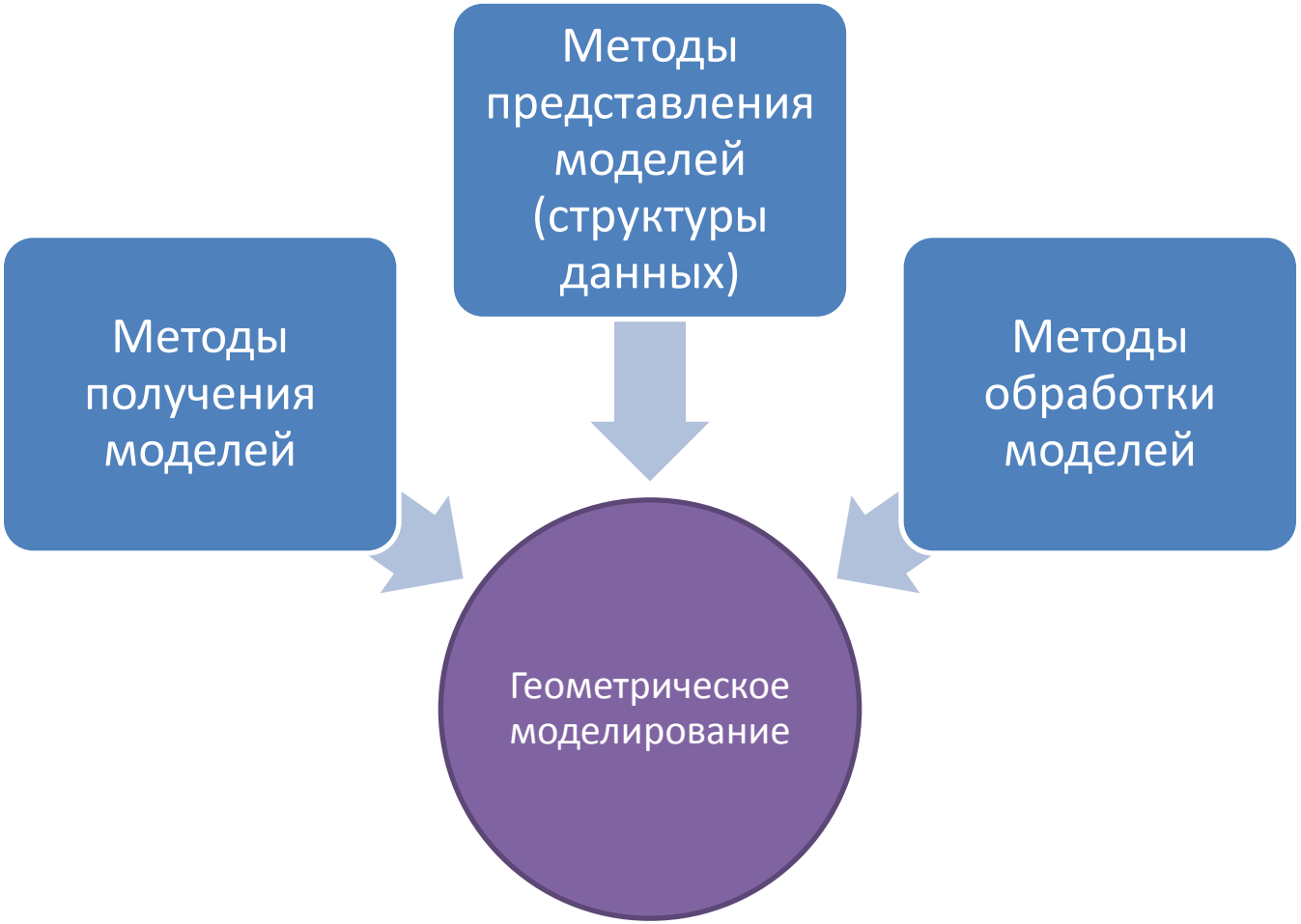


Модель → Алгоритм синтеза → Изображение



Получение проекции
Заливка

В геометрическое моделирование входят методы получения, представления и обработки моделей



Существует три основных способа получения геометрических моделей

Ручное моделирование

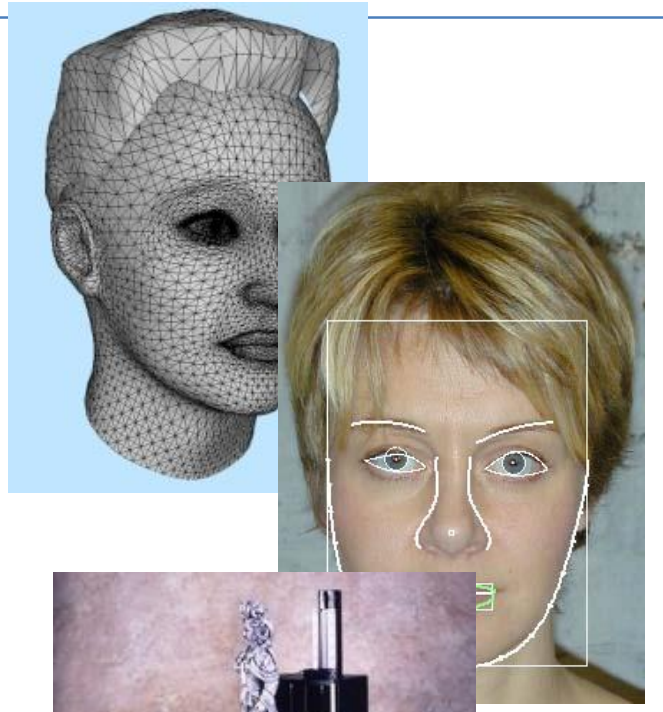
- Пакеты моделирования Maya, AutoCad и т.п.

Автоматизированное моделирование

- 3D-сканирование
- Реконструкция по фотографиям

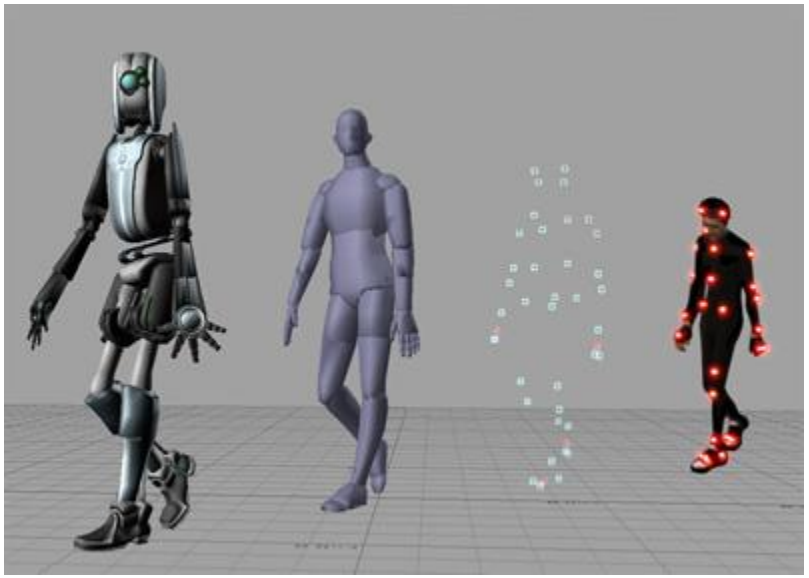
Библиотеки моделей

- Повторное использование созданных моделей



Анимация – крайне сложная задача для ручного моделирования

- Ручное моделирование
- Синтезированная анимация
- Захват движения (motion capture)



Для освещения модели надо иметь модель материалов и модель источников света

Модель материалов + Модель источников света = Модель освещения

Материалы и источники света обычно имеют параметрические модели

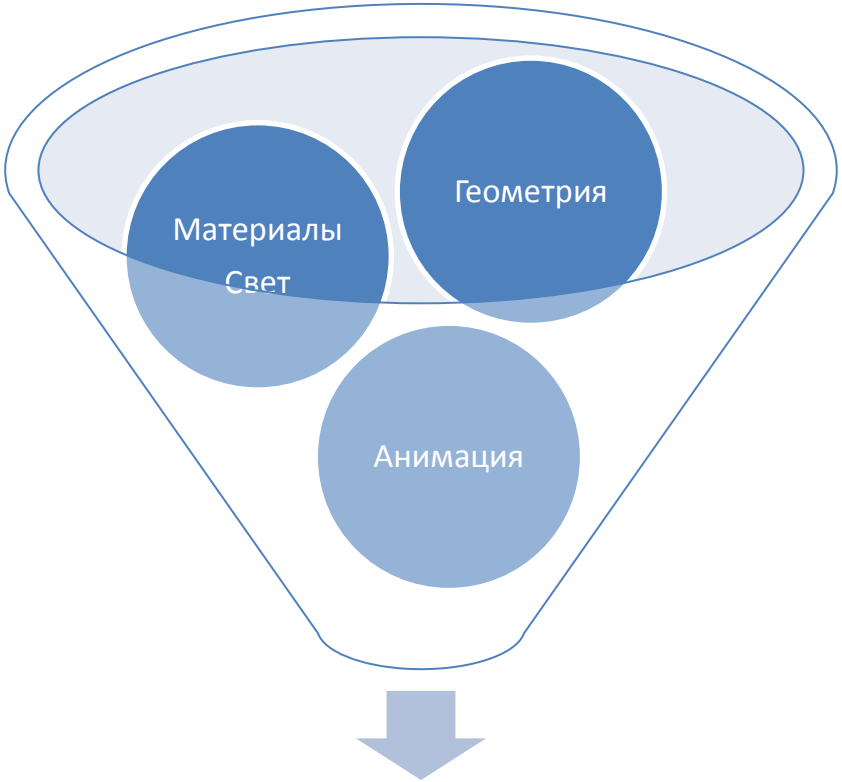
Материалы

- Модель отражения/пропускания
- Ручное моделирование
- Фотографии (или сканы)

Источники света

- Ручное моделирование
- Фотографии (HDR)

Алгоритмы синтеза изображений решают задачу создания изображения по набору моделей



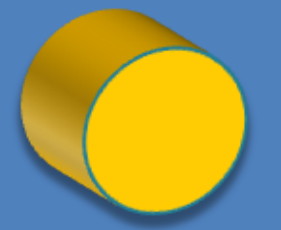
Задача этапа вывода – перевести цифровое изображение в свет

- Результирующее изображение выводится в:
  - Буфер кадра (монитор)
    - Преобразование изображения в излучение
    - Гамма-коррекция
    - Цветокоррекция
  - Файл
    - Сжатие изображений или видео
  - Принтер
    - Полутонирование и т.п.

Три части: две вводных + более подробно о геометрическом моделировании

Задача синтеза изображений

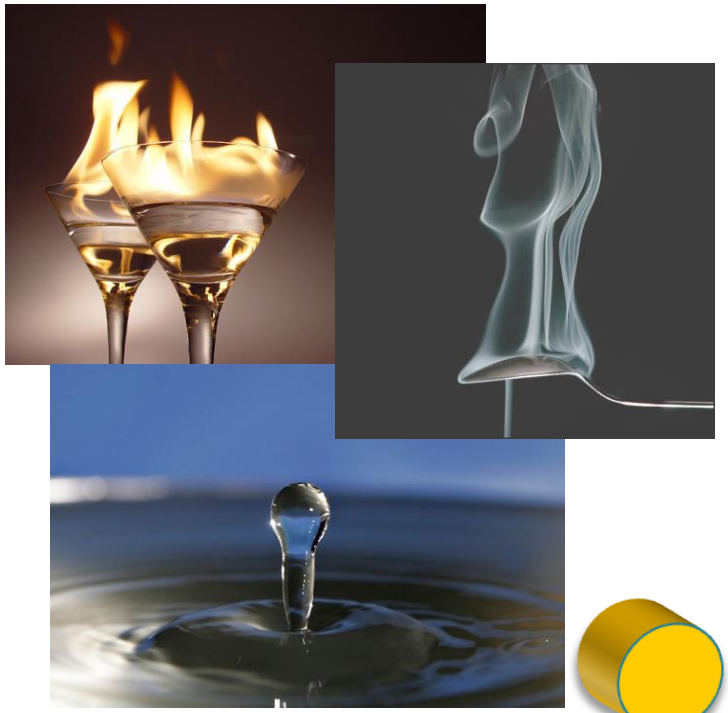
Графический процесс



Геометрическое моделирование

Модель – абстрактное представление сущности реального мира

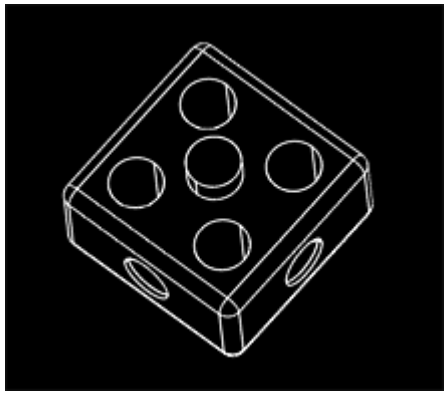
- Модель – это абстрактное представление сущности реального мира
  - Математическое моделирование физических, химических процессов и т.п.
- Компьютерное моделирование
  - Данные о физических объектах не могут быть целиком введены в компьютер
  - Нужно ограничить объем хранимой информации об объекте
- Задача: найти вид модели, наилучшим образом отвечающий решаемой задаче

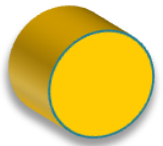


В компьютерной графике применяется геометрическое моделирование

- Моделирование объектов различной природы с помощью геометрических типов данных

- Принципы выбора модели
  - Соответствие поставленной задаче
  - Максимально использовать возможности графической системы
  - Учесть задачи обработки и редактирования модели





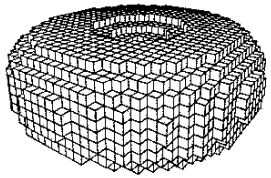
Существует огромное количество геометрических моделей и их видов

- Воксельное
- Точечное представление
- Конструктивная геометрия
- Каркасное представление
- Граничное представление первого порядка
- Граничные представление высших порядков
- На основе изображений
- Гибридные модели
- …

«Представление» -- синоним модели, но есть и тонкости

Один и тот же объект может иметь несколько представлений (моделей)

Представления могут быть получены как из исходного объекта, так и путем преобразования другого представления объекта



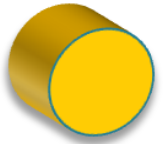
объект 1

объект 2

Модель 1 Модель 2

Модель 3 Модель 4

«Представления объекта 1» «Представления объекта 2»

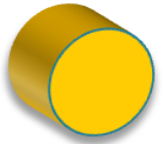


Будем рассматривать сплошные тела

Рассматриваем сплошные тела (solid models)

Можно выделить поверхность и объем

Модели трехмерных объектов



Различные характеристики модели нужны для правильного выбора типа модели

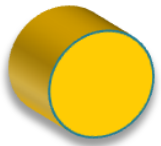
Структура данных

Алгоритм построения

Количество памяти, необходимое для хранения модели

Типичные свойства представления (алгоритмы)

Область применения моделей в данном представлении



Каждую модель можно характеризовать по набору параметров

Объем/Поверхность

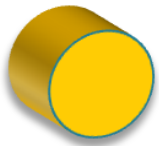
- Какие свойства трехмерного объекта описывает модель?

Дискретное/Непрерывное

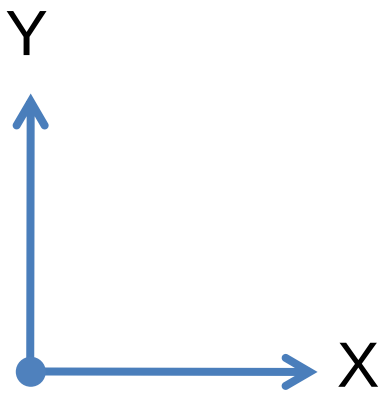
- Содержится ли в модели информация о дополнении дискретных данных до непрерывных?

Явное/Параметрическое

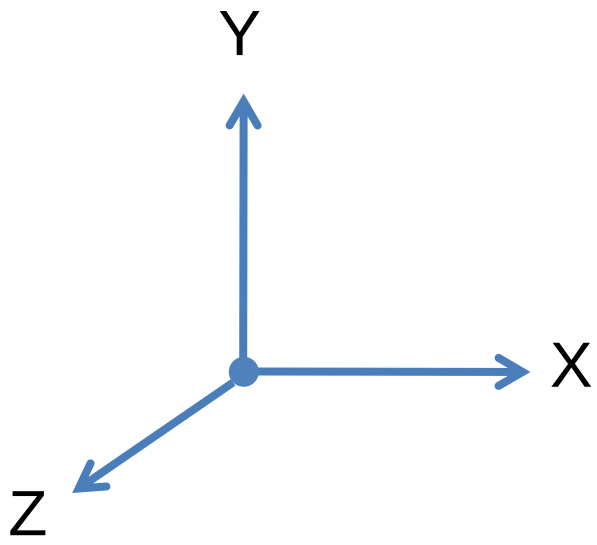
- Способ получения трехмерных координат точек, принадлежащих модели



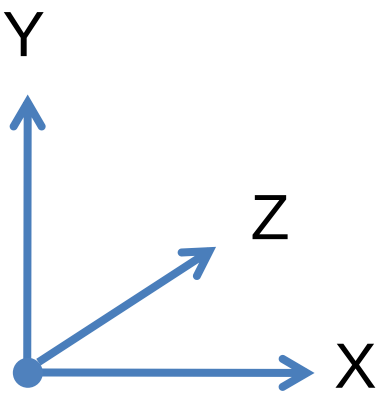
Системы координат необходима для любой трехмерной модели



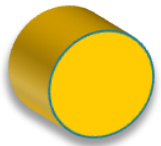
Двухмерная



Правосторонняя



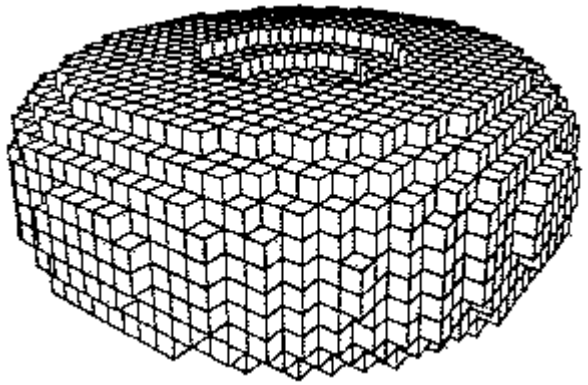
Левосторонняя



Воксельная модель – дискретизация пространства по равномерной сетке

Структура

- Равномерная сетка, каждый элемент которой показывает, если в нем часть объекта
- Ячейка называется воксель (voxel = volume element)
- Каждый воксель принимает значение 0 или 1
- Может также задавать плотность (0-1)

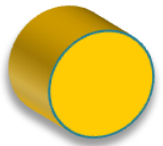


```
bool Voxels[X][Y][Z];
```

```
BYTE Voxels[X][Y][Z];
```

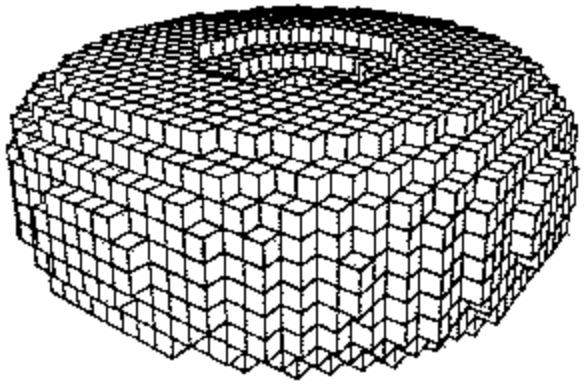
Способ получения

- Дискретизация трехмерных данных на равномерной сетке



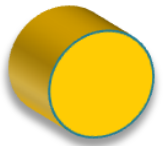
Воксельная модель: описывает объем

- Описывает объем
- Дискретное представление: приближение реального объекта!
- Плохо описываются части объекта, не параллельные сторонам воксельного куба
- Явное представление
- Размер данных пропорционален кубу разрешения сетки
  - 1 байт на точку:
    2000 x 2000 x 2000 = 7,45 Гб !



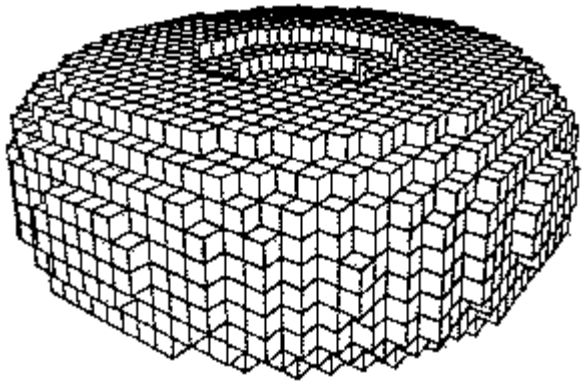
```
bool Voxels[X][Y][Z];
```

```
BYTE Voxels[X][Y][Z];
```



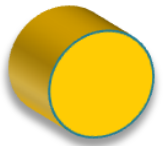
Воксельное представление: типичные алгоритмы

Пространственные алгоритмы



- Вычисление объема объекта
- Нахождение центра масс
- Булевы операции (пересечение, объединение)

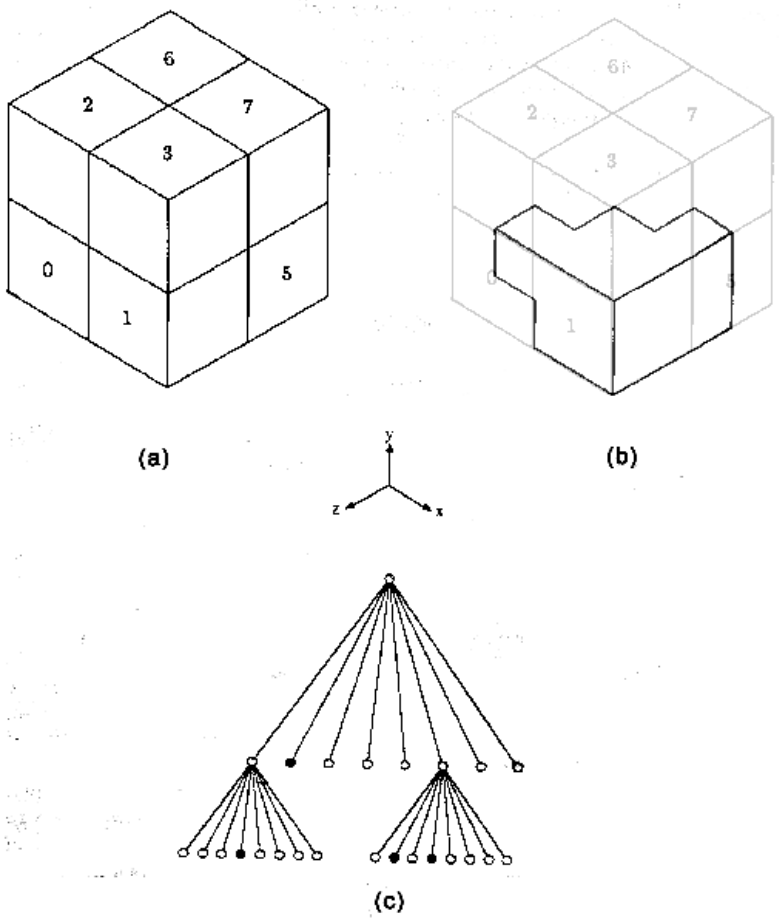
Плохо работают алгоритмы, требующие понятия поверхности!



Октодерево – иерархический вариант воксельной модели

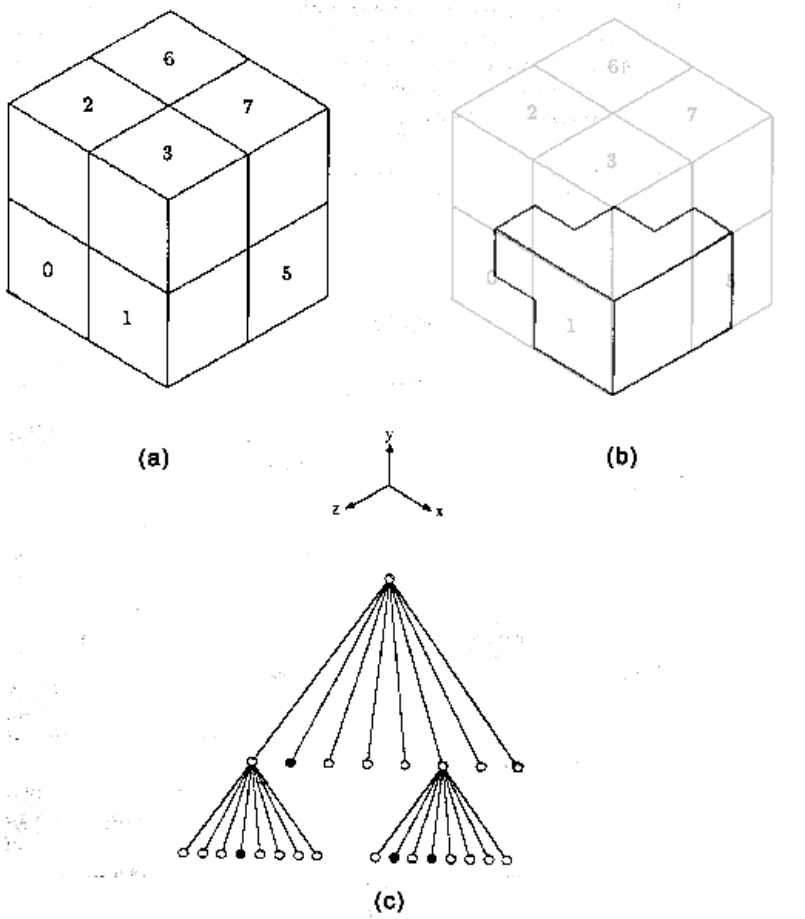
- Рекурсивное разбиение пространства на восемь октант

- Ветвь дерева:
  - Код = B (черный) - заполнено
  - Код = W (белый) - пусто
  - Код = G (серый) – частично
    - Эти подразбиты на 8 потомков

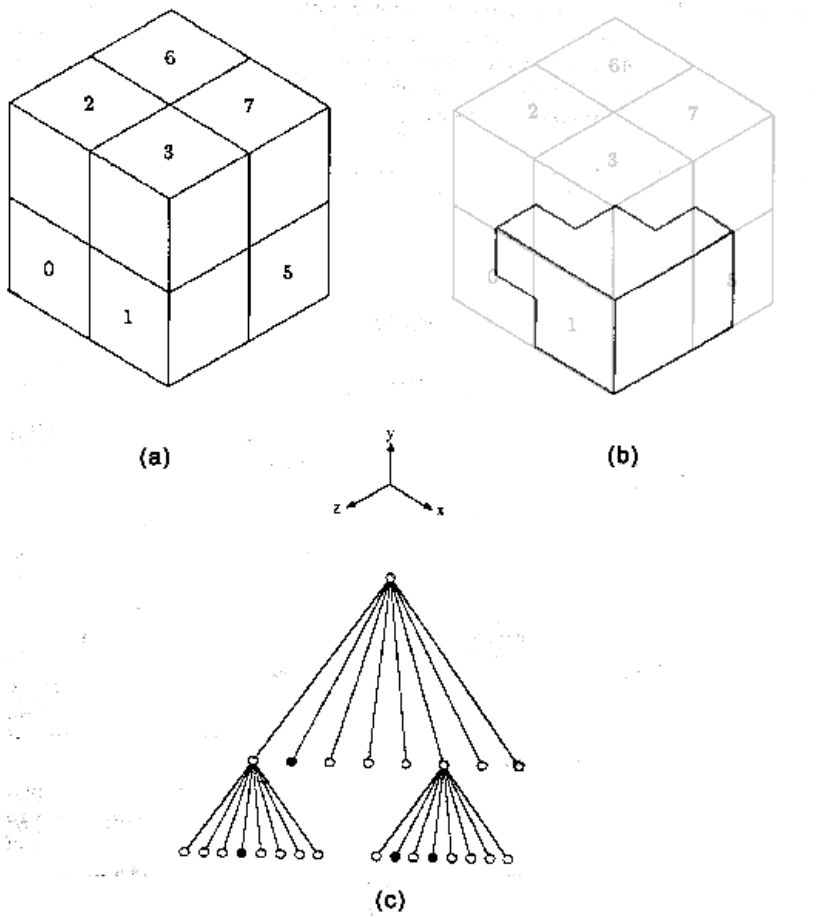


Структура данных октодерева: типичная рекурсивная иерархическая структура

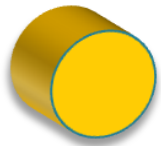
```
struct OctoTreeNode
{
        BYTE Value;
        OctoTreeNode* Children[8];
};
```



Октодерево применяется для оптимизации воксельного представления

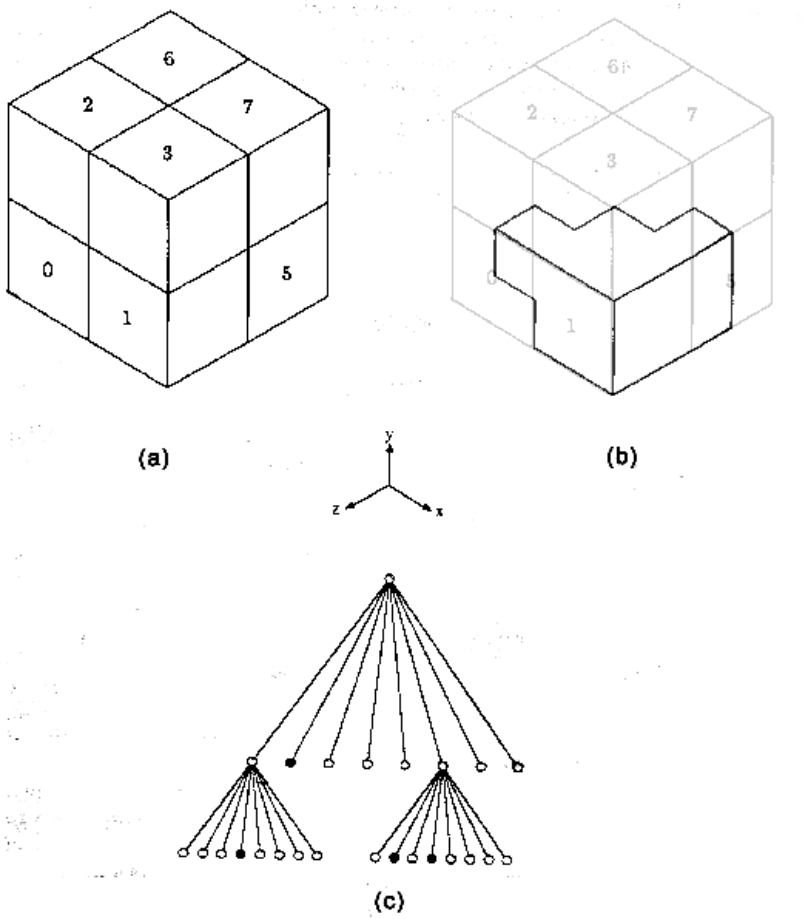


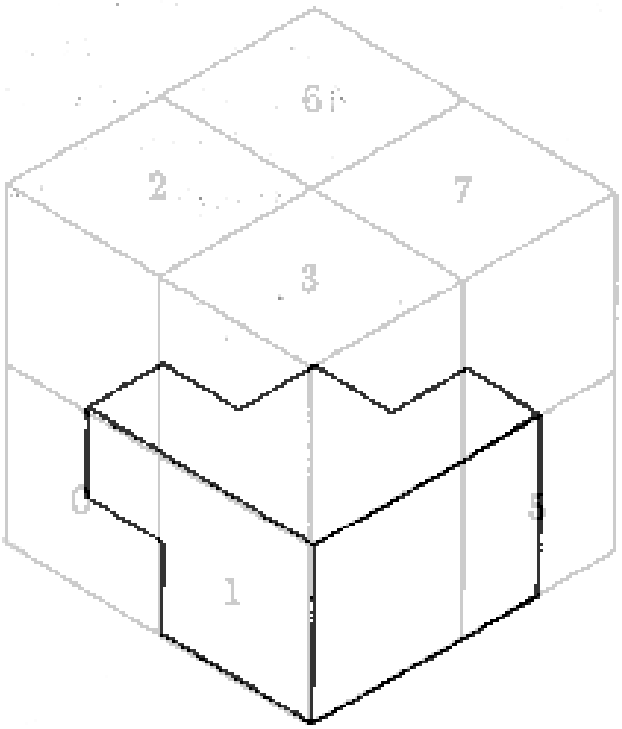
- Свойства:
  - Позволяет хранить информацию только о блоках, относящихся к объекту
  - Число элементов пропорционально площади поверхности объекта, т.е. квадрату разрешения
  - Для разреженных моделей позволяет уменьшить размер в тысячи раз!
- Способ получения
  - Из воксельного представления или напрямую, через дискретизацию

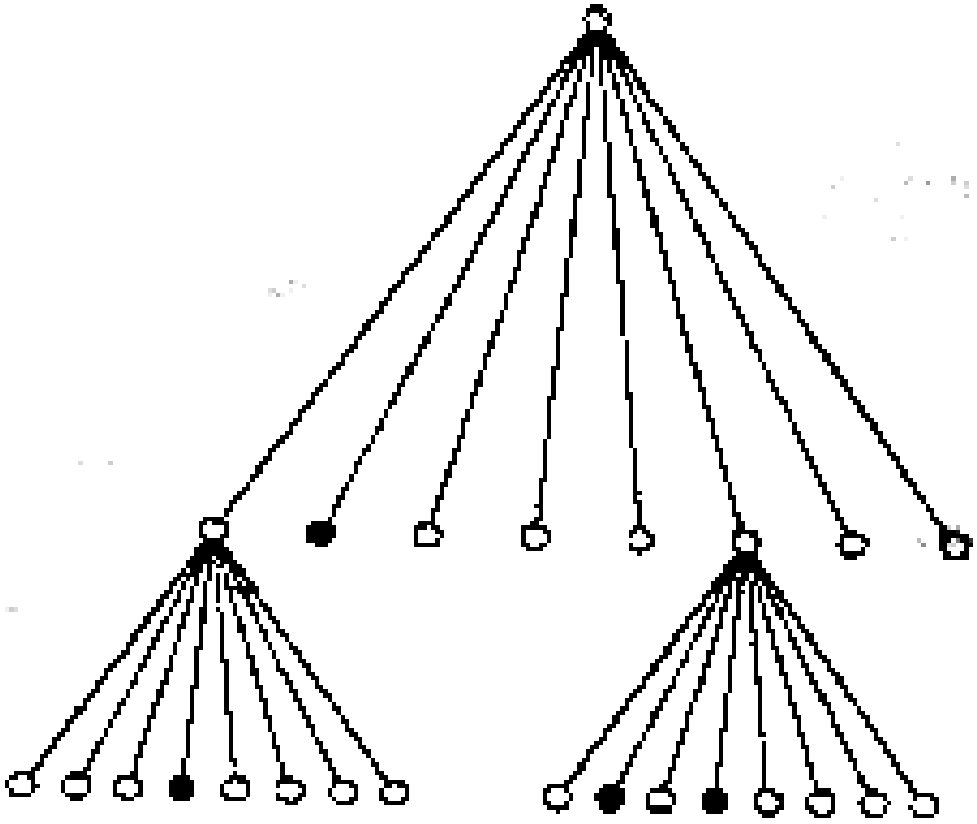


Вариант линейной записи 1: пути к листовым узлам

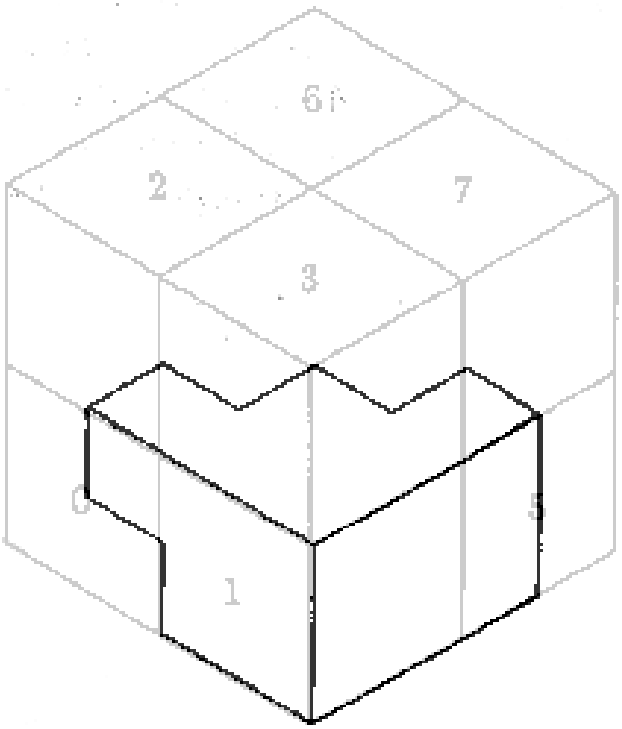
- Октанты дерева пронумерованы от 0 до 7
- Конструирование адреса каждой ветви дерева, кроме корня.
- Адрес ветви уровня i – последовательности i чисел от 0 до 7 – путь от корня к этой ветви
- Символ X: если в последовательности чисел меньше, чем максимальное разрешение
- Линейная запись дерева есть просто сортированный массив адресов ветвей с кодом "черный"

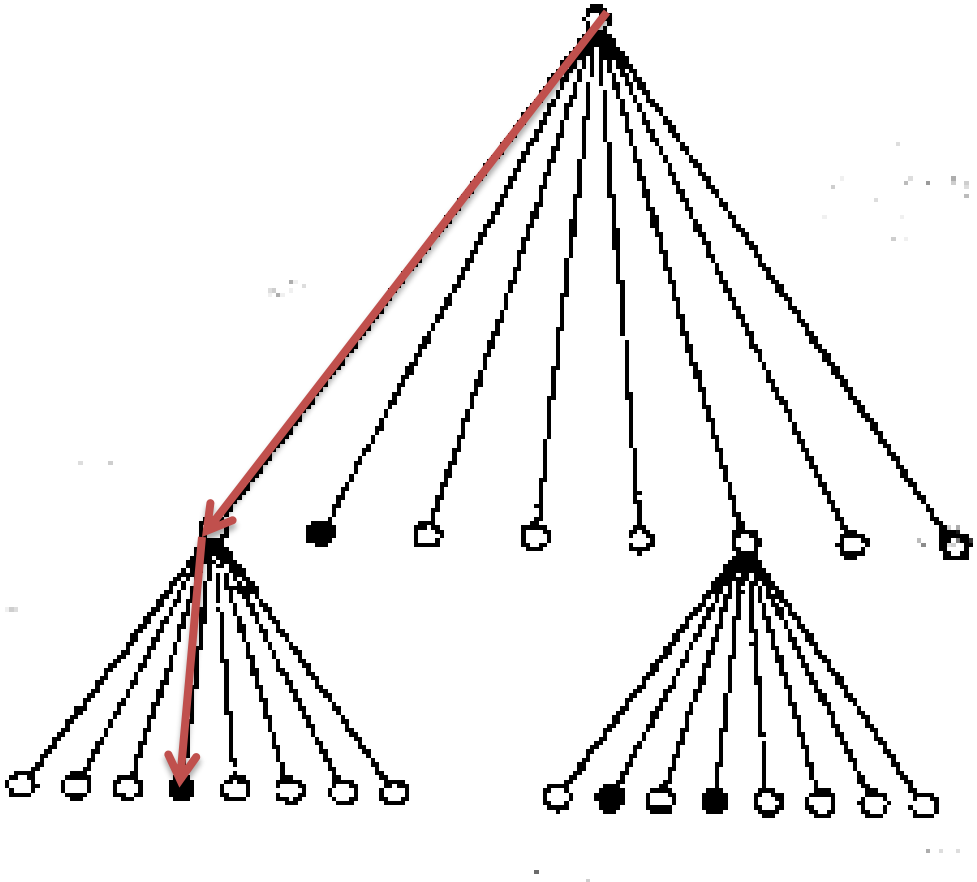




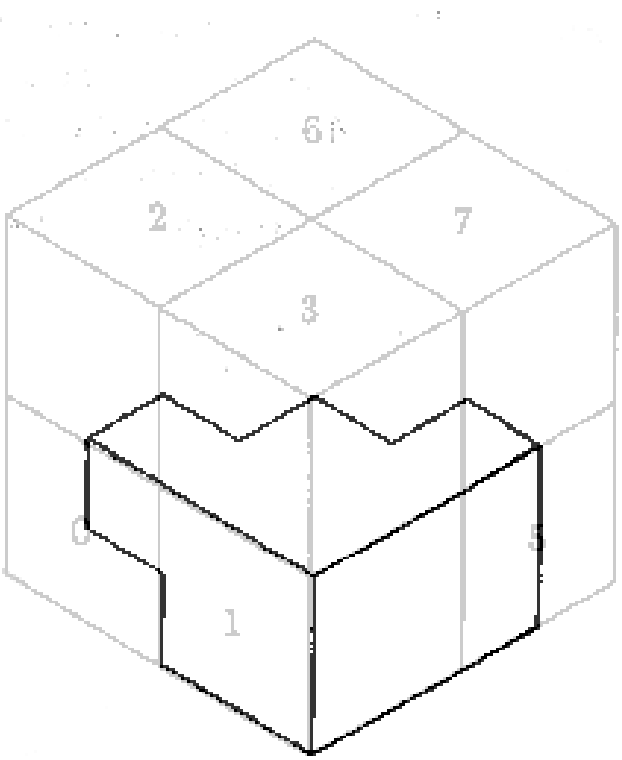


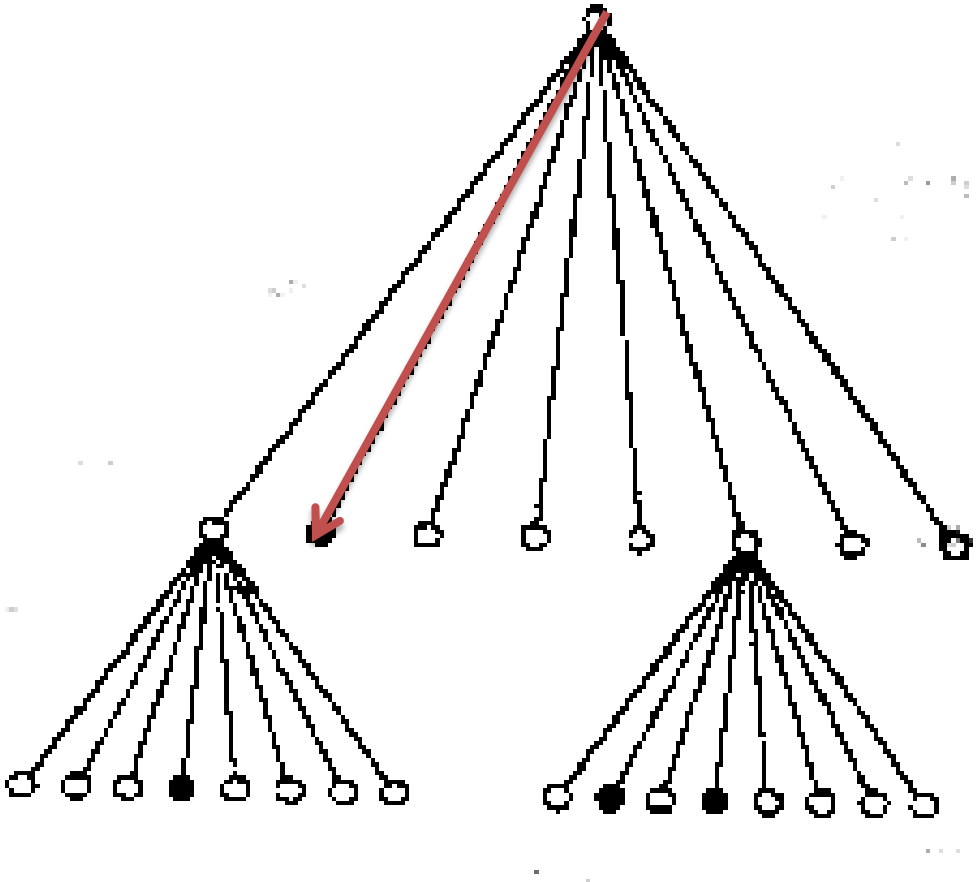
Линейная запись 1: {03,1X,51,53}



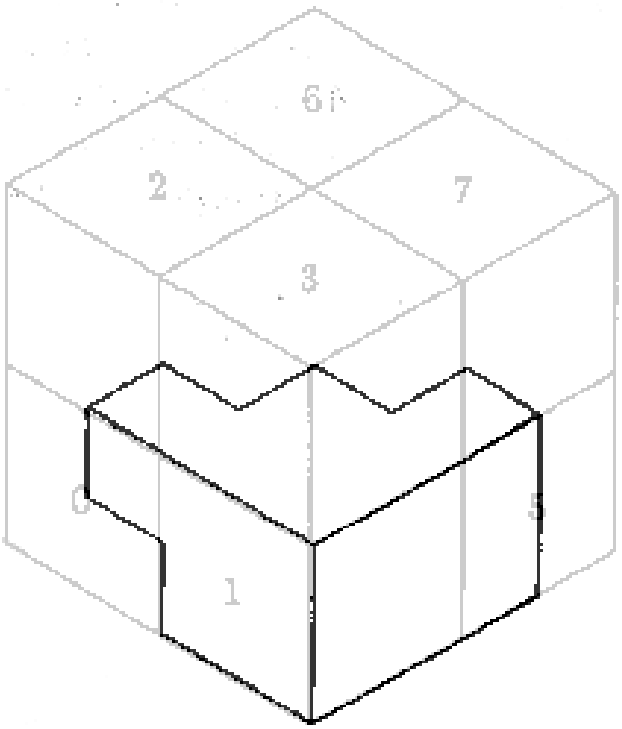


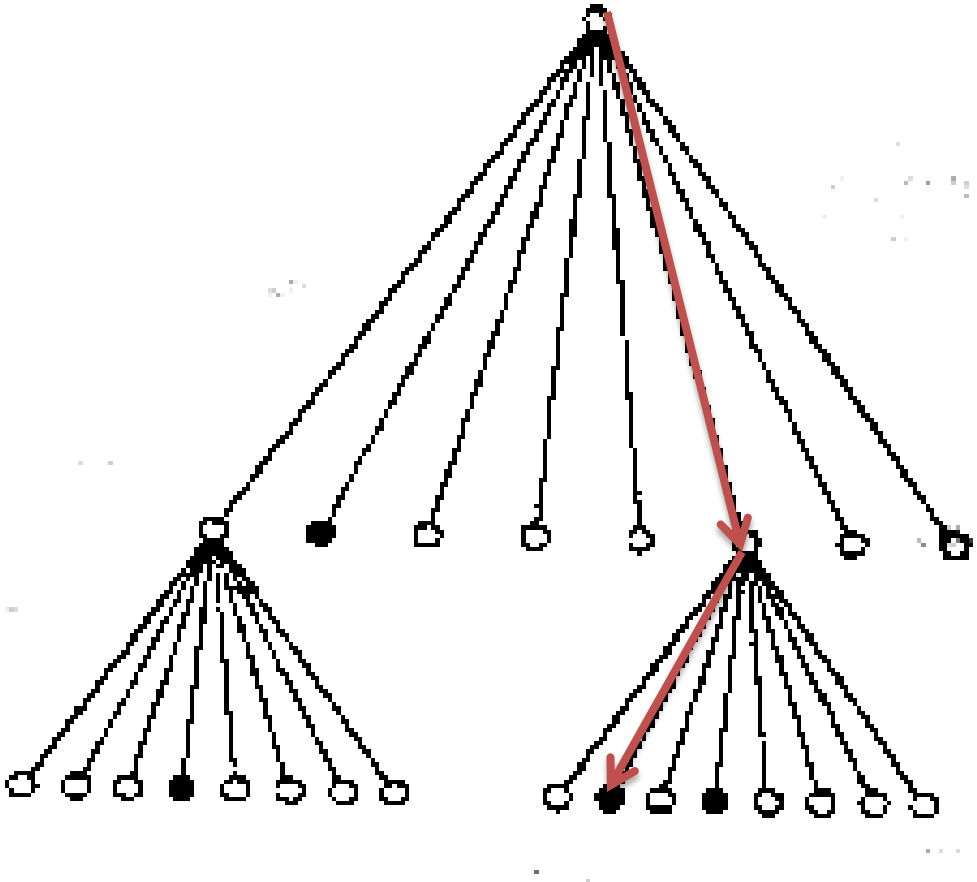
Линейная запись 1: {03,1X,51,53}



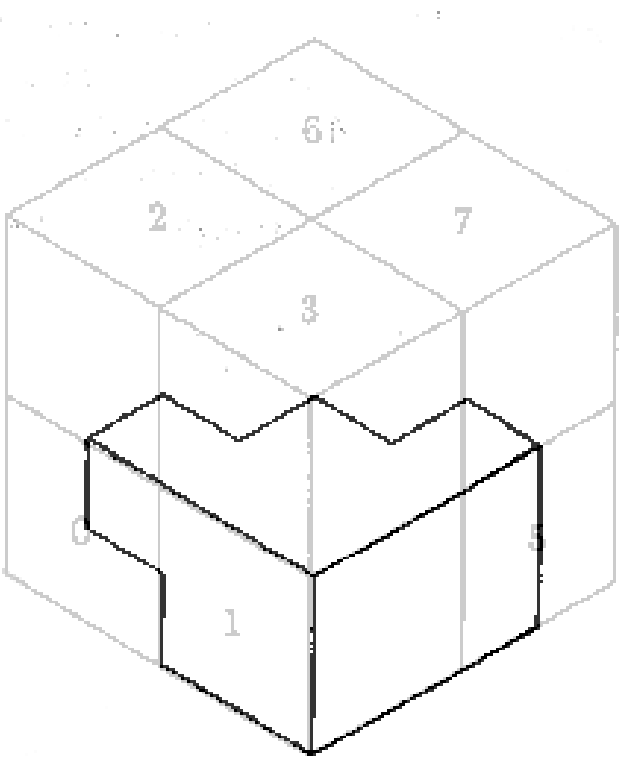


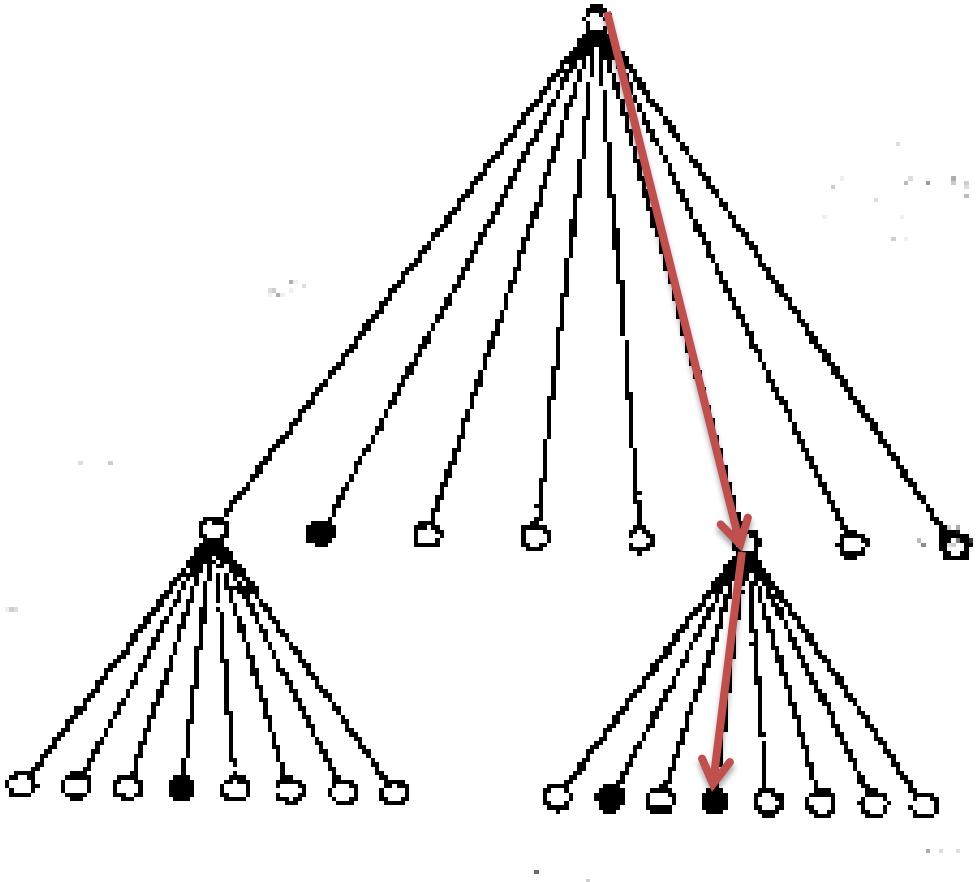
Линейная запись 1: {03,1X,51,53}





Линейная запись 1: {03,1X,51,53}

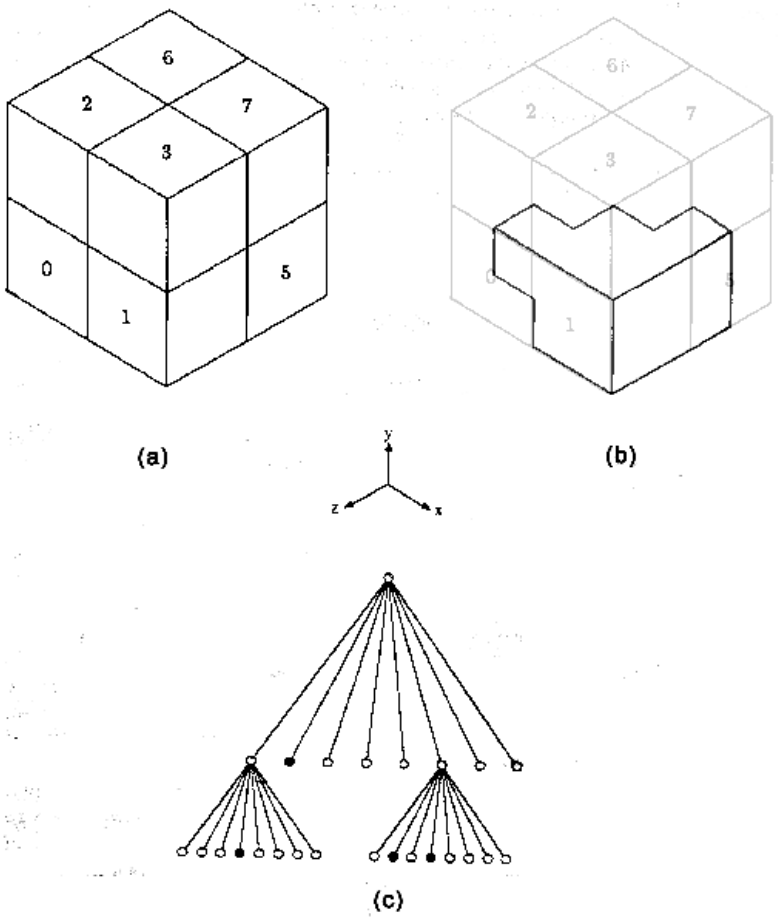


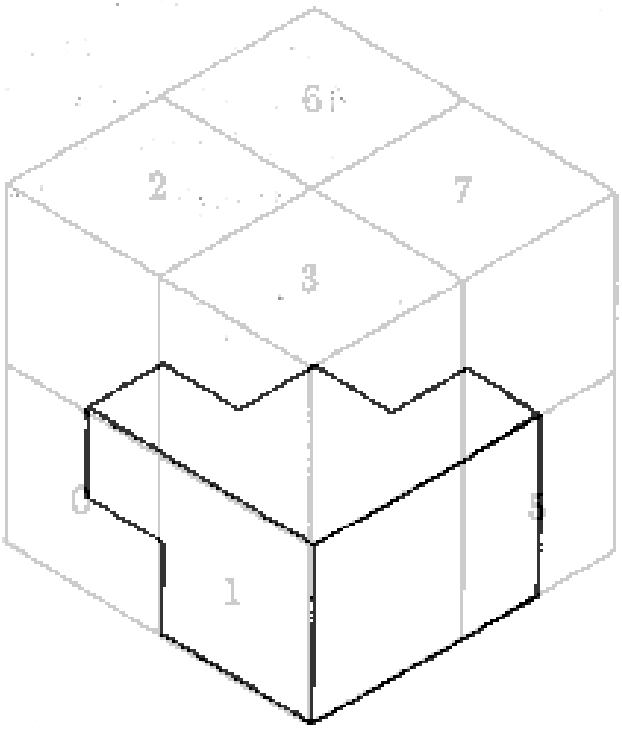


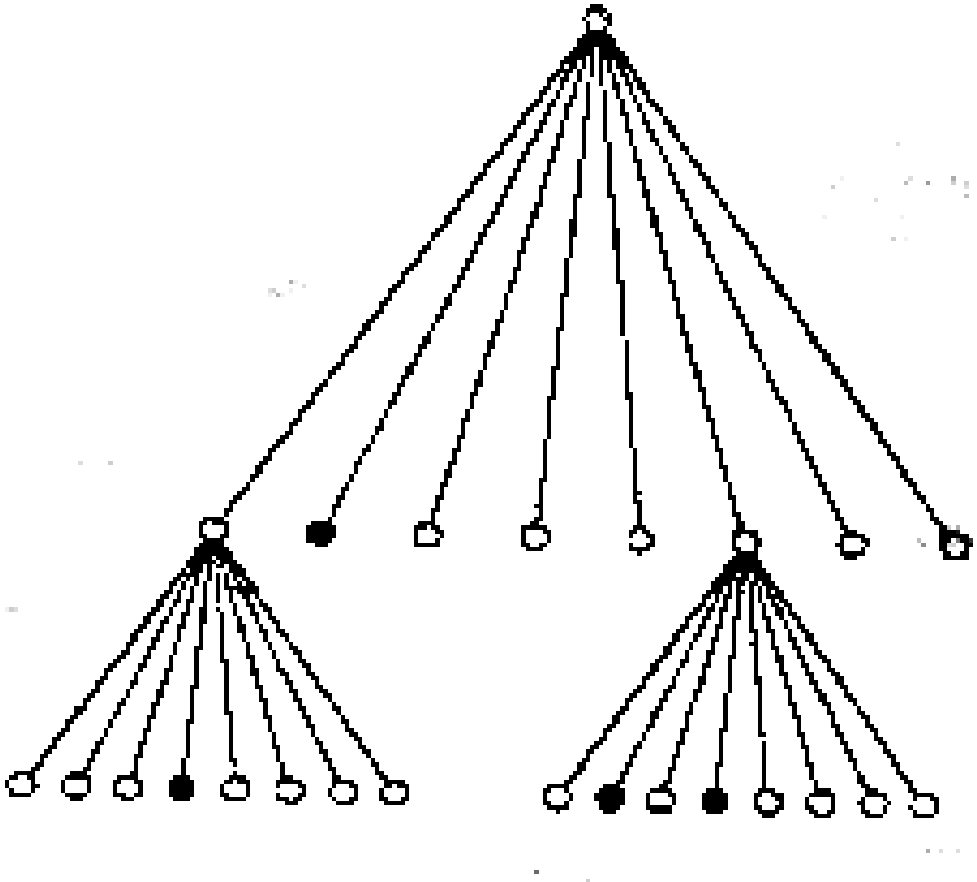
Линейная запись 1: {03,1X,51,53}

Вариант линейной записи 2: обход дерева в глубину

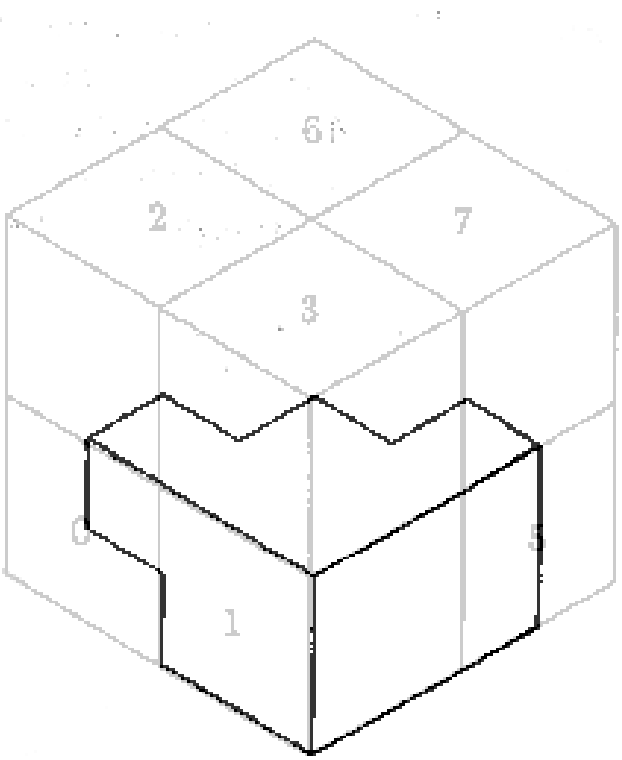
- Обход дерева в фиксированном порядке
- Трехсимвольная запись
  - «B»: черная ветвь
  - «W»: белая ветвь
  - «(» : серая ветвь

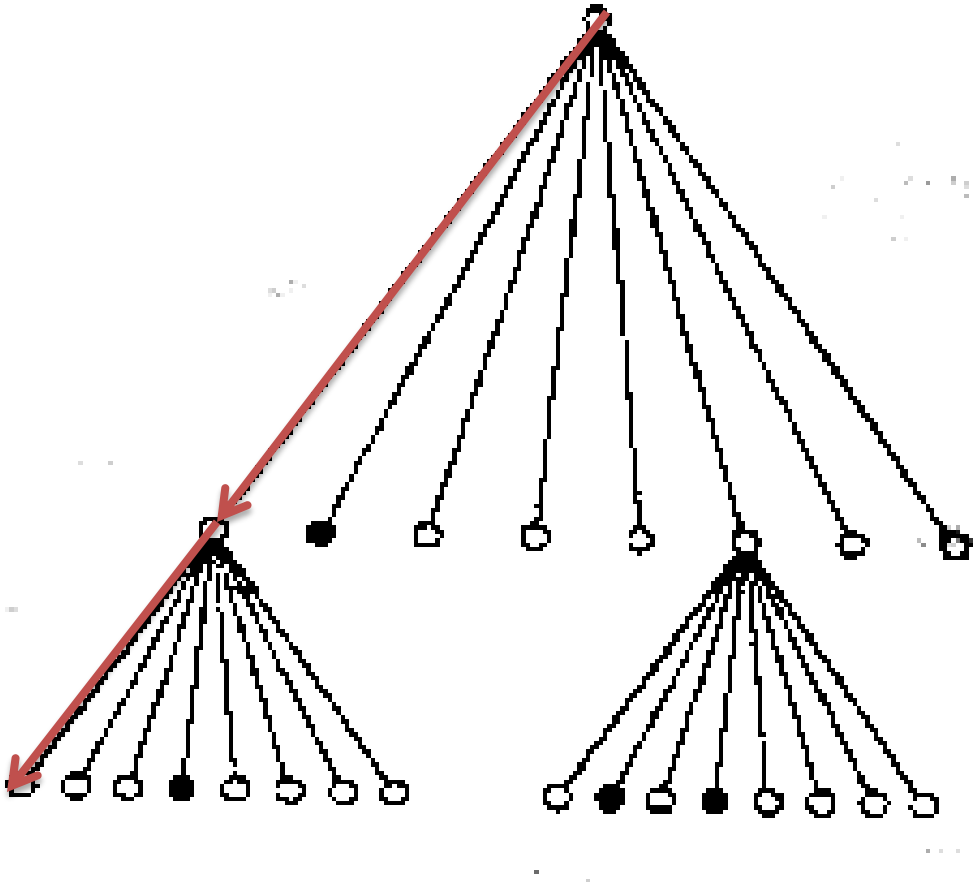




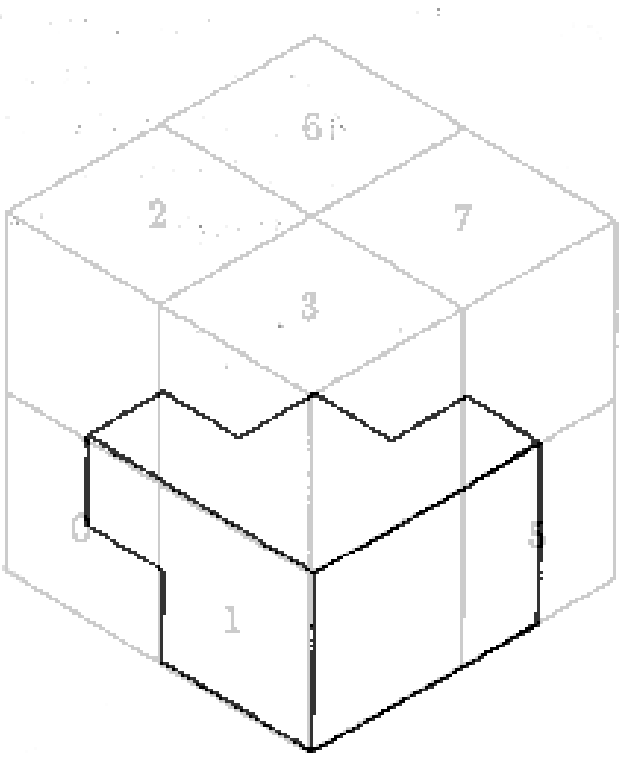


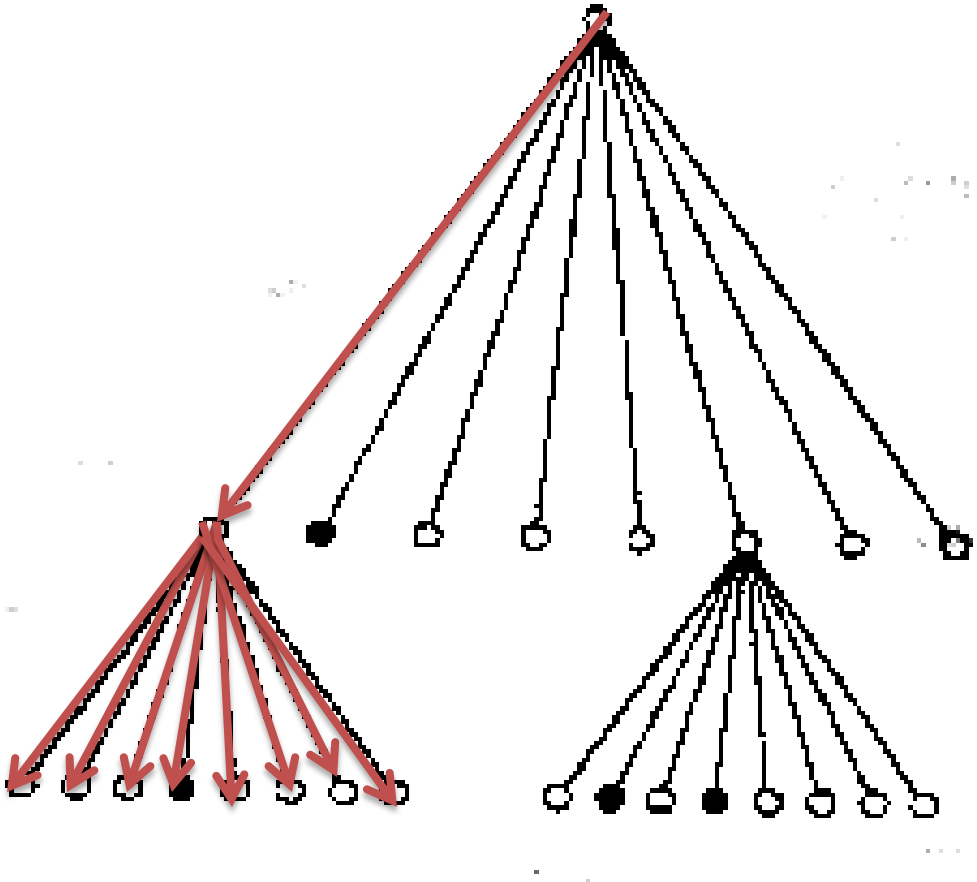
Линейная запись 2:
((WWWBWWWWBWWW(WBWBWWWWWW



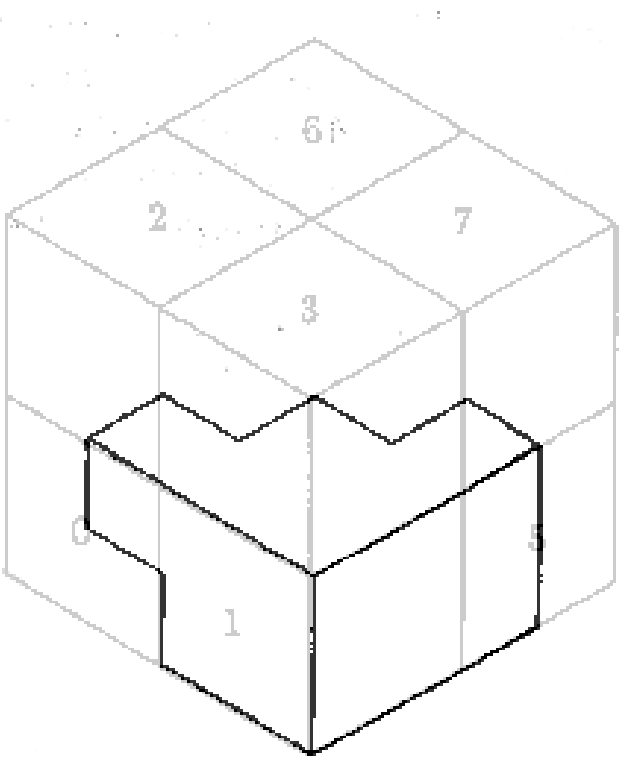


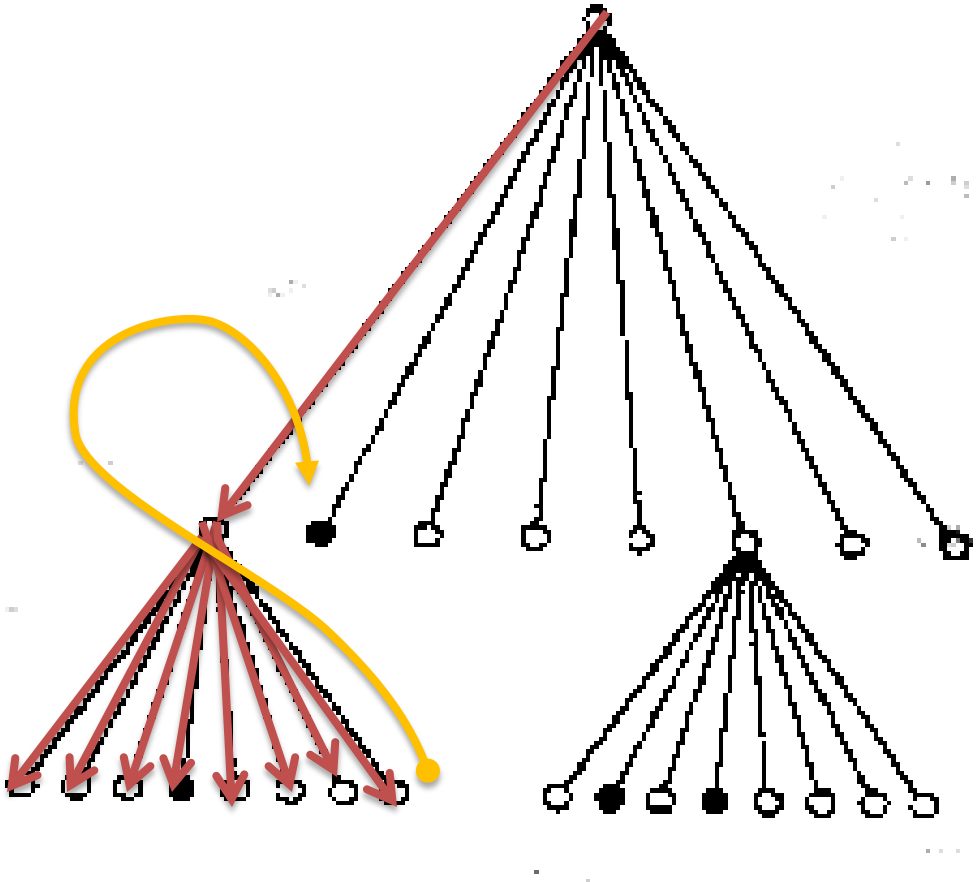
Линейная запись 2:
((WWWBWWWWBWWW(WBWBWWWWWW





Линейная запись 2:
((WWWBWWWWBWWW(WBWBWWWWWW

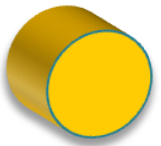




Линейная запись 2:
((WWWBWWWWBWWW(WBWBWWWWWW

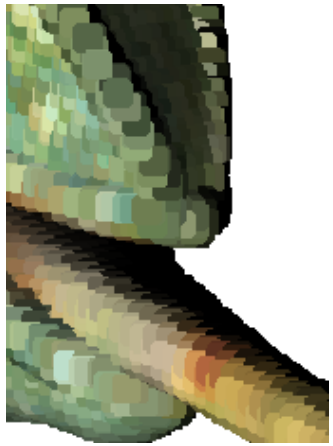
Октодерево часто применятся для хранения воксельных данных и при визуализации

- Удобно для синтеза:
  - Переменный уровень детализации
  - Вывод back-to-front
- Усложняются операции, требующие информации о смежных ячейках
- Сложно с анимацией, нужно перестраивать дерево



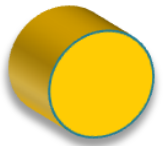
Точечное представление заменяет регулярную воксельную сетку на нерегулярную

- Набор неструктурированных точек
- Требования по памяти пропорциональны количеству точек
- Дискретное, явное представление

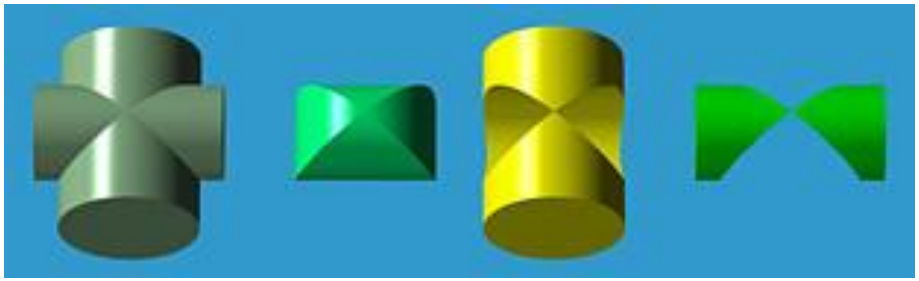


Точечное представление: структура

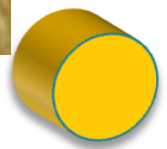
- Массив точек с атрибутами
  - Атрибуты: положение, цвет, нормаль, размер
- Описывает только принадлежащие объекту части пространства
- Явное хранение координат => возможное увеличение размера
- Нет связанности, инциндентности => для выполнения преобразований обычно строятся дополнительные структуры данных (октодерево)



Конструктивная геометрия: эффективно описывает и поверхность и объем

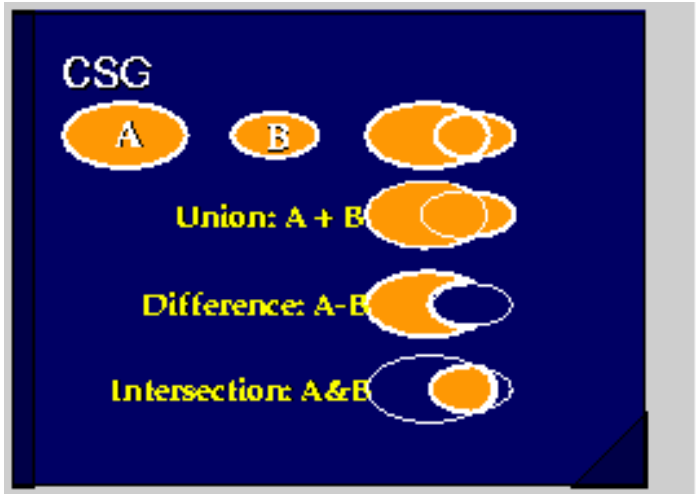


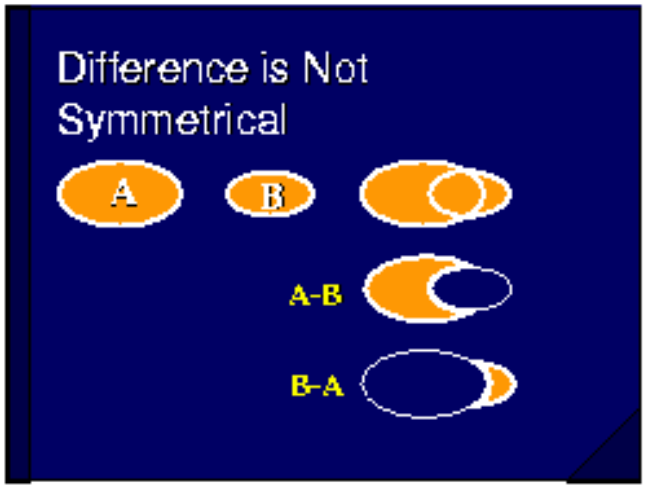
- Структура
  - Набор базовых примитивов
    - сфера, куб, цилиндр...
  - Операции по их комбинированию
- Способ получения
  - Ручное моделирование
- Свойства
  - Описывает объем и поверхность (!)
  - Непрерывное
  - Явное

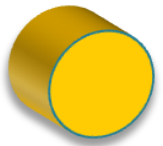


Задается набор операций для базовых примитивов

- Перенос/поворот/масштабирование
- Теоретико-множественные:
  - Объединение
  - Разность
  - Пересечение

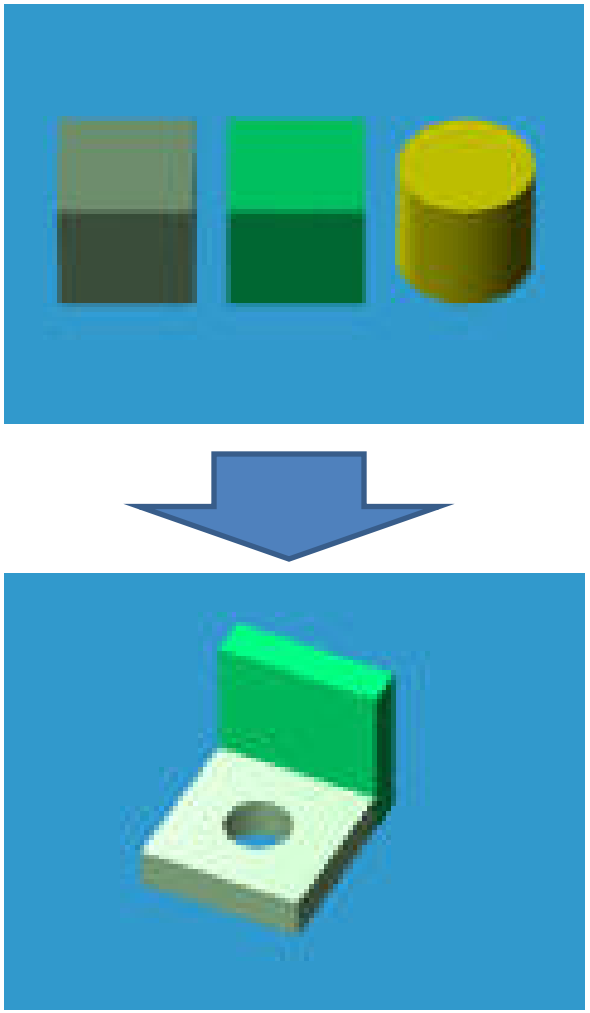






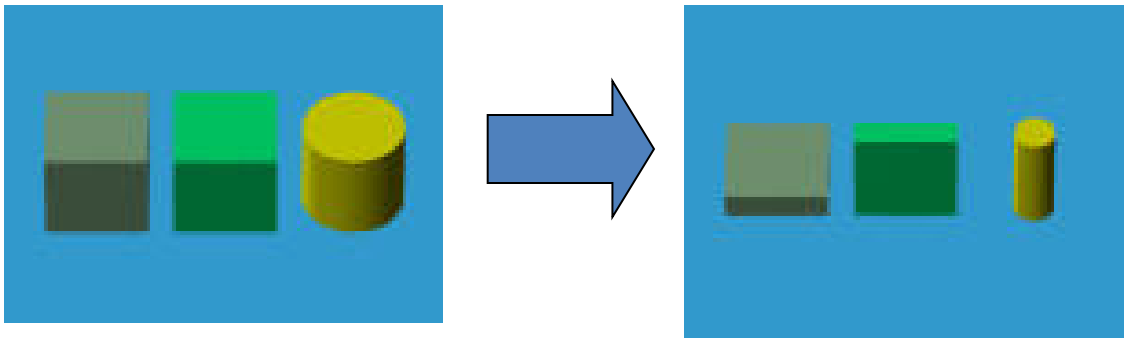
Последовательность операций задает финальный объект

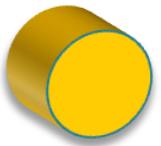
```
diff(
    union(
        trans1(Block1),
        trans2(Block2)
    ),
    trans3(Cylinder)
)
```



Разбор последовательности выполнения операций над телами

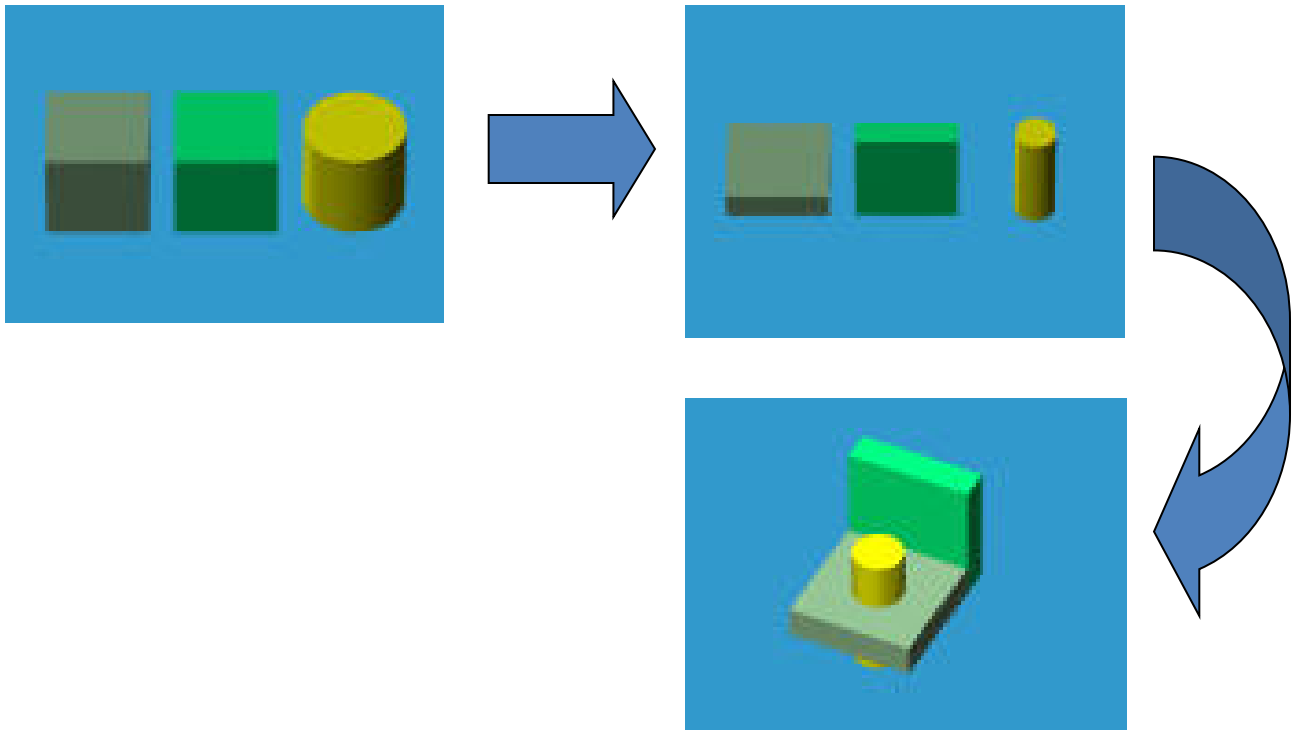
diff(union(**trans1(Block1), trans2(Block2)**),
trans3(Cylinder))

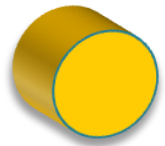




Разбор последовательности выполнения операций над телами

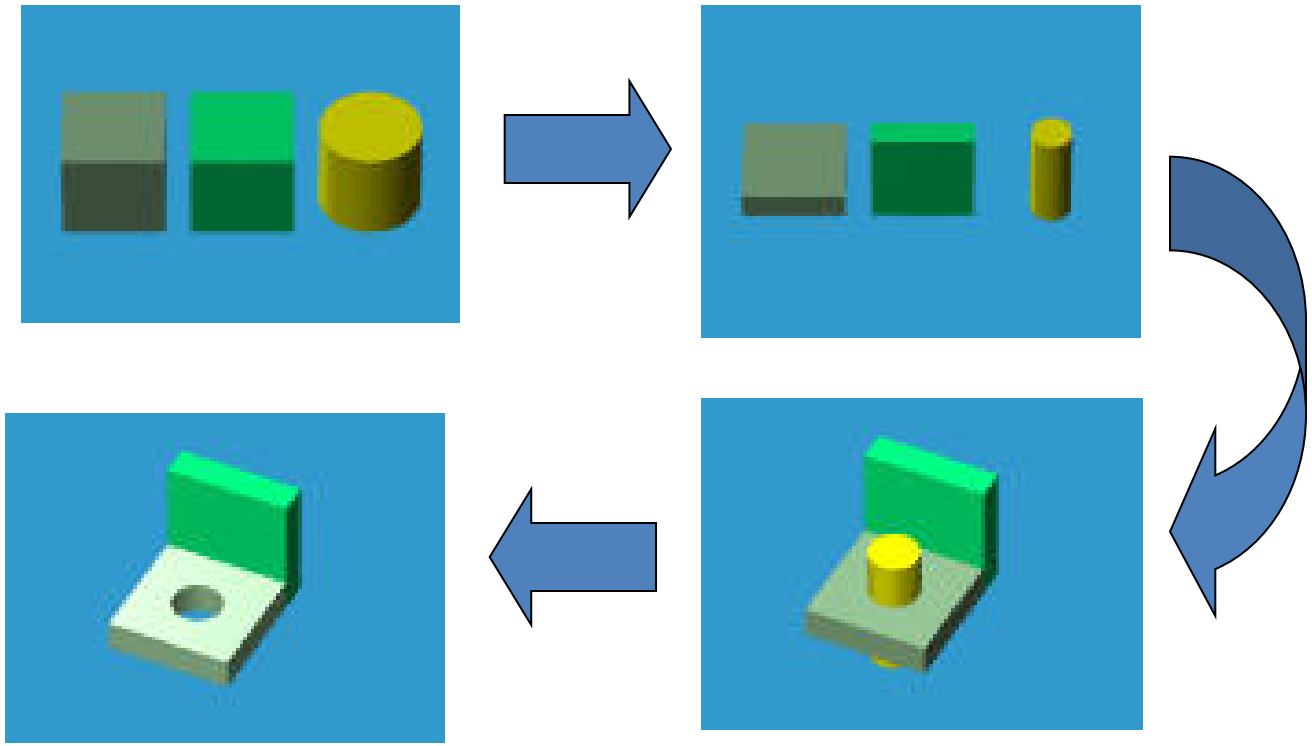
diff(**union(**trans1(Block1), trans2(Block2)**)**,
trans3(Cylinder))

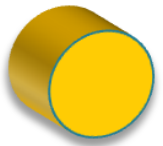




Разбор последовательности выполнения операций над телами

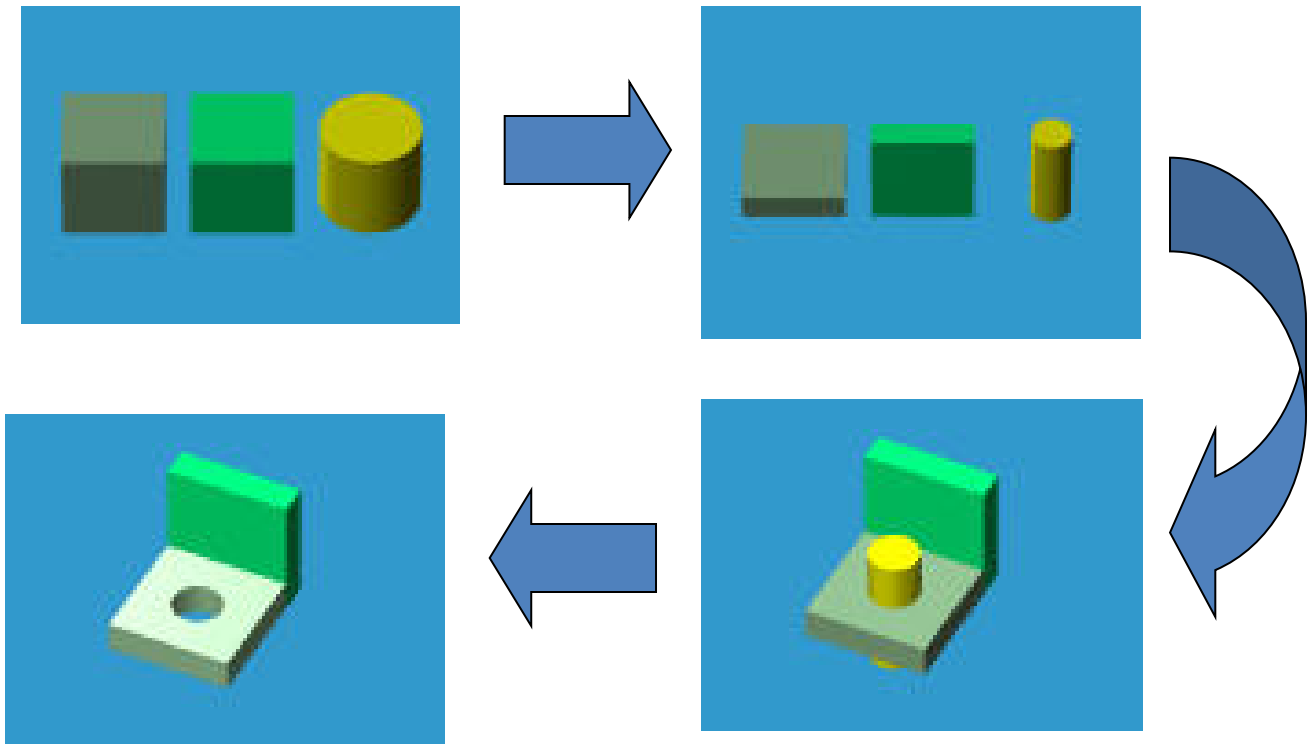
diff(union(trans1(Block1), trans2(Block2)), trans3(Cylinder)**)**





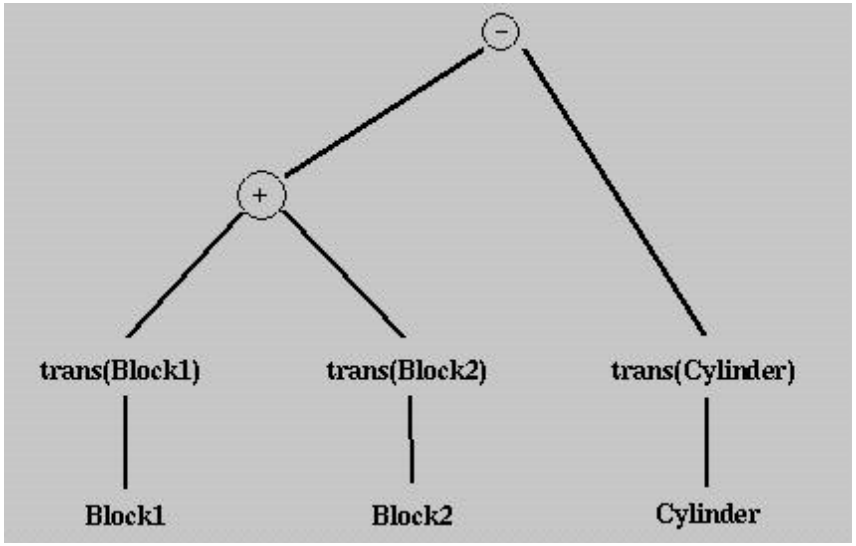
Разбор последовательности выполнения операций над телами

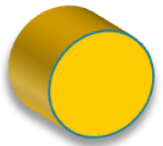
diff(union(trans1(Block1), trans2(Block2)),
trans3(Cylinder))



Структура данных для конструктивной геометрии – направленный ациклический граф

- Дерево из операций и базовых объектов
- Корень – результирующий объект
- Листья – базовые примитивы
- Число потомков равно числу операндов операции
- Из-за повторного использования превращается в направленный ациклический граф.

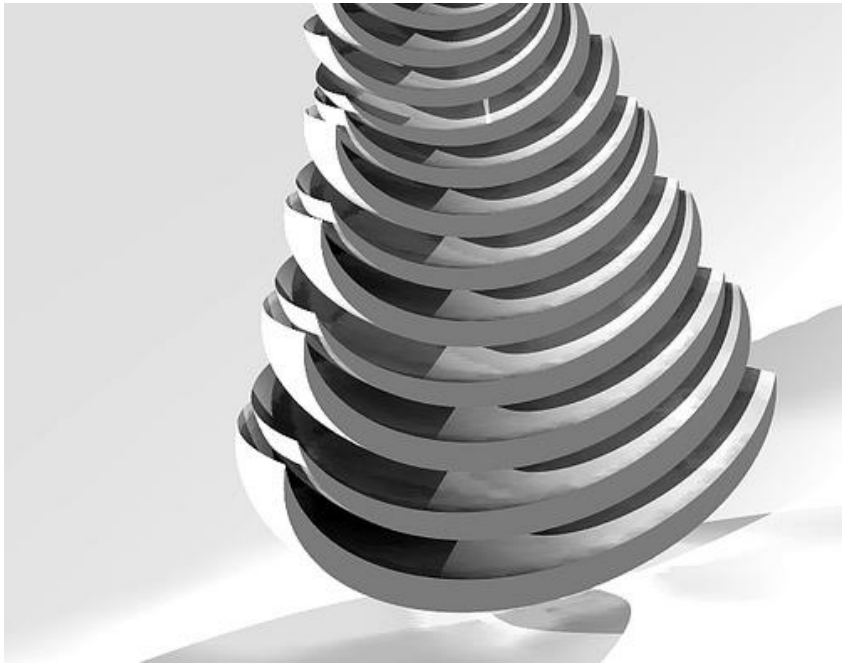




Конструктивная геометрия эффективно работает с «объемными» задачами, но также имеет понятие поверхности

Пространственные алгоритмы

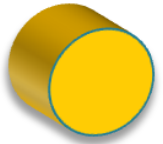
- вычисление объема объекта
- нахождение центра масс
- …



Есть понятие поверхности!

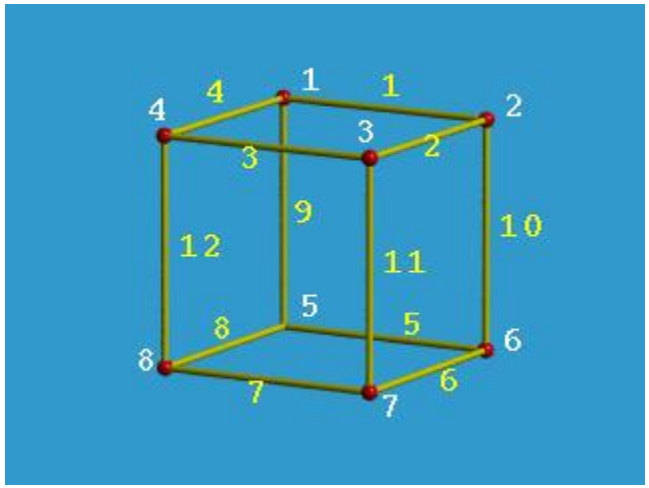
Основной недостаток – на практике можно построить только синтетически

Также сложно визуализировать



Объект можно моделировать через выделение характерных точек и ребер

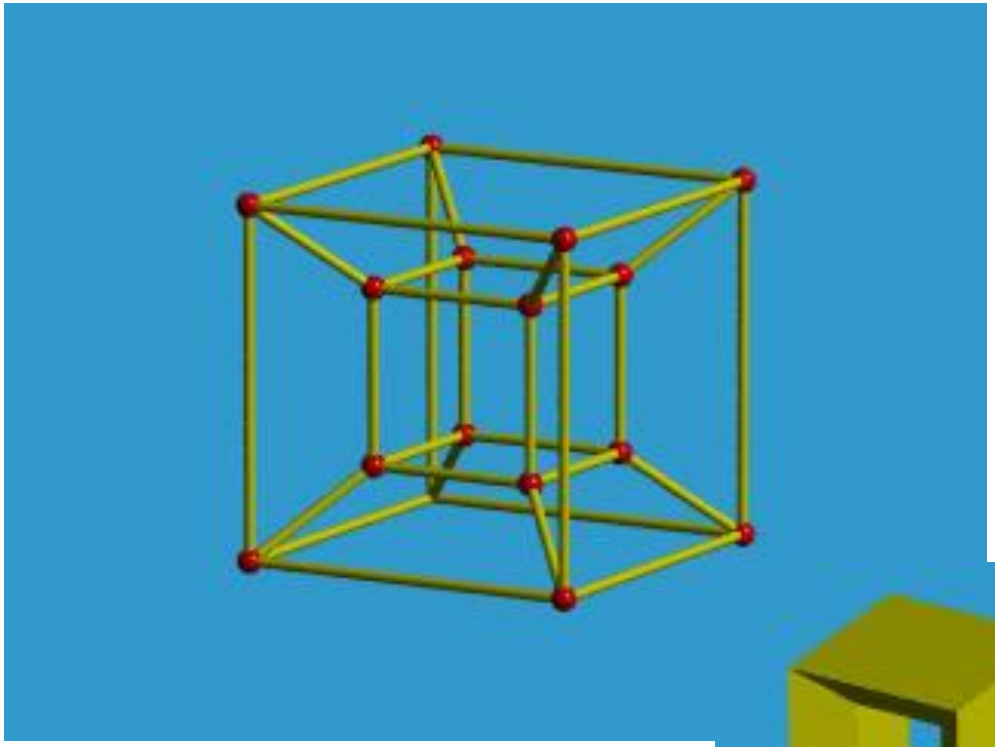
Каркасное представление



i	X	Y	Z
1	1	1	1
2	1	-1	1
3	-1	-1	1
4	-1	1	1
5	1	1	-1
6	1	-1	-1
7	-1	-1	-1
8	-1	1	-1

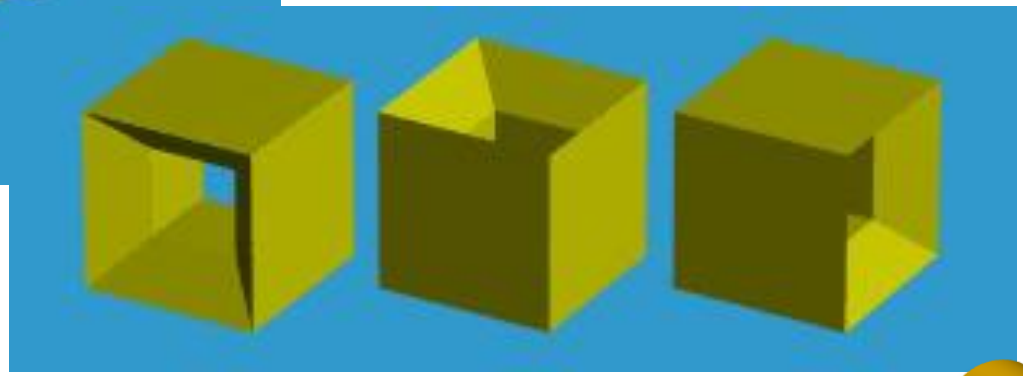
1	(1,2)	**2**	(2,3)	**3**	(3,4)
4	(5,6)	**5**	(5,6)	**6**	(6,7)
7	(7,8)	**8**	(8,5)	**9**	(1,5)
10	(2,6)	**11**	(3,7)	**12**	(4,8)

Каркасное представление: неоднозначная интерпретация для случая сплошных тел



Нужна дополнительная информация!

Как верно отобразить?

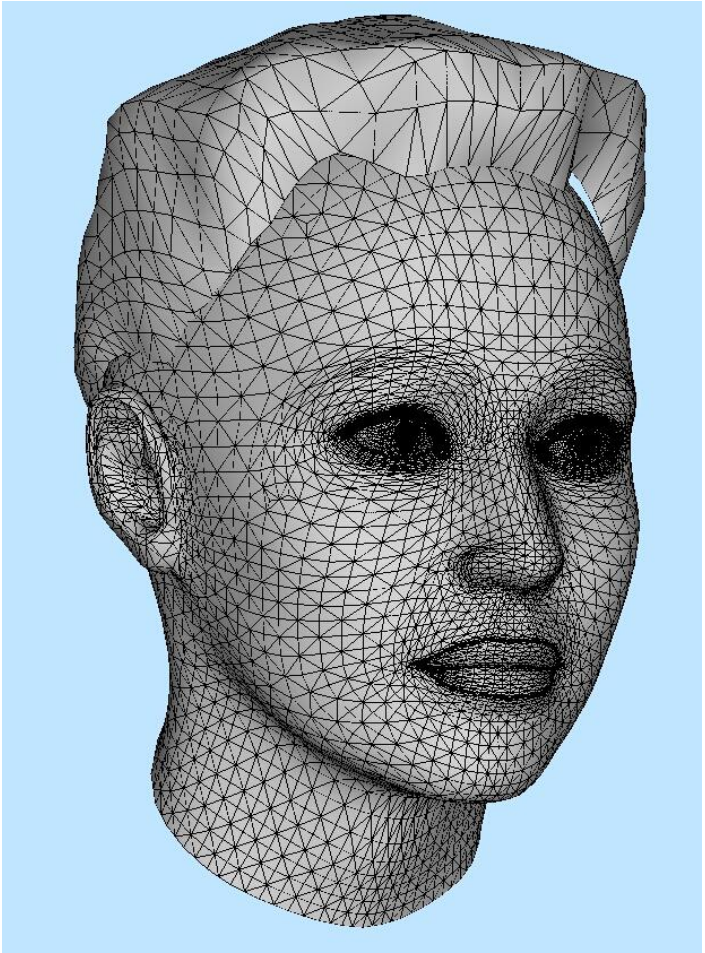


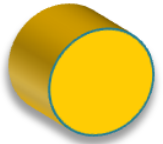
Граничное представление содержит не только каркас, но грани

Кусочная аппроксимация поверхности объекта

Рассматриваем представления первого порядка: линейная интерполяция

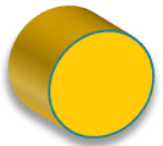
Структура данных: вершины + грани





Граничные представления различаются по способам хранения информации о гранях и ребрах

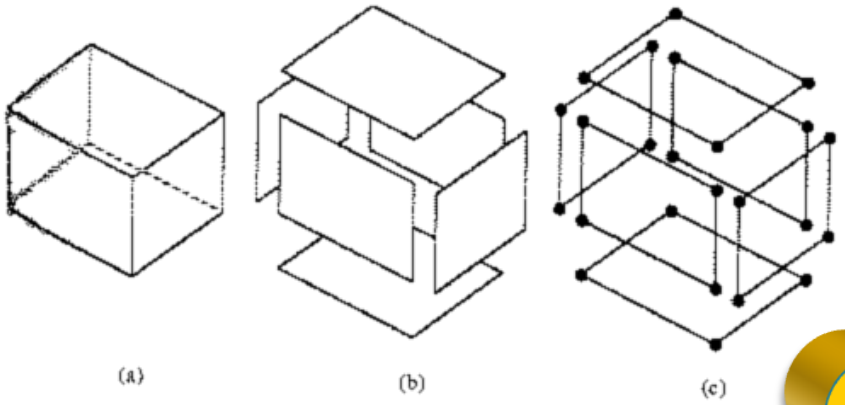
- Явное представление
- Индексированное по вершинам
- Индексированное по ребрам
- «Крылатое» представление



Явное представление хранит список граней

- Каждая грань – полигон, состоящий из последовательности координат вершин
- Объект состоит из набора граней
- Недостатки
  - Взаимоотношения граней заданы неявно
  - Координаты вершин дублируются
  - Алгоритмы поиска инцидентных ребер требуют полного перебора

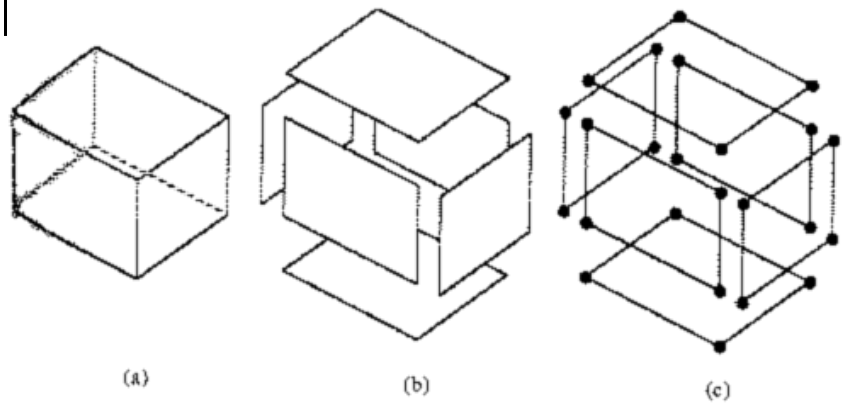
Грани	Координаты
f1	x1 y1 z1, x2 y2 z2, x3 y3 z3, x4 y4 z4
f2	x6 y6 z6, x2 y2 z2, x1 y1 z1, x5 y5 z5
f3	x7 y7 z7, x3 y3 z3, x2 y2 z2, x6 y6 z6
f4	x8 y8 z8, x4 y4 z4, x3 y3 z3, x7 y7 z7
f5	x5 y5 z5, x1 y1 z1, x4 y4 z4, x8 y8 z8
f6	x8 y8 z8, x7 y7 z7, x6 y6 z6, x5 y5 z5



Индексированное по вершинам хранит индексы вершин

- Выделение координат вершин в отдельную структуру
- С гранями ассоциируются не координаты вершин, а индексы в массиве координат вершин
- Недостатки
  - Аналогично явному представлению

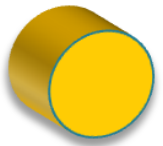
Вершины	Координаты	Грани	Вершины
v1	x1 y1 z1	f1	v1 v2 v3 v4
v2	x2 y2 z2	f2	v6 v2 v1 v5
v3	x3 y3 z3	f3	v7 v3 v2 v6
v4	x4 y4 z4	f4	v8 v4 v3 v7
v5	x5 y5 z5	f5	v5 v1 v4 v8
v6	x6 y6 z6	f6	v8 v7 v6 v5
v7	x7 y7 z7		



Индексированное по ребрам хранит индексы ребер и индексы вершин

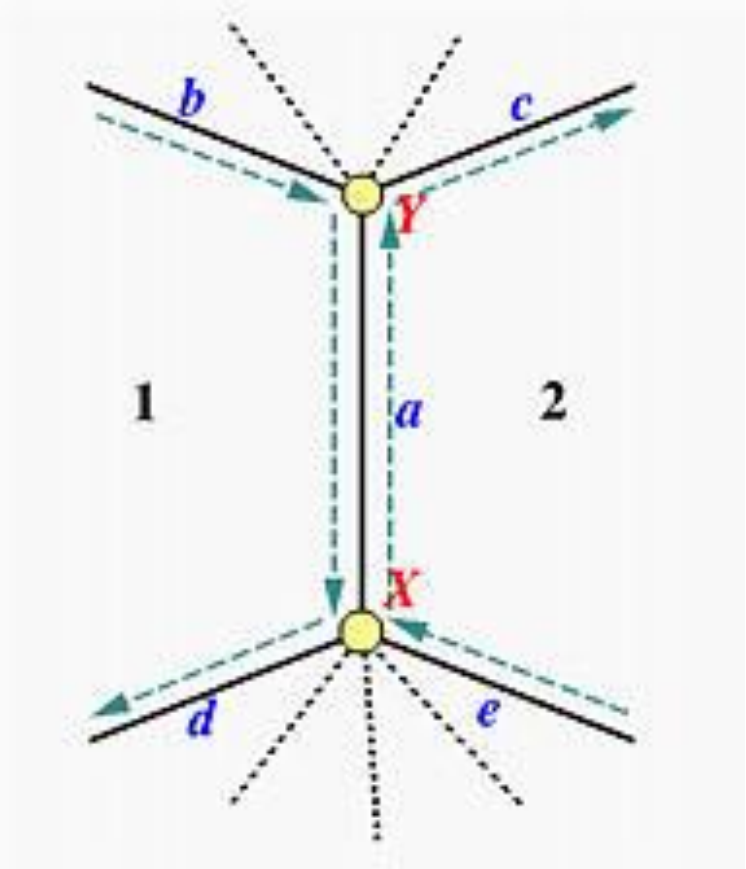
- Грани определяются через ребра
- Ребра задаются вершинами
- Вершины задаются положением в пространстве

Реб	Верш	Верш	Коорд	Грн	Ребра
e1	v1 v2	v1	x1 y1 z1	f1	e1 e2 e3 e4
e2	v2 v3	v2	x2 y2 z2	f2	e9 e6 e1 e5
e3	v3 v4	v3	x3 y3 z3	f3	e10 e7 e2 e6
e4	v4 v1	v4	x4 y4 z4	f4	e11 e8 e7 e3
e5	v1 v5	v5	x5 y5 z5	f5	e12 e5 e4 e8
e6	v2 v6	v6	x6 y6 z6	f6	e12 e11 e10 e9
e7	v3 v7	v7	x7 y7 z7		
e8	v4 v8	v8	x8 y8 z8		
e9	v5 v6				
e10	v6 v7				
e11	v7 v8				
e12	v8 v5				



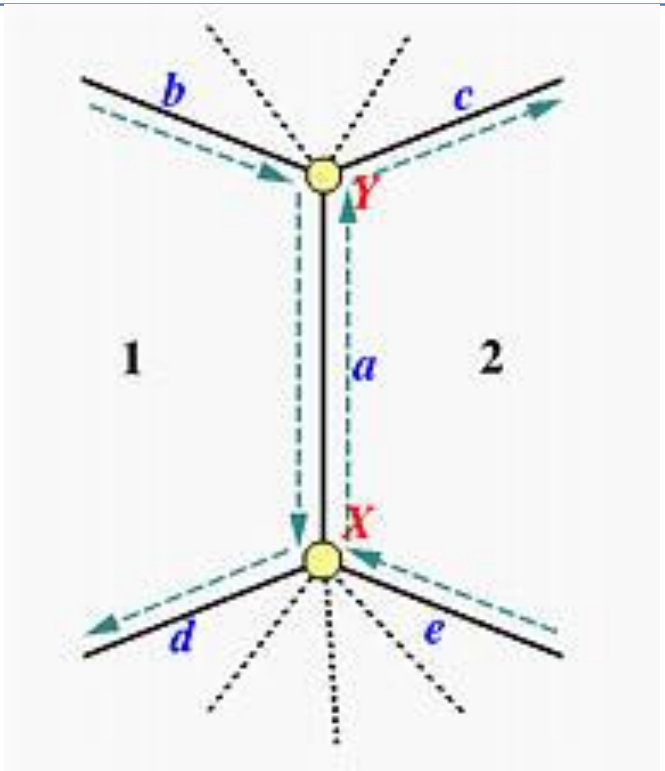
«Крылатое» представление связывает грани, ребра, вершины

- “Winded-Edge”
- Добавляется информация о взаимном расположении граней



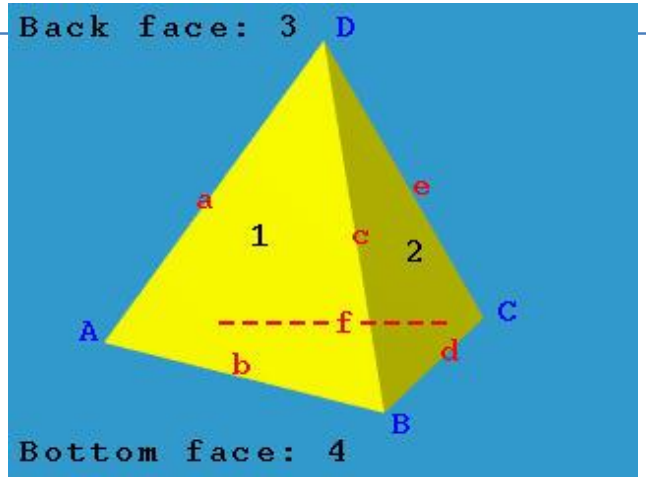
Структура данных построена вокруг ребра

```
struct Edge
{
        Vertex* X;
        Vertex* Y;

        Face* FaceLeft;
        Face* FaceRight;

        Edge* LeftPred;
        Edge* LeftSucc;
        Edge* RightPred;
        Edge* RightSucc;
};
```



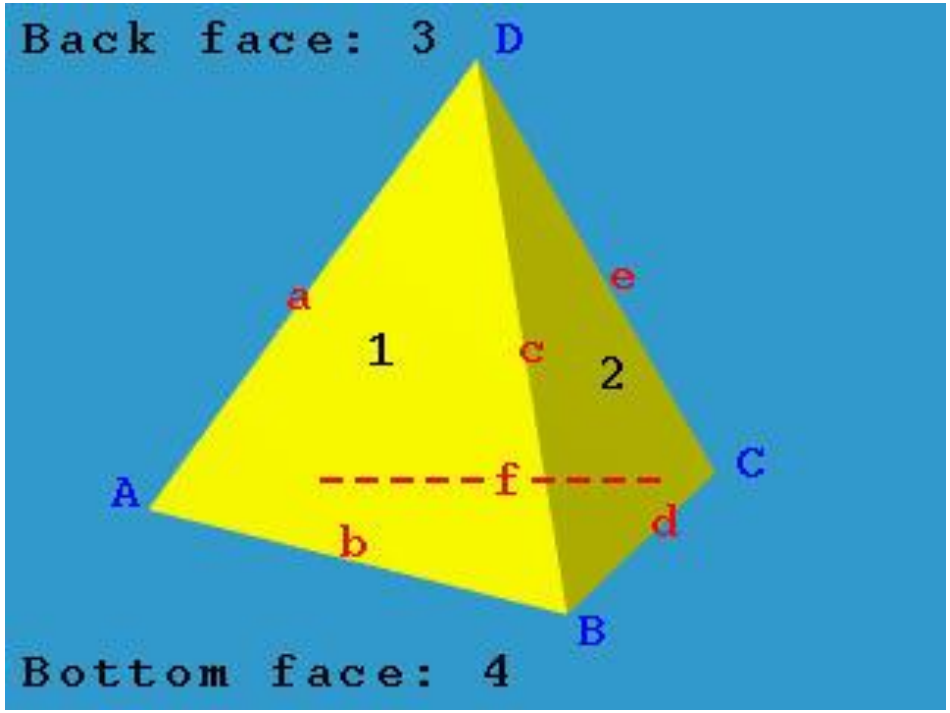
Ребро	Вершины		Грани		Левый обход		Правый обход	
Name	Begin	End	Left	Right	Pred	Succ	Pred	Succ
a	X	Y	1	2	b	d	e	c

Пример основной таблицы для пирамиды



Edge	Vertices		Faces		Left Traverse		Right Traverse	
Name	Start	End	Left	Right	Pred	Succ	Pred	Succ
a	A	D	3	1	e	f	b	c
b	A	B	1	4	c	a	f	d
c	B	D	1	2	a	b	d	e
d	B	C	2	4	e	c	b	f
e	C	D	2	3	c	d	f	a
f	A	C	4	3	d	b	a	e

Нужны еще две таблицы: для вершин и для ребер

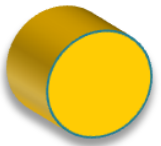


Vertex Name	Incident Edge
A	a
B	b
C	d
D	e

Face Name	Incident Edge
1	a
2	c
3	a
4	b

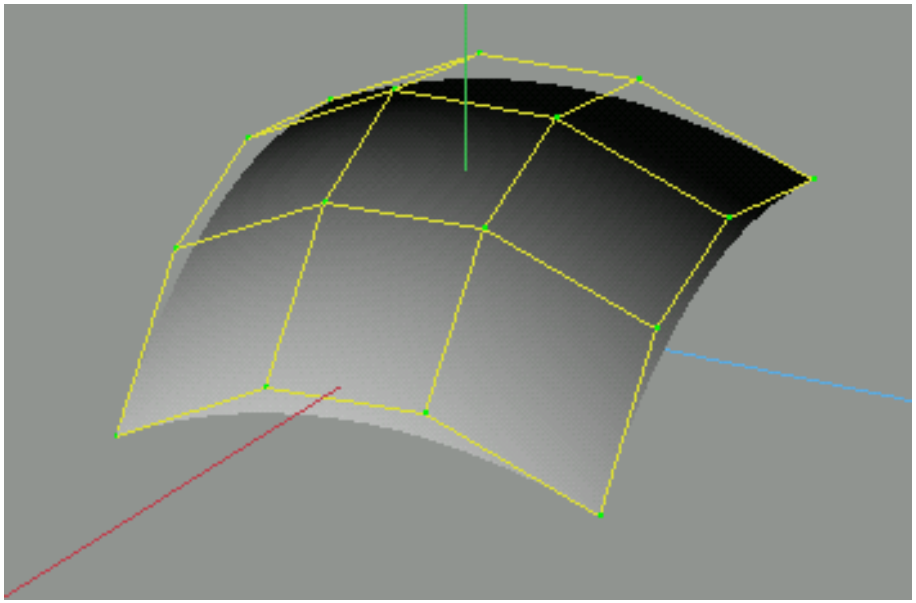
Граничное представление: типичные алгоритмы

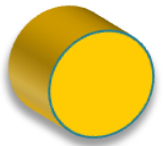
- Проверка правильности задания
- Вычисление габаритного объема
- Вычисление нормали в точке
- Вычисление кривизны поверхности
- Нахождение точки пересечения с лучом или кривой
- Определение положения точки относительно поверхности



Граничные представления высших порядков

Контрольные точки + способ интерполяции 2-го порядка и выше (полиномы Берштейна, например)





Итоги

- Задача синтеза изображений
- Графический процесс
- Геометрическое моделирование
- Модели трехмерных объектов
  - Воксельное (+октарное дерево)
  - Точечное представление
  - Конструктивная геометрия
  - Каркасное представление
  - Граничное представление (явное, индексированное по вершинам, ребрам, «крылатое»)
  - Граничные представление высших порядков