

Лекция 8.

Деревья разбиения пространства

Даниил Михайлович Берлизов

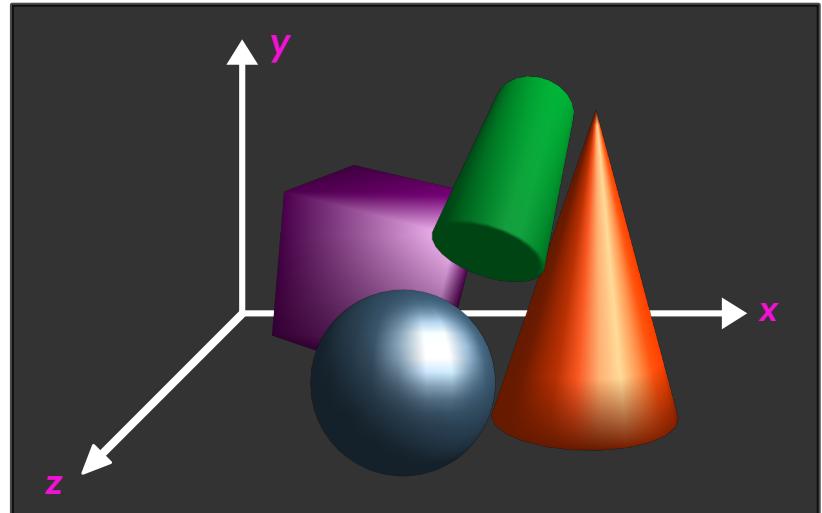
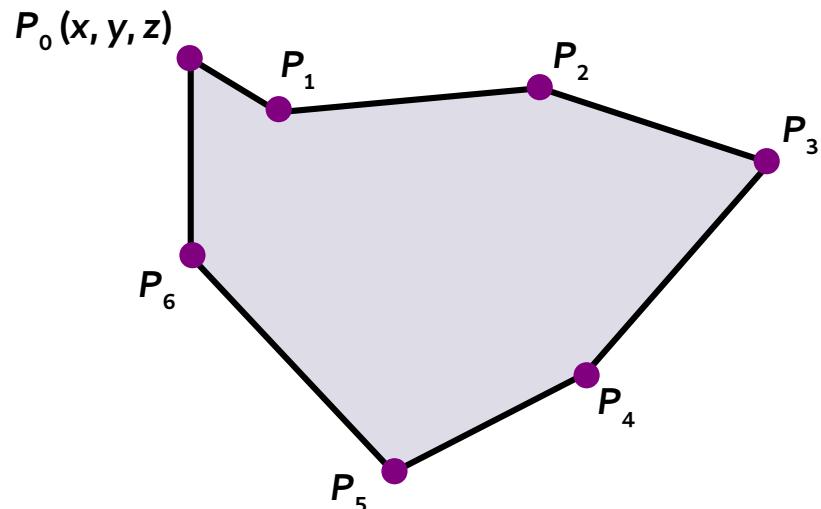
Старший преподаватель Кафедры вычислительных систем СибГУТИ

E-mail: sillyhat34@gmail.com

Курс «Структуры и алгоритмы обработки данных»
Осенний семестр, 2021 г.

Визуализация трёхмерных сцен

- Имеется трёхмерное пространство, в котором размещено n полигонов (многоугольников) и известны их координаты
- Полигоном (*polygon*) называется замкнутая ломаная линия, т. е. многоугольник
- Сложные поверхности представляются множеством полигонов



Визуализация трёхмерных сцен

- Имеется трёхмерное пространство, в котором размещено *n* полигонов (многоугольников) и известны их координаты
- Требуется решать следующие задачи:
 - Сортировать объекты (полигоны) в порядке удаления от наблюдателя (камеры)
 - Обнаруживать столкновения объектов



Изображение: INTERNET CLUB's Bandcamp page [<https://internetclub.bandcamp.com/>]

Визуализация трёхмерных сцен

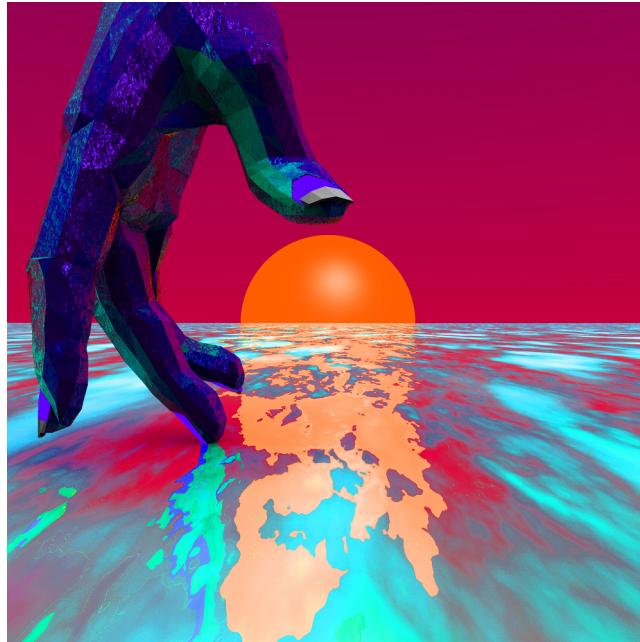
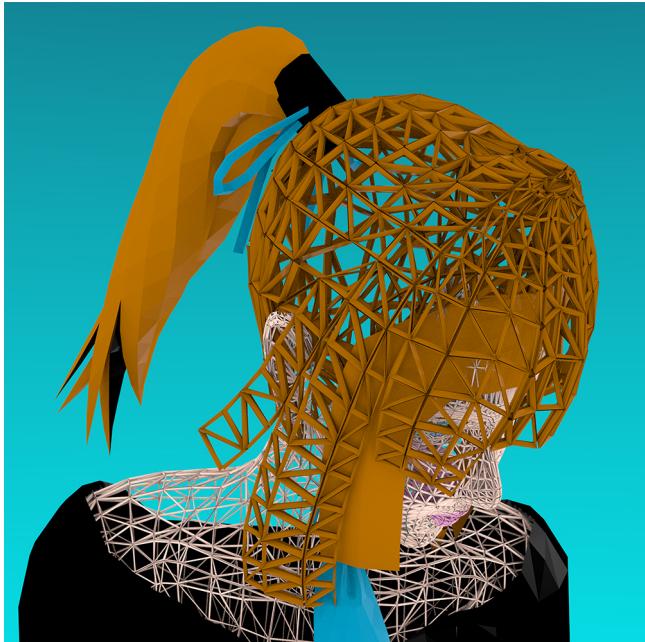
- Невидимые поверхности (полигоны) не отрисовываются
- Видимость определяется порядком расположения полигонов (от камеры)



Изображение: Vektroid's Bandcamp page [<https://vektroid.bandcamp.com/>]

Визуализация трёхмерных сцен

- Объекты нарисованы с учётом расположения относительно наблюдателя (камеры)



Изображения: death's dynamic shroud.wmv's Bandcamp page [<https://deathsdynamicshroud.bandcamp.com/>]

Деревья двоичного разбиения пространства

- **Дерево двоичного разбиения пространства** (*binary space partitioning tree, BSP tree*) — это бинарное дерево, разделяющее пространство на подмножества
- BSP-дерево хранит информацию о расположении объектов сцены в упорядоченном виде — от переднего плана (front) к заднему (back)
- Также применяется для определения столкновений объектов (*collision detection*) в компьютерных играх и робототехнике
- Использовались в играх Doom (1993), Quake (1996) и др.
- Авторы:
 - Schumacker R. A., Brand B., Gilliland M. G., Sharp W. H., **1969¹**
 - Fuchs H., Kedem Z. M., Naylor B. F., **1980²**

¹ Schumacker R. A. et al. **Study for applying computer-generated images to visual simulation.** – GENERAL ELECTRIC CO DAYTONA BEACH FL APOLLO AND GROUND SYSTEMS, 1969.

² Fuchs H., Kedem Z. M., Naylor B. F. **On visible surface generation by a priori tree structures** // ACM Siggraph Computer Graphics. – ACM, 1980. – T. 14. – №. 3. – С. 124-133.

Построение BSP-дерева

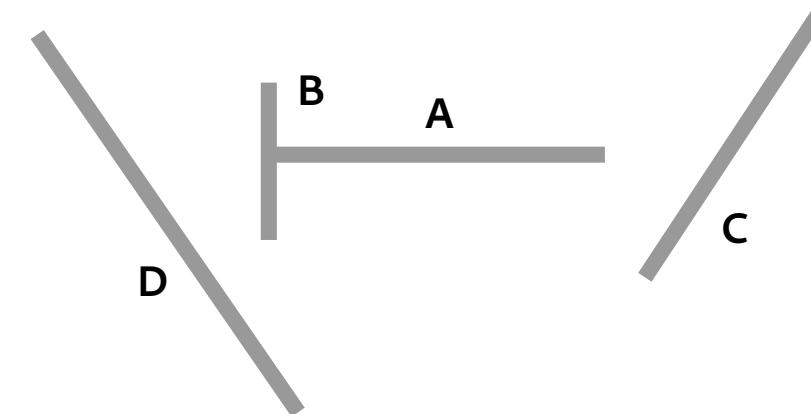
- Из заданного списка P полигонов выбираем **разбивающий полигон S** , задающий разбивающую плоскость (*plane*)
- Создаём в дереве новый узел *node*
- Для каждого полигона p в списке полигонов:
 - Если полигон p **находится с фронтальной стороны S** , заносим его в список полигонов фронтальной стороны (*front list*)
 - Если полигон p **находится с обратной стороны S** , заносим его в список полигонов обратной стороны (*back list*)
 - Если полигон p **пересекается** плоскостью S , разделяем его на два и заносим полученные полигоны в списки *front* и *back*
 - Если полигон p **лежит в плоскости S** , добавляем его в список полигонов узла *node*
- Рекурсивно применяем алгоритм к списку *front*, затем к *back*

Пример построения BSP-дерева в случае 2D-сцены

- В двумерном пространстве задан список прямых $\{A, B, C, D\}$ — список P
- Из заданного списка прямых требуется построить BSP-дерево

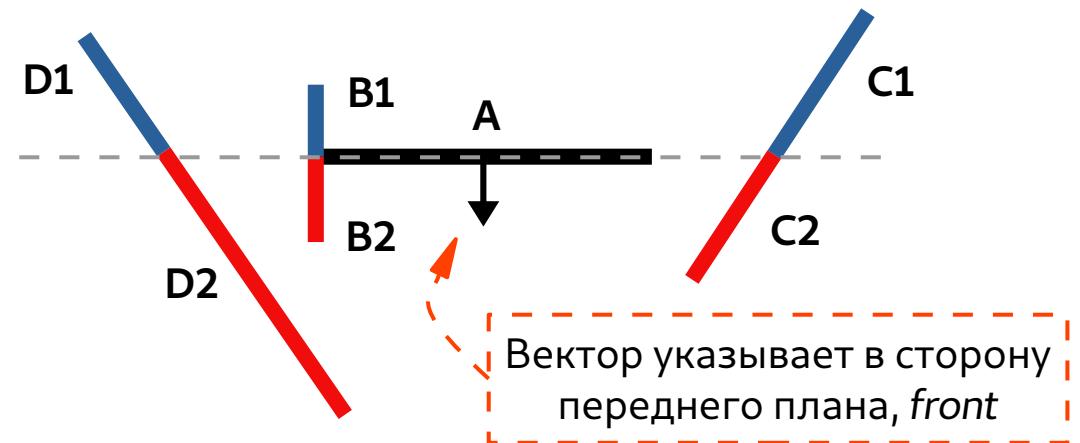
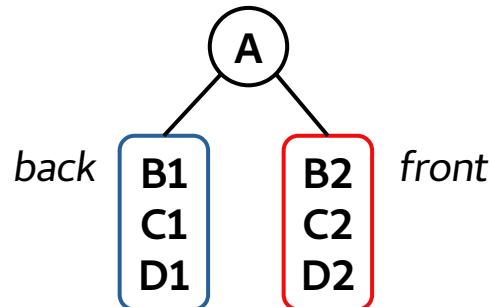
list P

| |
|---|
| A |
| B |
| C |
| D |



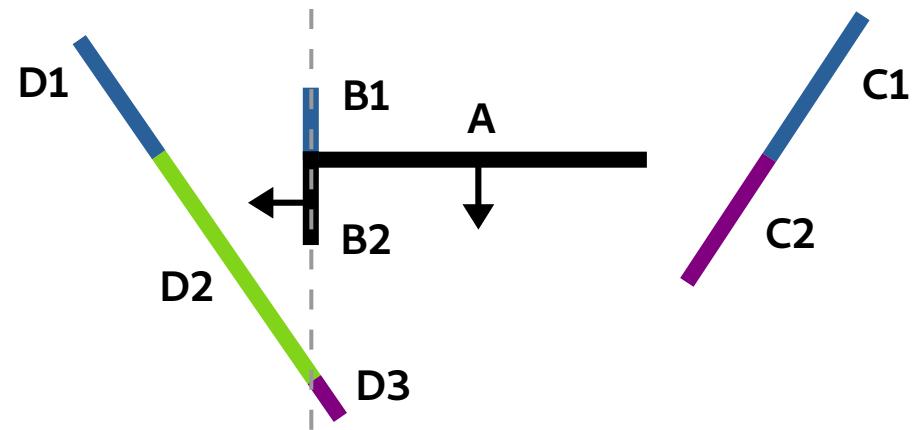
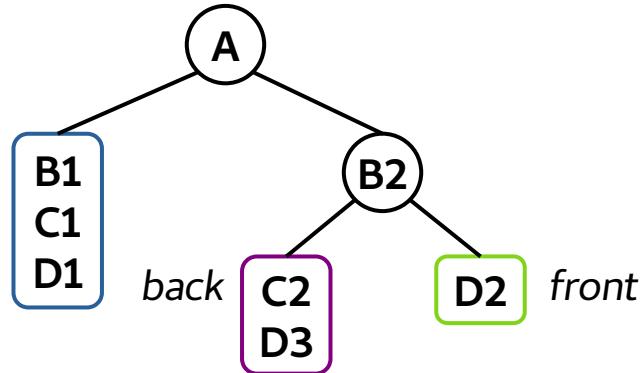
Пример построения BSP-дерева в случае 2D-сцены

- Из списка $\{A, B, C, D\}$ выбираем разделяющую прямую A
- Создаём новый узел A — корень BSP-дерева
- Формируем списки *front* и *back*
- Рекурсивно обрабатываем списки *front* и *back*



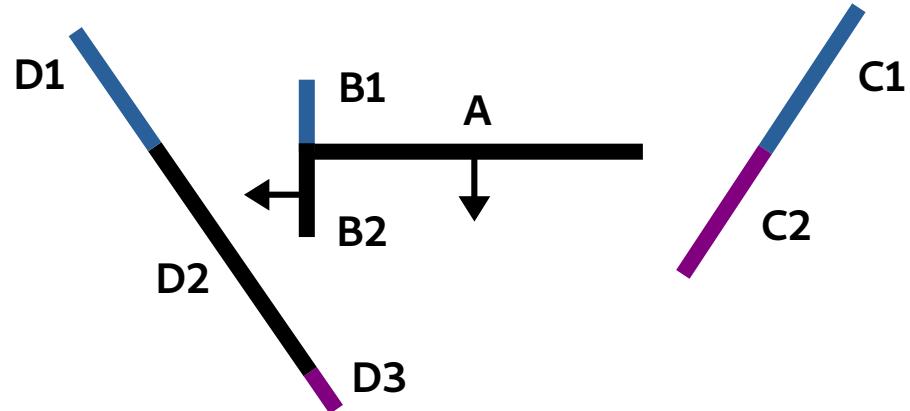
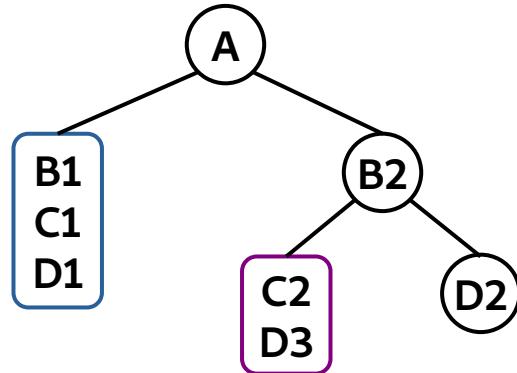
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *front* узла *A* выбираем разделяющую прямую B_2
- Создаём узел B_2 и формируем его списки *front* и *back*



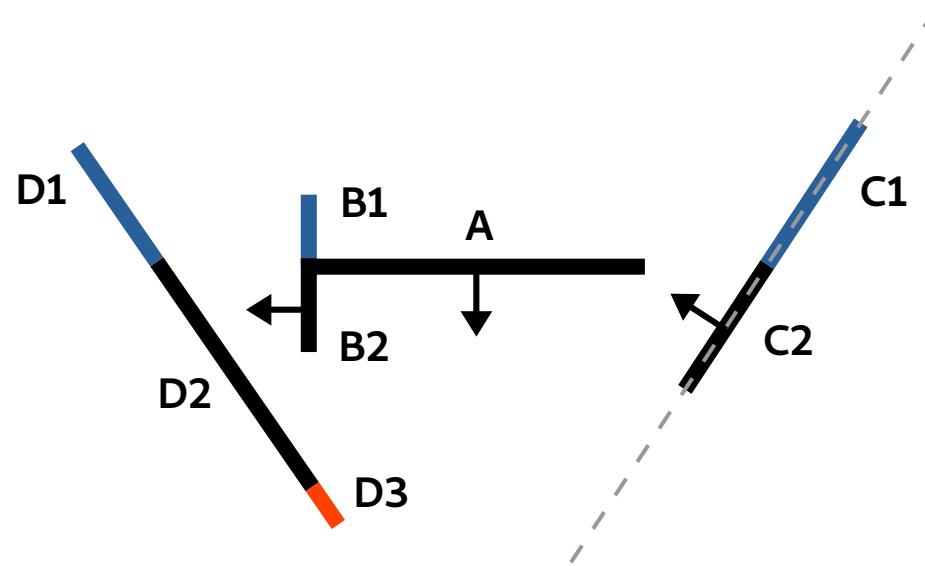
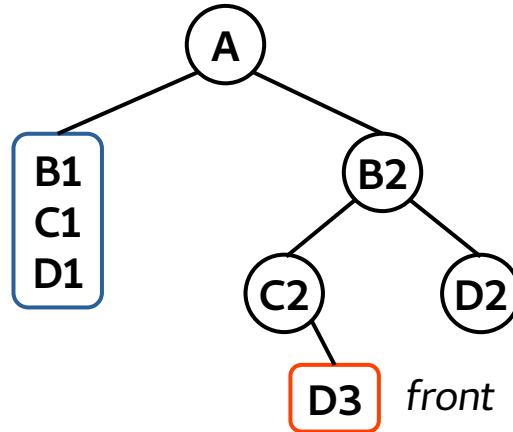
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *front* узла B_2 выбираем разделяющую прямую D_2
- Формируем узел D_2 (его списки *front* и *back* пусты)



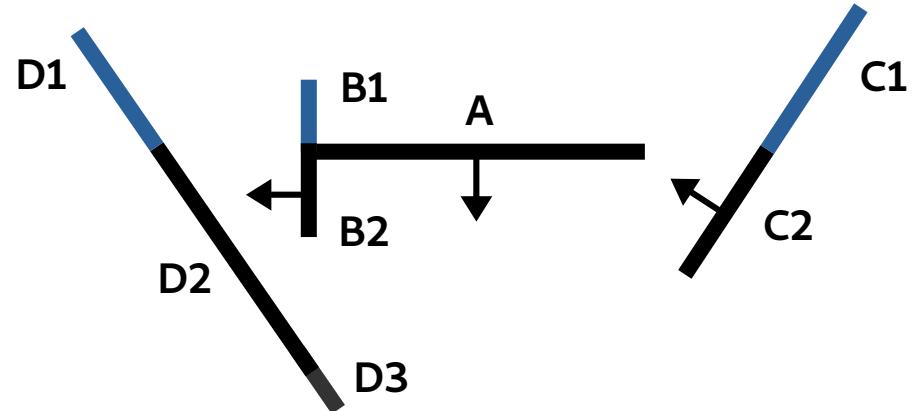
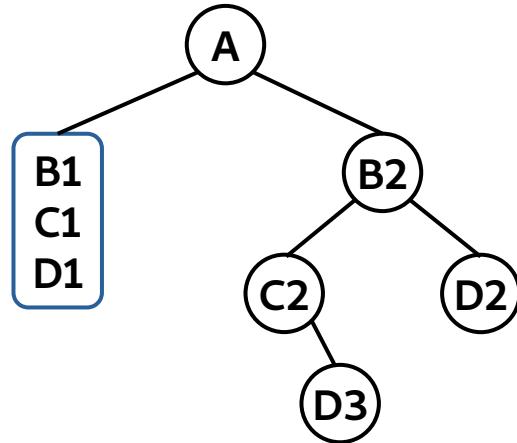
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *back* узла B_2 выбираем разделяющую прямую C_2
- Формируем узел C_2 и его список *front* (список *back* пуст)



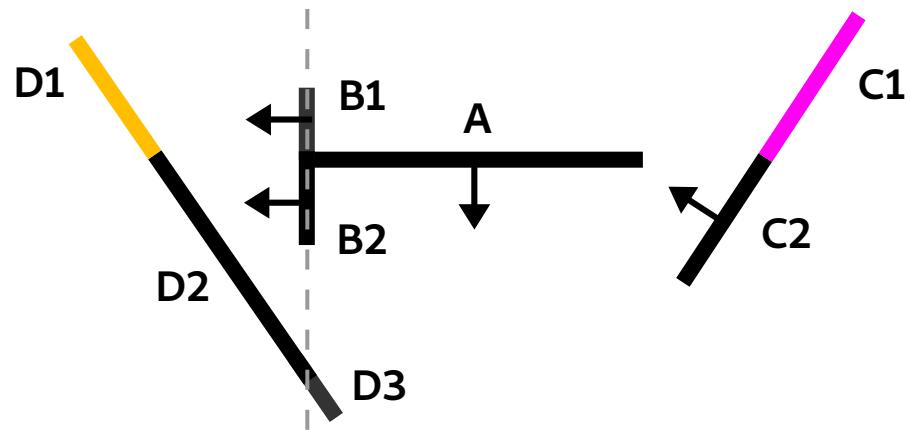
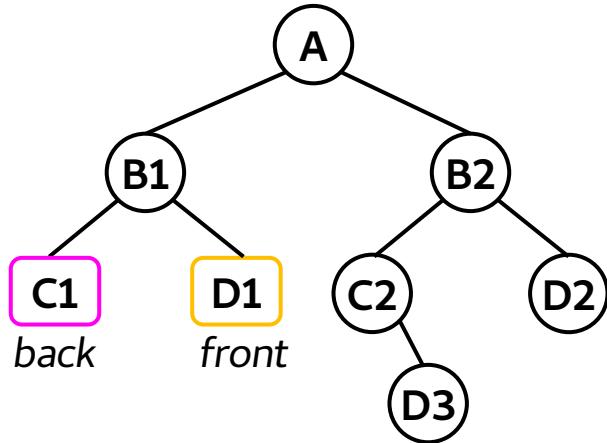
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *front* узла C_2 выбираем разделяющую прямую D_3
- Формируем узел D_3



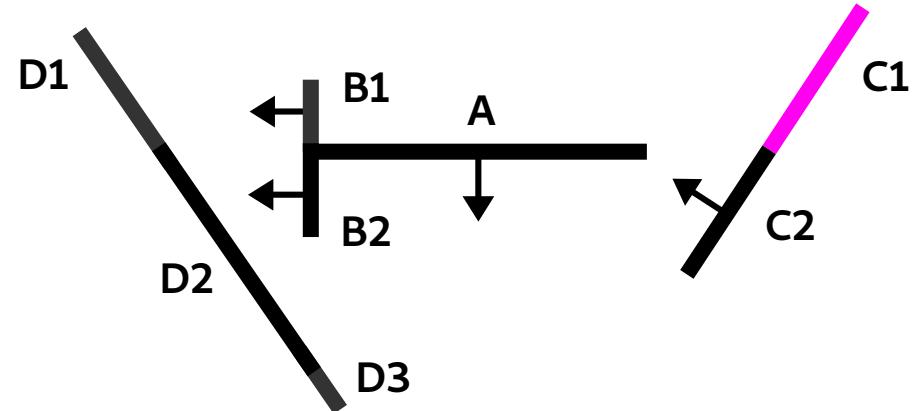
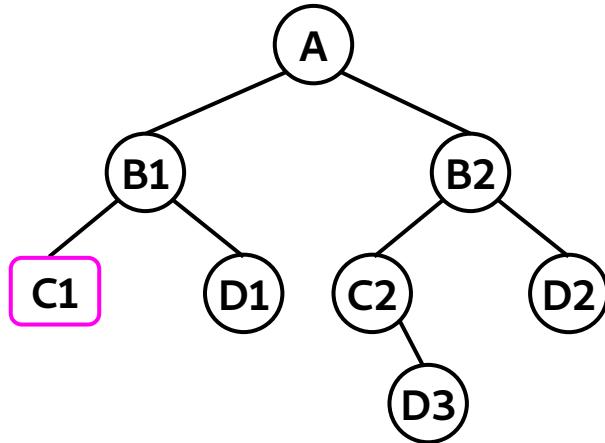
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *back* узла *A* выбираем разделяющую прямую B_1
- Формируем узел B_1 и его списки *front* и *back*



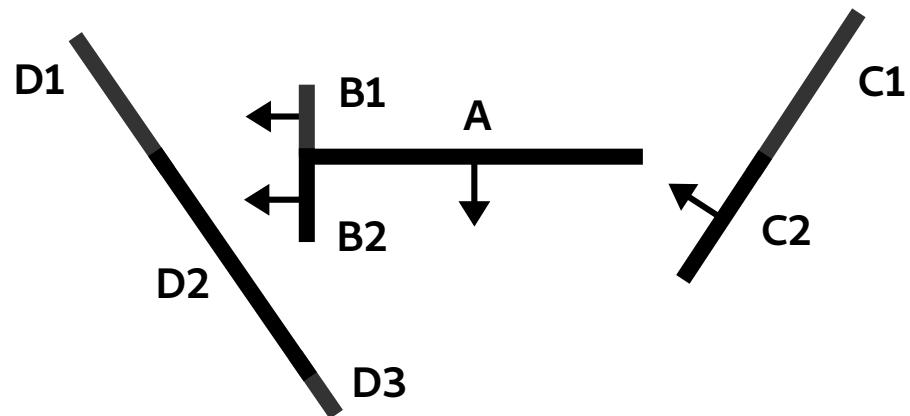
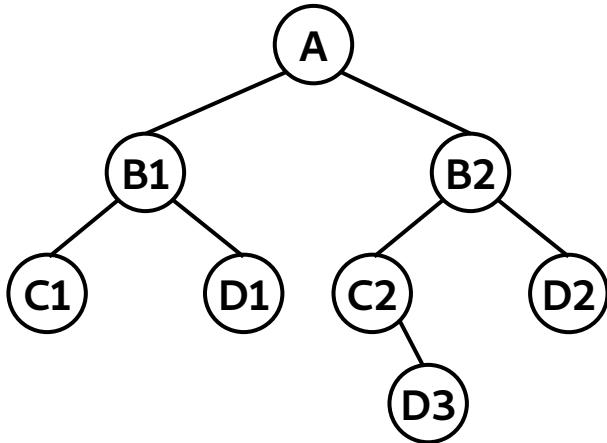
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *front* узла B_1 выбираем разделяющую прямую D_1
- Формируем узел D_1



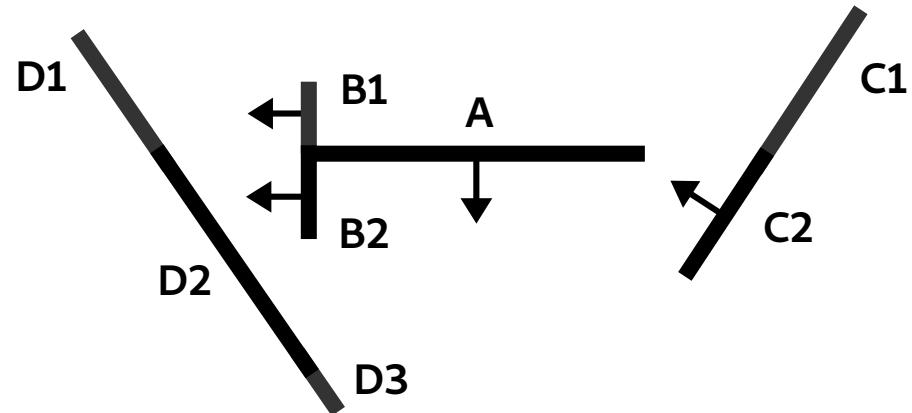
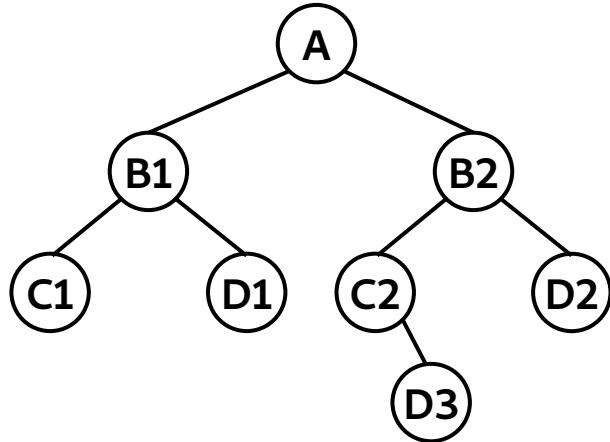
Пример построения BSP-дерева в случае 2D-сцены

- Из списка *back* узла B_1 выбираем разделяющую прямую C_1
- Формируем узел C_1



Пример построения BSP-дерева в случае 2D-сцены

- Количество полигонов в дереве больше их начального числа
- На плоскости были 4 прямые, в дереве — 8



Сортировка полигонов

- Задано BSP-дерево
- Заданы координаты наблюдателя — POV (point of view)
- **Как изобразить на экране полигоны в порядке удаления от наблюдателя (упорядочить их)?**

Обход BSP-дерева (сортировка полигонов)

- **Если текущий узел — лист**, нарисовать его полигоны
- **Если наблюдатель (камера) расположен перед текущим узлом:**
 1. Рекурсивно обойти поддерево с узлами, находящимися позади текущего узла
 2. Нарисовать полигоны текущего узла
 3. Рекурсивно обойти поддерево с узлами, находящимися впереди текущего узла
- **Если наблюдатель расположен позади текущего узла:**
 1. Рекурсивно обойти поддерево с узлами, находящимися впереди текущего узла
 2. Нарисовать полигоны текущего узла
 3. Рекурсивно обойти поддерево с узлами, находящимися позади текущего узла
- **Если наблюдатель расположен в плоскости текущего узла:**
 1. Рекурсивно обойти поддерево с узлами, находящимися впереди текущего узла
 2. Рекурсивно обойти поддерево с узлами, находящимися позади текущего узла

Поиск столкновений

- Задано BSP-дерево
- Заданы координаты объекта
- Границы объекта заданы ограничивающей сферой (или окружностью) для упрощения вычислений
- Требуется найти полигон, пересекающий ограничивающую сферу объекта — полигон, с которым объект столкнулся

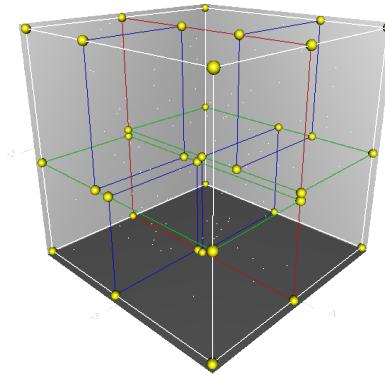


Поиск столкновений

```
function FindCollision(node, object)
    if node = NULL then
        return NULL
    if Dist(node.polygon, object.sphere.center) > object.sphere.radius then
        if DotProduct(object.center, node.normal) >= 0 then
            /* Объект находится с фронтальной стороны плоскости,
               обходим только фронтальное поддерево */
            return FindCollision(node.front, object)
        else
            /* Объект находится с обратной стороны разбивающей плоскости,
               обходим только обратное поддерево */
            return FindCollision(node.back, object)
        end if
    else
        /* Столкновение с полигоном узла node */
        return node
    end if
end function
```

k-мерное дерево (**k-d tree**)

- **k**-мерное дерево (**k-d tree**) — это бинарное дерево разбиения пространства для упорядочивания точек в **k**-мерном пространстве
- **k-d дерево** — это разновидность дерева поиска
- **Автор:** Дж. Бентли, **1975***

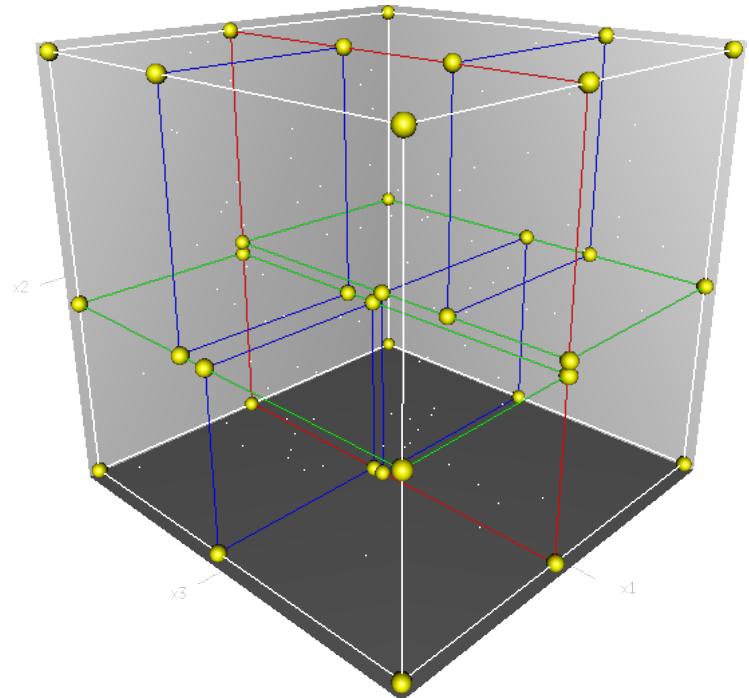


| Операция | Сложность в среднем случае | Сложность в худшем случае |
|--|----------------------------|---------------------------|
| Insert | $O(\log n)$ | $O(n)$ |
| Lookup | $O(\log n)$ | $O(n)$ |
| Delete | $O(\log n)$ | $O(n)$ |
| Объём требуемой памяти: $O(n)$ | | |

* Bentley J. L. Multidimensional binary search trees used for associative searching // Communications of the ACM. – 1975. – Т. 18. – №. 9. – С. 509-517.

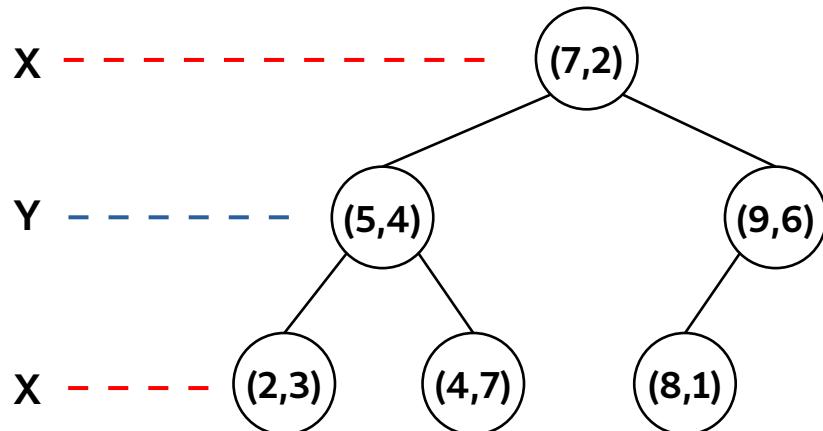
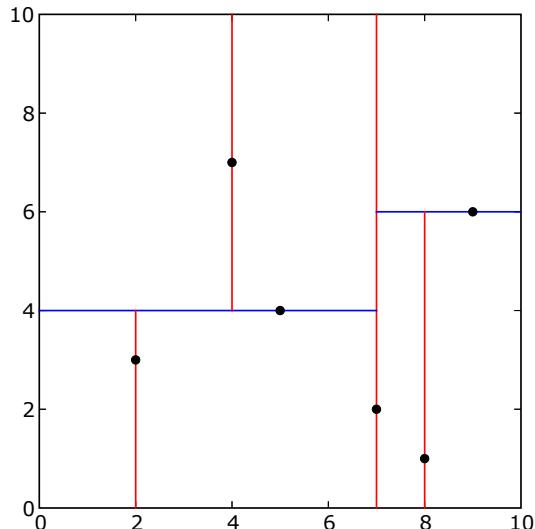
k-мерное дерево (*k*-d tree)

- ***k*-мерное дерево (*k*-d tree)** — это бинарное дерево разбиения пространства для упорядочивания точек в *k*-мерном пространстве
- Пространство разбивается гиперплоскостью на два подпространства
- *k*-d дерево строится для заданного множества точек



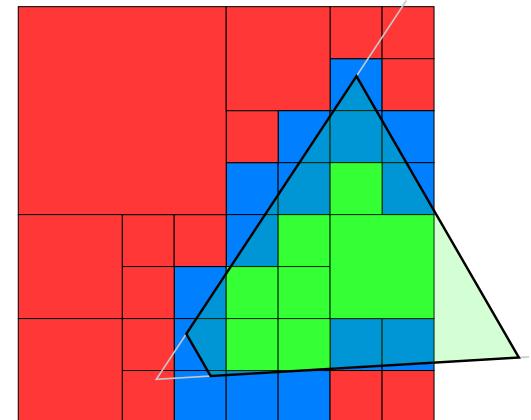
k-мерное дерево (*k*-d tree)

- Заданы точки: (2, 3), (5, 4), (9, 6), (4, 7), (8, 1), (7, 2)
- Построение *k*-d дерева производится путём рекурсивного разбиения плоскости поочерёдно прямыми $x = k$, $y = k$
- За k принимается медиана среди координат X и Y точек



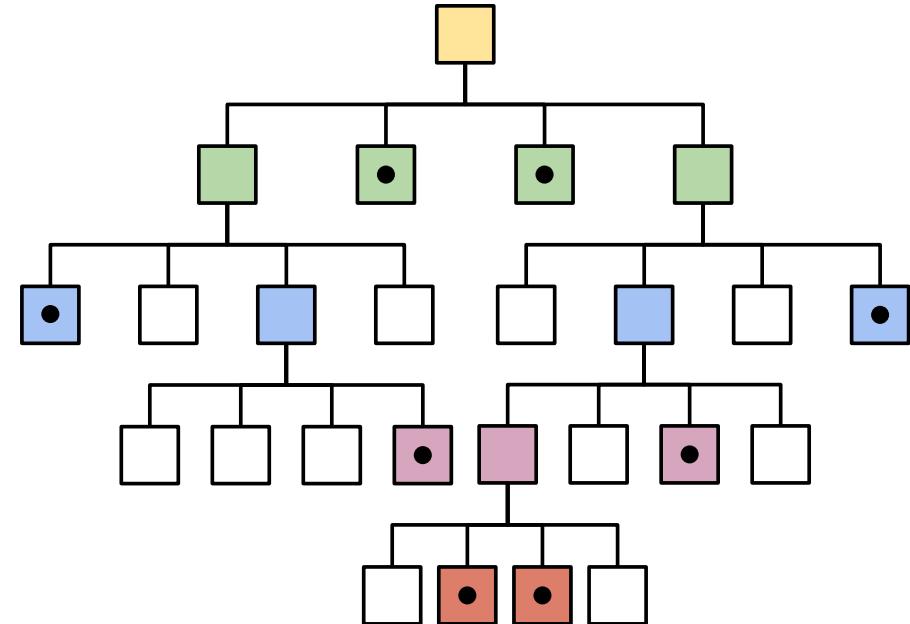
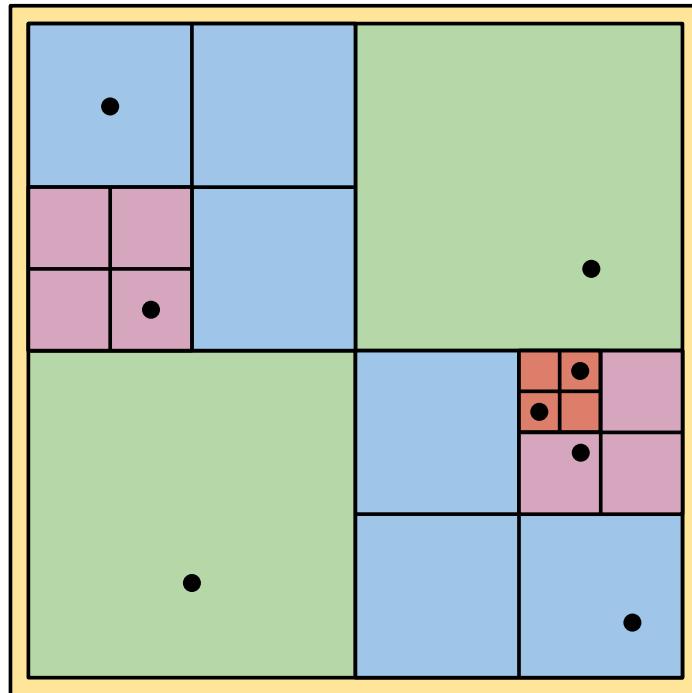
Дерево квадрантов (quadtree)

- **Дерево квадрантов (quadtree)** — это дерево, в котором каждый внутренний узел содержит 4 дочерних элемента
- Дерево квадрантов применяется для рекурсивного разбиения двумерного пространства по четыре области (квадранта)
- **Авторы:** Raphael A. Finkel, Jon L. Bentley, **1974***
- **Задачи:**
 - Представление изображений
 - Обнаружение коллизий в двумерном пространстве
 - Хранение данных для табличных или матричных вычислений
 - Пространственная индексация (например, в СУБД)



* Finkel R. A., Bentley J. L. **Quad trees a data structure for retrieval on composite keys** // Acta informatica. – 1974. – Т. 4. – №. 1. – С. 1-9.

Дерево квадрантов (quadtree)

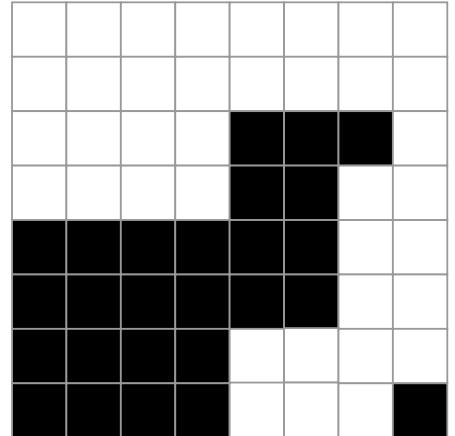


Дерево квадрантов (quadtree)

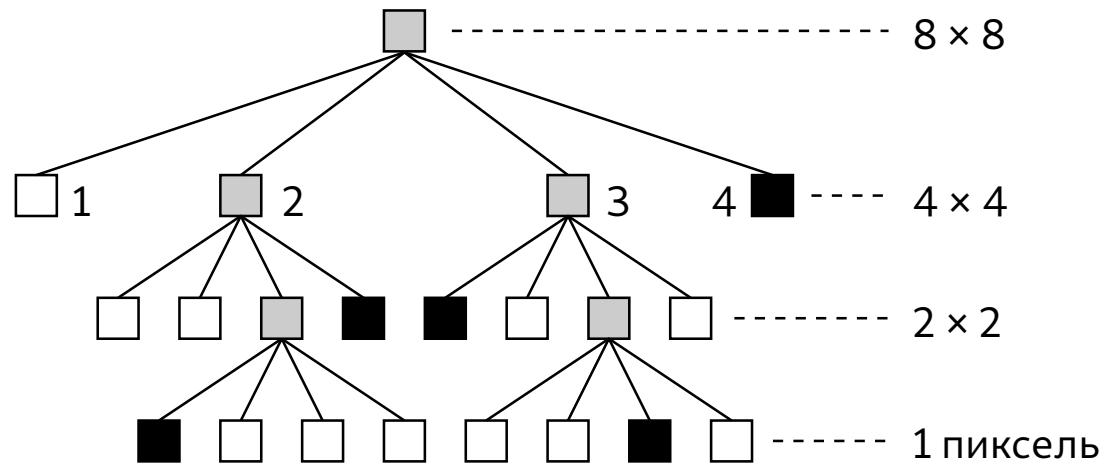
- Деревья квадрантов классифицируются в соответствии с представляемым ими типом данных:
 - Точки (*point quadtree*)
 - Прямые или кривые линии (*edge quadtree*)
 - Полигоны (*polygonal map quadtree*)
 - Двумерное пространство (*region quadtree*)
- Дерево квадрантов, разбивающее двумерное пространство, является вариацией **префиксного дерева (trie)**

Дерево квадрантов: представление изображений

- Деревом квадрантов разбиения двумерного пространства с глубиной p может быть представлено изображение, состоящее из $2^n \times 2^n$ пикселей, принимающих значение 0 или 1:
 - Корнем дерева представлено всё пространство изображения
 - Если значения пикселей в одном узле различаются, узел разбивается на квадранты
 - Листовой узел такого дерева соответствует множеству пикселей: либо нулевых, либо единичных

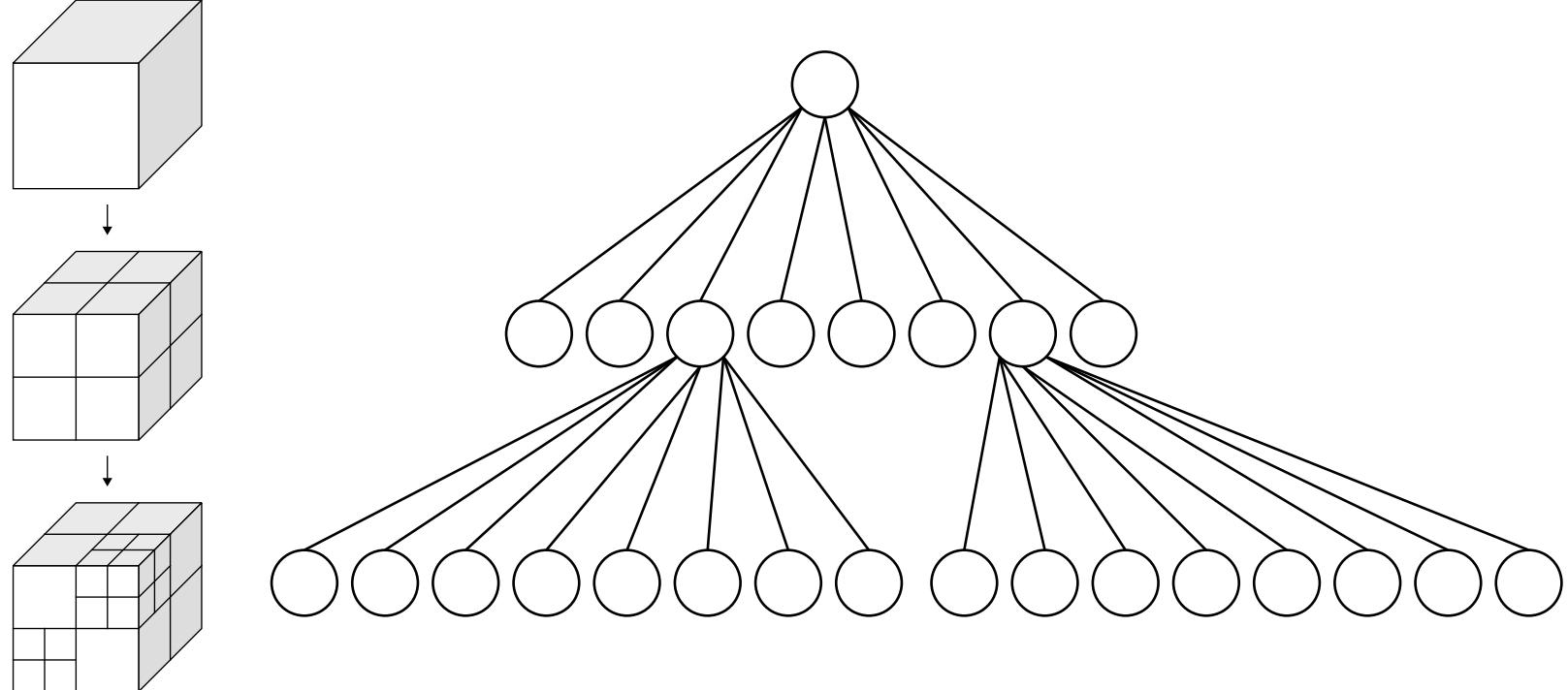


Дерево квадрантов: представление изображений



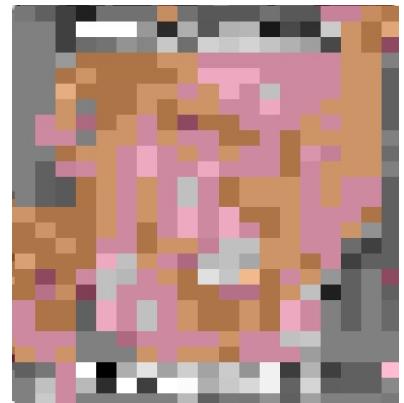
Октодерево (octree)

- Аналогом дерева квадрантов для разбиения трёхмерных пространств является **октодерево** (octree)



Дальнейшее чтение

- Изучить алгоритм сжатия изображений (*image compression*) на основе дерева квадрантов
- Оценить вычислительную сложность операции сжатия изображения при помощи дерева квадрантов и объём требуемой памяти
- Ознакомиться с **деревьями отрезков** (*interval tree*) [[CLRS 3ed.](#), раздел 14.3]
- Найти и просмотреть материалы о принципах устройства R-деревьев* (*R-tree*)



* Guttman A. **R-trees: A dynamic index structure for spatial searching.** – ACM, 1984. – Т. 14. – №. 2. – С. 47-57.

ご清聴ありがとうございました!

Даниил Михайлович Берлизов

Старший преподаватель Кафедры вычислительных систем СибГУТИ

E-mail: sillyhat34@gmail.com

Курс «Структуры и алгоритмы обработки данных»
Осенний семестр, 2021 г.