Иерархическая дискретная трассировка лучей в октантных деревьях

Михаил Цыганков

Softgraph Intl. GmbH

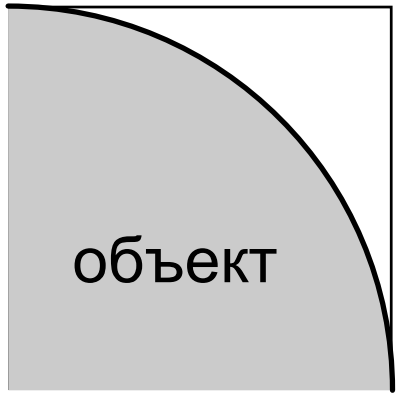
mailto:Mike@Softgraph.com

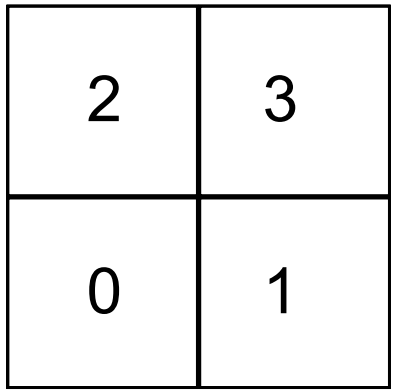
International Conference Graphicon 1998, Moscow, Russia, http://www.graphicon.ru/

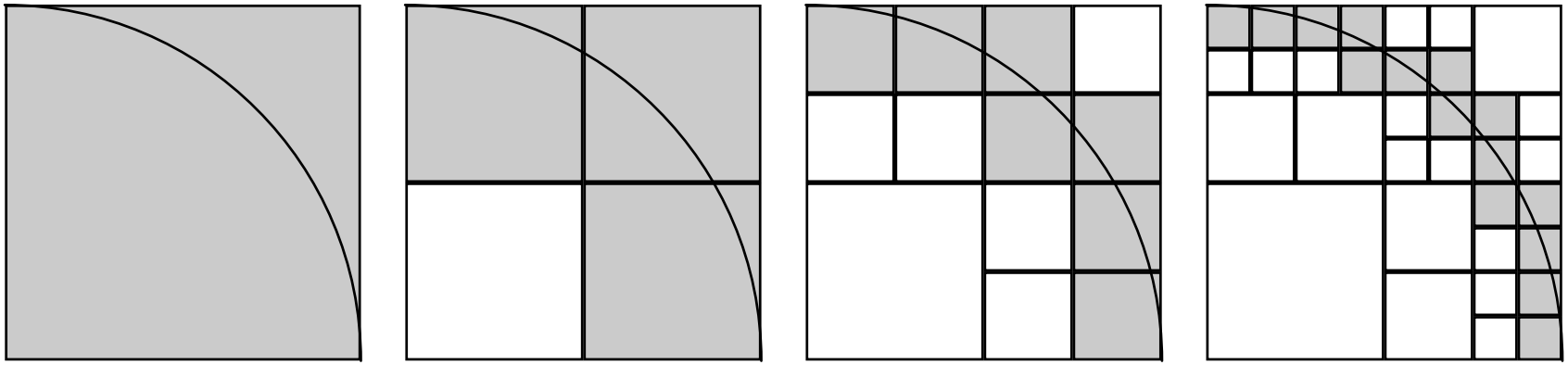
Классификация алгоритма

- front-to-back
- image order
- volume visualization

Иерархическое разбиение пространства

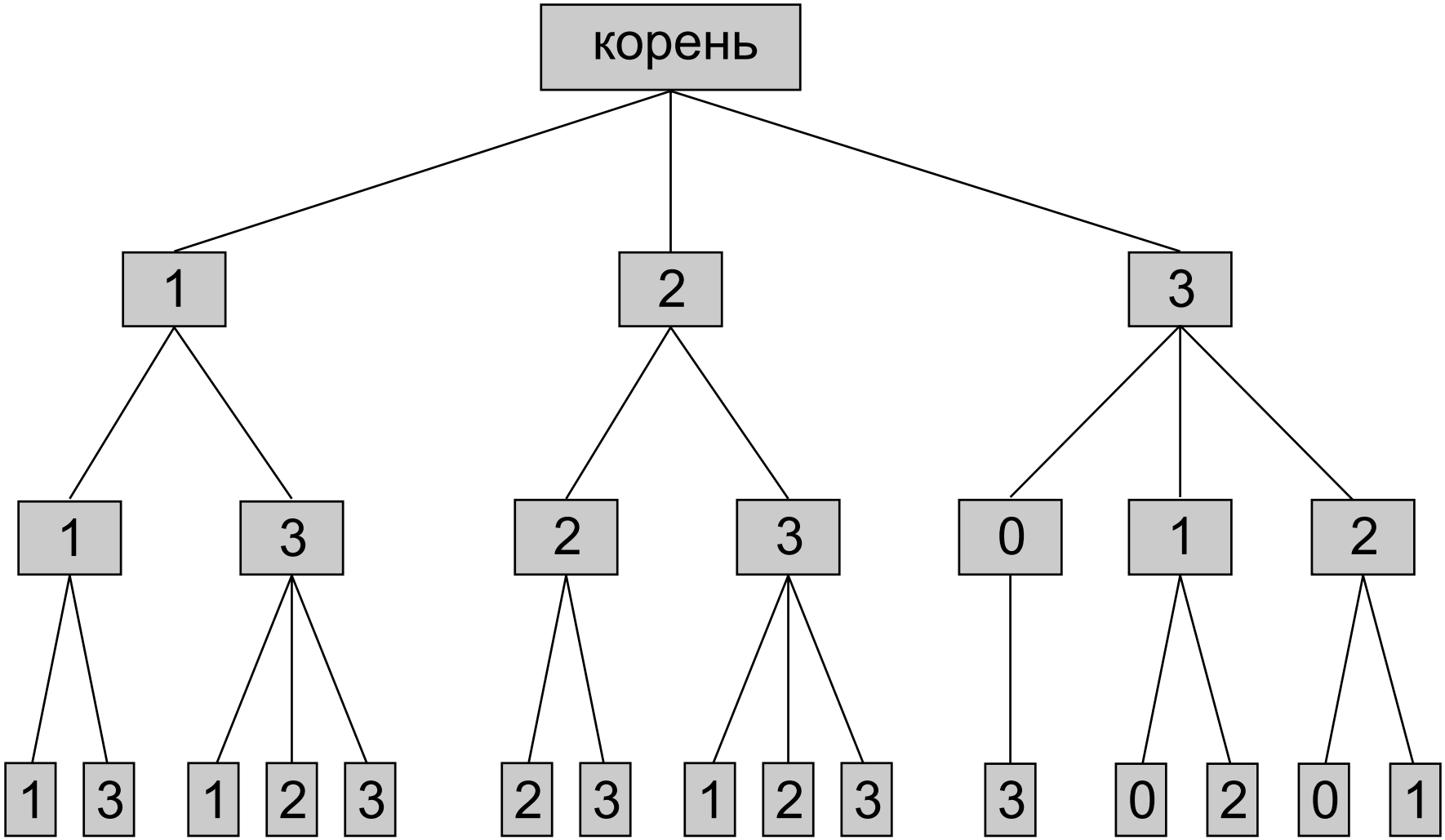




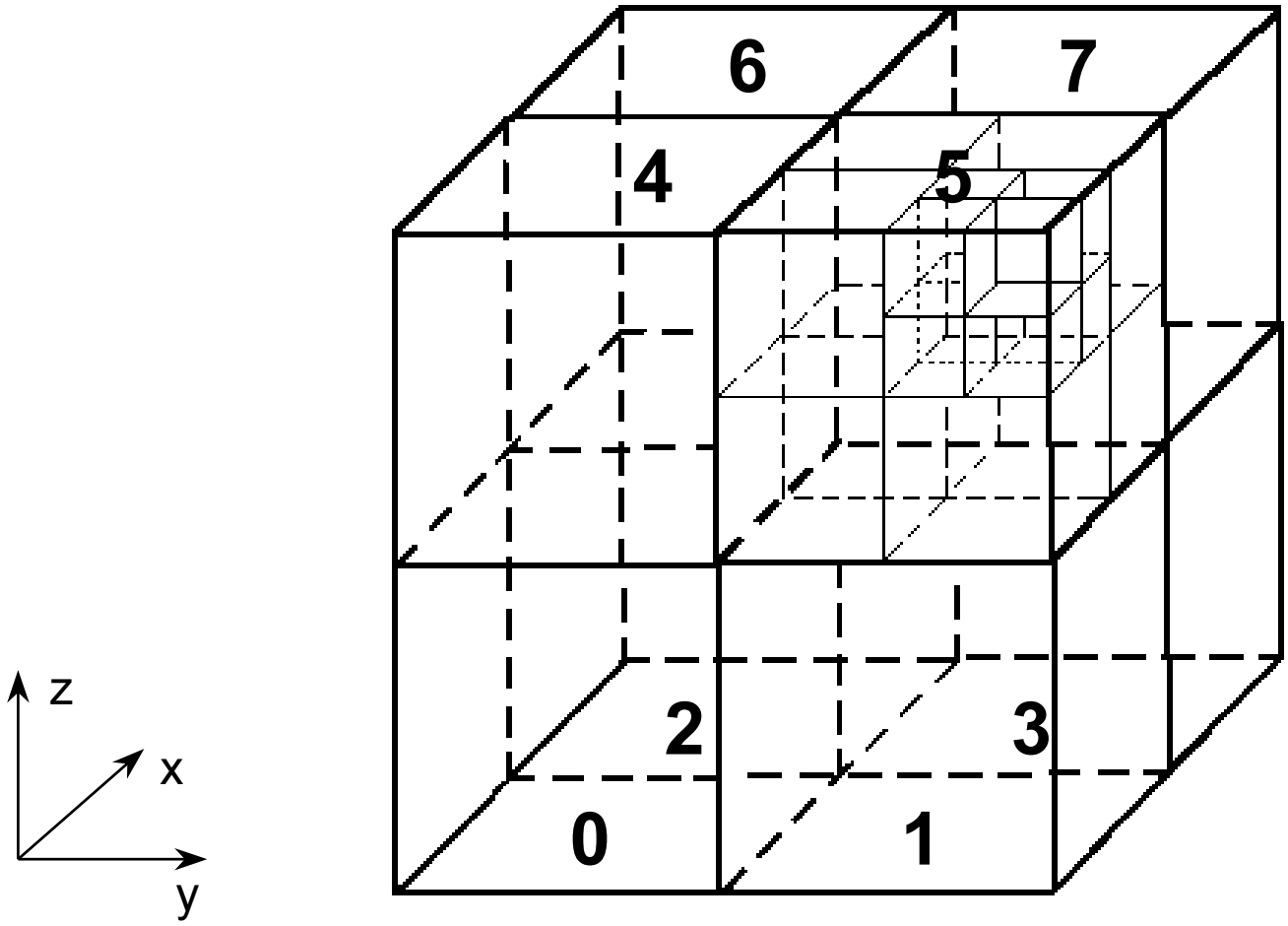


глубина 0 глубина 1 глубина 2 глубина 3

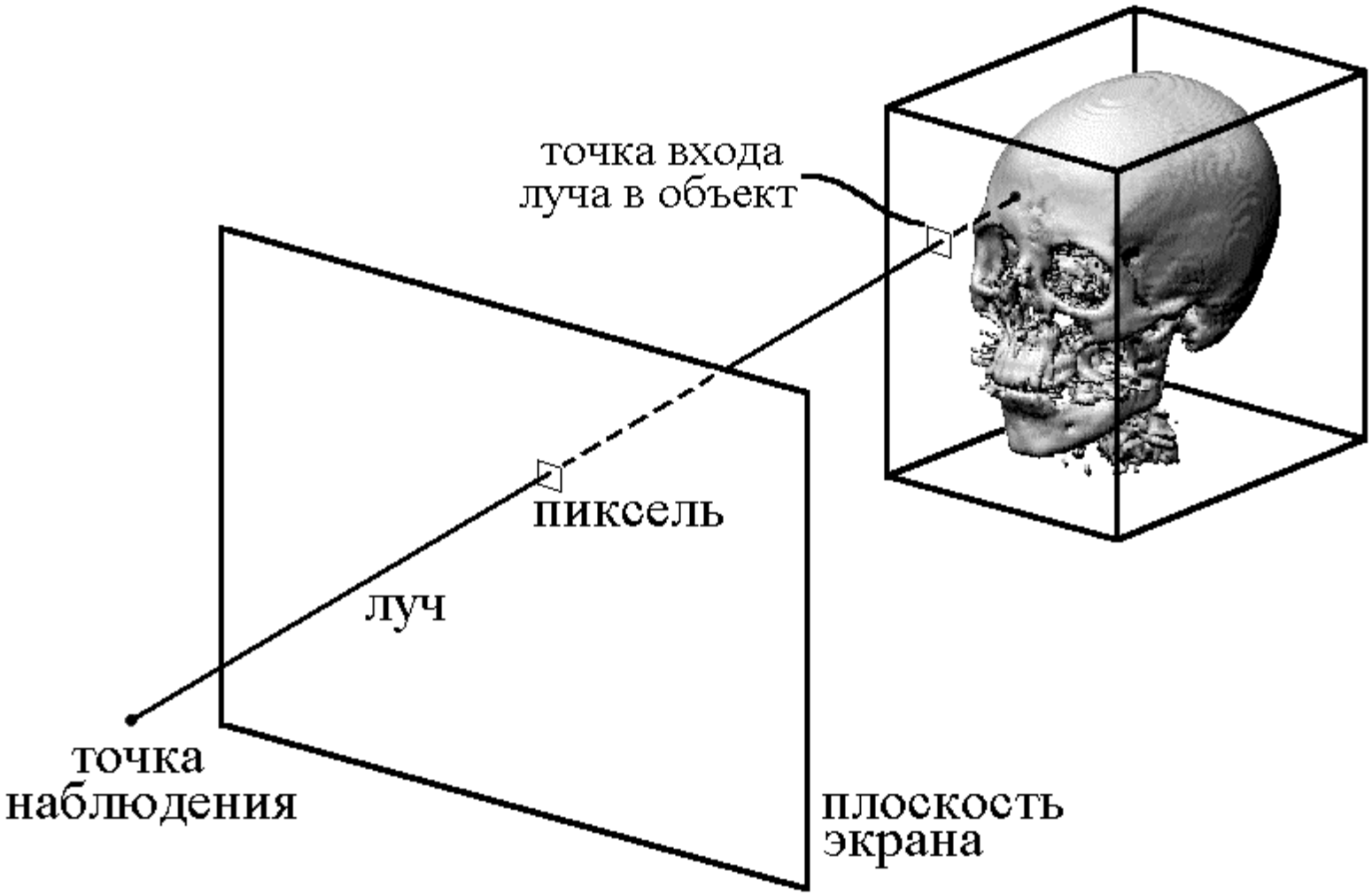
Иерархическое разбиение пространства



Октантные деревья (octree)



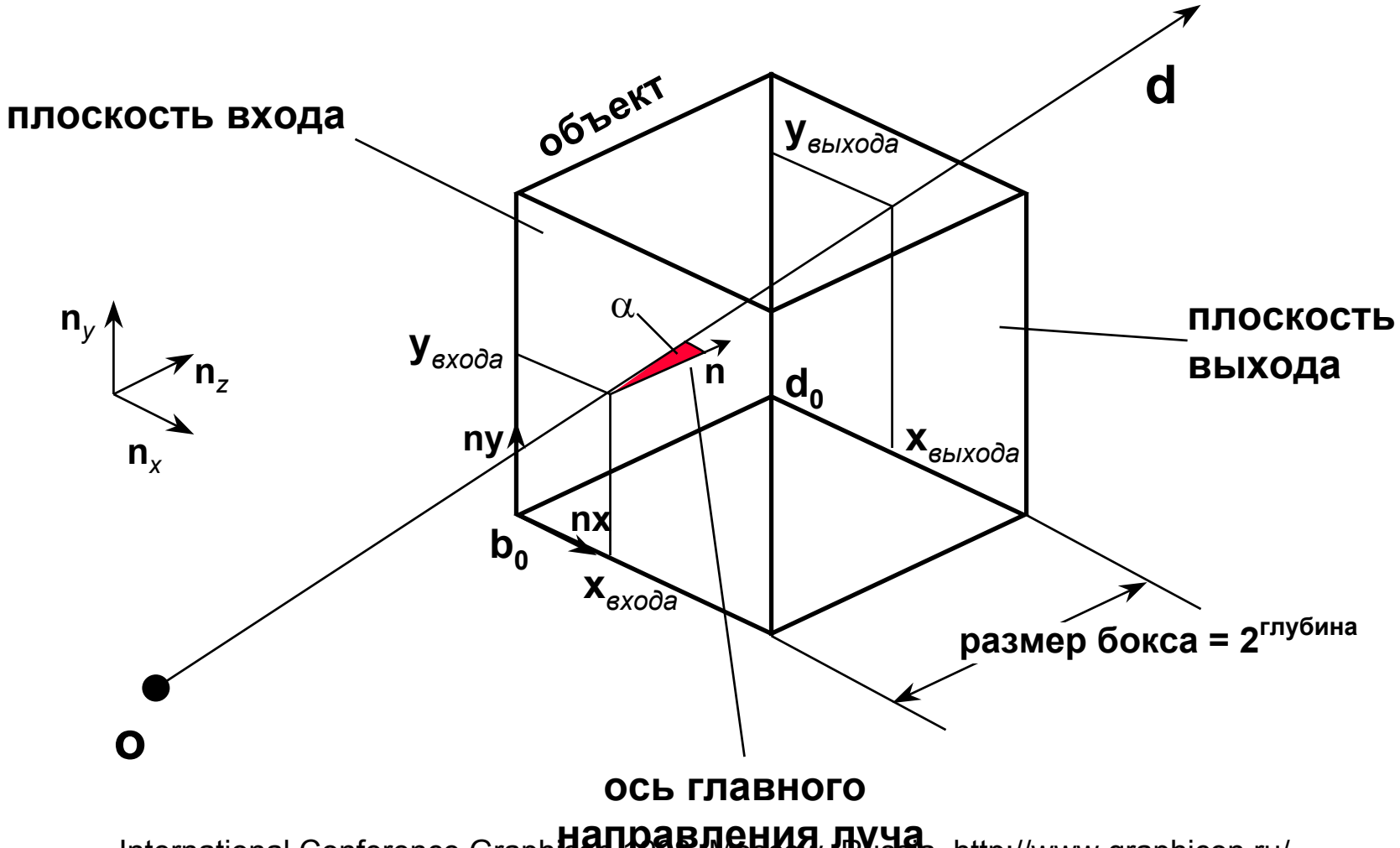
Общая схема


точка входа
луча в объект
пиксель
луч
точка
наблюдения
плоскость
экрана

Трассировка луча

- вычислить главное направление луча внутри объекта
- вычислить точки пересечения луча и оболочки объекта (точки входа и выхода луча из бокса)
- найти воксель, в котором луч пересекает объект

Координатные вычисления



Главное направление луча

$$|\mathbf{n}_{x+}| = |\mathbf{n}_{z+}| = |\mathbf{n}_{z+}| = |\mathbf{d}| = 1$$

$$\mathbf{n}_{x-} = -\mathbf{n}_{x+} \;\; ; \;\; \mathbf{n}_{y-} = -\mathbf{n}_{y+} \;\; ; \;\; \mathbf{n}_{z-} = -\mathbf{n}_{z+}$$

$$\cos \alpha_{x+} = \mathbf{n}_{x+} \cdot \mathbf{d} \;\; ; \;\; \cos \alpha_{x-} = \mathbf{n}_{x-} \cdot \mathbf{d}$$
$$\cos \alpha_{y+} = \mathbf{n}_{y+} \cdot \mathbf{d} \;\; ; \;\; \cos \alpha_{y-} = \mathbf{n}_{y-} \cdot \mathbf{d}$$
$$\cos \alpha_{z+} = \mathbf{n}_{z+} \cdot \mathbf{d} \;\; ; \;\; \cos \alpha_{z-} = \mathbf{n}_{z-} \cdot \mathbf{d}$$

$$\text{ОГНЛ} = \mathit{argmax}\,(\mathbf{n} \cdot \mathbf{d})$$

$$\mathbf{n} \in \{\mathbf{n}_{x\pm}, \mathbf{n}_{y\pm}, \mathbf{n}_{z\pm}\}$$

Точки входа и выхода луча из бокса

$p = o + t \cdot \mathbf{d}$ (произвольная точка на луче)

$$\mathbf{n} \cdot p = \mathbf{n} \cdot b_0$$

$$\mathbf{n} \cdot p_{входа} = \mathbf{n} \cdot (o + t_{входа} \cdot \mathbf{d}) = \mathbf{n} \cdot b_0$$

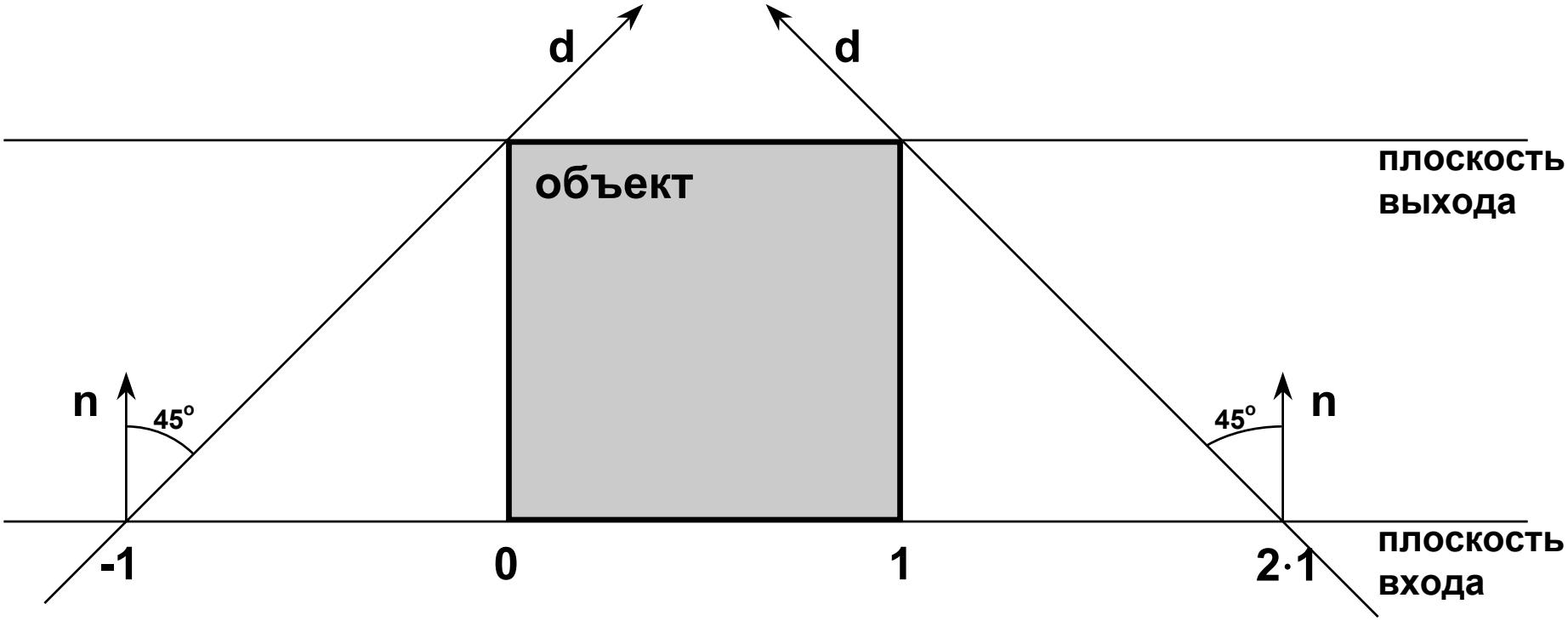
$$t_{входа} = (\mathbf{n} \cdot b_0 - \mathbf{n} \cdot o) / (\mathbf{n} \cdot \mathbf{d})$$

$$p_{входа} = o + t_{входа} \cdot \mathbf{d}$$

$$x_{входа} = \mathbf{n}_x \cdot p_{входа} - \mathbf{n}_x \cdot b_0$$

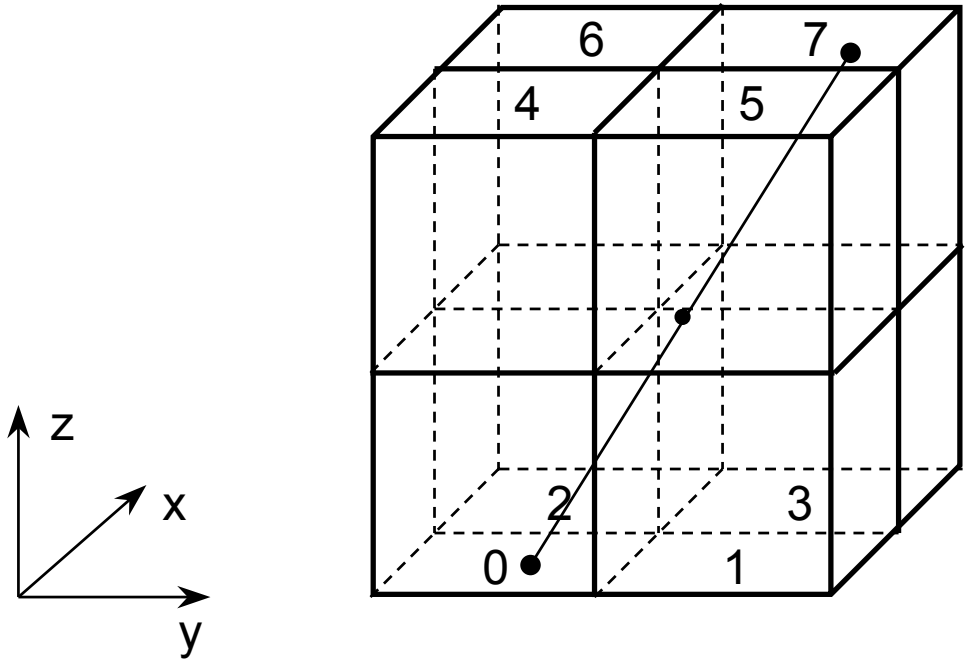
$$y_{входа} = \mathbf{n}_y \cdot p_{входа} - \mathbf{n}_y \cdot b_0$$

Диапазоны значений



$x_{входа}, y_{входа} \in$ [-размер_бокса, 2 · размер_бокса]

Порядок прохождения вокселей



Ориентация: **ОГНЛ = Z; X+; Y+; Z+;**

Порядок вокселей: **0 → 1 → 2 → 3 → 4 → 5 → 6 → 7**

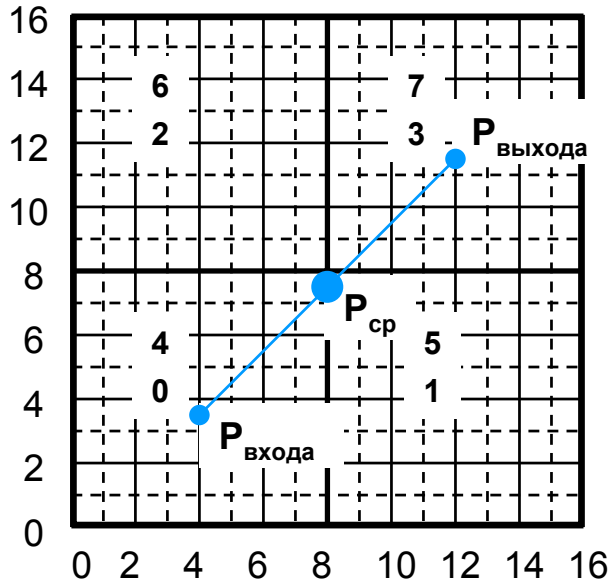
Поиск по дереву

```
bool trace_ray(octree_node, p_enter, p_exit)
{
   ray_path = get_ray_path(p_enter, p_exit);
   voxel_path = octree_node.infobyte & ray_path;

   for(every subnode in voxel_path)
   {
      p_0, p_1 = calc_subnode(octree_node, p_enter, p_exit, subnode);
      if(trace_ray(subnode, p_0, p_1))
        return true;
   }

   return false;
}
```

Путь луча



D = 1 2 3 4

$P_{входа}$ y = 3 : 0 0 1 1

x = 5 : 0 1 0 1

$2^{(00)} = 1$

$P_{ср}$ y = 7 : 0 1 1 1

x = 9 : 1 0 0 1

для “нижнего этажа“: $2^{(01)} = 2$

для “верхнего этажа“: $2^{4 + (01)} = 32$

$P_{выхода}$ y = 11 : 1 0 1 1

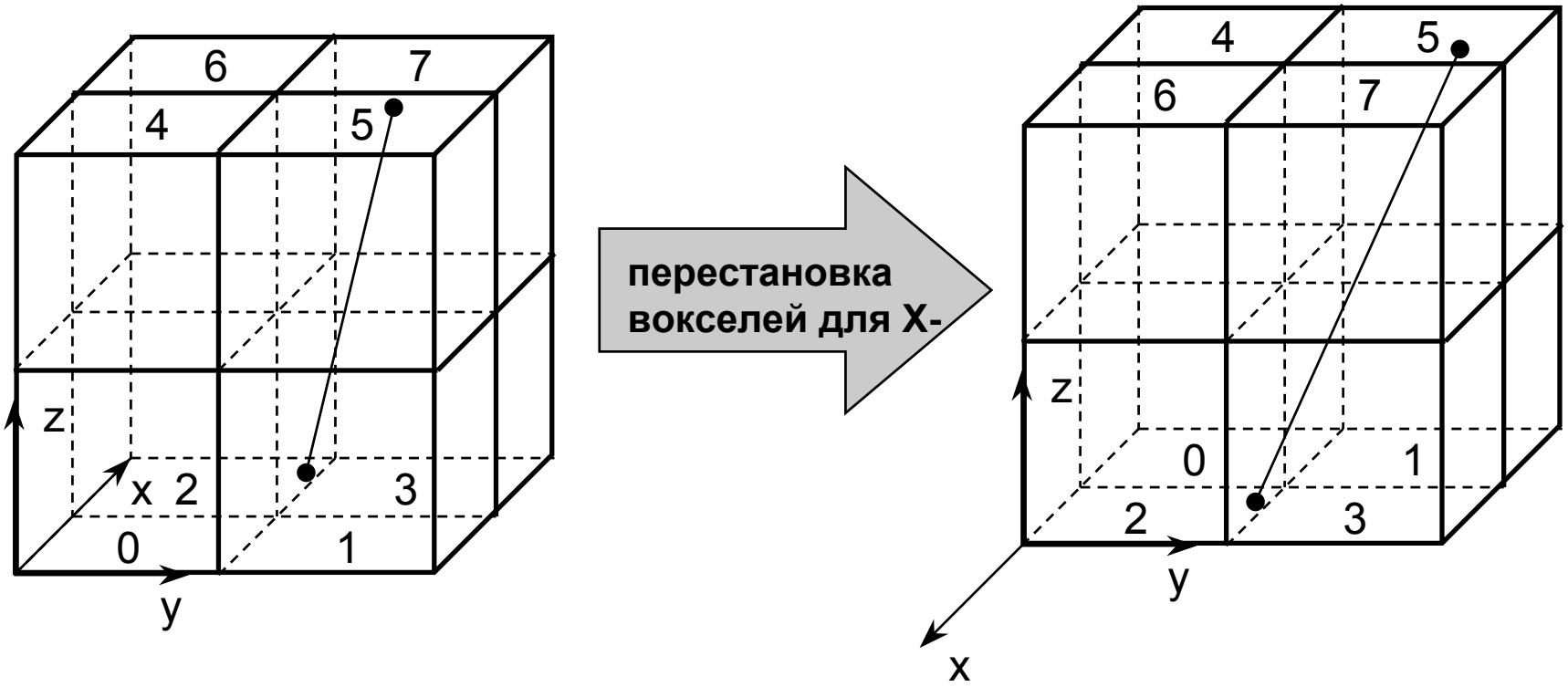
x = 13 : 1 1 0 1

$2^{4 + (11)} = 128$

путь луча	1	0	1	0	0	0	1	1
инфобайт	0	0	1	1	0	1	0	1
AND	0	0	1	0	0	0	0	1

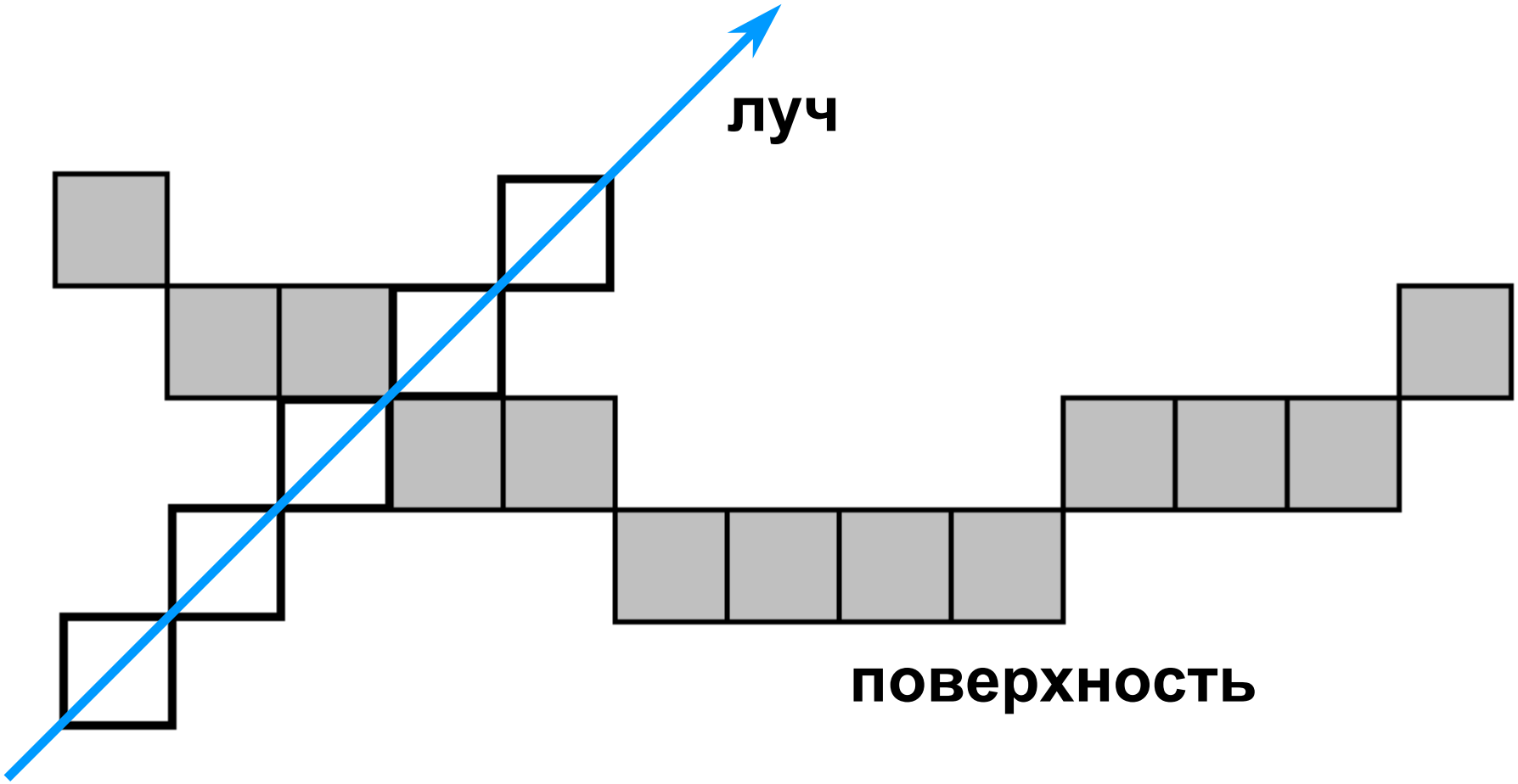
Перестановка вокселей

Ориентация: ОГНЛ = Z; X-; Y+; Z+;

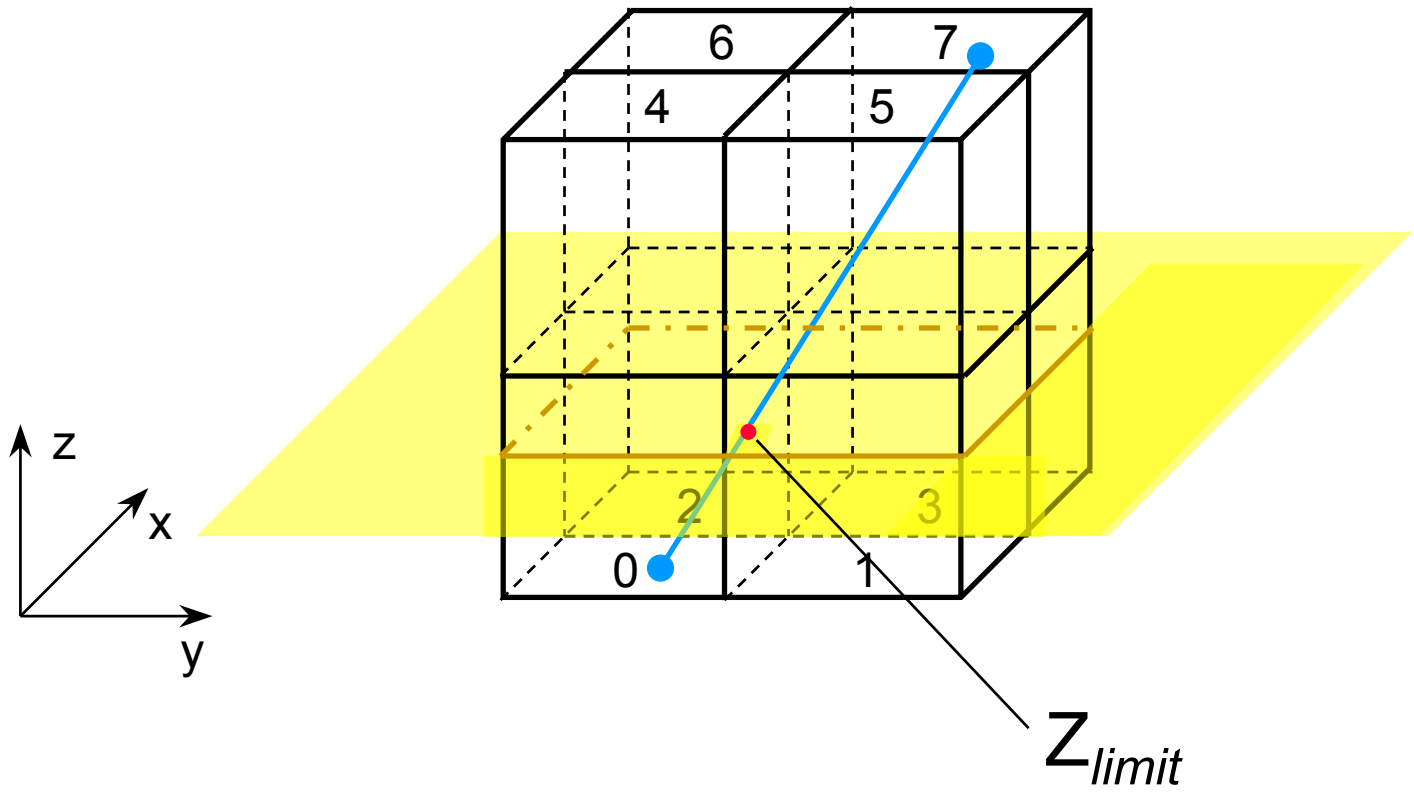


Новый порядок вокселей: 2 → 3 → 0 → 1 → 6 → 7 → 4 → 5

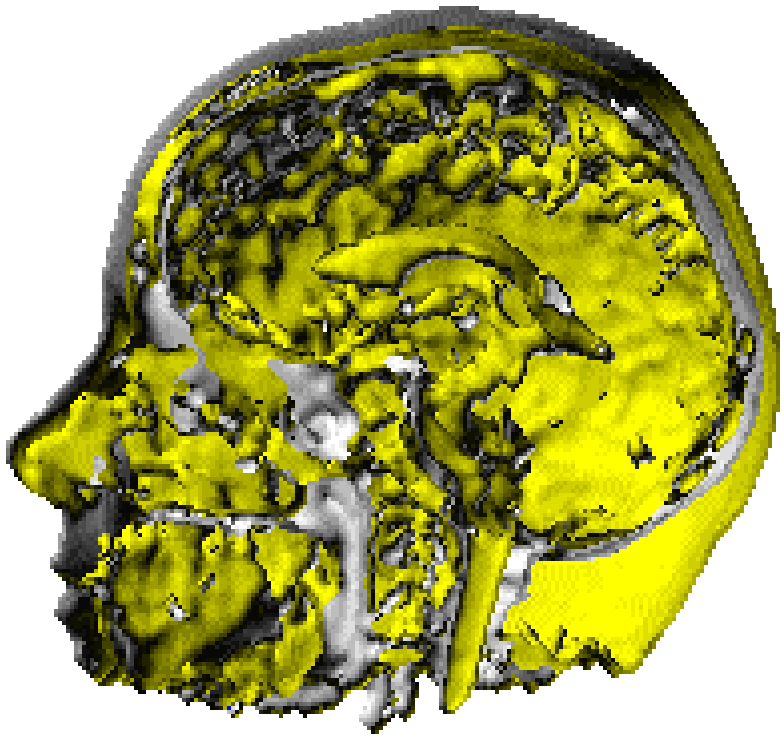
Туннельный эффект



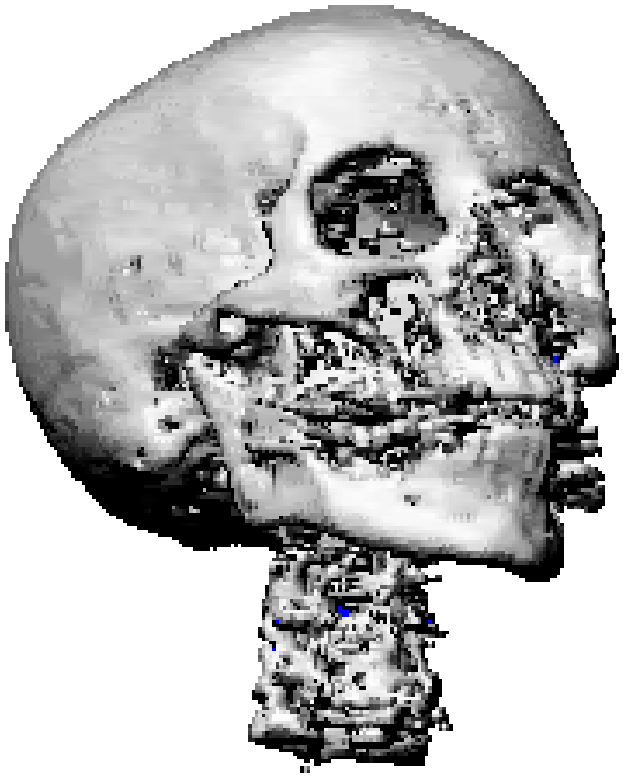
Z-limit



Z-limit

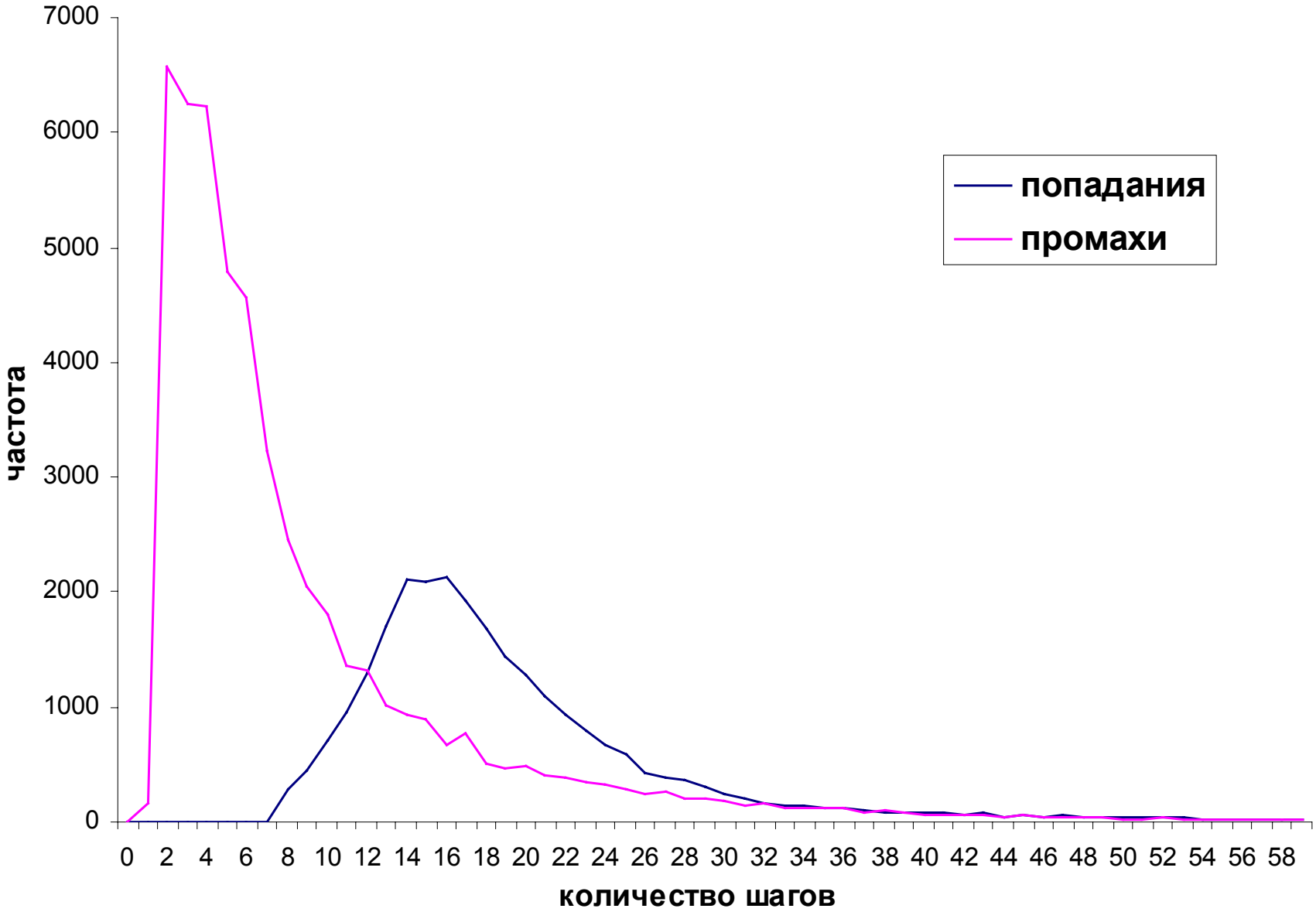


Вычислительные затраты



International Conference Graphicon 1998, Moscow, Russia, http://www.graphicon.ru/

Статистические результаты



Методы оптимизации алгоритма

- аппроксимация параметров алгоритма поиска (координат точек входа и выхода)
- отслеживание только нужных лучей (k-dop)
- использование когерентности цветов соседних пикселей
- кэширование пути луча в дереве

Методы оптимизации структур данных

- гранулирование дерева:
  - сокращение расходов на ссылки внутри дерева
  - послойная обработка
  - возможность неполной загрузки
  - кэширование данных

Свойства метода

- компактное хранение объектов (экономия ресурсов/памяти)
- интенсивно применяется только целочисленная арифметика
- меньшая стоимость операции поиска точки пересечения луча и объекта
- независимость скорости от сложности модели
- нет необходимости делать текстурирование в процессе визуализации
- нет необходимости вычислять нормали в процессе визуализации
- можно визуализировать данные с измерительных приборов без полигонизации
- простота реализации алгоритмов поиска пересечений объектов для игр и САПР
- можно комбинировать геометрические данные с измерительных устройств с CSG и полигональными моделями

Михаил Цыганков

Softgraph Intl. GmbH

Email: Mike@Softgraph.com

Телефоны: +7-812-3279900

+7-812-5877331

Факс: +7-812-3279865

Адрес: Россия, 199053, Санкт-Петербург, п/я 703