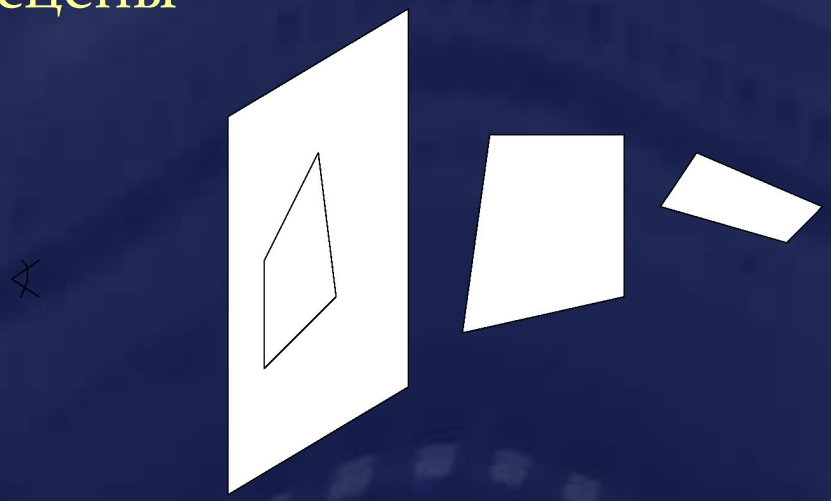
Удаление невидимых линий и поверхностей

Лекция 9

Астана 2004

Методы удаления невидимых линий и поверхностей

При проецировании трехмерных объектов на картинную плоскость (экран) часто оказывается, что отдельные части объектов (или даже некоторые объекты сами) оказываются скрытыми от наблюдателя другими объектами сцены



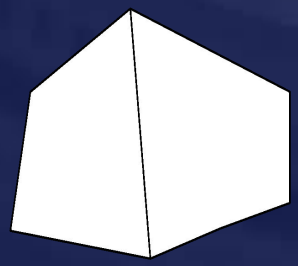
Классификация методов удаления невидимых линий и поверхностей

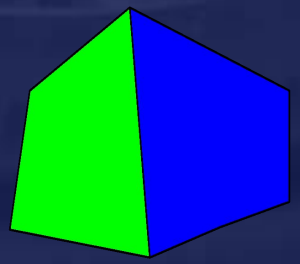
По способу изображения объекта:

- Каркасное (wireframe)
- Сплошное (solid)

По пространству, в котором решается задача

- В пространстве сцены
- На картинной плоскости



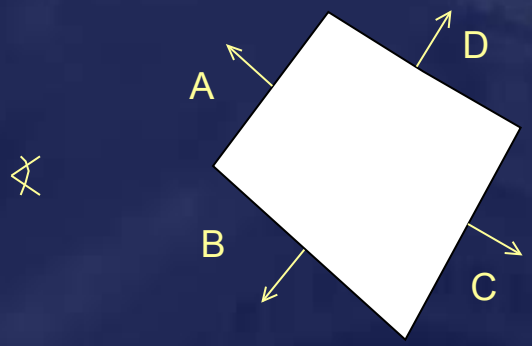


По точности получаемого решения

- Точные аналитические методы (*continuous*)
- Приближенные методы (*point-sampling*)

Лицевые и нелицевые грани

Если грани являются границей тела (или нескольких тел), то для каждой из них можно определить вектор внешней нормали



Нормали к граням *A* и *B* смотрят в сторону наблюдателя (наблюдатель находится в положительном полупространстве по отношению к плоскости, проходящей через соответствующую грань). Такие грани называются *лицевыми (front-faced*).

Для граней *C* и *D* нормали направлены от наблюдателя, их называ нелицевыми (*back-faced*).

Свойства (не)лицевых граней

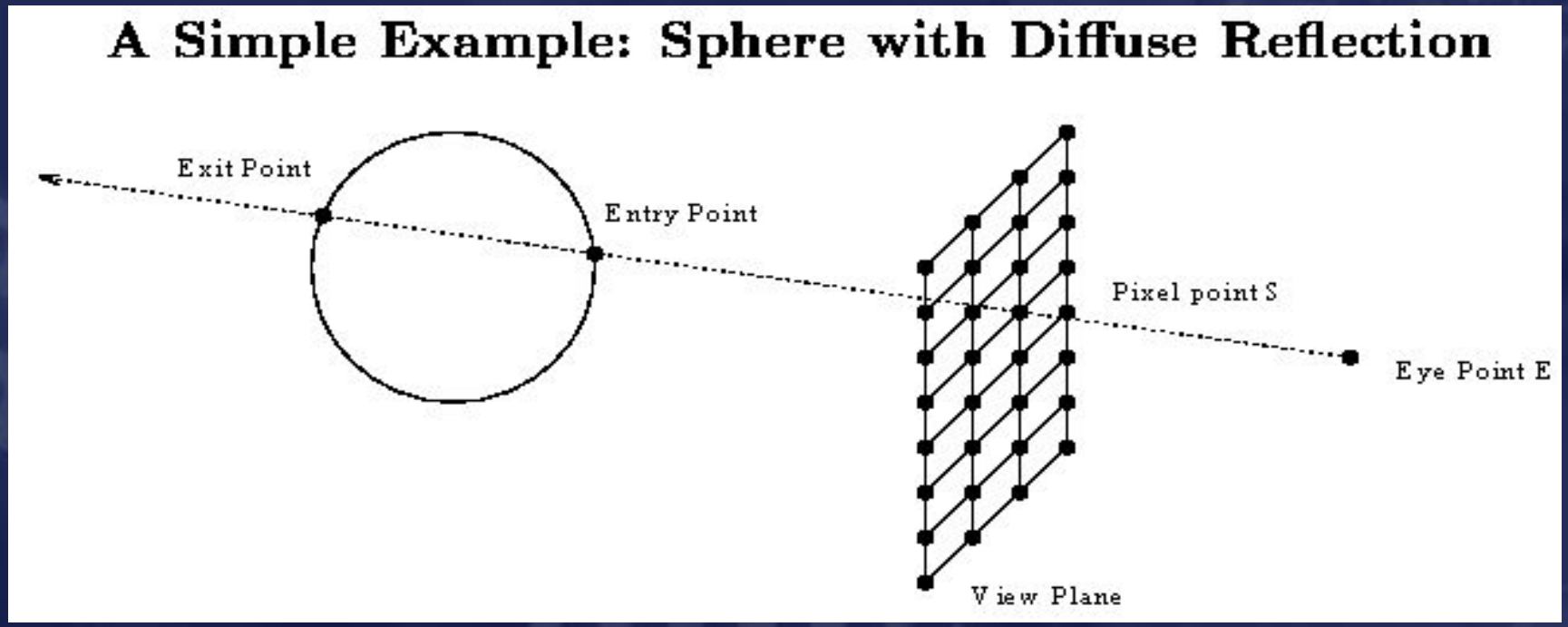
В случае, когда грани являются границей тела (или нескольких тел), то ни одна из нелицевых граней не может быть видна даже частично – любая из них всегда будет закрываться от наблюдателя лицевыми гранями.

При определении видимости все нелицевые грани можно всегда отбрасывать, что сокращает число рассматриваемых граней примерно вдвое (в общем случае количество лицевых граней примерно равно количеству лицевых, т.е. составляет половину от общего числа граней).

Когда вся сцена состоит из одного выпуклого объекта, то все лицевые грани и только они будут видны, причем полностью.

Трассировка лучей

При использовании метода трассировки лучей через каждый пиксел картинной плоскости выпускается луч (из положения наблюдателя) в сцену. Далее находится ближайшее пересечение этого луча с объектами сцены – оно и определяет, что именно будет видно в данном пикселе.



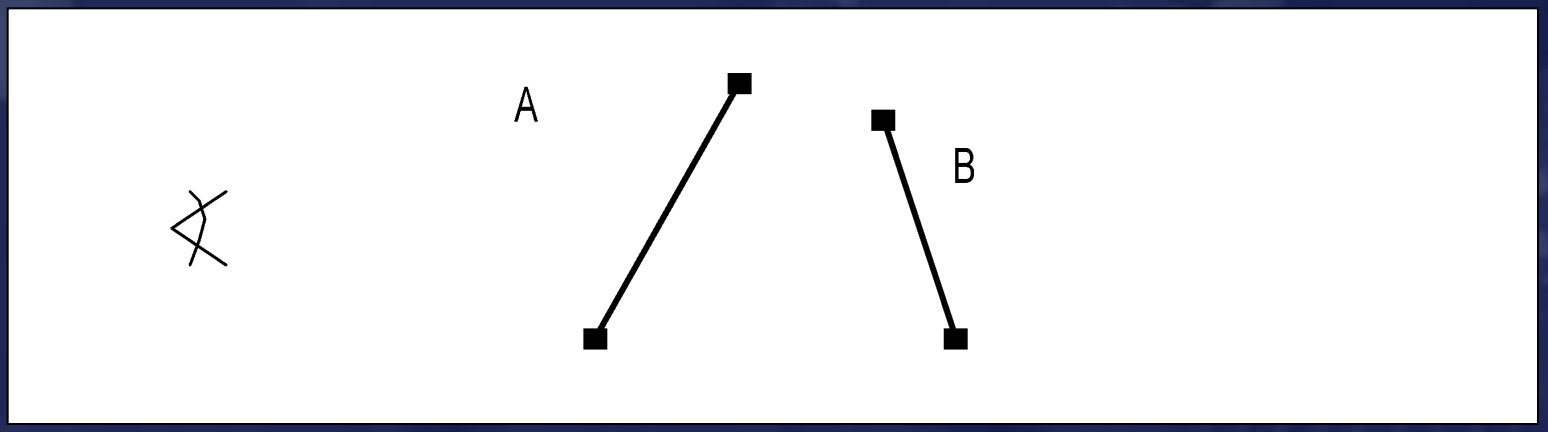
Метод буфера глубины

Каждому пикселу картинной плоскости, кроме значения цвета, хранящемуся в буфере кадра, сопоставляется еще значение глубины (расстояние вдоль направления проектирования от картинной плоскости до соответствующей точки пространства).

```
foreach(p in pixels)
   if(p.z < zBuffer[p.x, p.y])
      draw( p );
      zBuffer[p.x, p.y] = p.z;
    }
```

Алгоритм художника

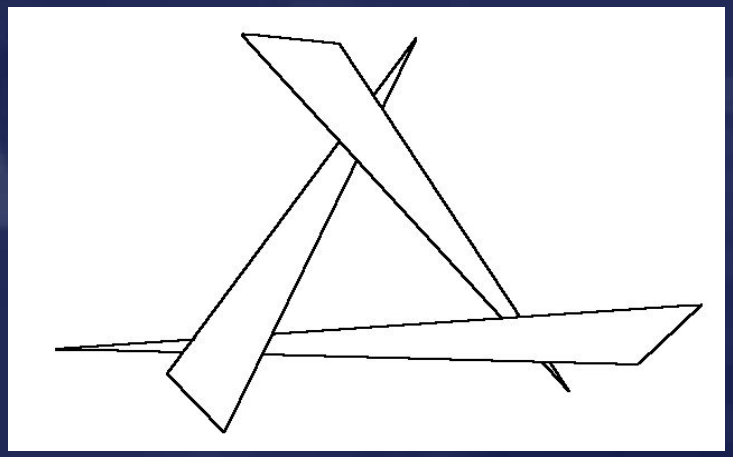
Алгоритм художника (*painter's algorithm*) явно сортирует все грани сцены в порядке их приближения к наблюдателю (*back-to-front*) и выводит их в этом порядке.



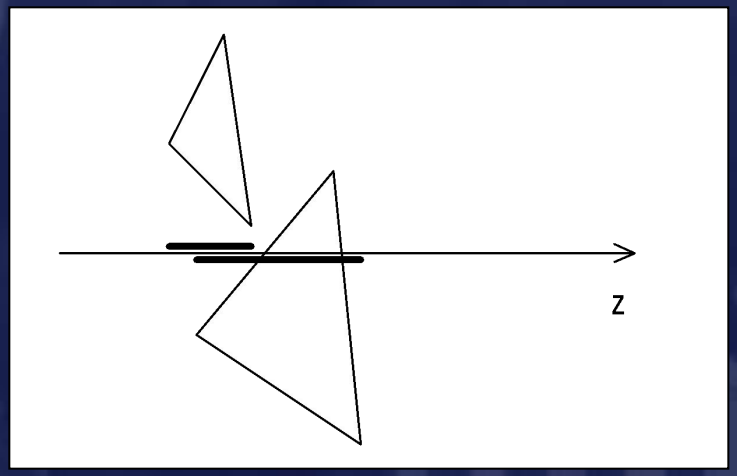
Если сперва вывести объект *B*, а потом вывести объект *A* поверх него, то в результате получится корректное изображение (для тех пикселов, которые принадлежат как проекции грани *A*, так и проекции грани *B*, последним будет выведено значение, соответствующее грани *A*, которое и должно быть видно).

Алгоритм художника: проблемы

- Не всегда грани возможно упорядочить



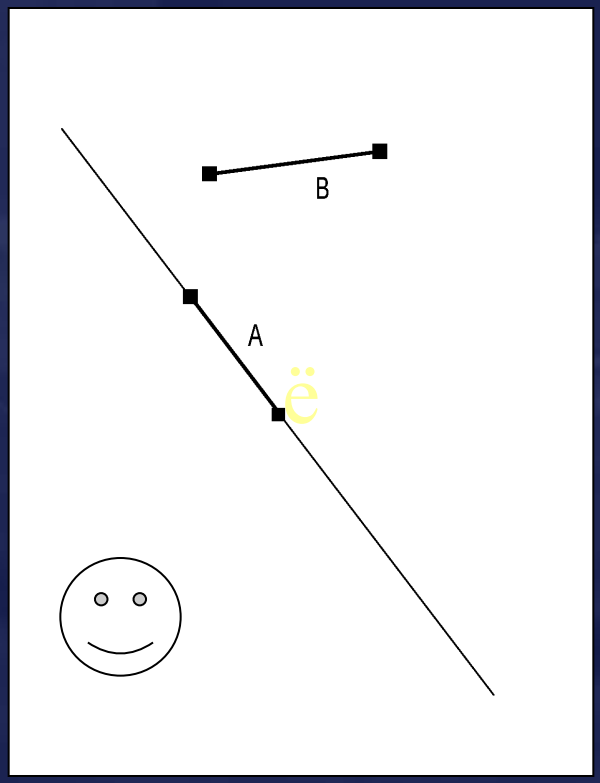
- Не всегда грани возможно сравнить по координате z



Упорядочивание граней

Проведем через одну из граней плоскость и проверим, лежит ли другая грань целиком по одну сторону относительно этой плоскости.

Например, грань *B* не может закрывать грань *A* от наблюдателя, поскольку находится в другом полупространстве относительно плоскости, проходящей через грань *A*.

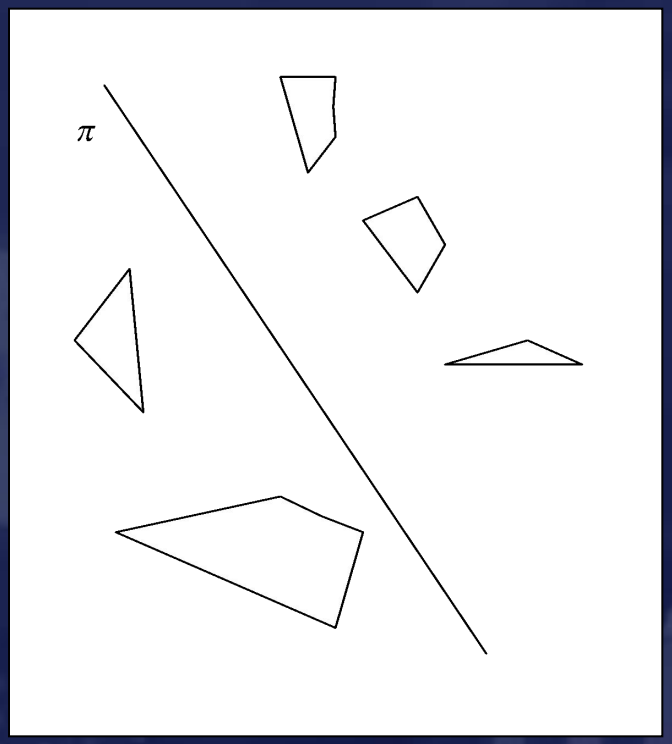


Пять проверок в алгоритме художника

1. Накладываются ли *x*-габариты мн-ков?
2. Накладываются ли *y*-габариты мн-ков?
3. P полностью за плоскостью Q по отношению к наблюдателю?
4. Q полностью перед плоскостью P по отношению к наблюдателю?
5. Пересекаются ли проекции многоугольников на плоскость *(x, y)*?

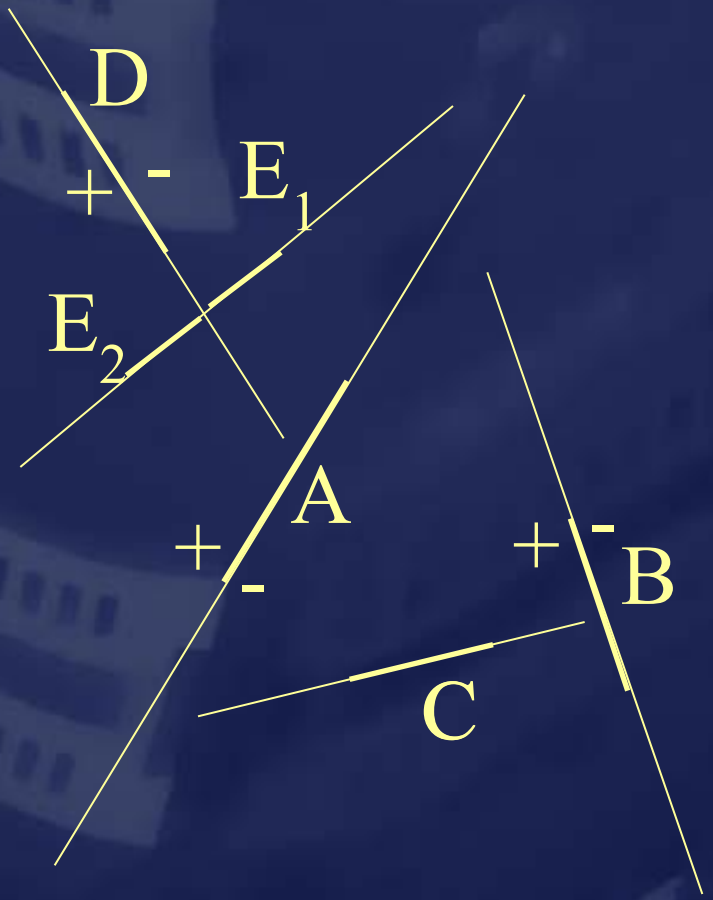
Метод двоичного разбиения пространства (1/3)

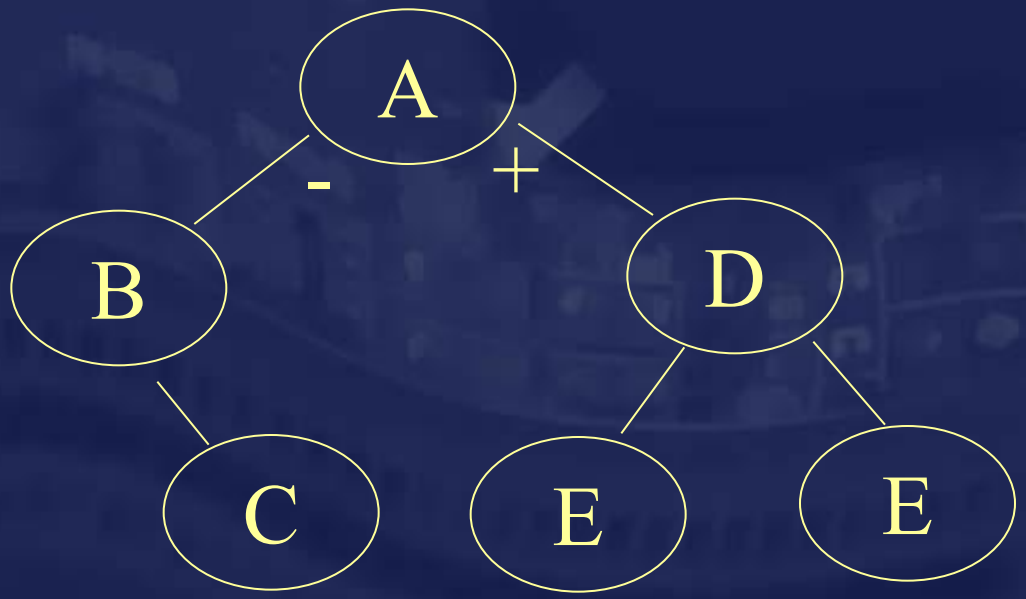
Пусть известно, что плоскость π разбивает все грани (объекты) сцены на два непересекающихся множества в зависимости от того, в каком полупространстве по отношению к данной плоскости они лежат



Тогда ни одна из граней, лежащих в том же полупространстве, что и наблюдатель, не может быть закрыта ни одной из граней из другого полупространства. Таким образом, удалось осуществить частичное упорядочение граней исходя из возможности загораживания друг друга.

Метод двоичного разбиения пространства (2/3)



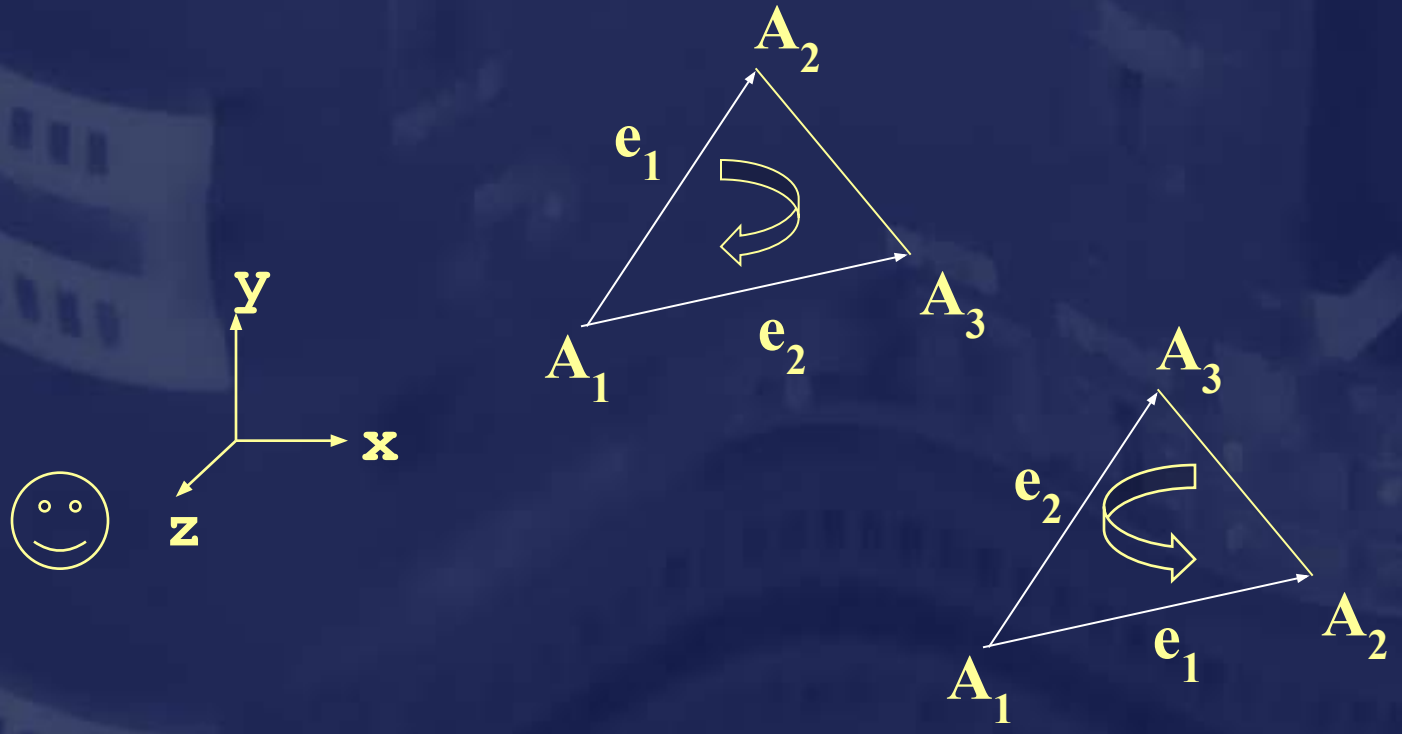


Метод двоичного разбиения пространства (3/3)

```
class BSPNode {
    Face *face; // Грань объекта
    BSPNode *positive;
    BSPNode *negative;
    …
}
```

```
void BSPNode::Draw() {
    if(face->Sign(viewer) == 1) {
        if(negative) negative->Draw();
        face->Draw();
        if(positive) positive->Draw();
    } else {
        if(positive) positive->Draw();
        face->Draw();
        if(negative) negative->Draw();
    }
}
```

Лицевые и нелицевые грани в OpenGL



$$e_1 = A_2 - A_1,$$
$$e_2 = A_3 - A_1,$$
$$n = [e_1, e_2]$$

```
glEnable(GL_CULL_FACE); glDisable(GL_CULL_FACE);

void glFrontFace(GLenum type);
     type = {GL_CW|GL_CCW}

void glCullFace(GLenum type);
     type = {GL_FRONT|GL_BACK (по умолчанию)}
```

Z-буфер

- Необходимо создать z-буфер

```
glutDisplayMode(GLUT_DEPTH|/*…*/);
```

- Перед рисованием сцены очистить z-буфер

```
glClear(GL_DEPTH_BUFFER_BIT|/*…*/);
```

- Включить или выключить сравнение z координат

```
glEnable(GL_DEPTH_TEST);
glDisable(GL_DEPTH_TEST);
```

- Возможно включить или выключить запись в z-буфер

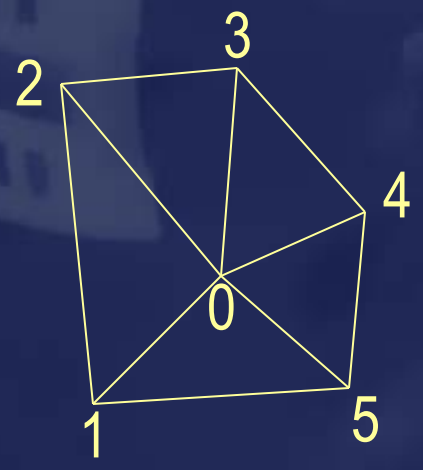
```
glDepthMask(TRUE); или glDepthMask(FALSE);
```

- Возможно задать операцию сравнения

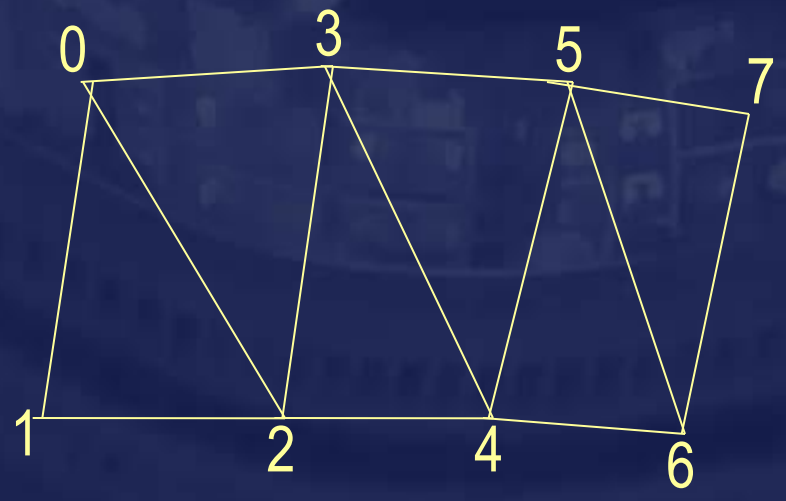
```
glDepthFunc(GLenum type);
type = {GL_ALWAYS|GL_NEVER|GL_LESS|GL_GREATER|
        GL_EQUAL|GL_NOTEQUAL|GL_LEQUAL|GL_GEQUAL}
```

Уменьшение количества вершин

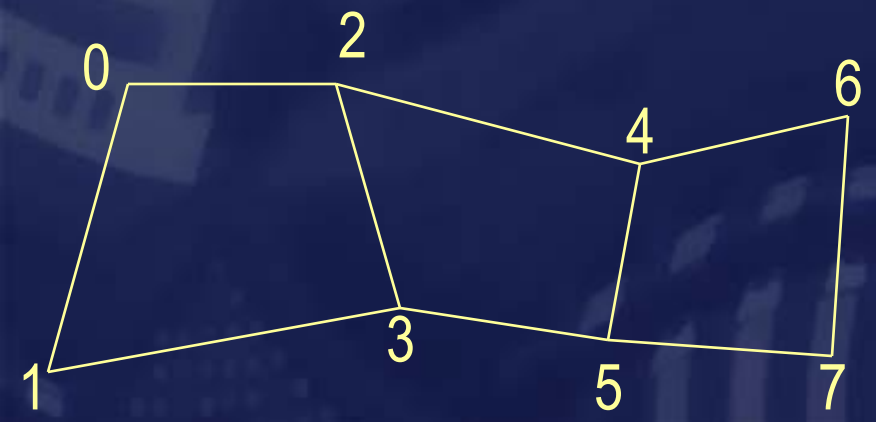
- GL_TRIANGLE_FAN: 3n vs. 1+n, n>1



- GL_TRIANGLE_STRIP: 3n vs. 2+n



- GL_QUAD_STRIP: 4n vs. 2+2n



Дисплейные списки

- Дисплейный список (display list) – запомненная последовательность команд OpenGL.
- Находим неиспользуемый номер дисплейного списка

```
GLuint n = glGenLists(1);
```

- Сохраняем последовательность команд

```
glNewList(n,GL_COMPILE);
<…вызовы функции OpenGL…>
glEndList();
```

- Воспроизводим сохраненную последовательность команд (с теми же самыми параметрами!)

```
glCallList(n);
```

- Освобождаем номер дисплейного списка

```
glDeleteLists(n,1);
```