Компьютерная графика

Лекция 9: Геометрические шейдеры, shadow volumes, shadow mapping и его разновидности

2021

Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным

Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным
- Создаётся как `glCreateShader(GL_GEOMETRY_SHADER)`

Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным
- Создаётся как `glCreateShader(GL_GEOMETRY_SHADER)`
- Встраивается после вершинного шейдера, до perspective divide

Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным
- Создаётся как `glCreateShader(GL_GEOMETRY_SHADER)`
- Встраивается после вершинного шейдера, до perspective divide
- Оперирует целыми примитивами (точками/линиями/треугольниками), т.е. наборами вершин

Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным
- Создаётся как `glCreateShader(GL_GEOMETRY_SHADER)`
- Встраивается после вершинного шейдера, до perspective divide
- Оперирует целыми примитивами (точками/линиями/треугольниками), т.е. наборами вершин
- Может менять тип примитива и количество вершин

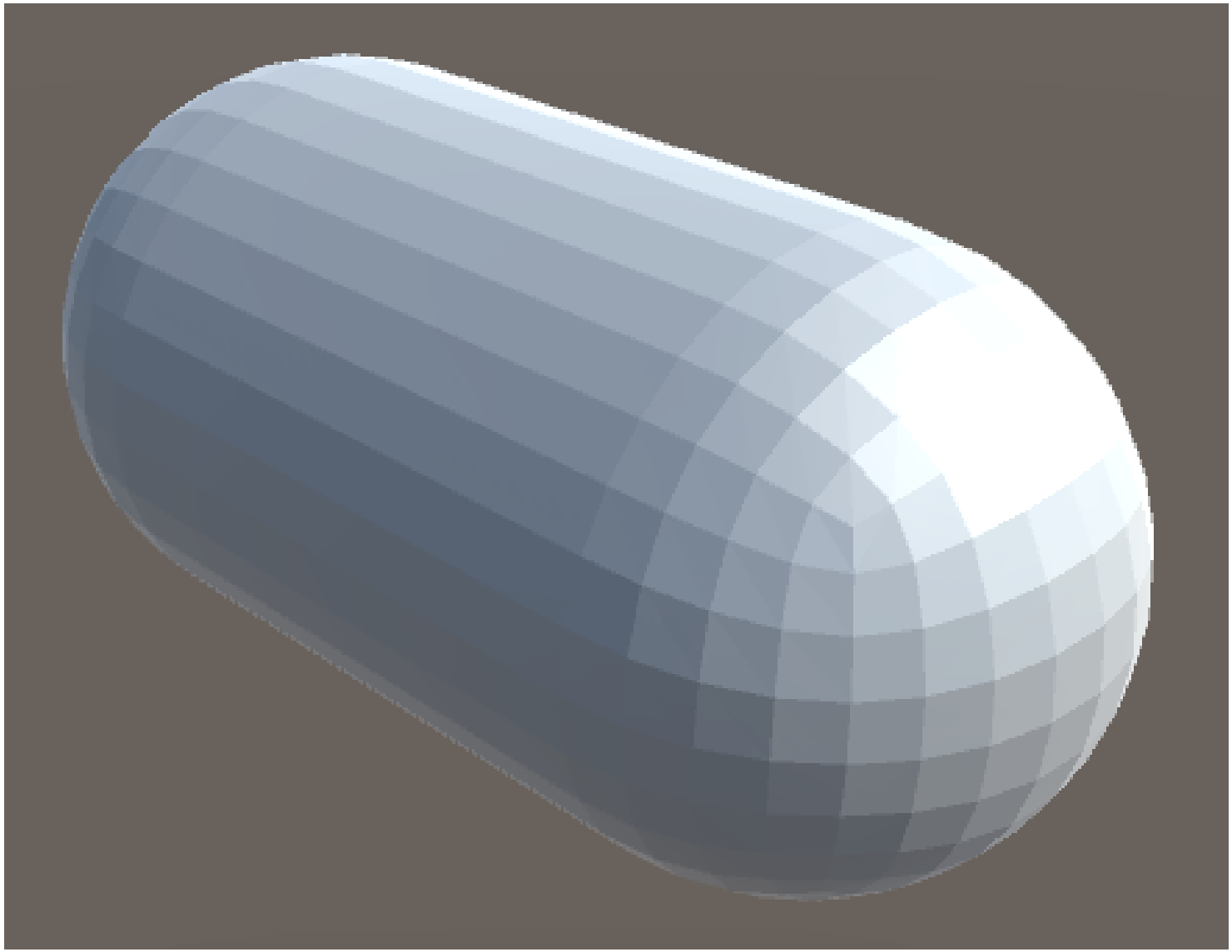
Геометрические шейдеры

- Тип шейдера, наравне в вершинным и фрагментным
- Создаётся как `glCreateShader(GL_GEOMETRY_SHADER)`
- Встраивается после вершинного шейдера, до perspective divide
- Оперирует целыми примитивами (точками/линиями/треугольниками), т.е. наборами вершин
- Может менять тип примитива и количество вершин
- Может варьировать количество вершин на выходе

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а

Flat shading



Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью

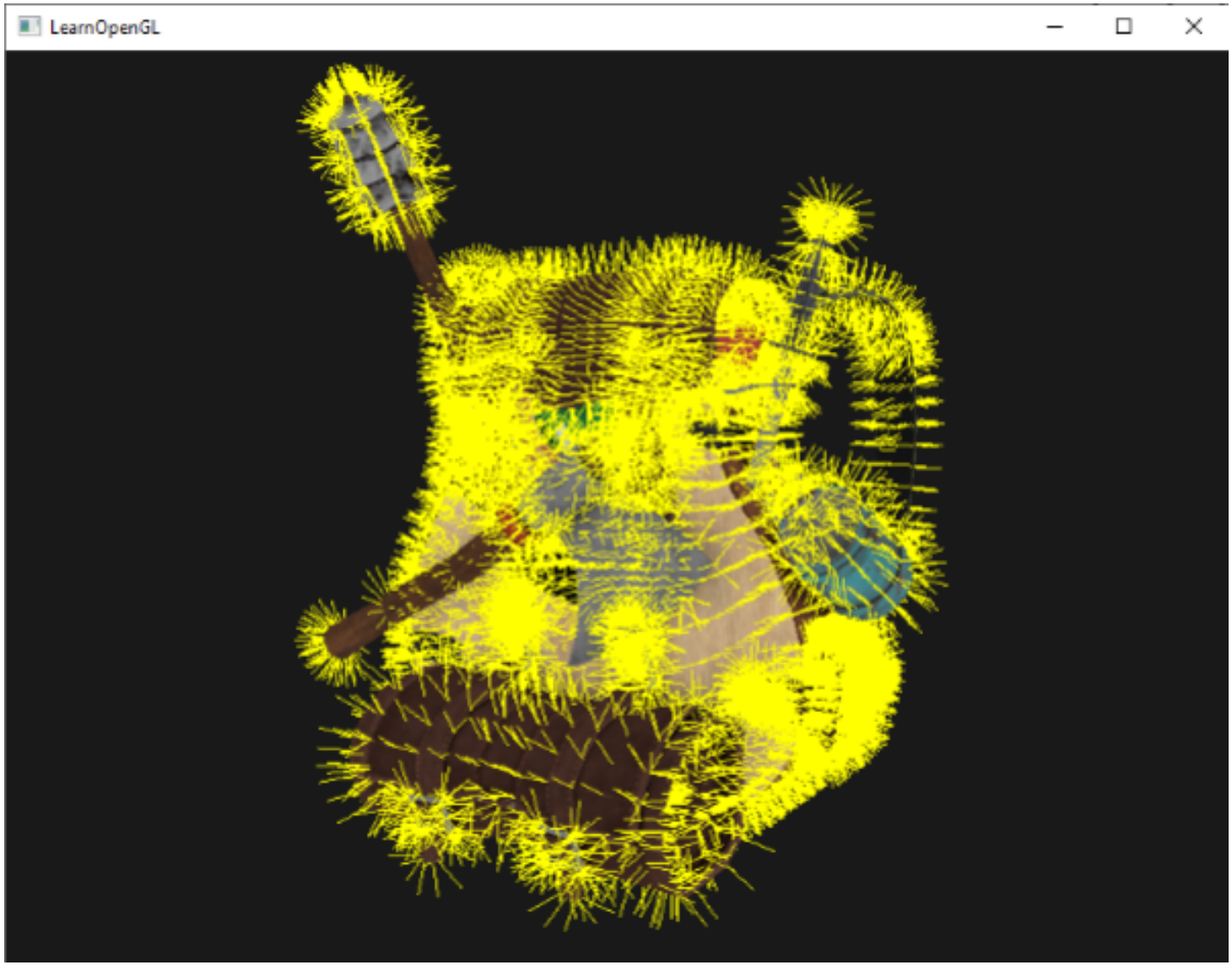
Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль

Визуализация нормалей



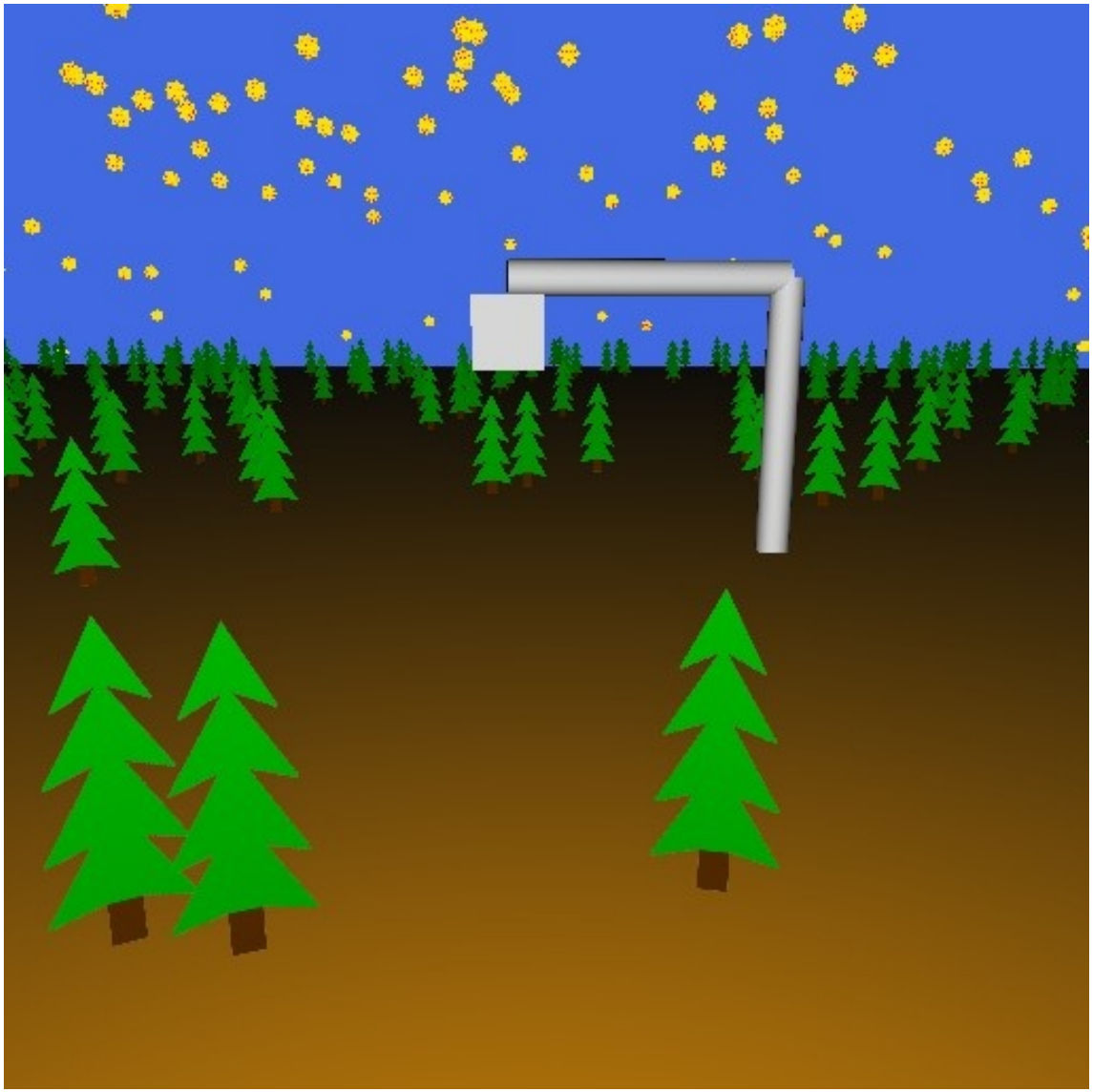
Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры

Billboards



Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц

Billboards: система частиц (дым)

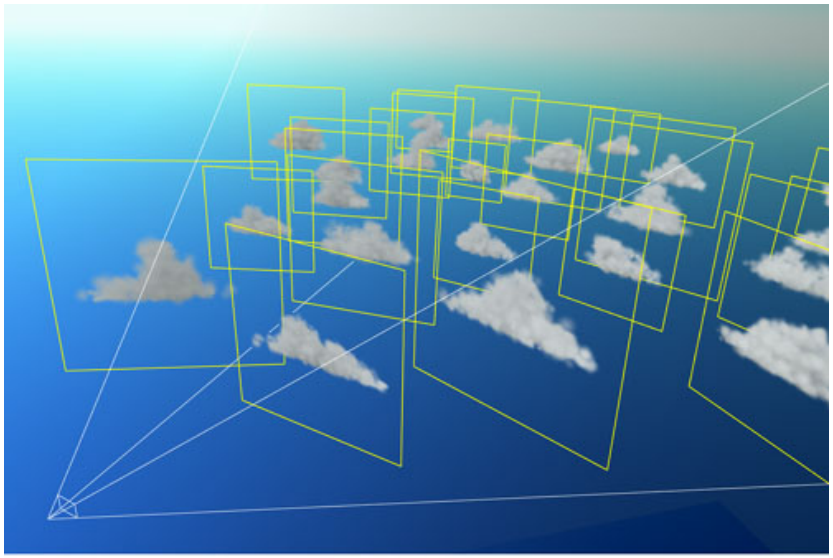
Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц
  - Облака

Billboards: облака



Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц
  - Облака

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц
  - Облака
  - Деревья

Billboards: деревья

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц
  - Облака
  - Деревья

Геометрические шейдеры: примеры использования

- Расчёт нормалей для flat shading'а
  - Вход: треугольник (тройка вершин)
  - Выход: те же вершины с посчитанной нормалью
- Визуализация нормалей
  - Вход: вершина с нормалью
  - Выход: линия из двух вершин - исходная вершина, исходная вершина + нормаль
- Shadow volumes
- Billboards - плоские фигуры, всегда смотрящие в сторону камеры
  - Вход: одна вершина (точка)
  - Выход: набор треугольников
  - Системы частиц
  - Облака
  - Деревья
  - Трава

Billboards: трава

Геометрические шейдеры: пример

```
#version 330 core
uniform mat4 transform;
// Входные примитивы - точки
layout (points) in;
// Выходные примитивы - линии, в сумме не больше 2х вершин
layout (line_strip, max_vertices = 2) out;
// Данные из вершинного шейдера
in vec3 normal[];

void main() {
    gl_Position = transform * gl_in[0].gl_Position;
    EmitVertex();

    gl_Position = transform * (gl_in[0].gl_Position
        + vec4(normal[0], 0));
    EmitVertex();

    EndPrimitive();
}
```

Геометрические шейдеры: пример

```
#version 330 core
// Входные примитивы - линии
layout (lines) in;
// Выходные примитивы - треугольники, в сумме не больше 4х вершин
layout (triangle_strip, max_vertices = 4) out;
// Данные для фрагментного шейдера
out vec4 color;

void main() {
    gl_Position = gl_in[0].gl_Position + vec4(-1.0, -1.0, 0.0, 0.0);
    color = vec4(1.0, 0.0, 0.0, 1.0);
    EmitVertex();

    gl_Position = gl_in[0].gl_Position + vec4( 1.0, -1.0, 0.0, 0.0);
    color = vec4(0.0, 1.0, 0.0, 1.0);
    EmitVertex();

    gl_Position = gl_in[1].gl_Position + vec4(-1.0,  1.0, 0.0, 0.0);
    color = vec4(0.0, 0.0, 1.0, 1.0);
    EmitVertex();

    gl_Position = gl_in[1].gl_Position + vec4( 1.0,  1.0, 0.0, 0.0);
    color = vec4(1.0, 1.0, 1.0, 1.0);
    EmitVertex();

    EndPrimitive();
}
```

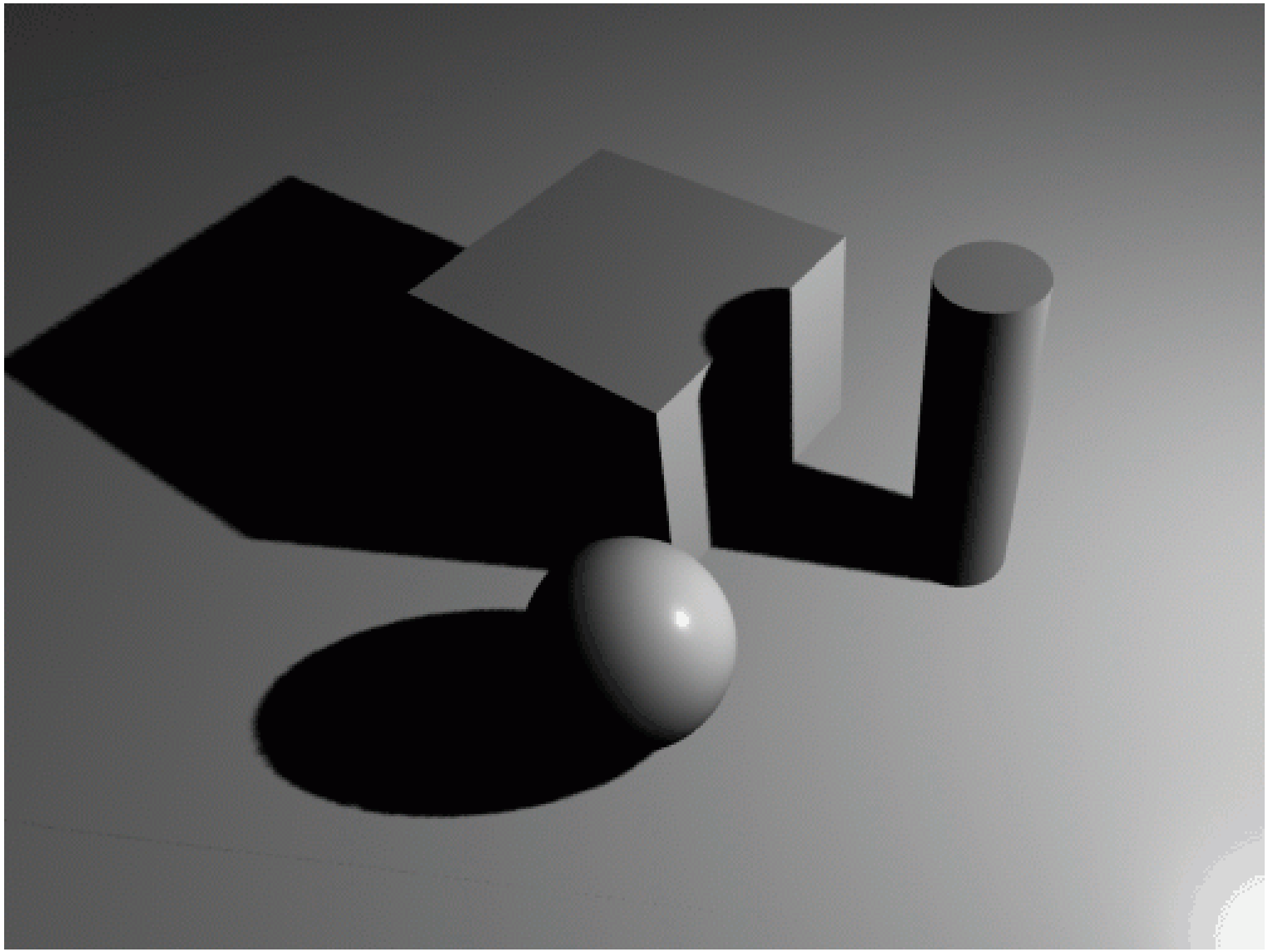
Геометрические шейдеры: ссылки

- khronos.org/opengl/wiki/Geometry_Shader
- learnopengl.com/Advanced-OpenGL/Geometry-Shader
- open.gl/geometry
- lighthouse3d.com/tutorials/glsl-tutorial/geometry-shader
- GPU Gems, Chapter 7. Rendering Countless Blades of Waving Grass

Тени: теория

- Точка сцены, в которую не попадает (заблокирован чем-то) прямой свет из конкретного источника света
- Свойство точки по отношению к конкретному источнику света

Тени



Тени: теория

- Если источник света точечный (или бесконечно удалённый), тень - бинарное свойство: луч из точки сцены в источник света или пересекает что-то (точка в тени), или нет (точка не в тени) $\Rightarrow$ жёсткие тени (hard shadows)

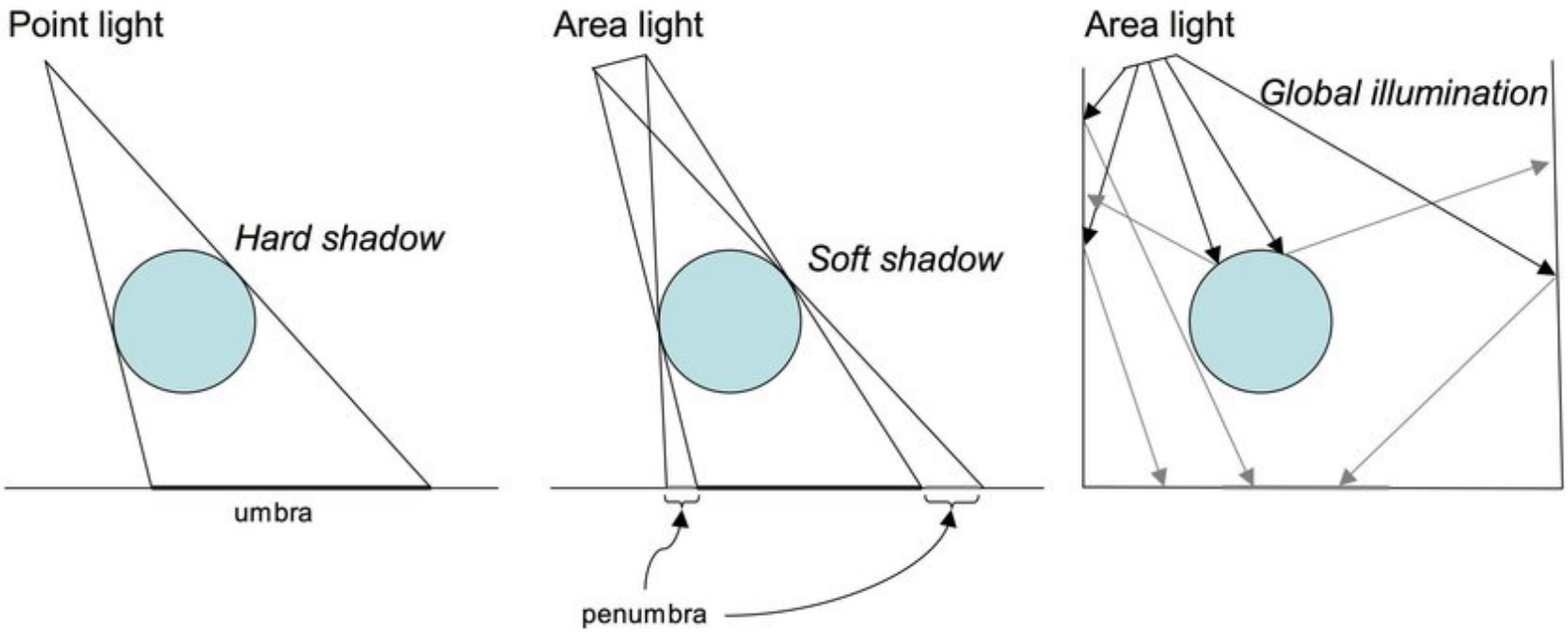
Тени: теория

- Если источник света точечный (или бесконечно удалённый), тень - бинарное свойство: луч из точки сцены в источник света или пересекает что-то (точка в тени), или нет (точка не в тени) $\Rightarrow$ жёсткие тени (hard shadows)
- Если источник света объёмный, точка сцены может находиться в тени относительно части источника света, и не находиться в тени относительно другой его части $\Rightarrow$ мягкие тени (soft shadows)

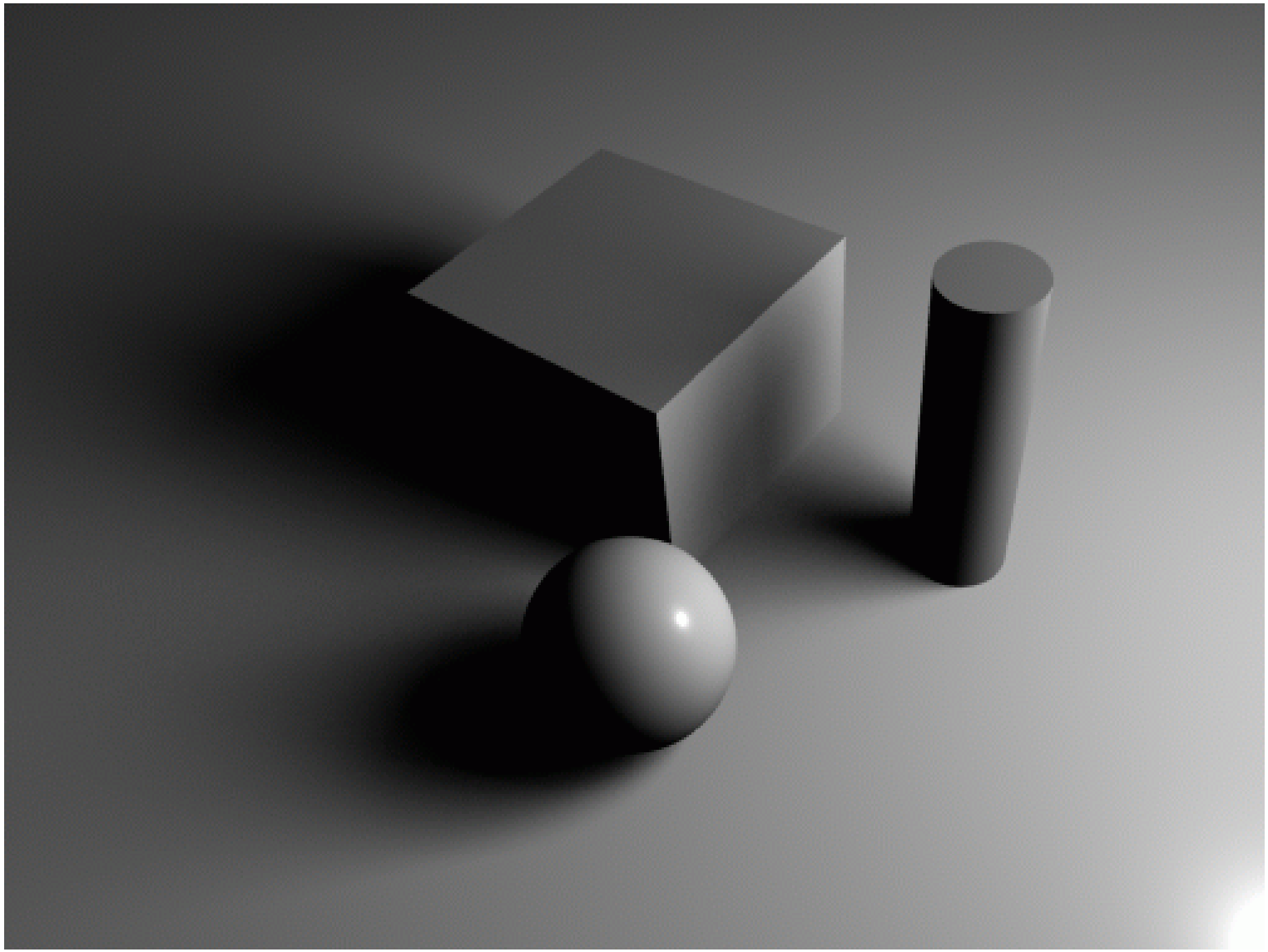
Тени: теория

- Если источник света точечный (или бесконечно удалённый), тень - бинарное свойство: луч из точки сцены в источник света или пересекает что-то (точка в тени), или нет (точка не в тени) $\Rightarrow$ жёсткие тени (hard shadows)
- Если источник света объёмный, точка сцены может находиться в тени относительно части источника света, и не находиться в тени относительно другой его части $\Rightarrow$ мягкие тени (soft shadows)
  - Точки, полностью находящиеся в тени - *umbra*
  - Точки, частично находящиеся в тени - *penumbra*

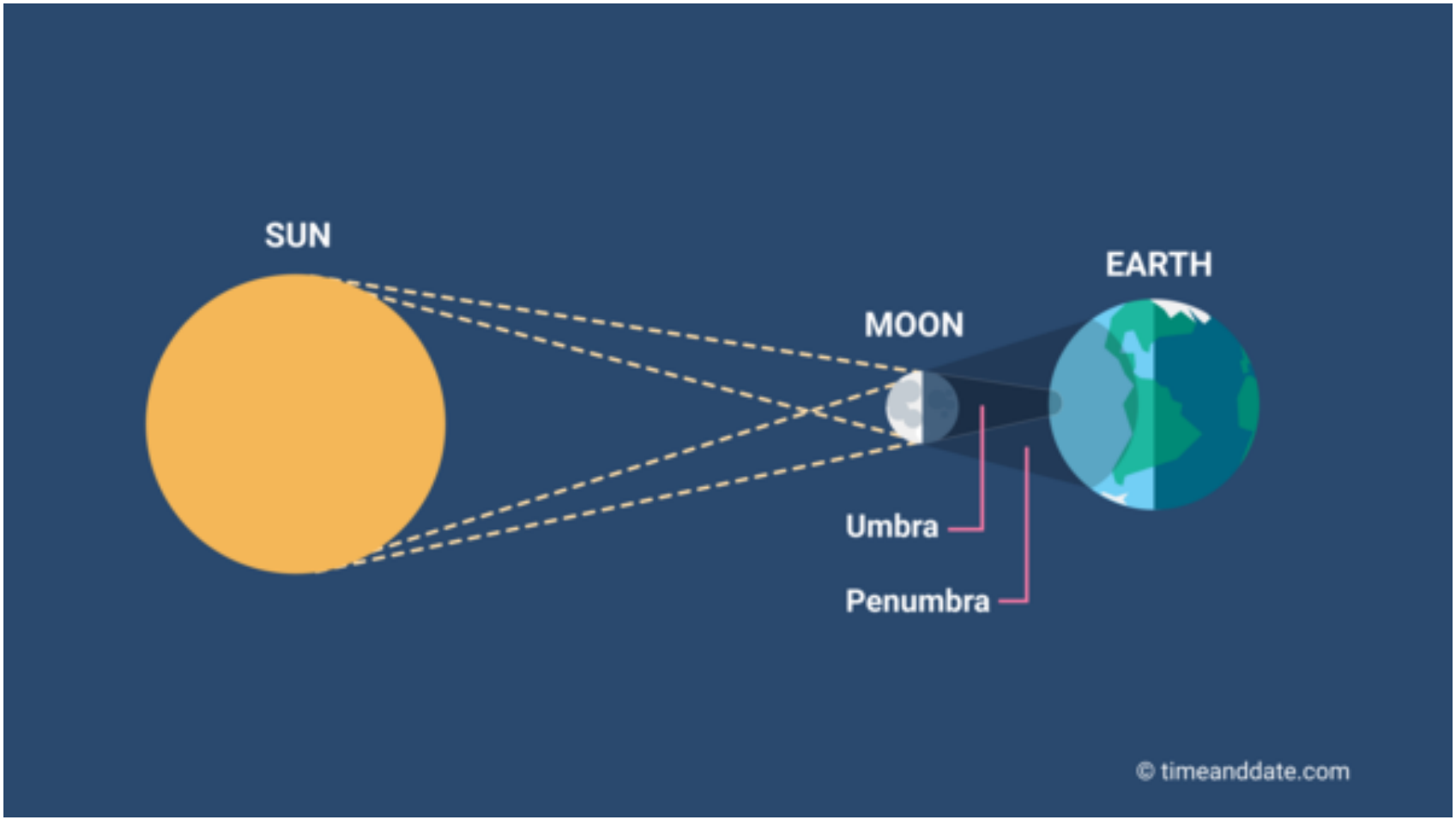
Тени: мягкие vs жёсткие



Мягкие тени



Солнечное затмение



Тени: теория

- Реальные источники света - объёмные

Тени: теория

- Реальные источники света - объёмные
- Размер и форма penumbra зависит от размеров и формы источника света, а также от расстояния до него (чем дальше, тем меньше penumbra)

Тени: теория

- Реальные источники света - объёмные
- Размер и форма penumbra зависит от размеров и формы источника света, а также от расстояния до него (чем дальше, тем меньше penumbra)
  - В пределе расстояния $\to \infty$ мягкая тень вырождается в жёсткую

Тени: теория

- Реальные источники света - объёмные
- Размер и форма penumbra зависит от размеров и формы источника света, а также от расстояния до него (чем дальше, тем меньше penumbra)
  - В пределе расстояния $\to \infty$ мягкая тень вырождается в жёсткую
- В real-time графике получить правильные тени (как жёсткие, так и мягкие) довольно сложно

Тени: алгоритмы

- Shadow volumes

Тени: алгоритмы

- Shadow volumes
  - \+ Идеальные жёсткие тени
  - — Алиасинг
  - — Сложно сделать мягкие тени
  - — Очень большой fill rate (количество обрабатываемых пикселей), плохо предсказумая производительность

Тени: алгоритмы

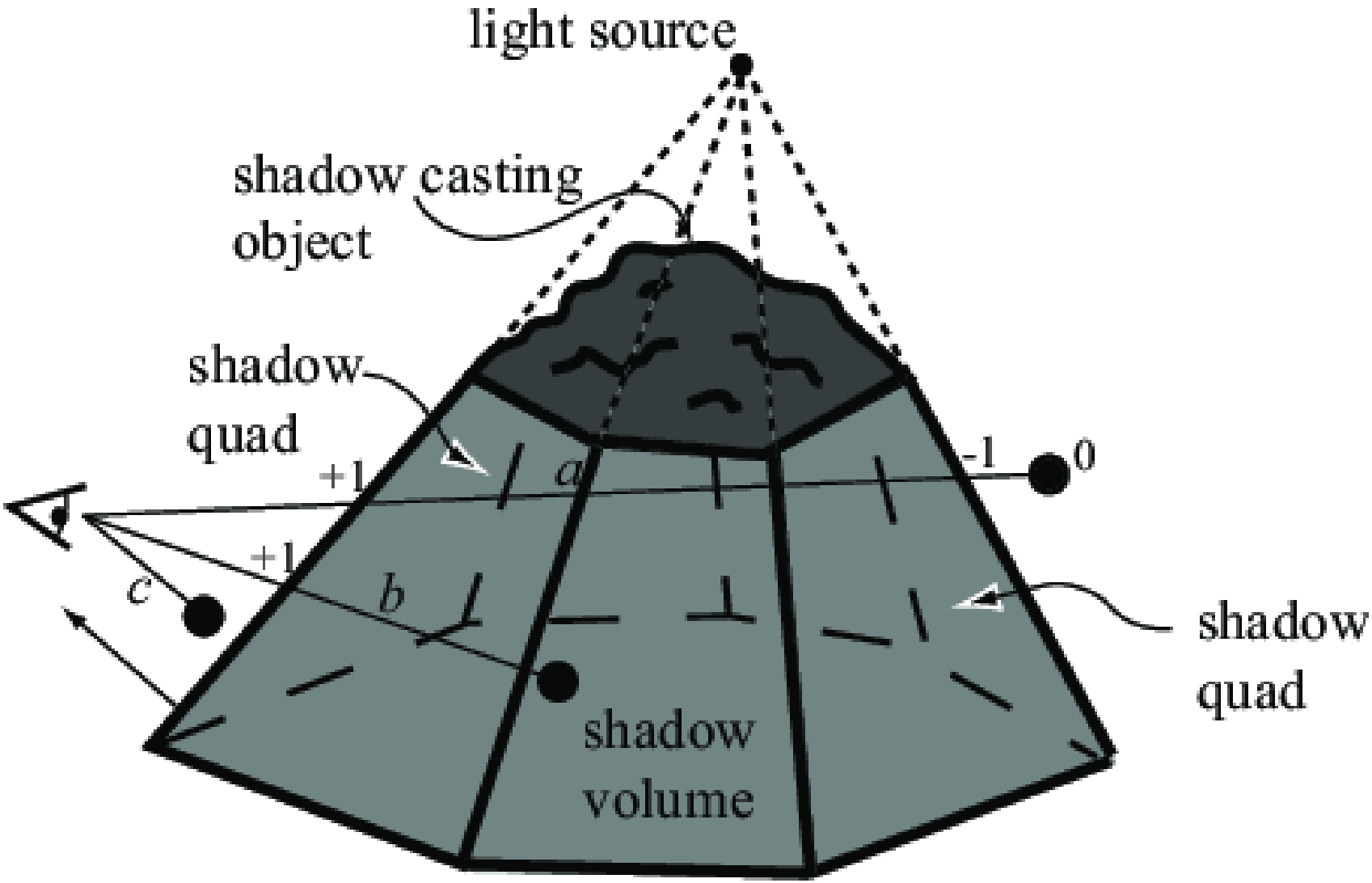
- Shadow volumes
  - \+ Идеальные жёсткие тени
  - — Алиасинг
  - — Сложно сделать мягкие тени
  - — Очень большой fill rate (количество обрабатываемых пикселей), плохо предсказумая производительность
- Shadow mapping

Тени: алгоритмы

- Shadow volumes
  - + Идеальные жёсткие тени
  - — Алиасинг
  - — Сложно сделать мягкие тени
  - — Очень большой fill rate (количество обрабатываемых пикселей), плохо предсказумая производительность
- Shadow mapping
  - + Производительность растёт +/- линейно с ростом сложности сцены
  - — Крупный алиасинг ("пиксельные"тени)
  - + Много вариаций, улучшающих качество и позволяющих делать как жёсткие, так и мягкие тени

Shadow volumes (они же stencil shadows)

- Тень от конкретного объекта - некая (полубесконечная) трёхмерная область пространства (shadow volume)



Shadow volumes (они же stencil shadows)

- Тень от конкретного объекта - некая (полубесконечная) трёхмерная область пространства (shadow volume)

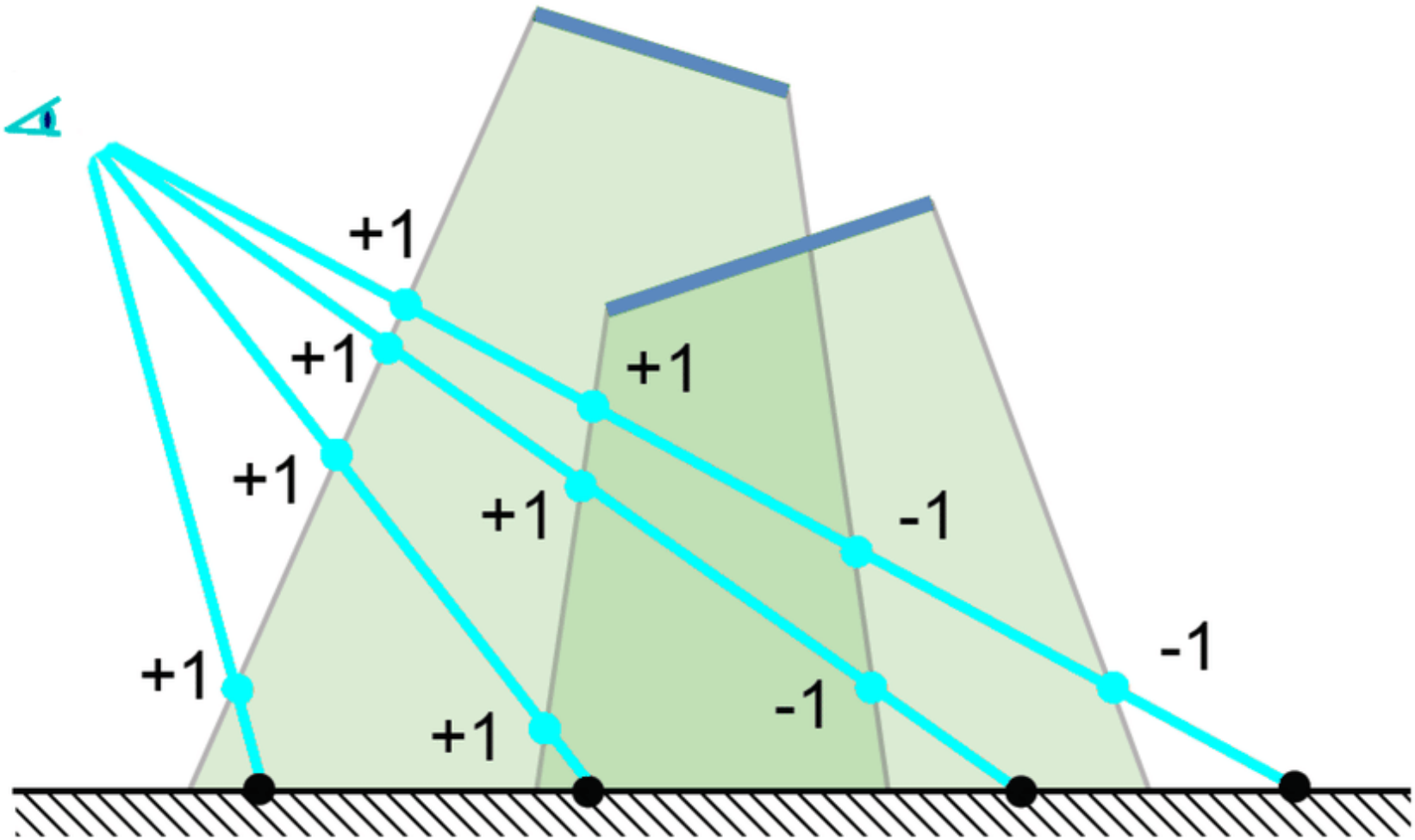
Shadow volumes (они же stencil shadows)

- Тень от конкретного объекта - некая (полубесконечная) трёхмерная область пространства (shadow volume)
- Нарисуем её как трёхмерный объект (построим и нарисуем её грани)

Shadow volumes (они же stencil shadows)

- Тень от конкретного объекта - некая (полубесконечная) трёхмерная область пространства (shadow volume)
- Нарисуем её как трёхмерный объект (построим и нарисуем её грани)
- Если в конкретный пиксель попало одинаковое количество front-facing и back-facing граней, то он не в тени, иначе - в тени

Shadow volumes (они же stencil shadows)



Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)

Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)
- Настраиваем stencil буфер: `+1` для front-facing треугольников, `-1` для back-facing

Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)
- Настраиваем stencil буфер: `+1` для front-facing треугольников, `-1` для back-facing
- Вычисляем и рисуем shadow volume

Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)
- Настраиваем stencil буфер: `+1` для front-facing треугольников, `-1` для back-facing
- Вычисляем и рисуем shadow volume
- Обратно включаем рисование цвета и глубины

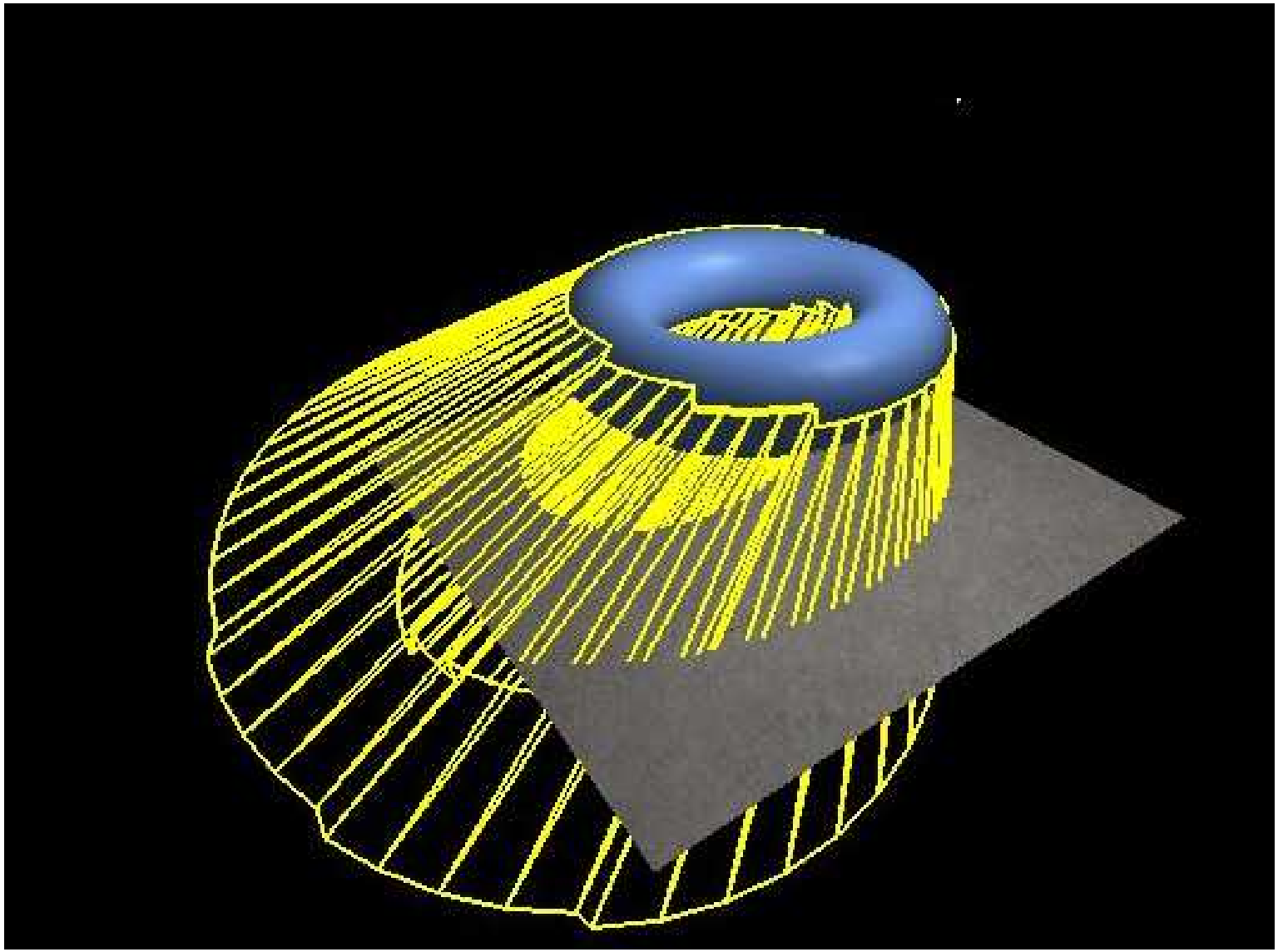
Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)
- Настраиваем stencil буфер: `+1` для front-facing треугольников, `-1` для back-facing
- Вычисляем и рисуем shadow volume
- Обратно включаем рисование цвета и глубины
- Настраиваем stencil-тест так, чтобы рисовать только пиксели, в которых `stencil=0`, и рисуем сцену шейдером, вычисляющим освещение

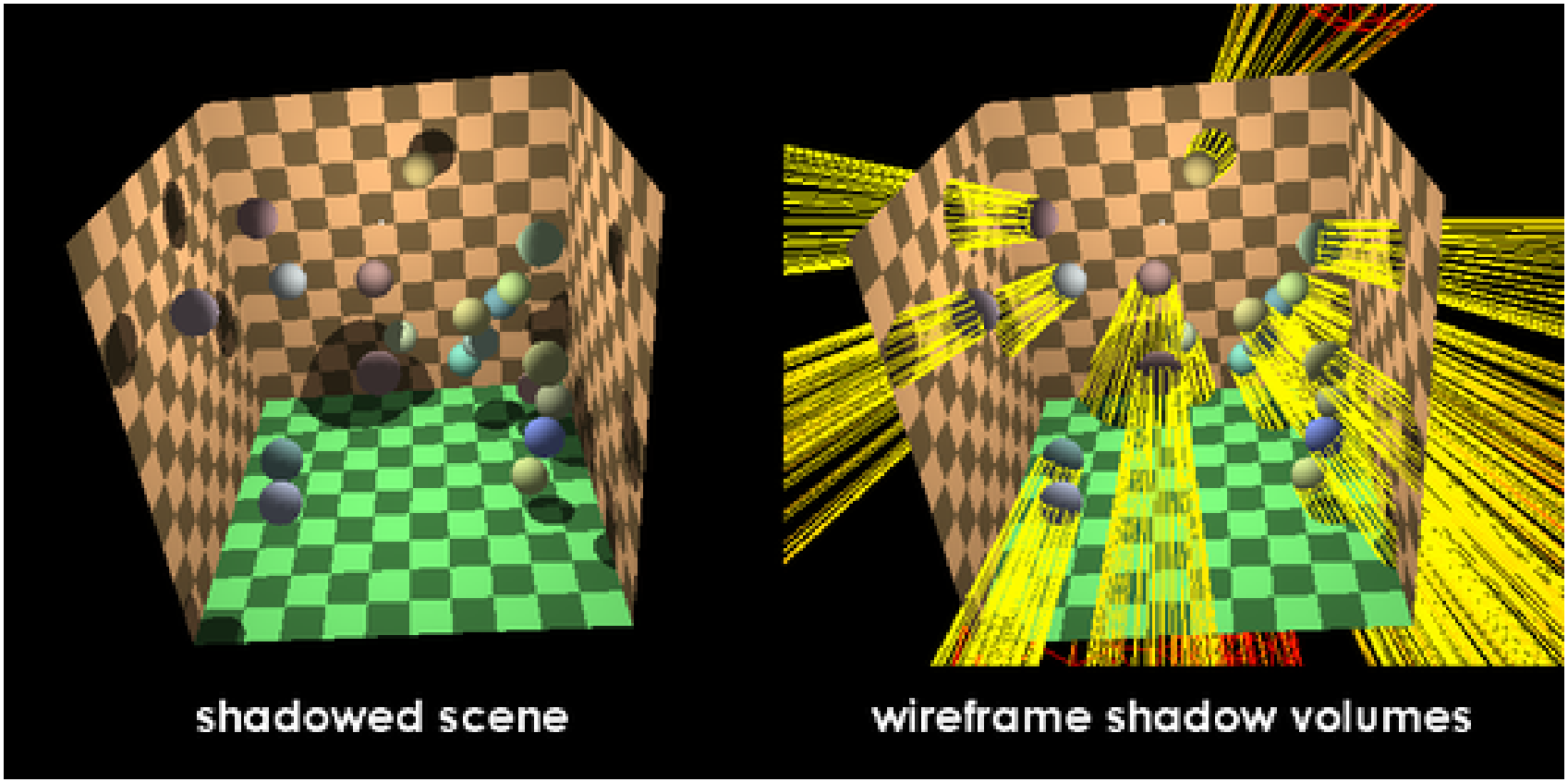
Shadow volumes: алгоритм

- Выключаем рисование в цветовой буфер и буфер глубины (`glColorMask`, `glDepthMask`) (но не выключаем тест глубины!)
- Настраиваем stencil буфер: `+1` для front-facing треугольников, `-1` для back-facing
- Вычисляем и рисуем shadow volume
- Обратно включаем рисование цвета и глубины
- Настраиваем stencil-тест так, чтобы рисовать только пиксели, в которых `stencil=0`, и рисуем сцену шейдером, вычисляющим освещение
- Настраиваем stencil-тест так, чтобы рисовать только пиксели, в которых `stencil!=0`, и рисуем сцену шейдером, игнорирующим освещение

Shadow volume



Shadow volume



Shadow volumes (Doom 3)

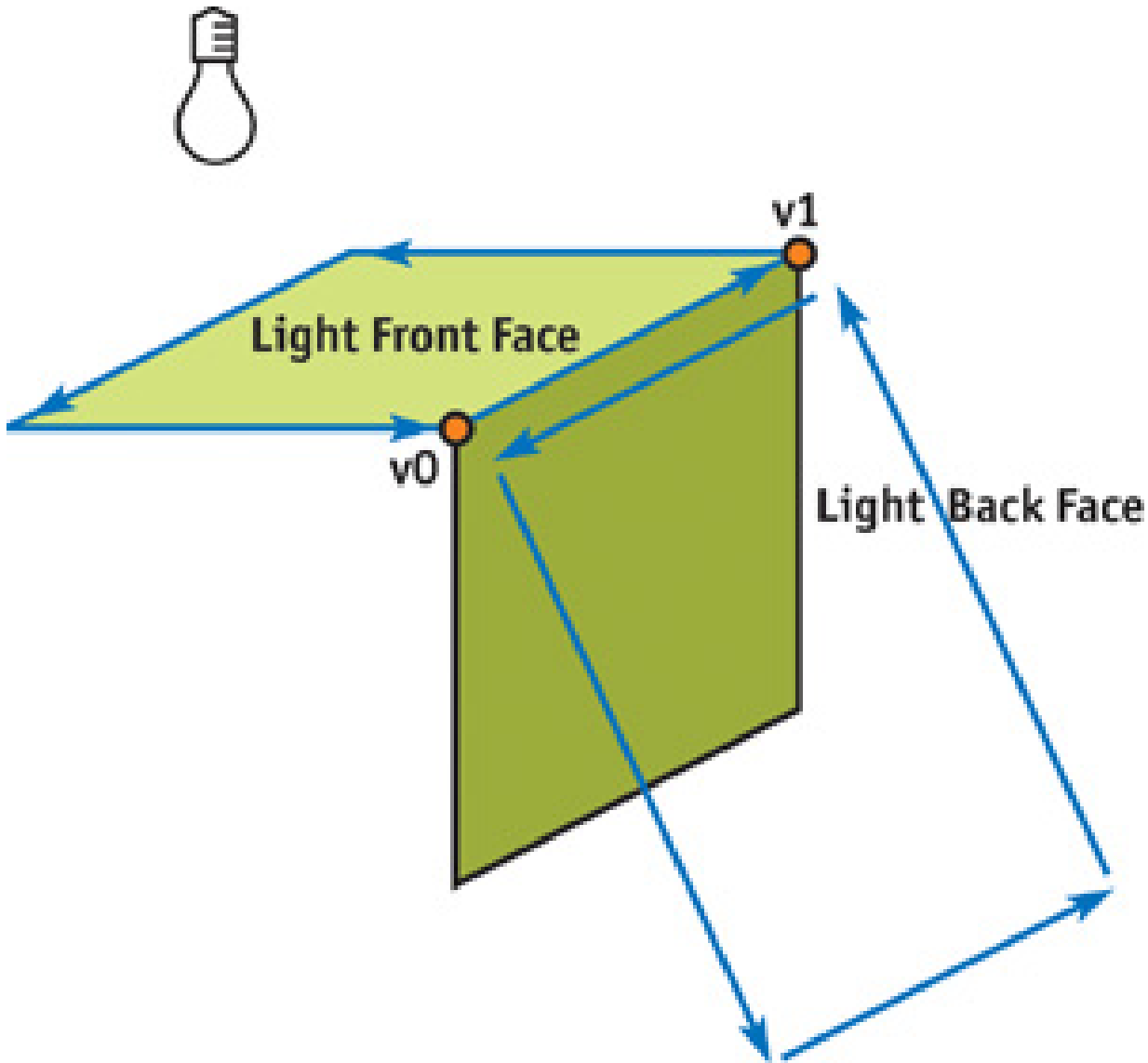
Shadow volumes: как вычислять shadow volume?

- Шаг 1: определить shadow edges - рёбра модели, у которых один соседний треугольник обращён к источнику света ($\vec{n} \cdot \vec{r} > 0$), а второй - против источника света ($\vec{n} \cdot \vec{r} \leq 0$)

Shadow volumes: как вычислять shadow volume?

- Шаг 1: определить shadow edges - рёбра модели, у которых один соседний треугольник обращён к источнику света ($\vec{n} \cdot \vec{r} > 0$), а второй - против источника света ($\vec{n} \cdot \vec{r} \leq 0$)
- Шаг 2: для каждого shadow edge построить четырёхугольную грань shadow volume - к точкам ребра p_1, p_2 добавлются точки $p_1 + D \cdot \frac{p_1 - o}{|p_1 - o|}$ и аналогично для p_2
  - o - координаты источника света
  - D - расстояние, до которого отбрасывается тень (в идеале мы хотим $D \to \infty$, для этого можно воспользоваться однородными координатами)

Shadow volumes (Doom 3)



Shadow volumes: как вычислять shadow volume?

- Два варианта: на CPU или на GPU

Shadow volumes: как вычислять shadow volume?

- Два варианта: на CPU или на GPU
- На CPU: очень дорого (положение модели и источника света могут меняться каждый кадр)

Shadow volumes: как вычислять shadow volume?

- Два варианта: на CPU или на GPU
- На CPU: очень дорого (положение модели и источника света могут меняться каждый кадр)
- На GPU: геометрические шейдеры!
  - Нужен тип примитива *lines adjacency*, позволяющий передать 4 вершины как один примитив (две вершины ребра + две вершины соседних треугольников)

Shadow volumes: как вычислять shadow volume?

- Два варианта: на CPU или на GPU
- На CPU: очень дорого (положение модели и источника света могут меняться каждый кадр)
- На GPU: геометрические шейдеры!
  - Нужен тип примитива *lines adjacency*, позволяющий передать 4 вершины как один примитив (две вершины ребра + две вершины соседних треугольников)
  - Шейдер проверяет, является ли ребро shadow edge, и генерирует грань shadow volume

Shadow volumes

- \+ Pixel-perfect жёсткие тени
- — Алиасинг
- — Сложно получить мягкие тени
- — Требует правильной геометрии (без дырок и дублирования)
- — Не работает, если камера находится внутри тени (исправляется с помощью т.н. Carmack's Reverse)
- — Ресурсозатратен: даже для маленького объекта его shadow volume может занимать значительную часть сцены $\Rightarrow$ слишком много растеризации, чтения/записи памяти (stencil buffer)

Shadow volumes с плохой геометрией

Shadow volumes

- Thief 3
- Doom 3
- Quake 4
- Prey
- Far Cry
- F.E.A.R 1, 2, 3
- S.T.A.L.K.E.R.
- Timeshift
- и др.

Shadow volumes: ссылки

- en.wikipedia.org/wiki/Shadow_volume
- ogldev.org/www/tutorial40/tutorial40.html
- GPU Gems, Chapter 9. Efficient Shadow Volume Rendering