

Sycles

Интеграция Cycles в Softimage

Руководство «Что нажимать...»
17 сентября 2020 г.

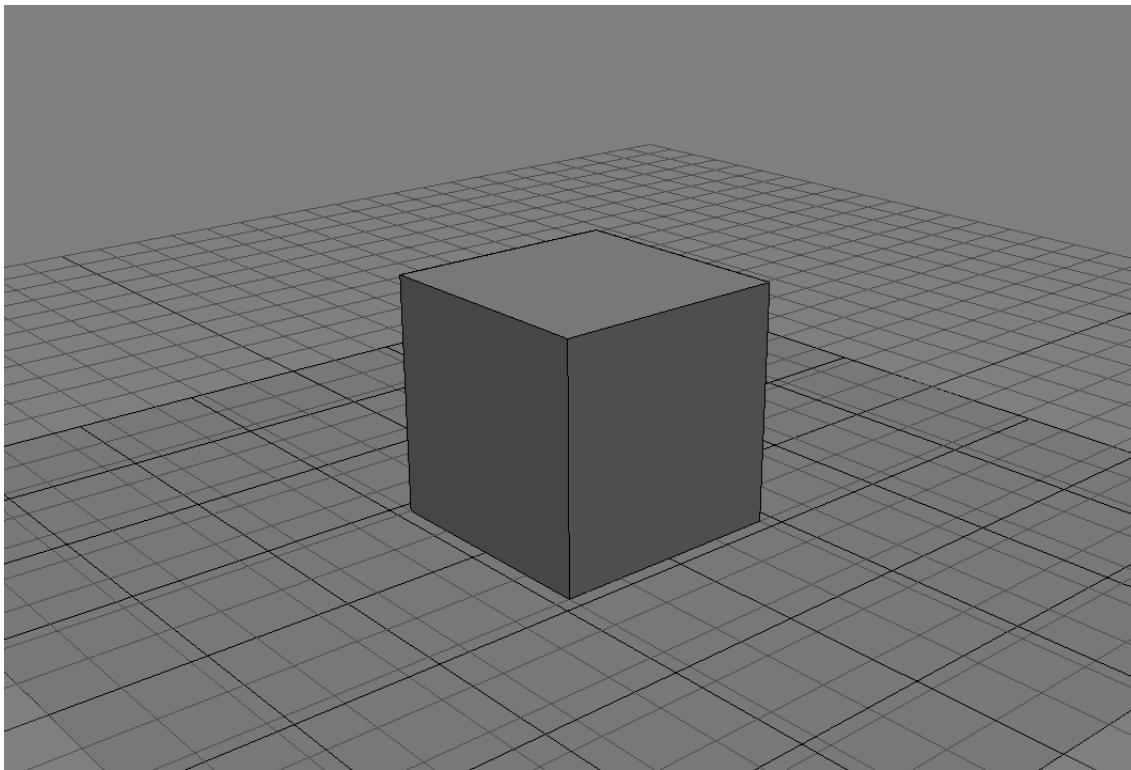
Содержание

1	Как начать рендерить	3
2	Как назначить шейдер	6
3	Как использовать небо и солнце для окружения	8
4	Как использовать HDR для окружения	12
5	Как использовать Bump	13
6	Как сделать настоящий Displacement	16
7	Как использовать Normal Map	18
8	Как рендерить волосы	19
9	Как рендерить DOF	21
10	Как рендерить Object ID pass	23
11	Как рендерить ICE-атрибуты	27
12	Как рендерить subdivise поверхности	29
13	Как рендерить ICE-инстансы	33
14	Как рендерить Vertex Color	36
15	Как сохранить результат в Multi-Layer EXR	38
16	Как использовать Stamp Output	41
17	Как использовать Cache	43
18	Как использовать цветовые профили	46
19	Как использовать shaderball-ы	49
20	Как использовать OSL-шейдеры	53
21	Как рендерить стандартные XSI-шные волосы	56
22	Как использовать видимость объектов	60
23	Как использовать несколько uv-координат	63
24	Что такое Camera Cull и Distance Cull	66
25	Как рендерить motion blur	68
26	Как использовать VDB Primitive	72
27	Как рендерить OpenVDB	75

28 Как рендерить emFluid	77
29 Как рендерить Explosia FX	79
30 Как рендерить и использовать Cryptomatte пассы	81
31 Как рендерить AOV-ы	85

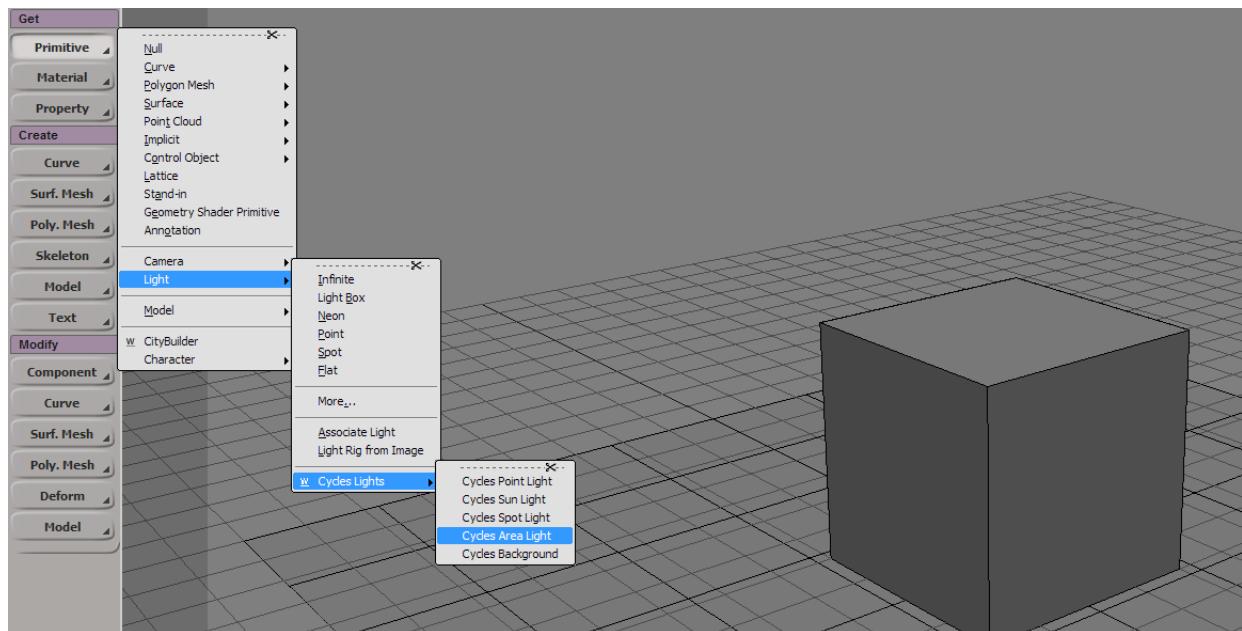
1 Как начать рендерить

Предположим у нас есть сцена: куб и плоскость.

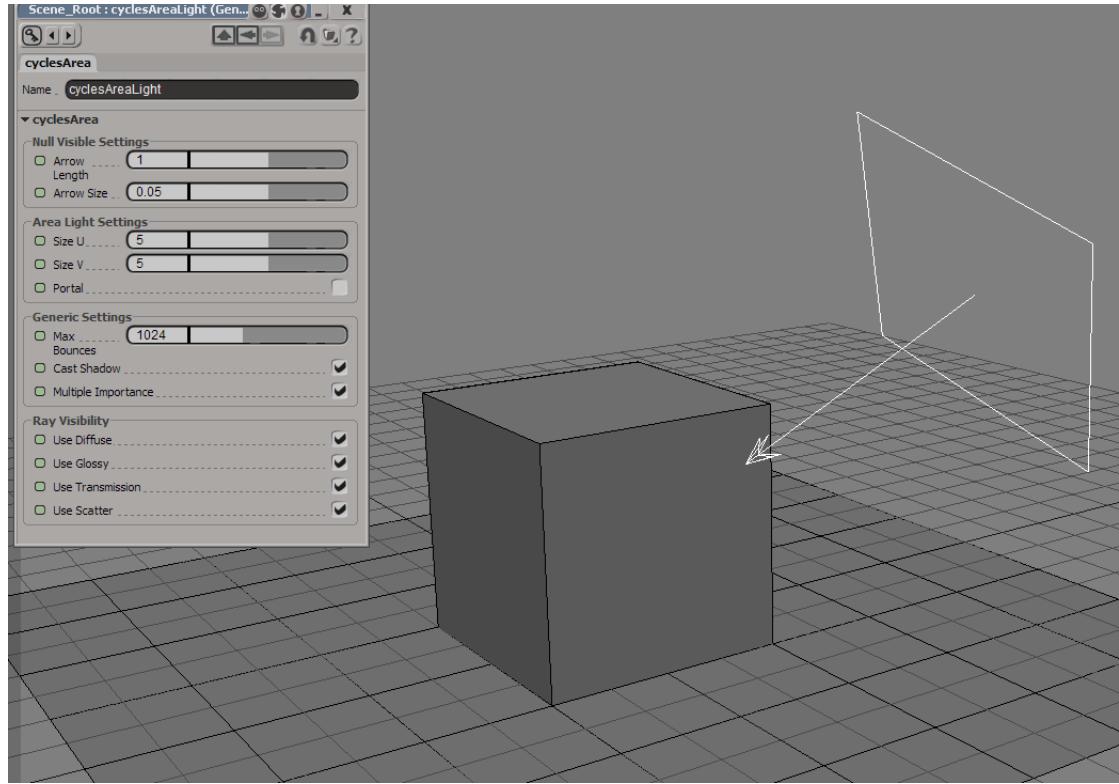


Добавляем прямоугольный источник света, выбирая

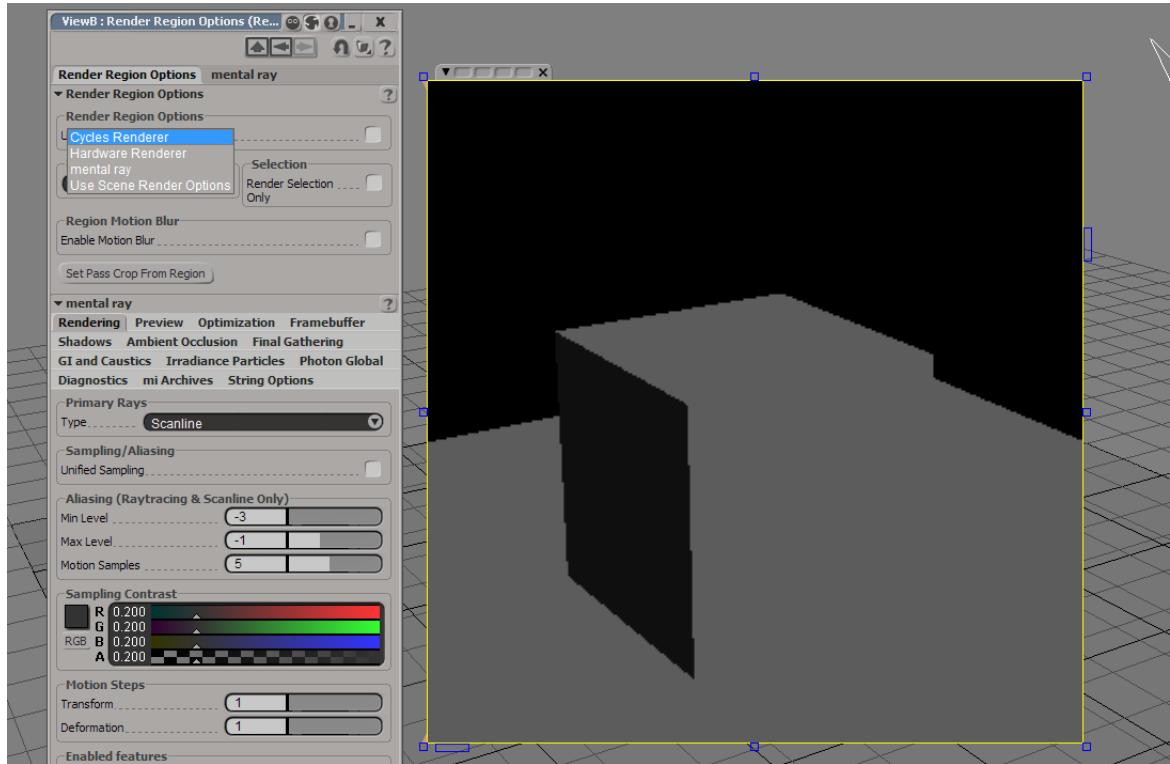
Primitive – Light – Cycles Lights – Cycles Area Light



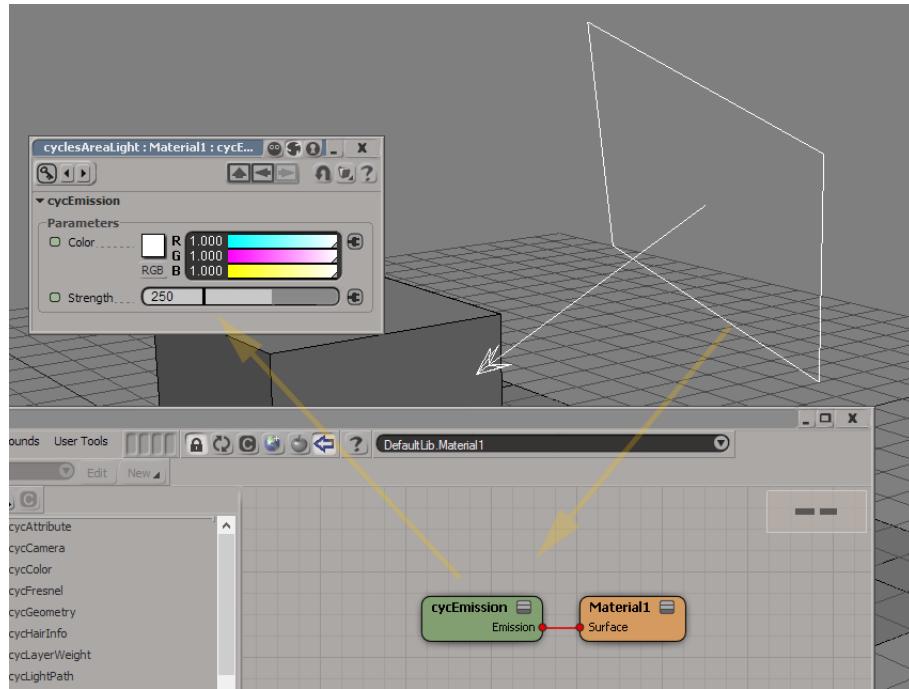
Выставляем у него размер 5x5.



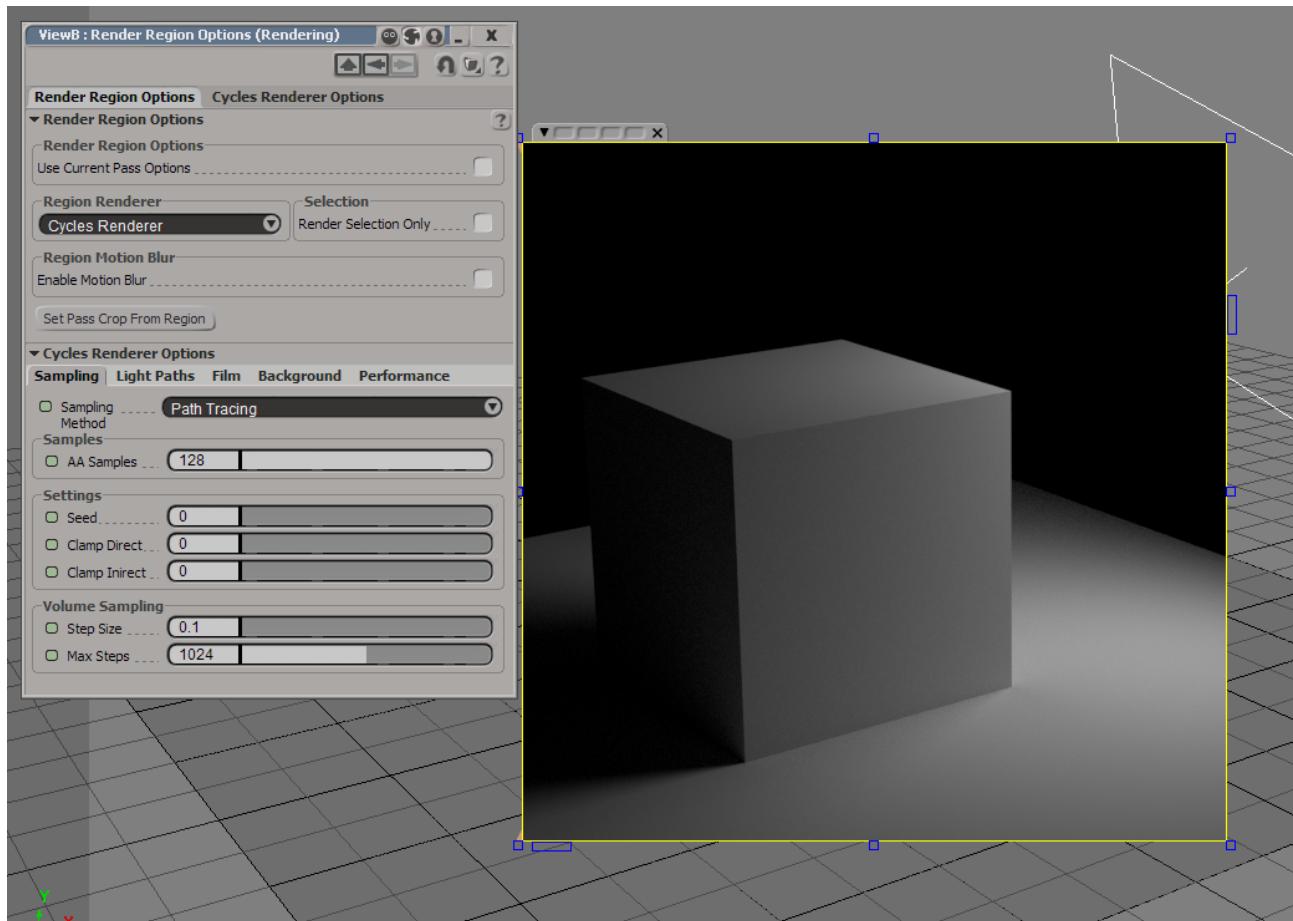
Потом выбираем область рендера и в настройках указываем Cycles Renderer.



Ничего не видно. Это потому что сила источника света слишком маленькая. Заходим в Render Tree источника света и задаём значение Strength ноды Emission равной 250.

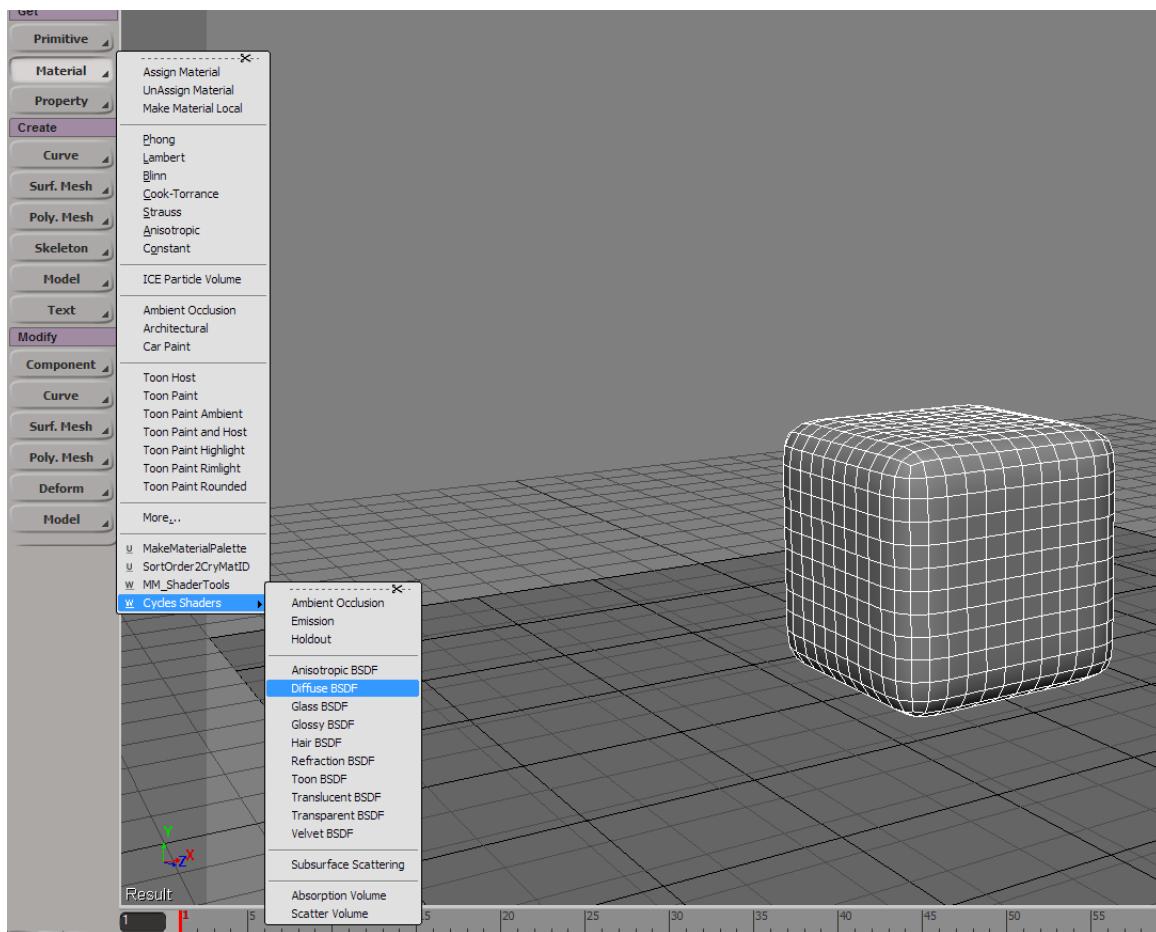


Получаем результат.

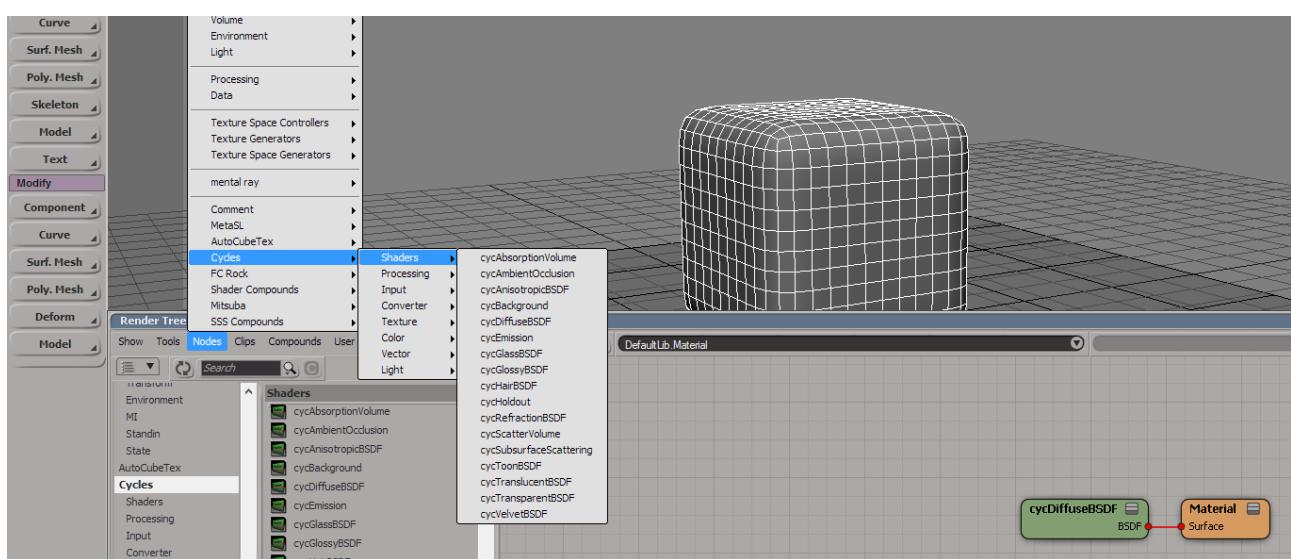


2 Как назначить шейдер

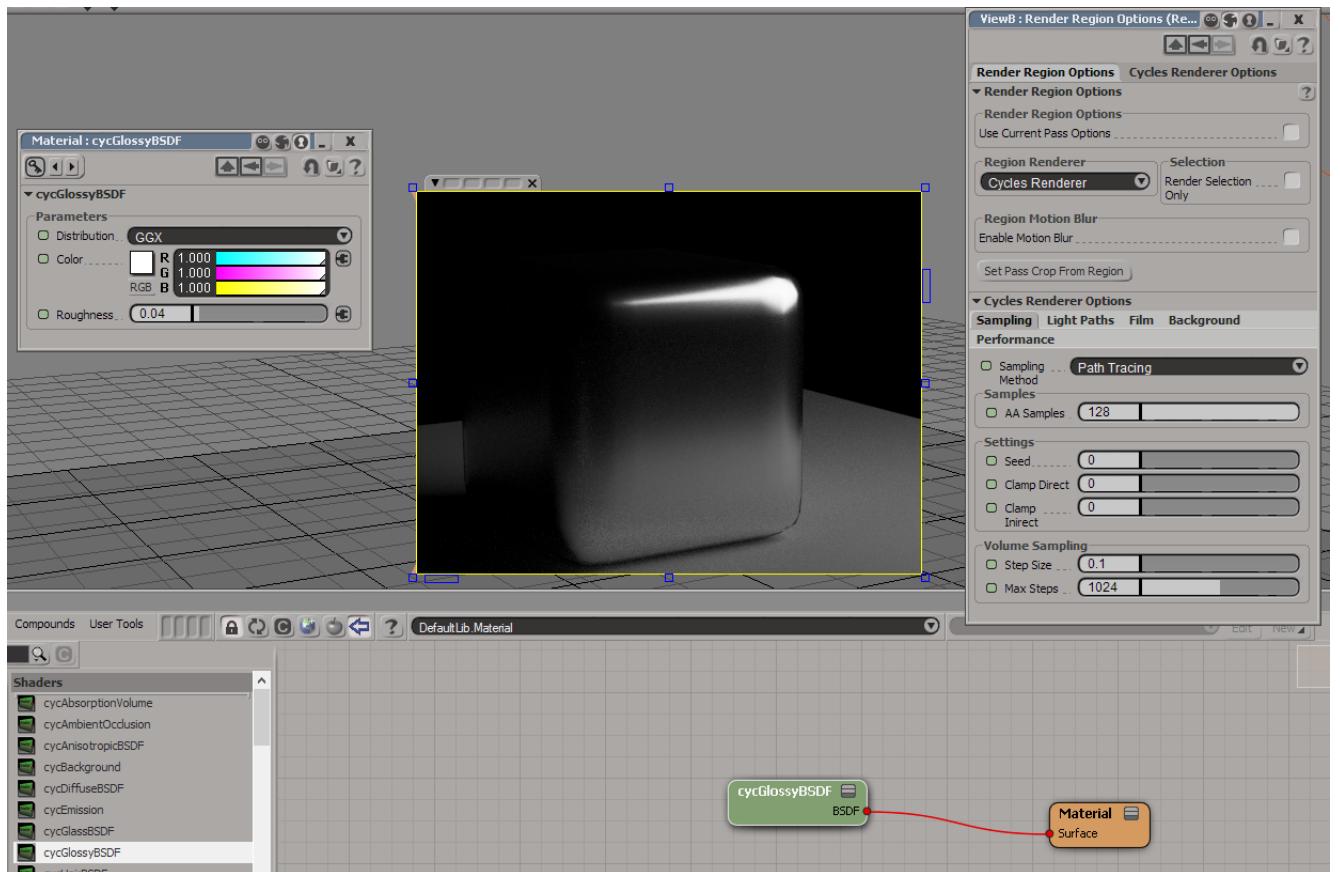
Два способа. Можно через команду **Material – Cycles Shaders** выбрать какой шейдер назначить на выделенный объект.



Либо непосредственно в **Render Tree** подключать ноды. Всё, что понимает Cycles, находится в отдельном разделе.

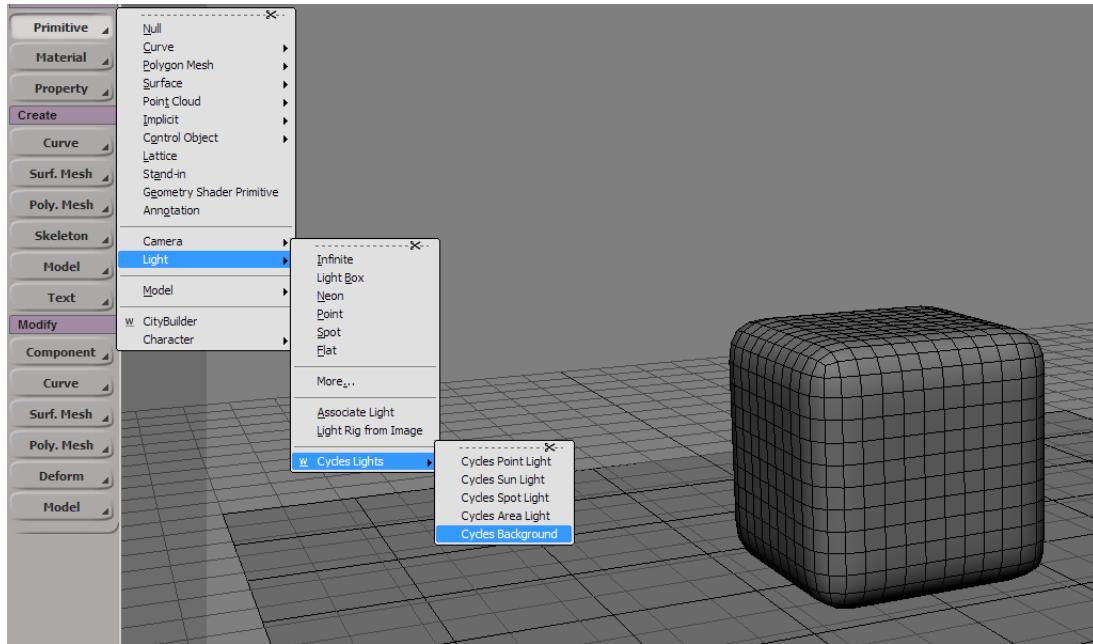


В качестве примера подсоединим ноду **GlossyBSDF** к порту **Surface** корневой ноды материала. Получим результат.

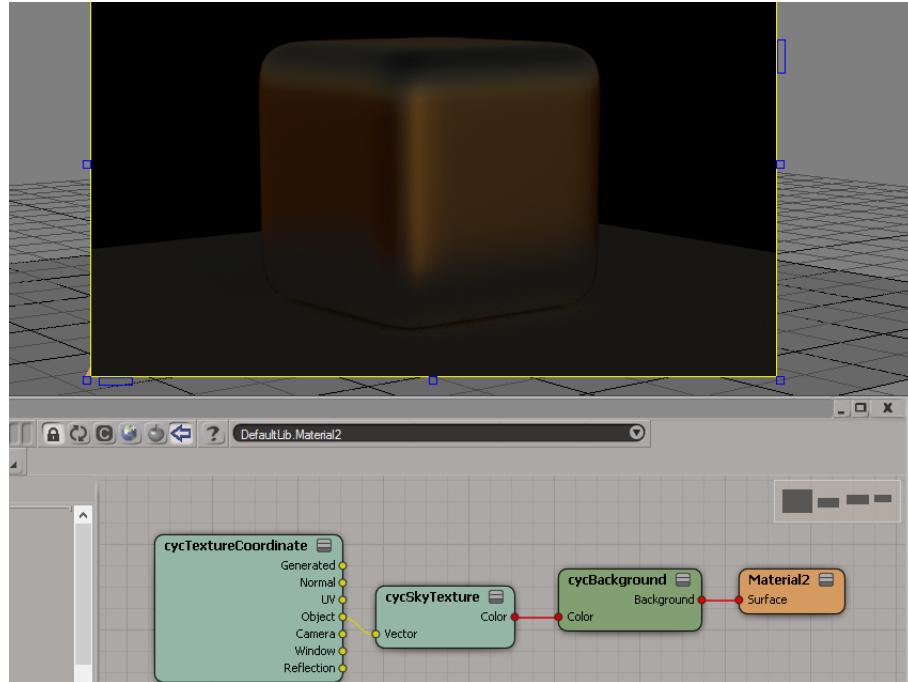


3 Как использовать небо и солнце для окружения

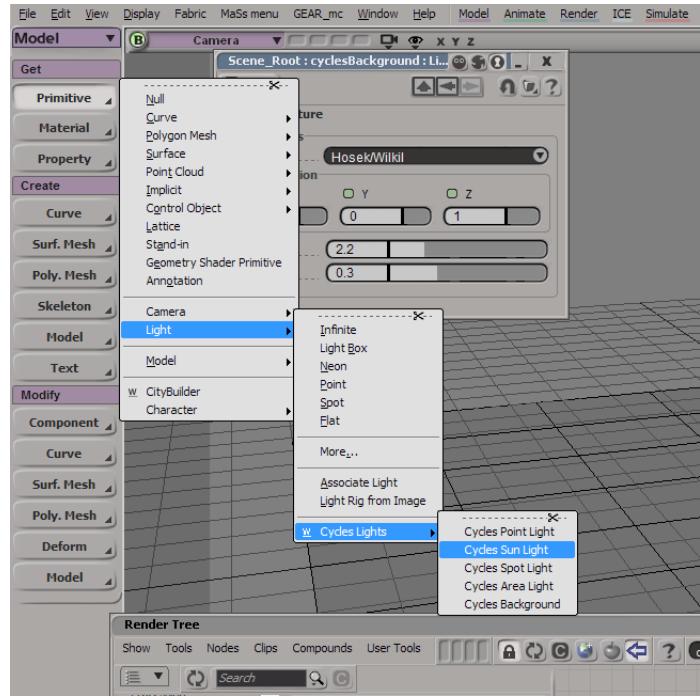
Предположим у нас есть сцена: куб и плоскость. Добавляем источник света типа Background.



В шейдер этого источника света кидаем ноду `SkyTexture`. Подключаем её к порту `Color` ноды `Background`. В порт `Vector` ноды `SkyTexture` подключаем выход `Object` ноды `TextureCoordinate`. Это надо для того, чтобы явно сказать рендеру, какие текстурные координаты для окружения использовать. Рендерим.

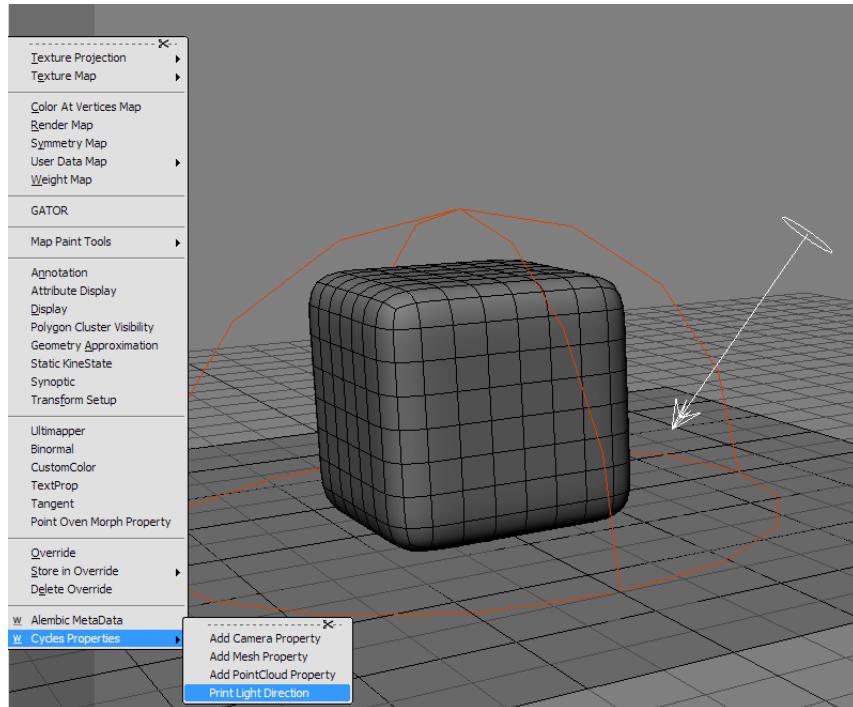


Что-то не то показывает. Это из-за того, что вектор для `Sun Direction` имеет значение $(0, 0, 1)$. То есть как будто солнце светит горизонтально. Это надо исправить. Добавляем источник света `Cycles Sun Light`.

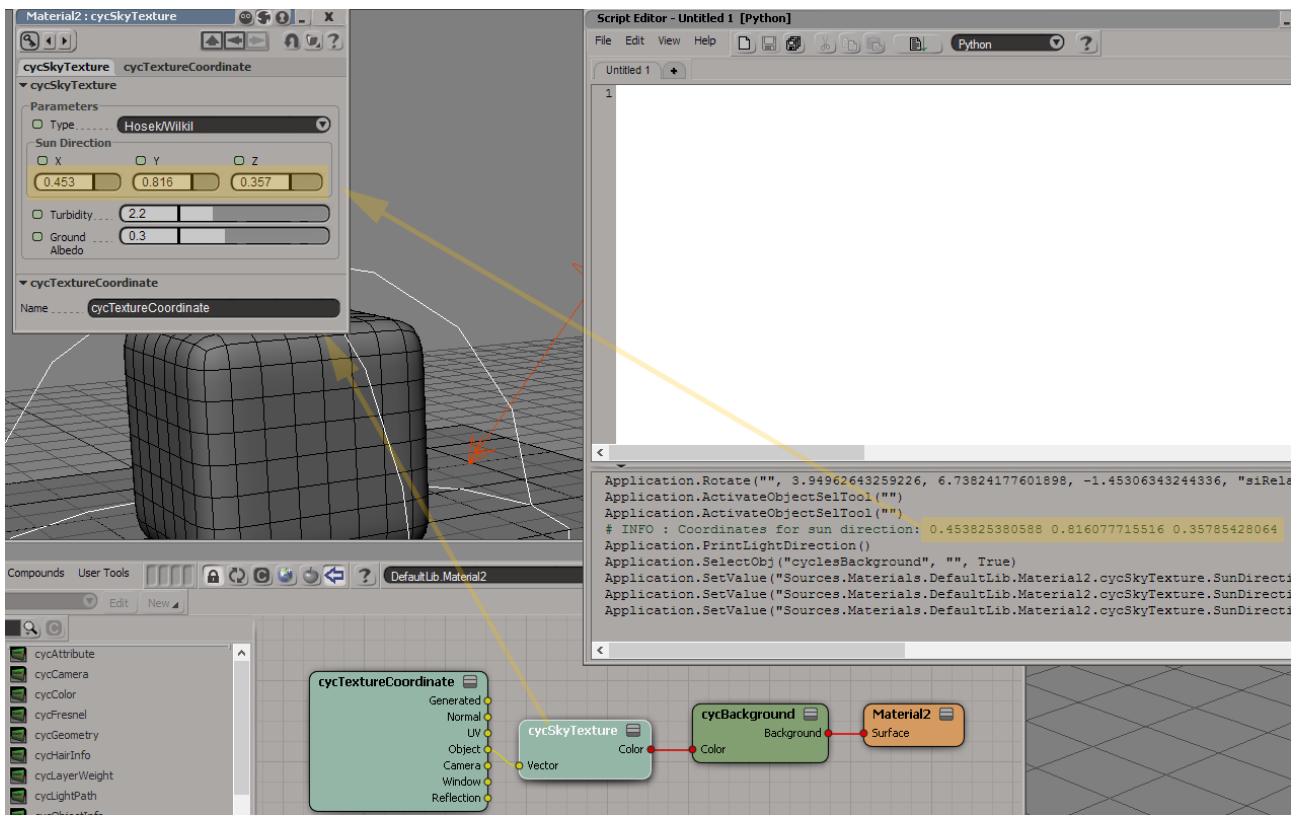


Дальше надо узнать его направление. Для этого с выделенным солнцем выбираем пункт

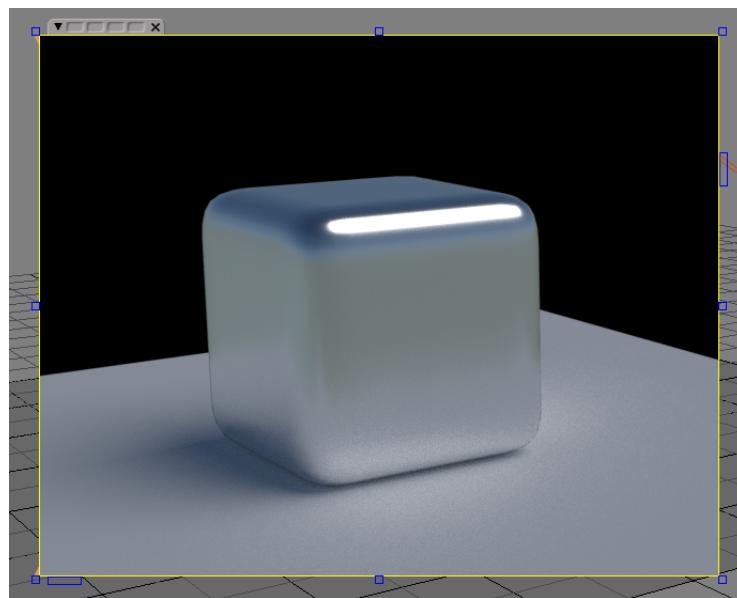
Property – CyclesProperties – Print Light Direction.



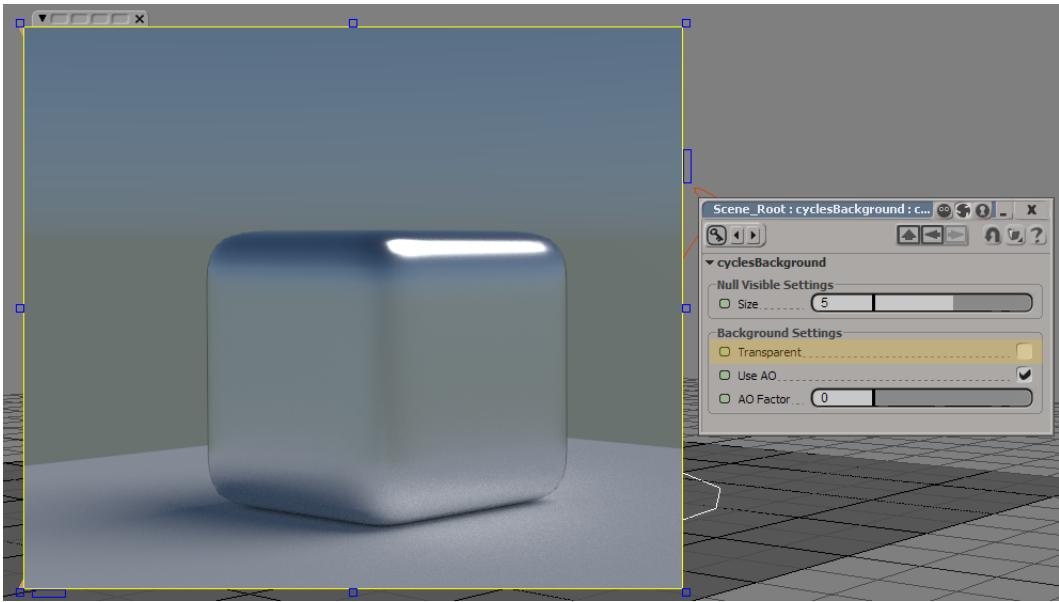
В лог выводятся три числа, которые и надо вставить в показания направления солнца в node SkyTexture.



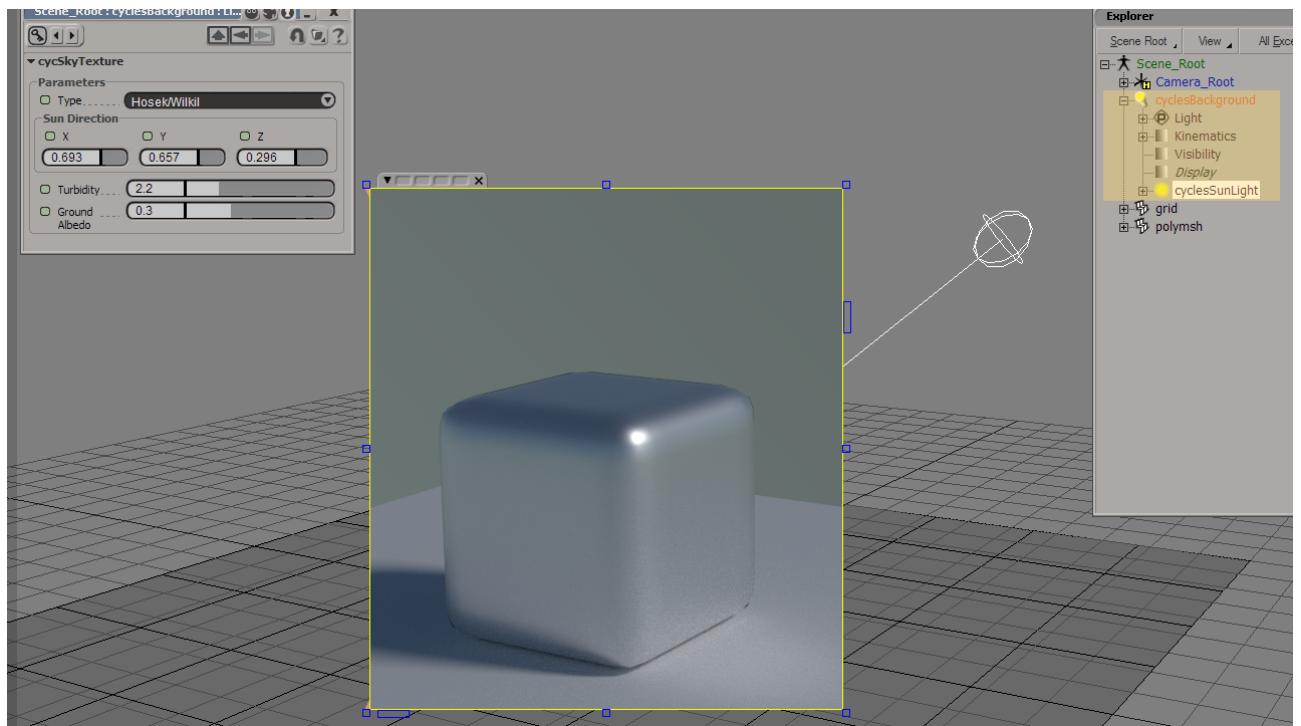
Рендерим. Теперь цвет неба и направление солнца согласованы.



Да, чтобы увидеть собственно цвет неба надо в свойствах источника света типа Background снять птицу с параметра Transparent.

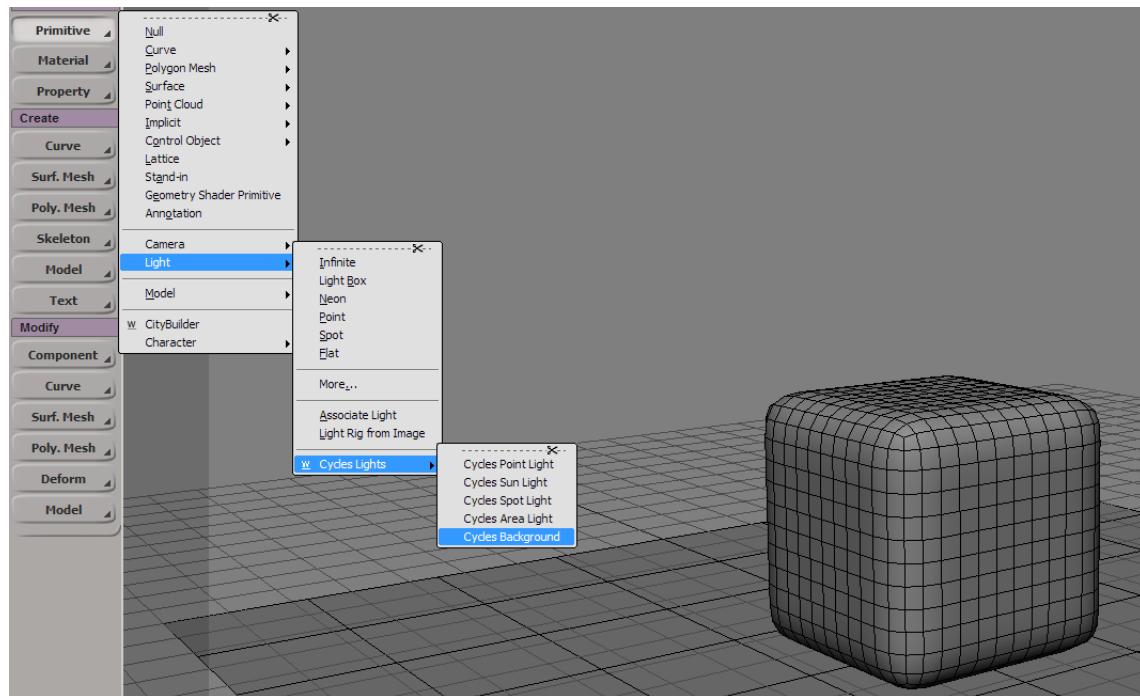


Возможно кто-то скажет: “Да что же это такое! Каждый раз после сдвига солнца вручную переписывать значения направления?!” Нет, ответим мы. Если солнце сделать дочерним к источнику-фону, то рендер будет игнорировать значения в полях **Sun Direction**, и при этом будет использовать значения поворота первого попавшегося дочернего объекта, который имеет тип солнца.

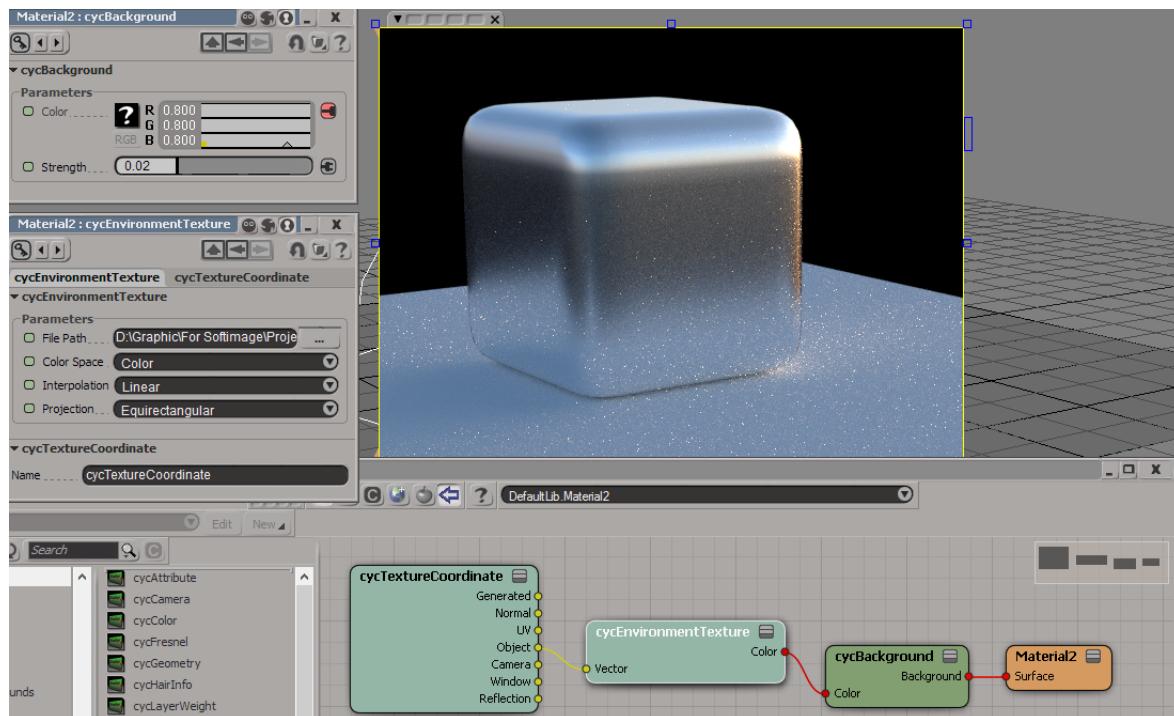


4 Как использовать HDR для окружения

Предположим у нас есть сцена: куб и плоскость. Добавляем источник освещения типа Cycles Background.

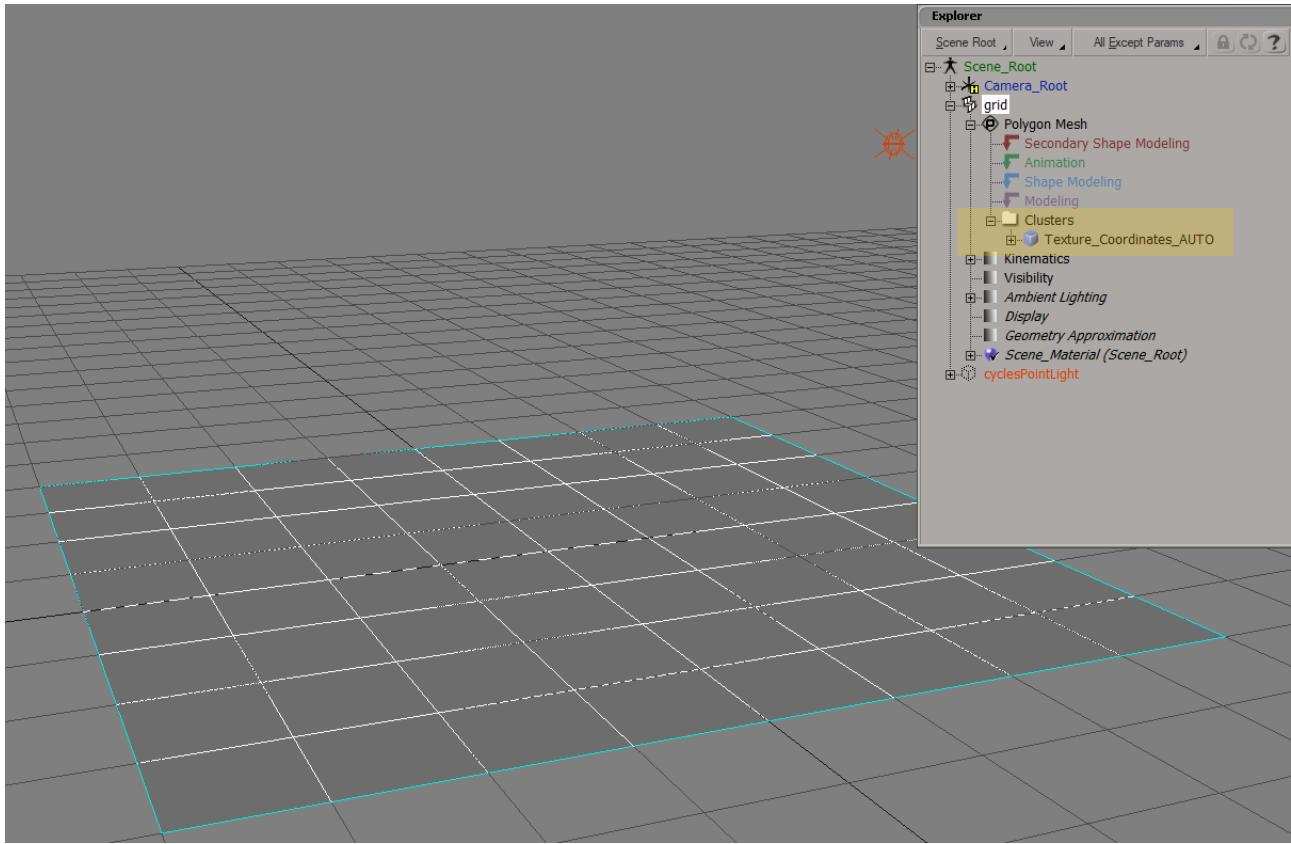


Кидаем в шейдер ноду `EnvironmentTexture` и подсоединяем её к порту `Color` ноды `Background`. В порт `Vector` подсоединянем выход ноды `TextureCoordinates`. Далее выбираем в параметрах ноды `EnvironmentTexture` какую-нибудь картинку, выставляем силу источника света и рендерим.

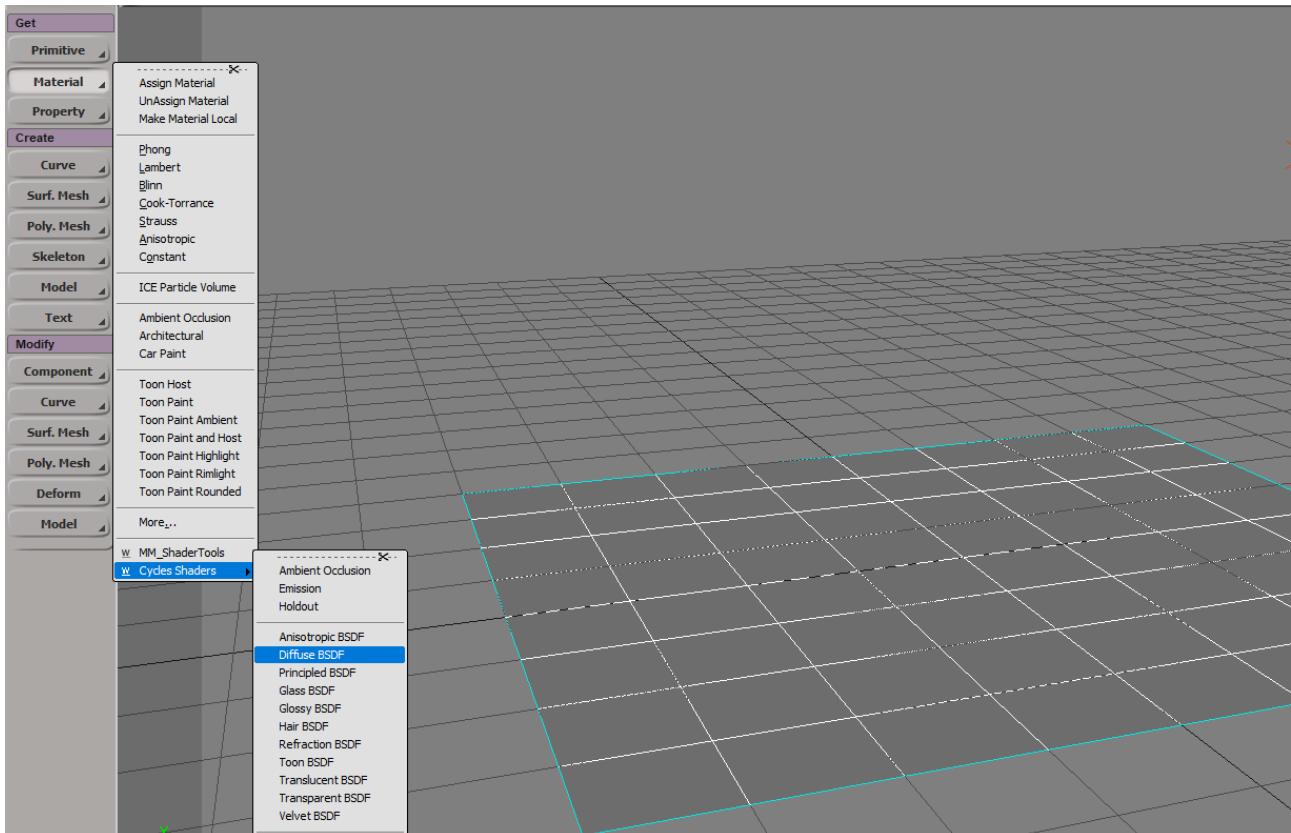


5 Как использовать Bump

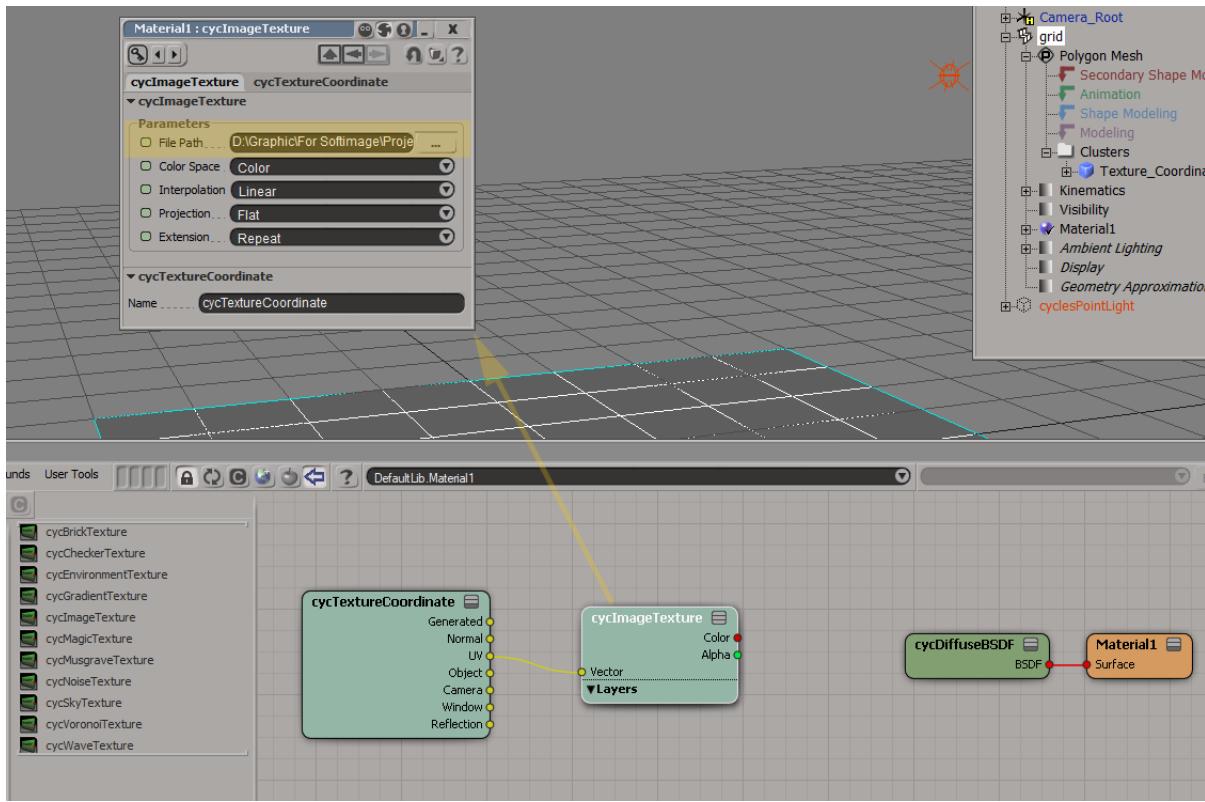
Предположим у нас есть сцена: плоскость и источник света. У плоскости есть текстурные координаты.



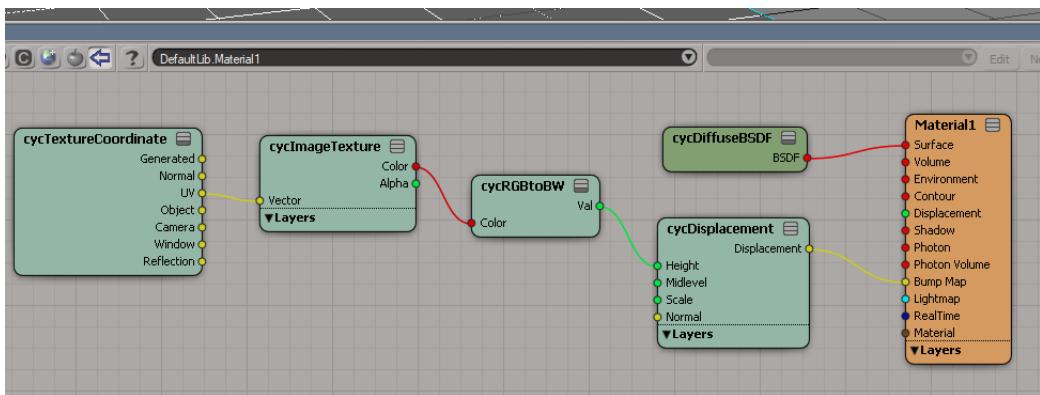
Назначаем плоскости шейдер DiffuseBSDF.



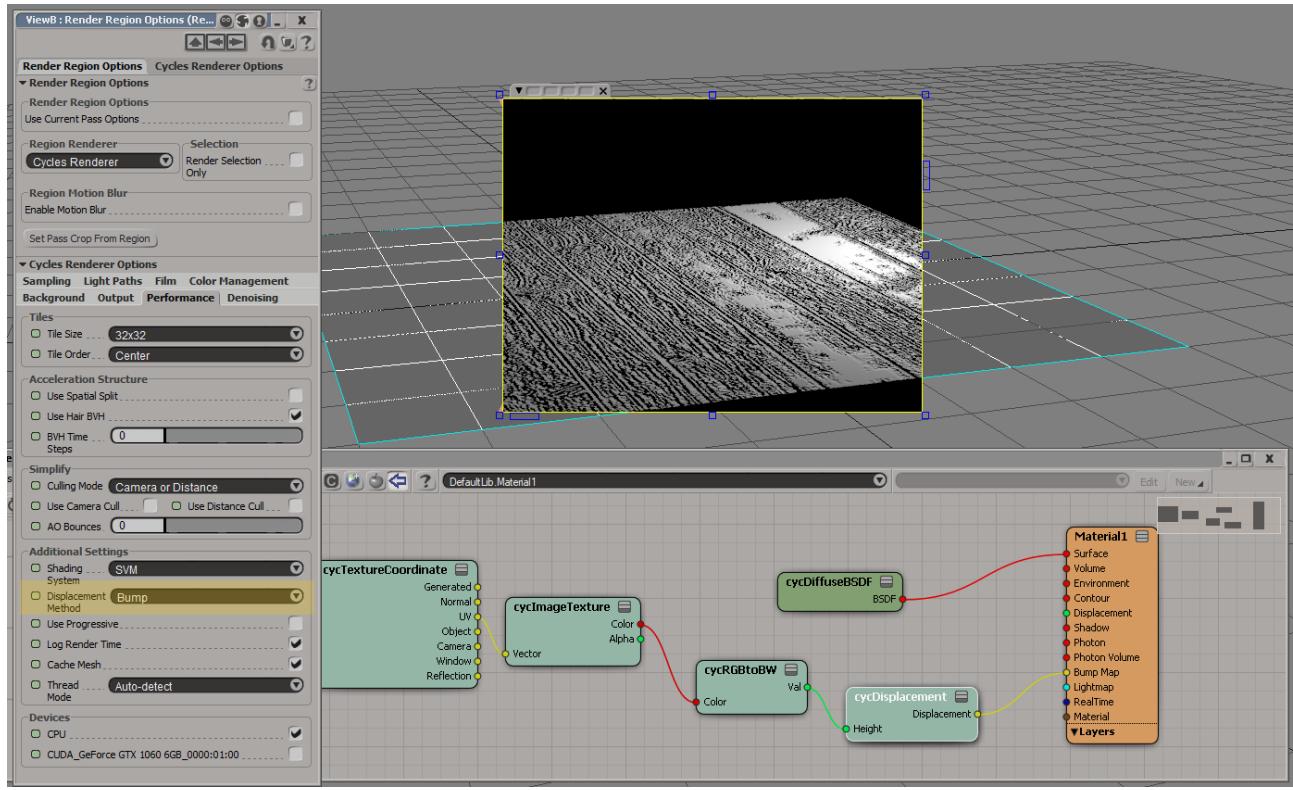
Потом кидаем в этот шейдер ноды **TextureCoordinate** и **ImageTexture**, соединяя выход **UV** первой ноды со входом **Vector** второй. В качестве текстурной карты выбираем картинку, которую хотим использовать в качестве bump-карты.



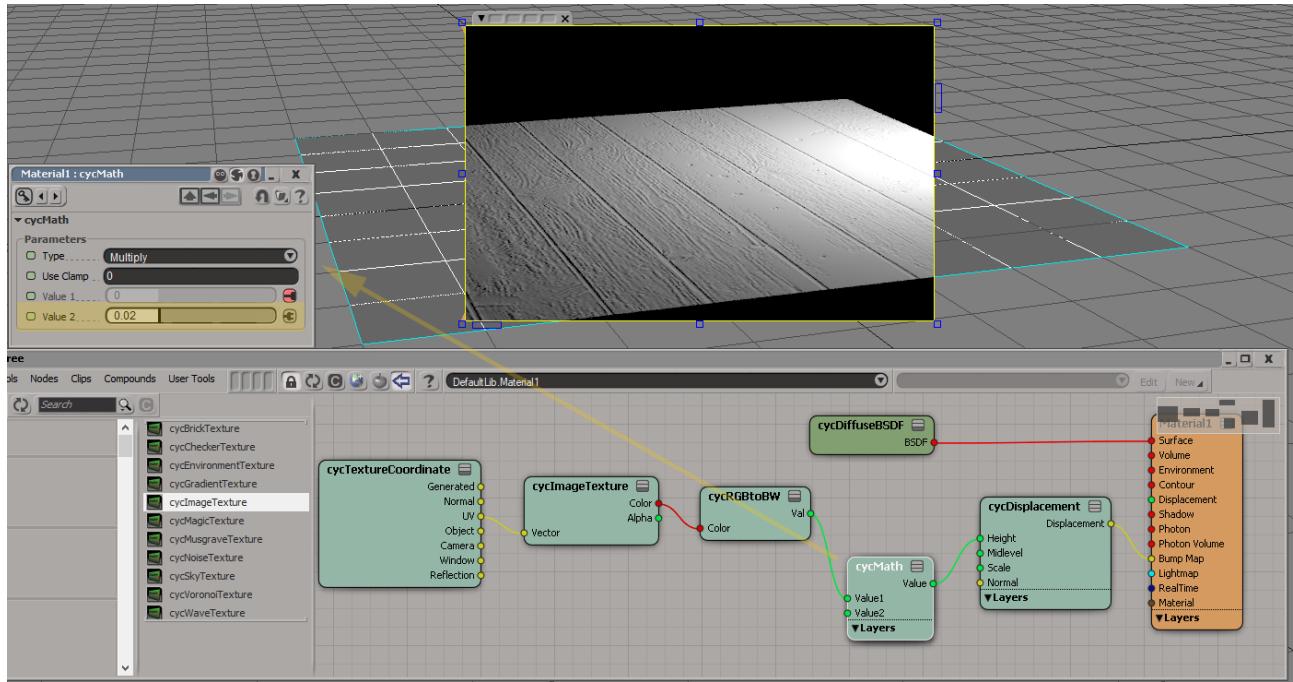
Конвертируем выход **Color** в чёрно-белое изображение с помощью ноды **RGBtoBW**, а результат подсоединяем ко входу **Height** ноды **Displacement** (вот это поворот!). Единственный выход этой ноды **Displacement** имеет векторный тип, поэтому подсоединяем его ко входу **Bump Map** корневой ноды материала. Это тоже единственный векторный порт, который у неё есть.



В настройках рендерера выбираем режим **Displacement Method** равный **Bump**. Рендерим.

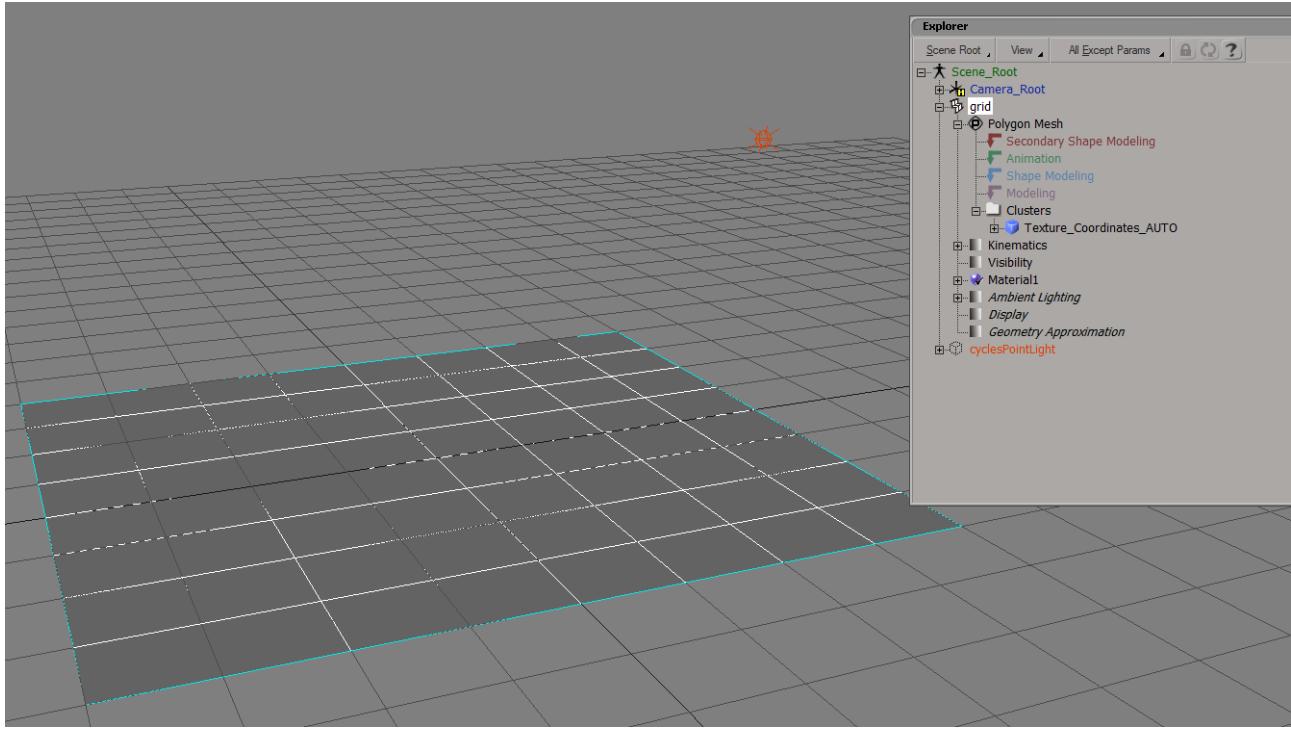


Для регулировки силы bump-а надо использовать ноду **Math**. Подключаем её непосредственно перед портом **Height** ноды **Displacement**, переключаем в режим **Multiply** и значение **Value 2** как раз будет отвечать за силу bump-а.

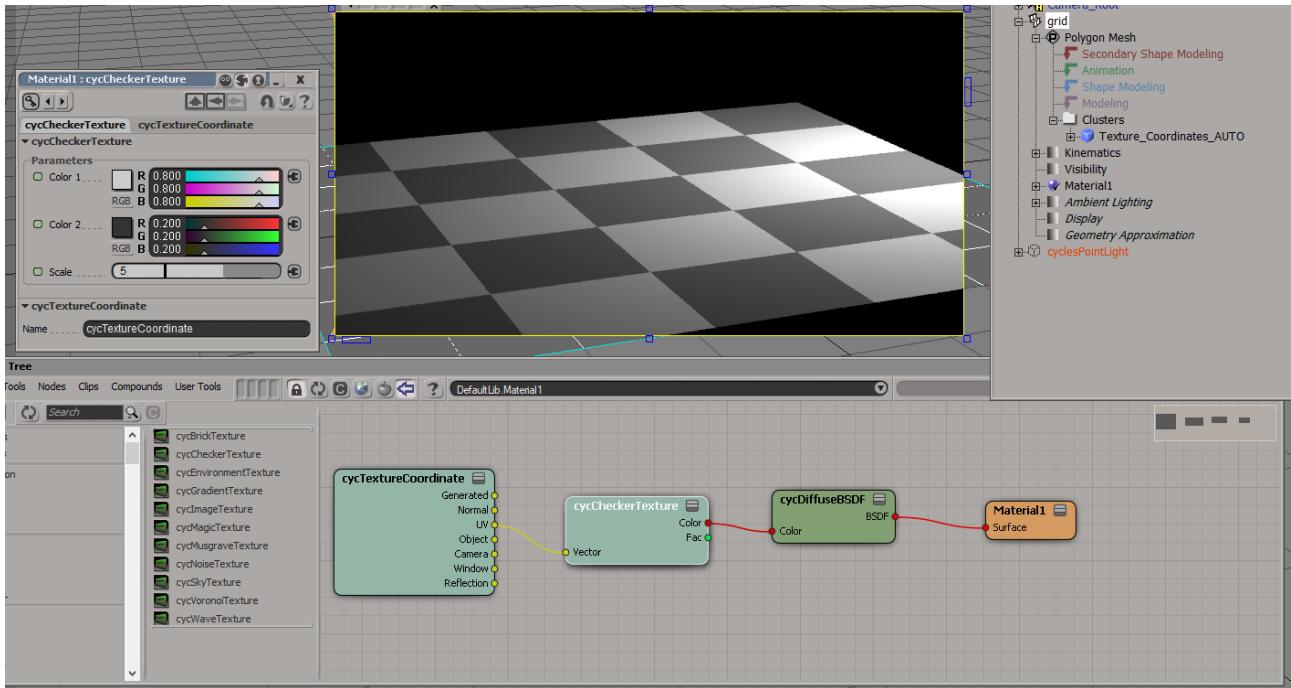


6 Как сделать настоящий Displacement

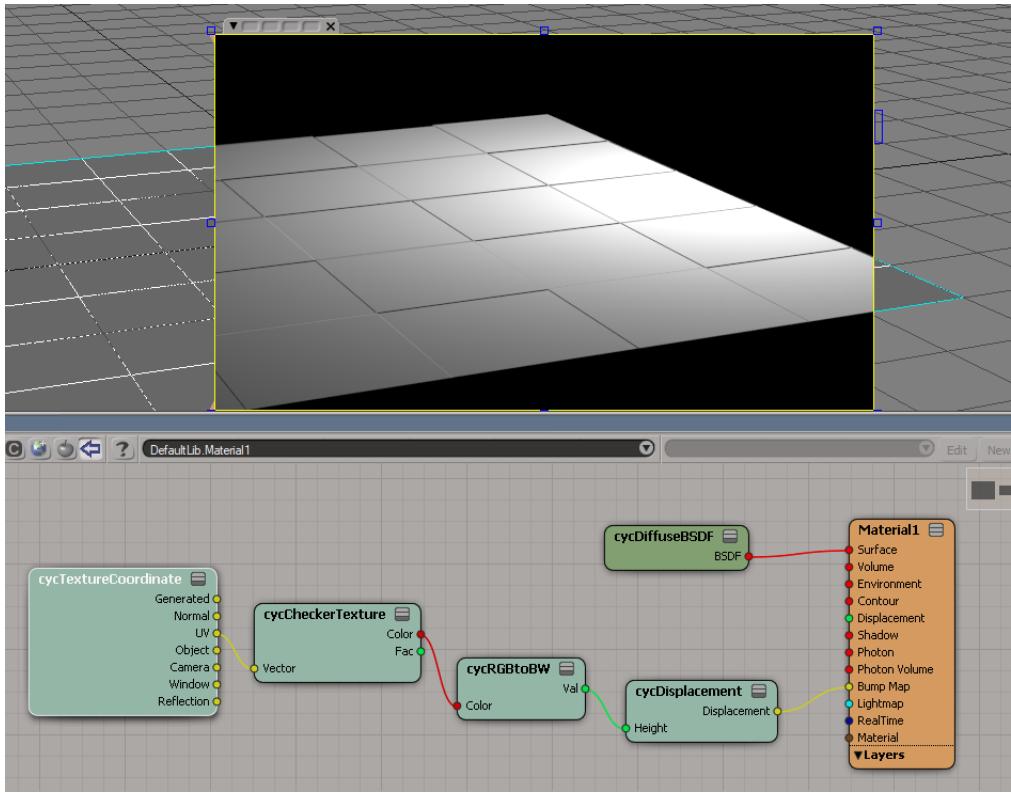
Предположим у нас есть сцена: плоскость и источник света. Плоскость с текстурными координатами.



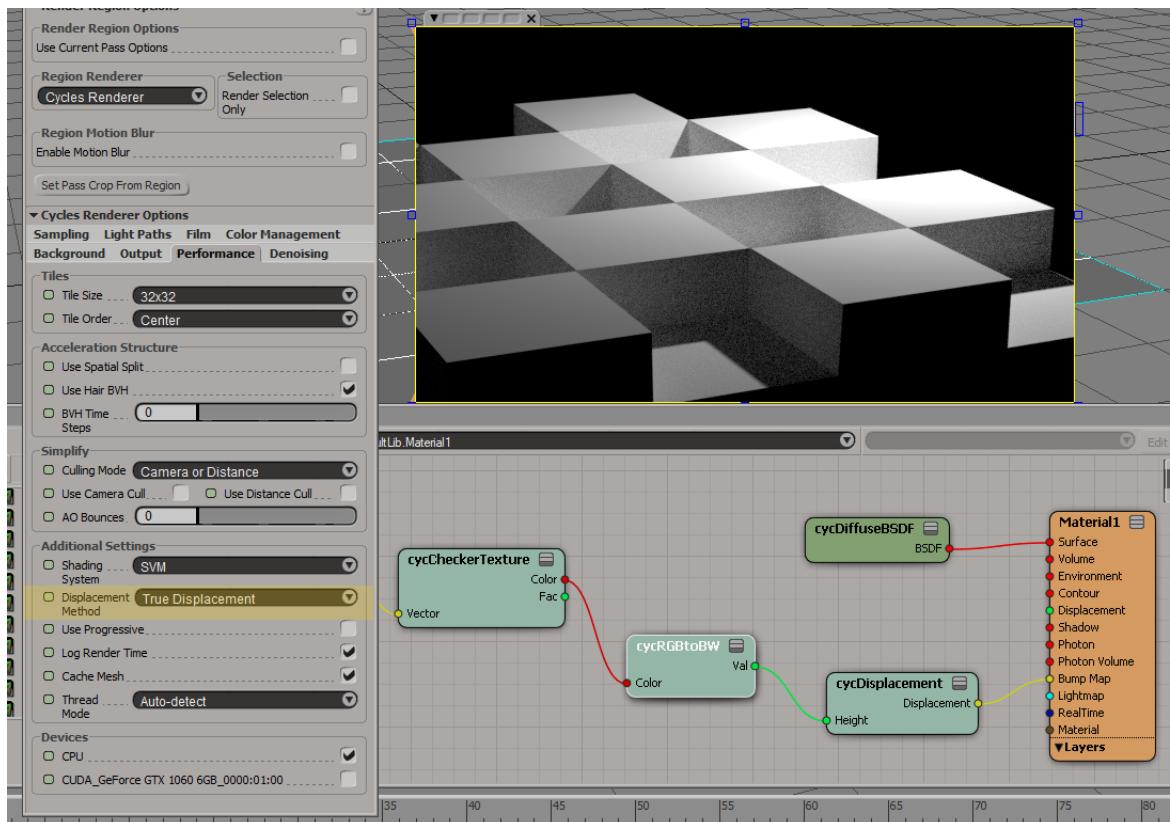
Назначаем плоскости шейдер DiffuseBSDF, в него кидаем ноду CheckerTexture и под соединяем её выход к порту Color ноды DiffuseBSDF. Чтобы текстура накладывалась с учётом текстурных координат, надо использовать ноду TextureCoordinate.



Подключаем выход Color к порту Height ноды Displacement, предварительно сконвертировав в чёрно-белый формат. Выход ноды Displacement подключаем к порту Bump Map корневой ноды материала. Рендерим и видим обычный bump.



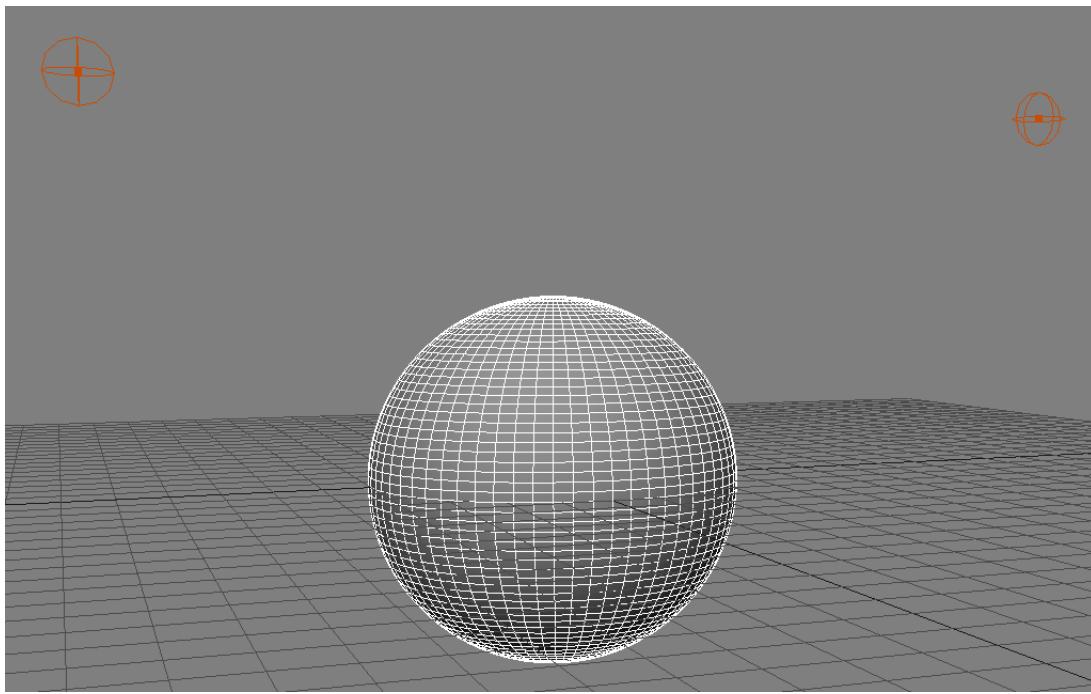
Но как же сделать displacement? Для этого надо зайти в настройки рендера и во вкладке **Performance** выбрать **True displacement** для параметра **Displacement Method**. Чтобы результат был корректным, надо увеличить число полигонов у плоскости.



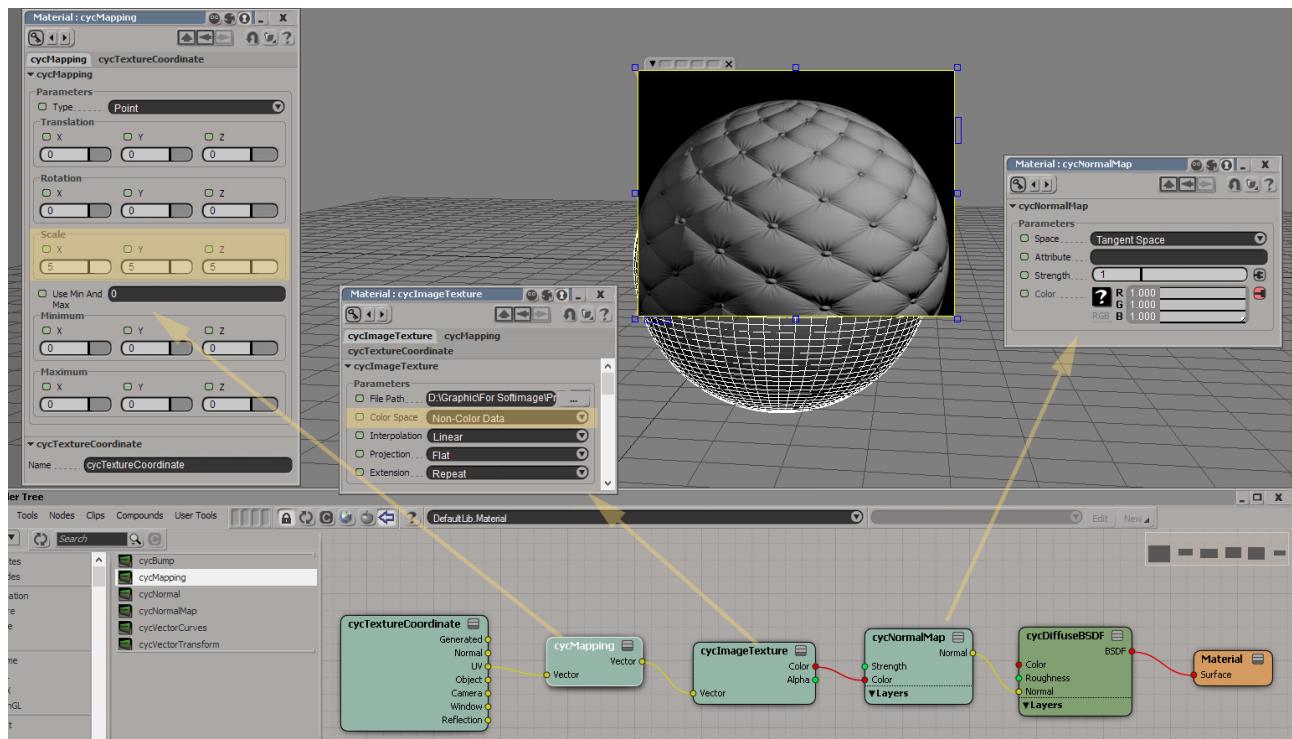
В большинстве случаев стоит использовать метод **Both Displacement**. В этом режиме геометрия деформируется в соответствии с текстурной картой, а в тех местах, где не хватает плотности полигонов, деформация применяется как bump map.

7 Как использовать Normal Map

Предположим у нас есть сцена: сфера и пара источников света. На сфере, конечно, есть текстурные координаты.

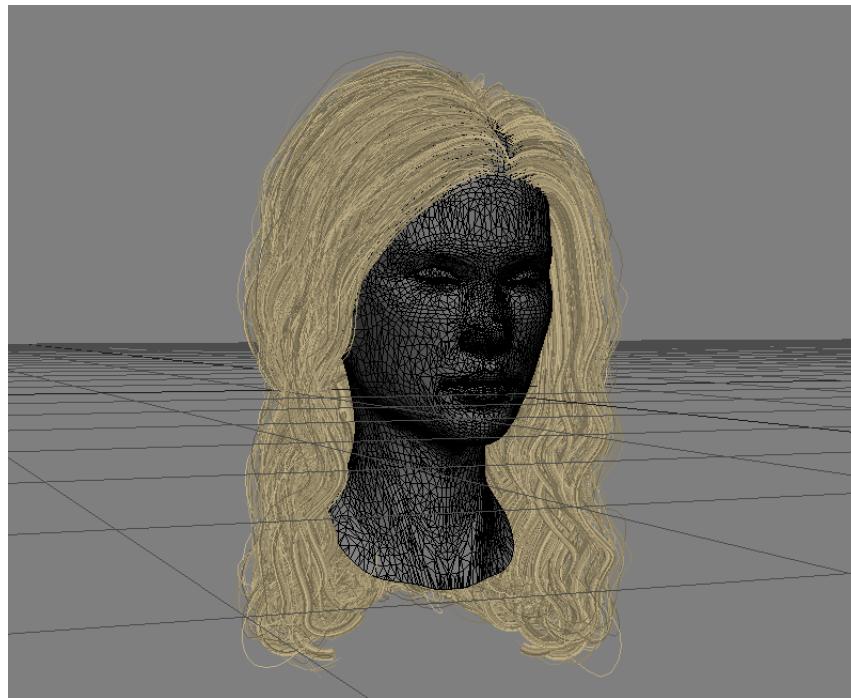


Кидаем в шейдер сферы ноду **NormalMap**. К ней в порт **Color** подсоединяем ноду **ImageTexture**. В node для текстуры выбираем собственно сам файл с картой нормалей и выбираем значение **Non-Color Data** для значения параметра **Color Space** (чтобы не происходила коррекция гаммы). Наконец указываем какие текстурные координаты надо использовать для карты нормалей. Соединение сделано через ноду **Mapping**, чтобы можно было указать пятикратный повтор карты.

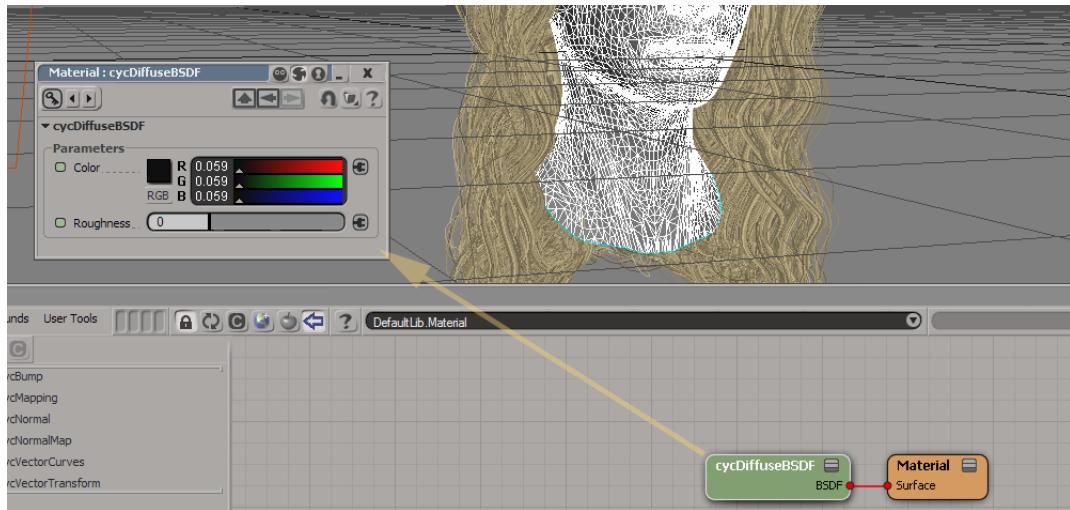


8 Как рендерить волосы

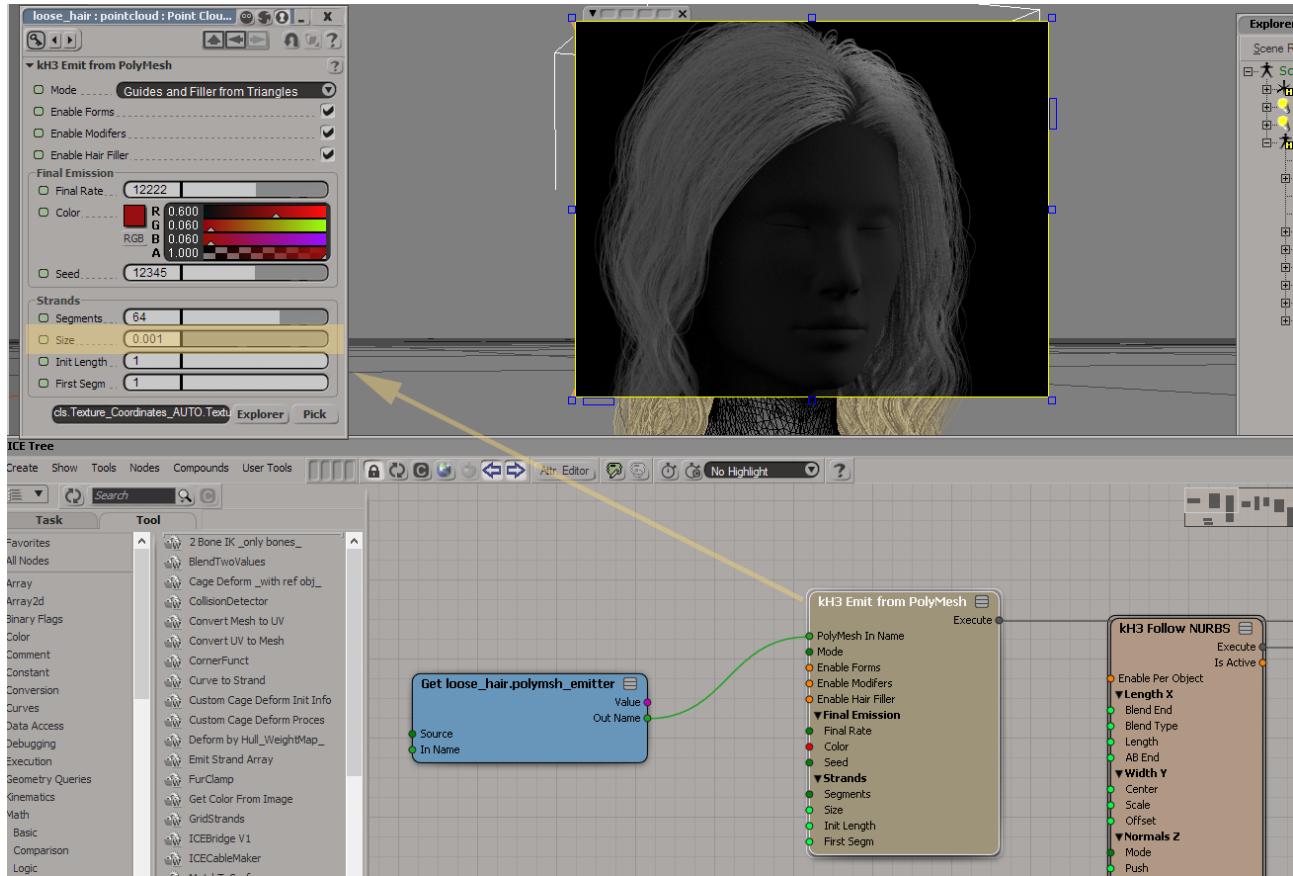
Предположим у нас есть голова с волосами, сделанными с помощью strand-ов.



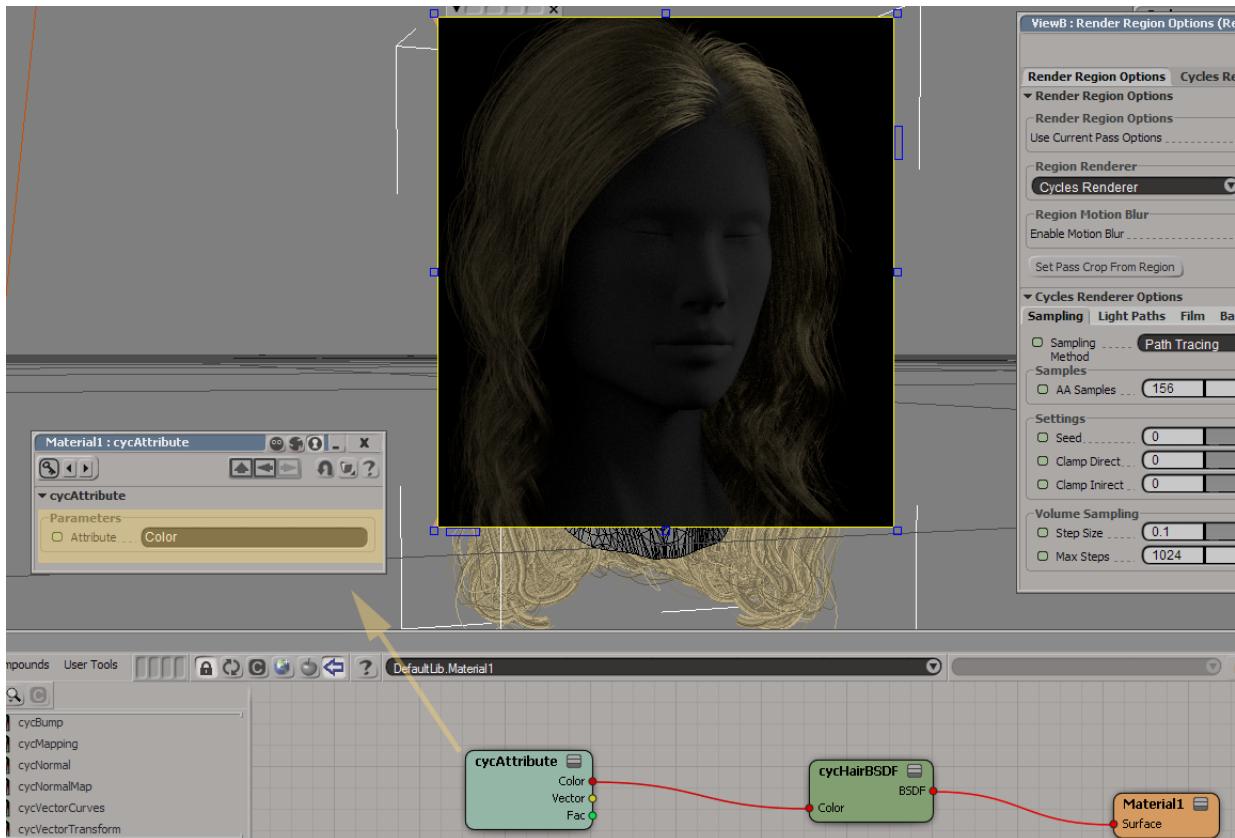
Чтобы голова не мешалась, назначаем ей тёмный материал.



Заходим в ICE-дерево для волос и указываем их толщину. В нашем случае задаем значение параметра **Size** = 0.001. Больше ничего не трогаем. Рендерим.

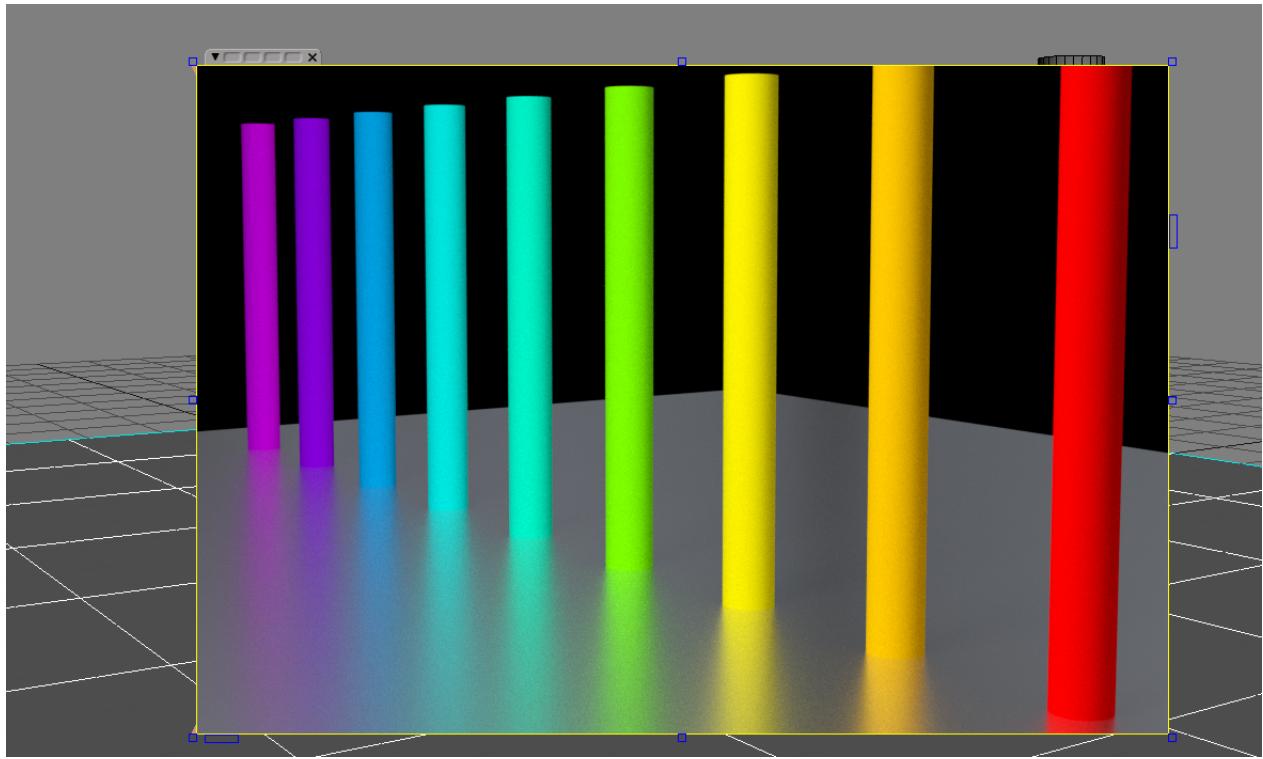


Сейчас на волосах стандартный **DiffuseBSDF** шейдер, и он никак не может понять какого цвета должны быть волосы. Назначаем шейдер **HairBSDF**, кидаем в него ноду **Attribute**. В ней указываем имя атрибута **Color** и подсоединяем выход **Color** к порту **Color** ноды **HairBSDF**. Вот теперь другое дело.



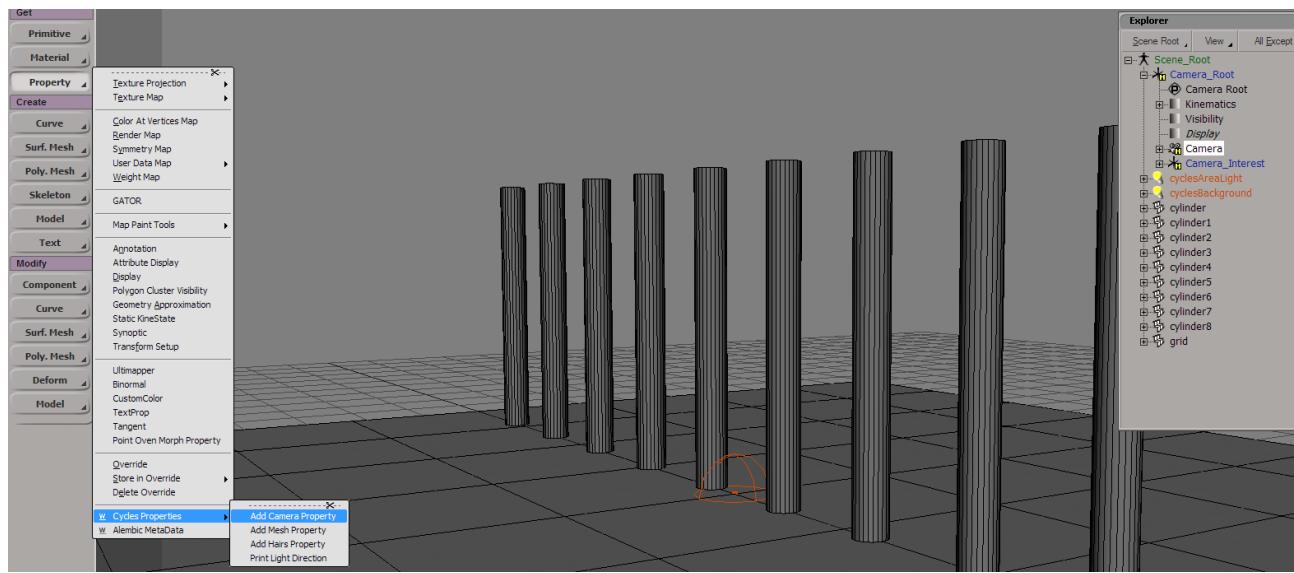
9 Как рендерить DOF

Предположим у нас есть сцена: плоскость и разноцветные столбики. Мы хотим, чтобы зелёный столбик был в фокусе, а остальные размыты.

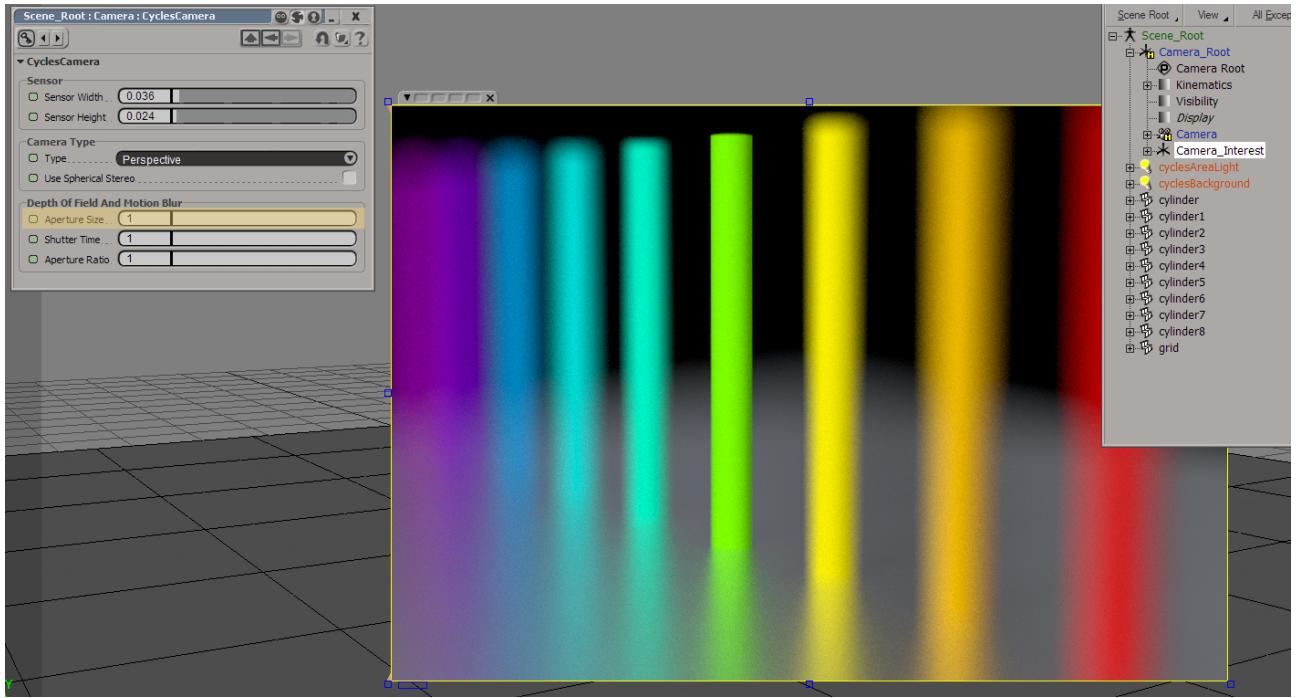


Выделяем камеру, из которой происходит рендер и добавляем на неё свойство командой

Property – Cycles Properties – Add Camera Property.

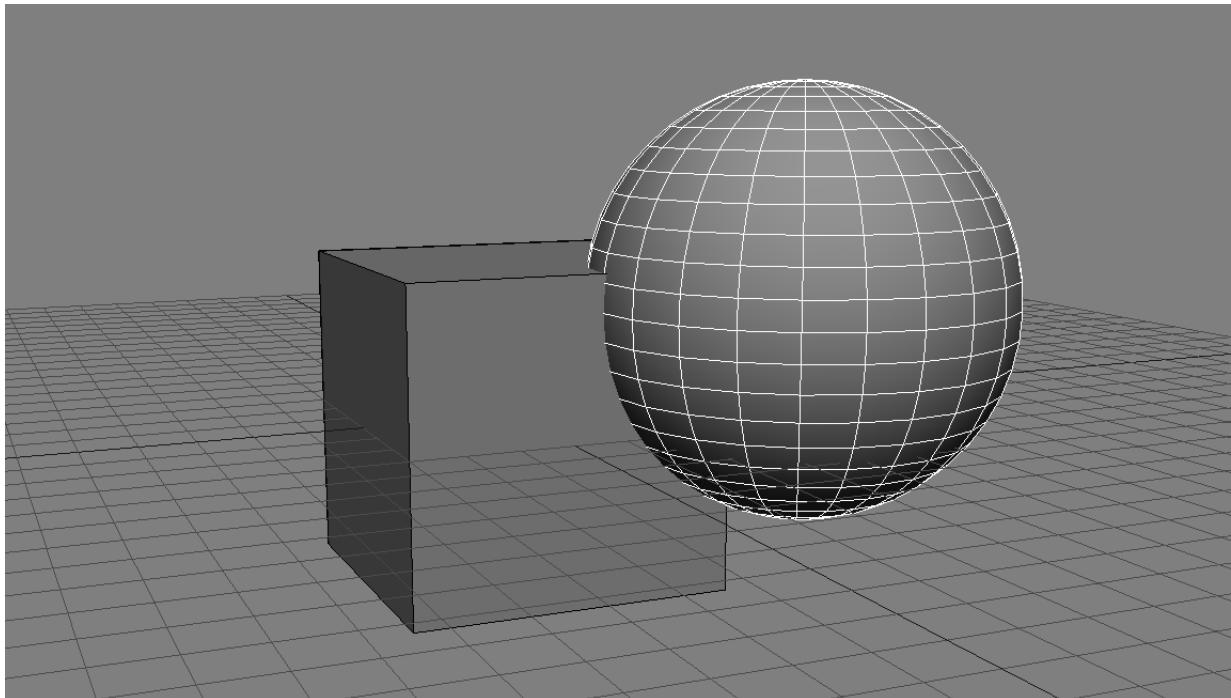


Указываем Aperture Size = 1 (для пущего эффекта). Фокус камеры будет там, где находится Camera_Interest.

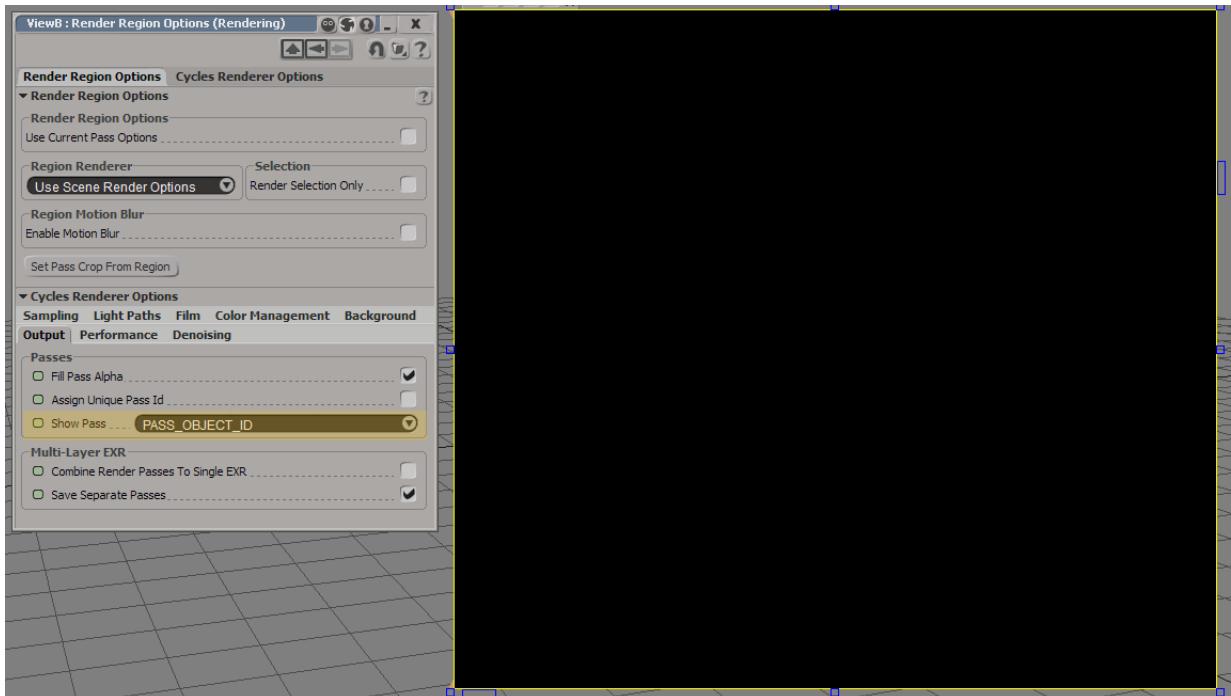


10 Какрендерить Object ID pass

Предположим у нас есть сцена: куб и сфера. И они как-то хитро расположены друг относительно друга.

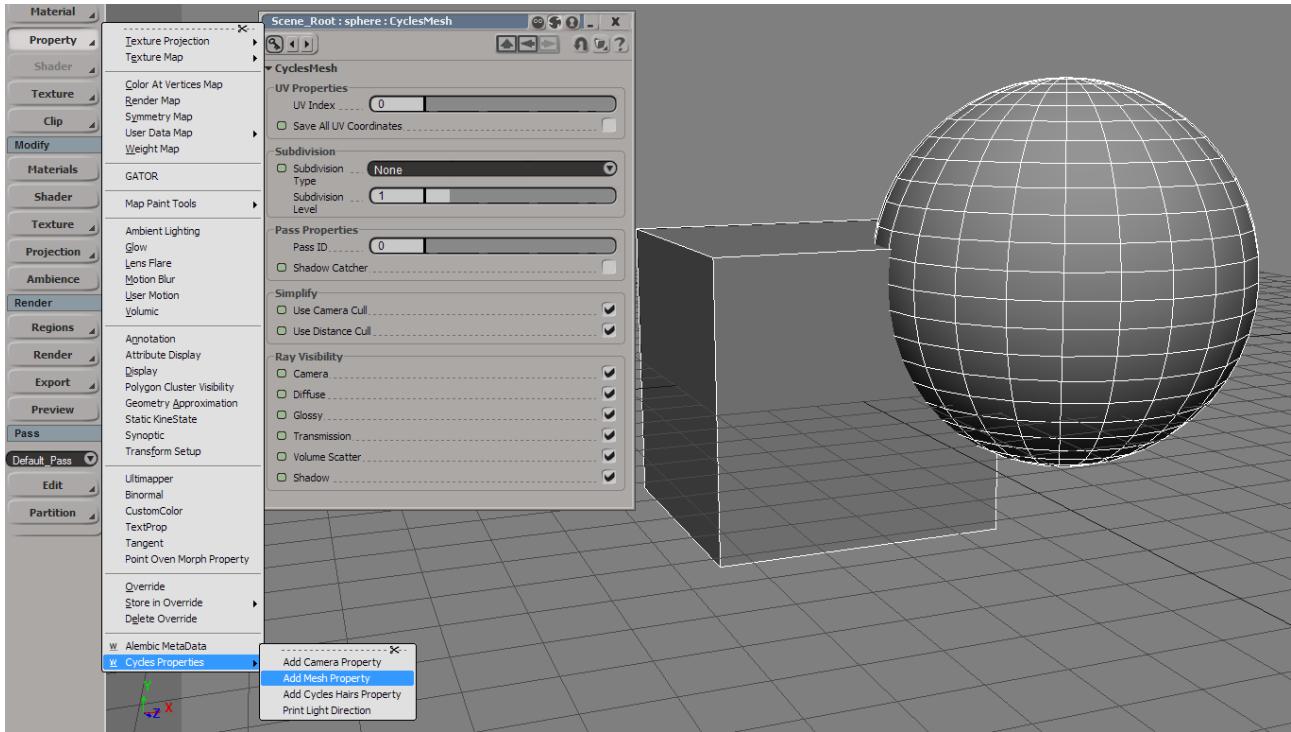


Мы хотим отрендерить **Object ID pass**, чтобы потом использовать его при композе. Сейчас этот канал пустой. Всё черным-черно.

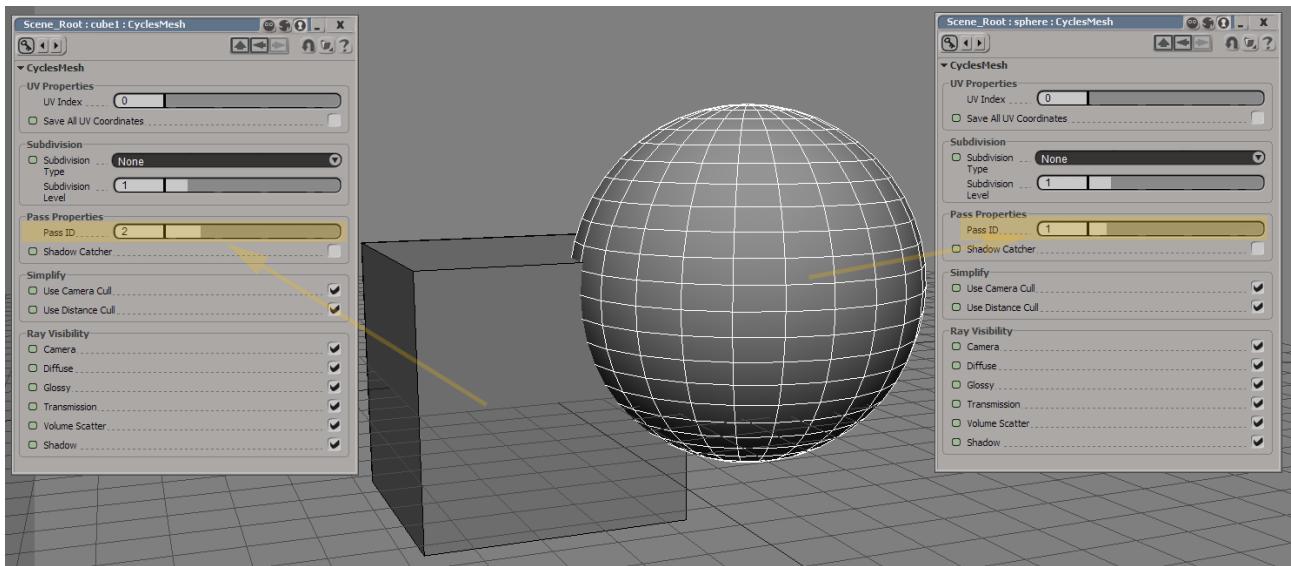


Выделяем оба объекта и назначаем им свойство **CyclesMesh**. Для этого выбираем команду

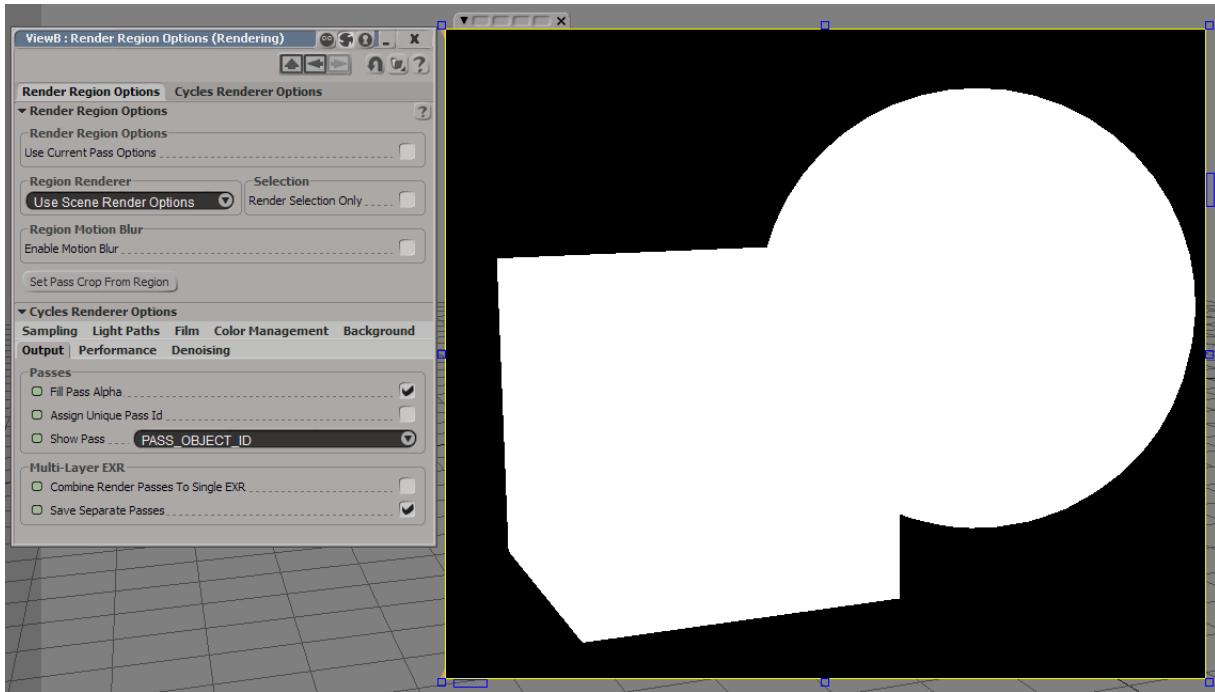
Property – Cycles Properties – Add Mesh Property.



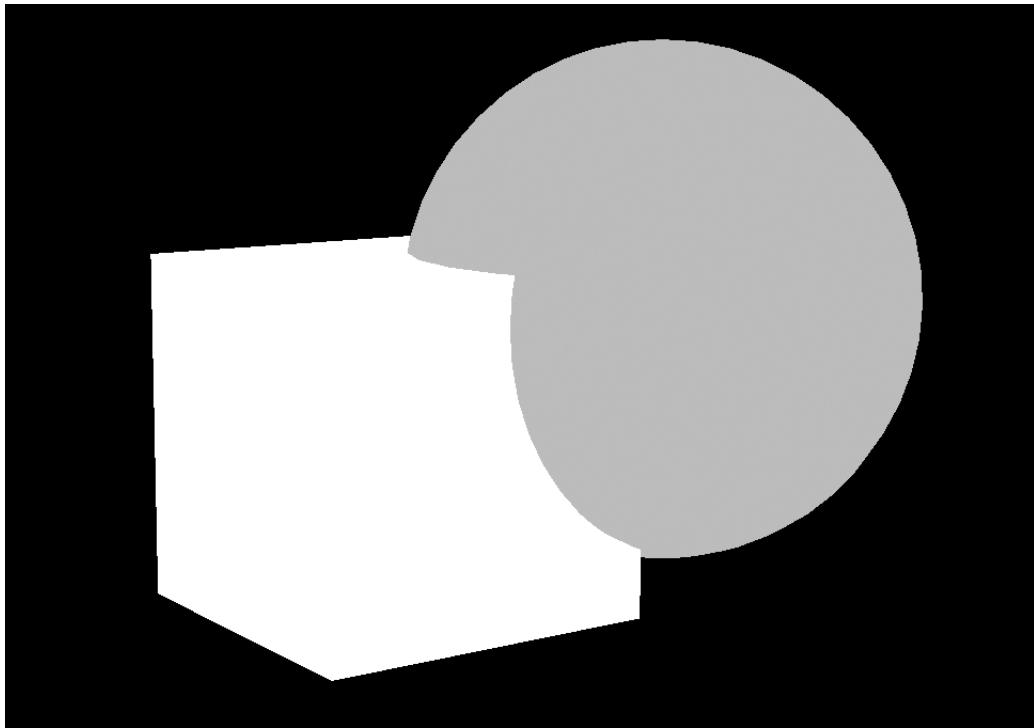
Для сферы указываем значение Pass ID = 1, а для куба Pass ID = 2.



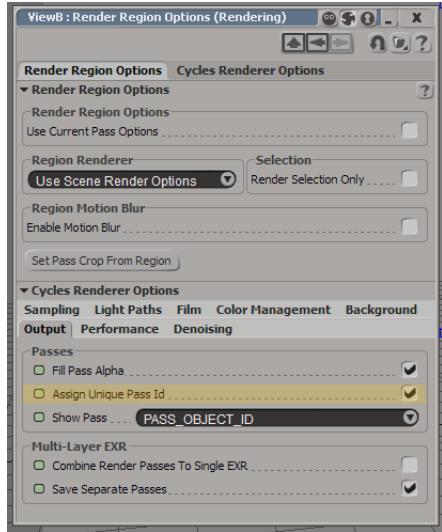
Рендерим, и видим, что теперь канал содержит данные.



Кажется, что оба объекта залиты одним и тем же цветом, но это не так. Это разные оттенки серого. Если отрендерить канал в 32 бита и сдвинуть уровни, то в этом легко можно будет убедиться. Вообще, смысл канала **Object ID** в том, что объекты с одинаковым значением **Pass ID** отображаются одним цветом. Те, у кого **Pass ID = 0** вообще не отображаются.



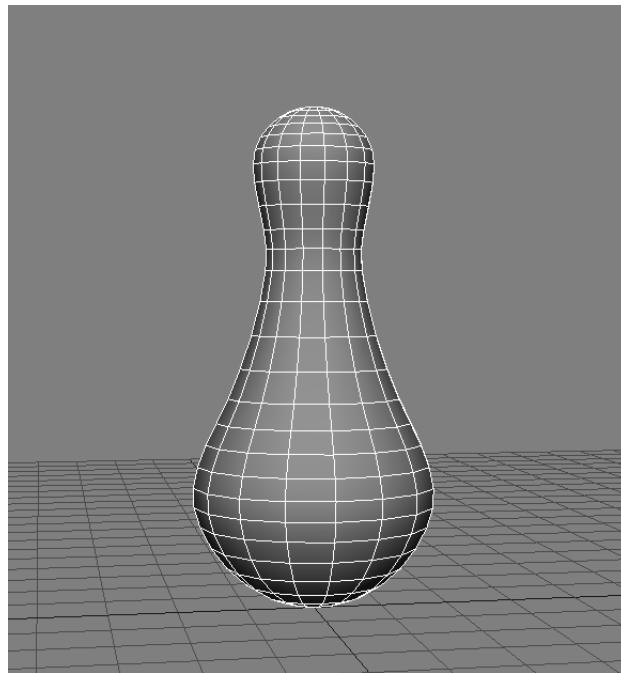
В свойствах рендера можно включить опцию **Output – Passes – Assign Unique Pass Id**. После этого всем объектам в сцене автоматически будут назначаться разные значения **Pass ID**.



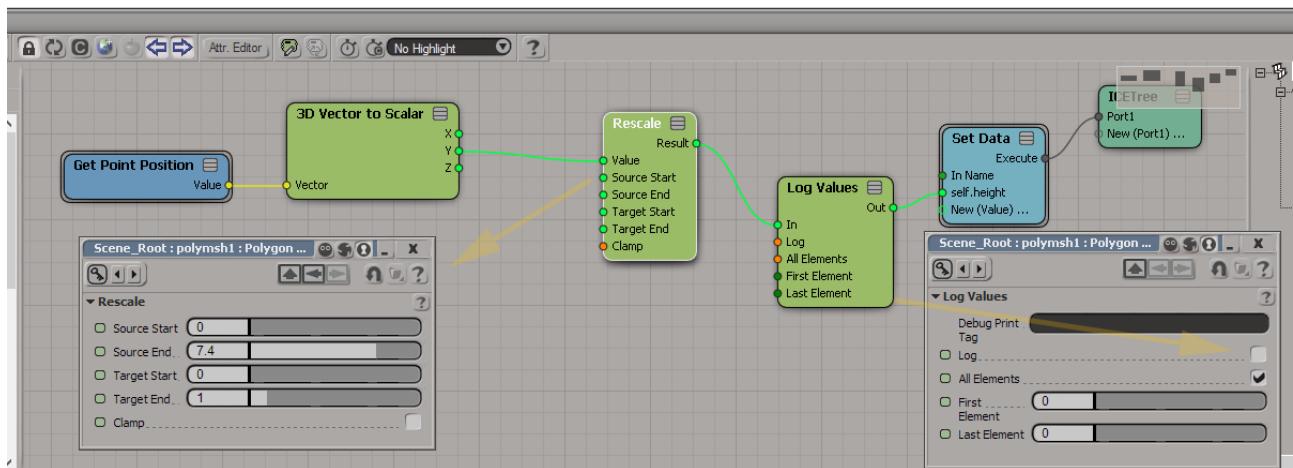
Аналогичным образом работает и вывод канала **Material ID**. Только он всегда имеет разные значения для разные материалов.

11 Какрендерить ICE-атрибуты

Предположим у нас есть сцена, на которой расположен простой объект.

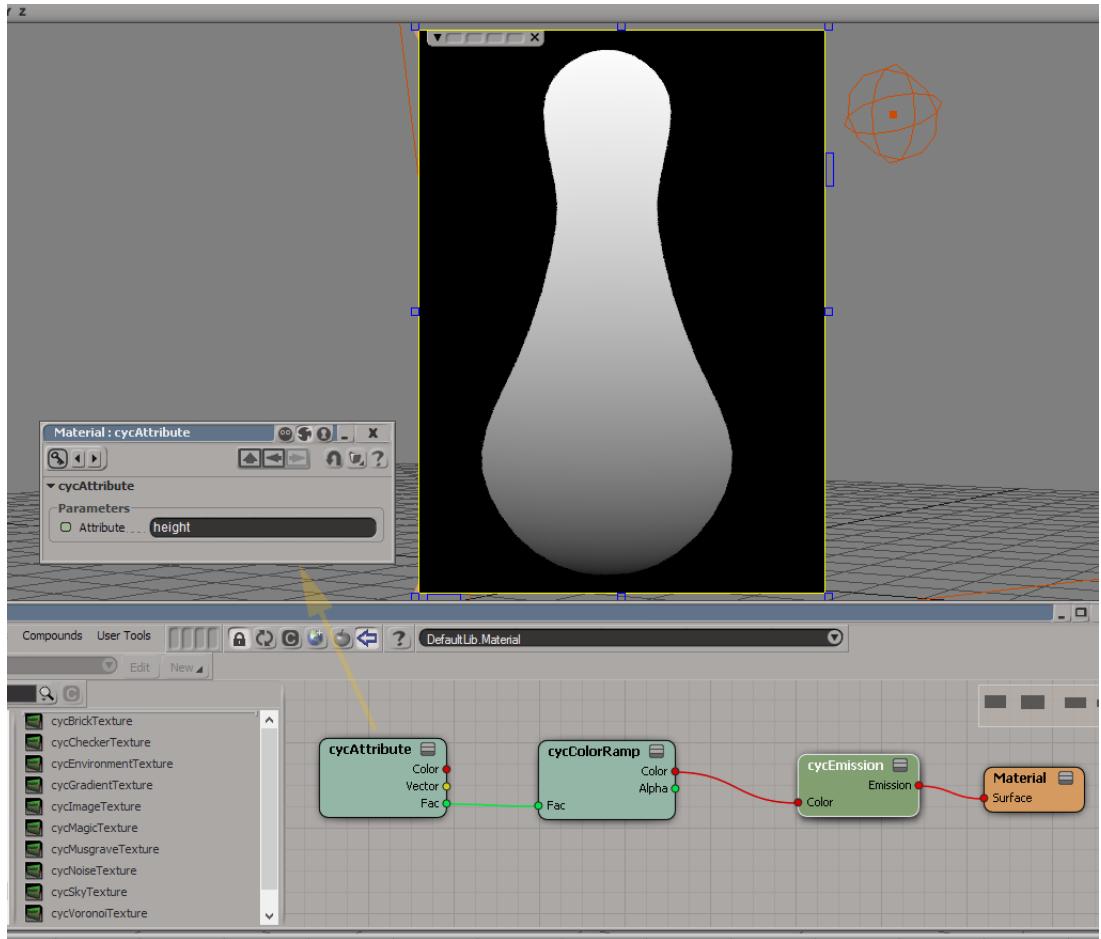


Этот объект содержит ICE-дерево следующего вида:



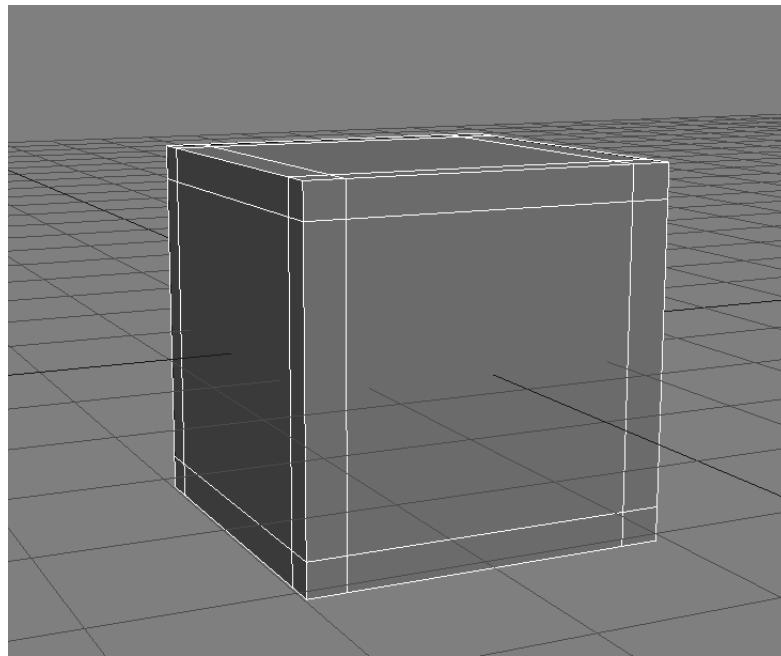
Это ICE-дерево записывает в каждую вершину значение от 0 до 1 в зависимости от того, как высоко вершина располагается над землёй. В ноде Rescale величина 7.4 – это примерная высота объекта. Все значения сохраняются в атрибут height. Нода Log Values используется для того, чтобы силой заставить ICE считать значения. Дело в том, что ICE сильно умный, и если видит, что атрибут нигде не используется, то часто его просто не считает. Нодой Log Values даже с отключенной опцией Log мы заставляем его проводить все вычисления.

Назначаем нашему объекту Emission-шейдер. Добавляем ноду Attribute, в которой указываем название ICE-атрибута (height в нашем случае). Наконец пропускаем float-выход этой ноды через Color Ramp и подключаем всё к порту Color. На рендре видно, как цвет объекта снизу вверх меняется от чёрного к белому.

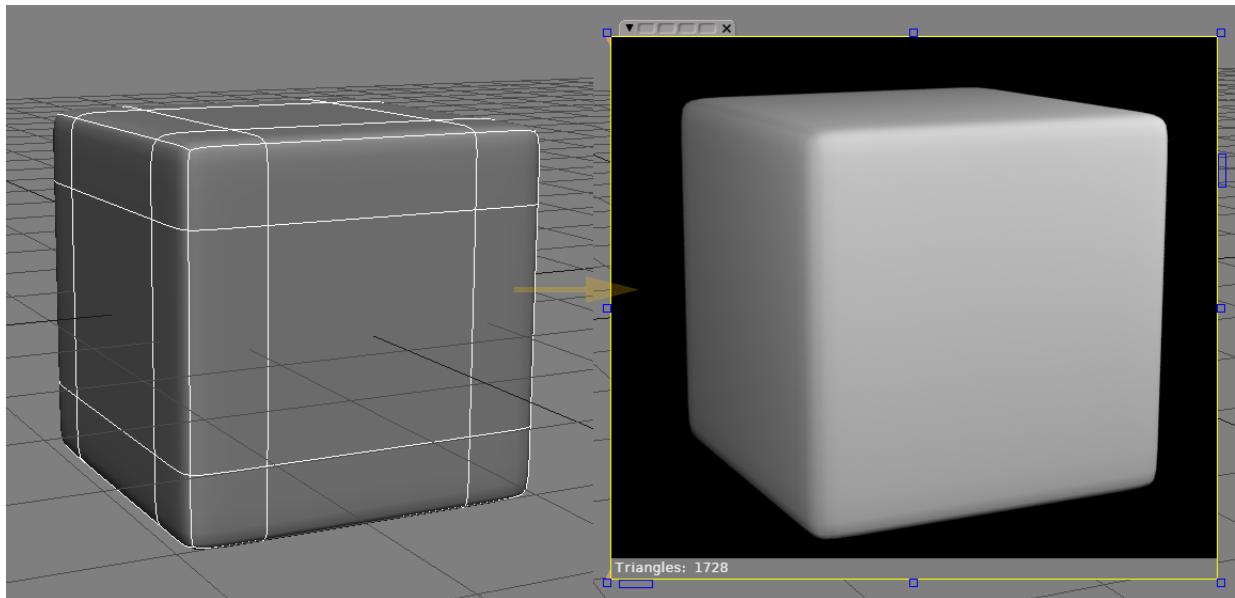


12 Какрендерить subdivide поверхности

Предположим у нас есть куб.

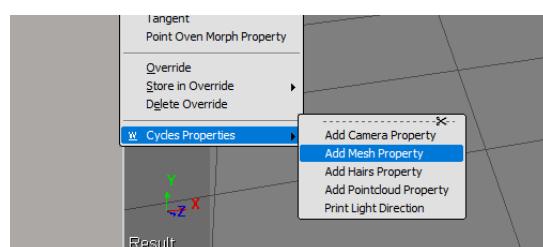


Если нажать 3 раза плюс на клавиатуре, то куб станет сглаженным. И на рендре он точно такой же.

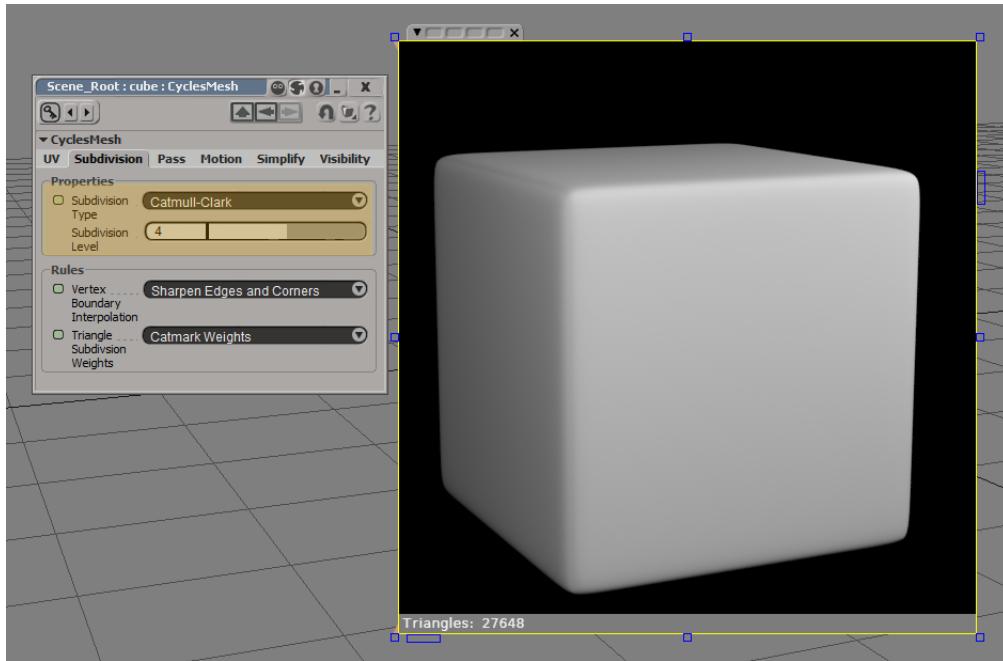


Если хочется делать подразбиение объекта только во время рендера, то на объект назначаем свойство **CyclesMesh**, выбрав команду

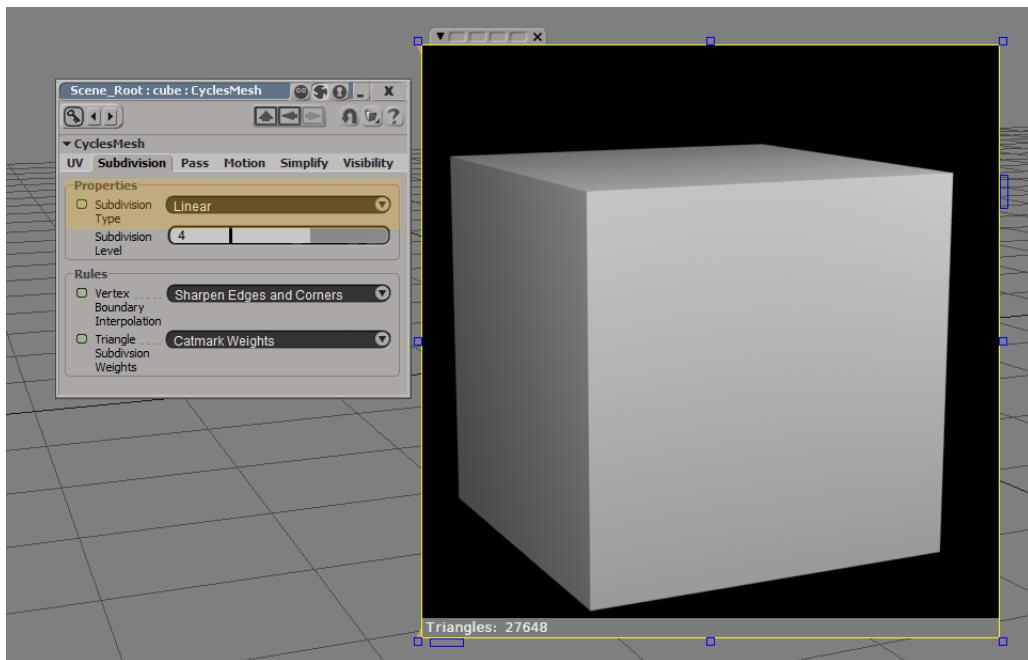
Property – Cycles Properties – Add Mesh Property.



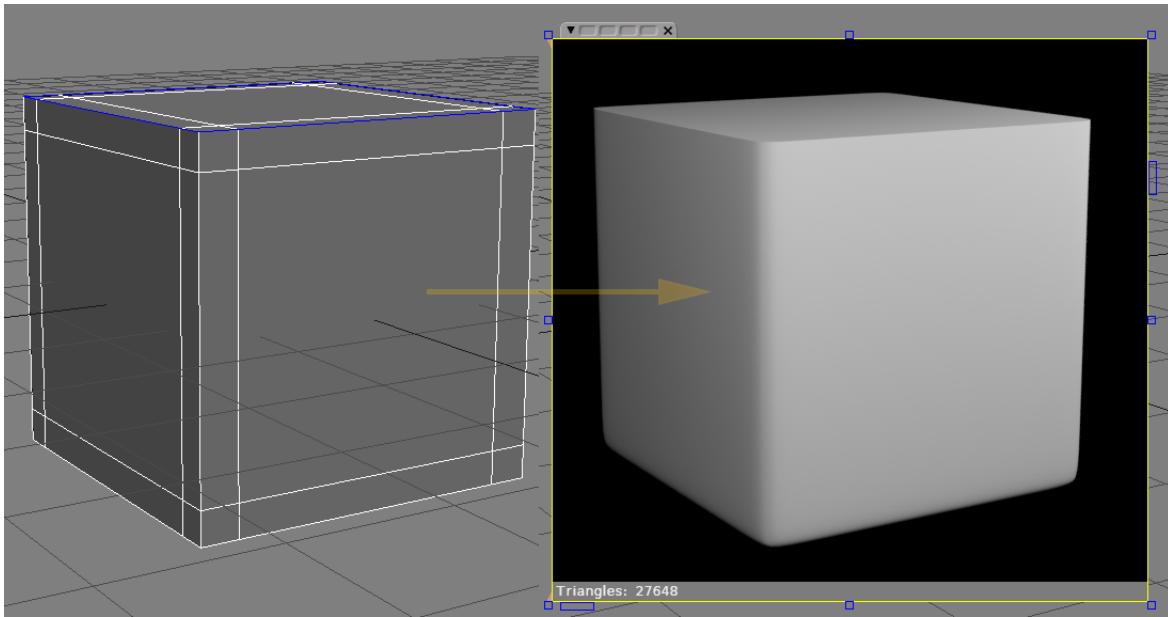
На вкладке Subdivision выбираем тип Catmull-Clark и указываем число шагов подразбиения равное 4. Рендерим, видим результат.



Если выбрать тип Linear, то полигоны будут разбиваться на более мелкие, но сама форма объекта останется без изменений. Полезно при использовании displacement.



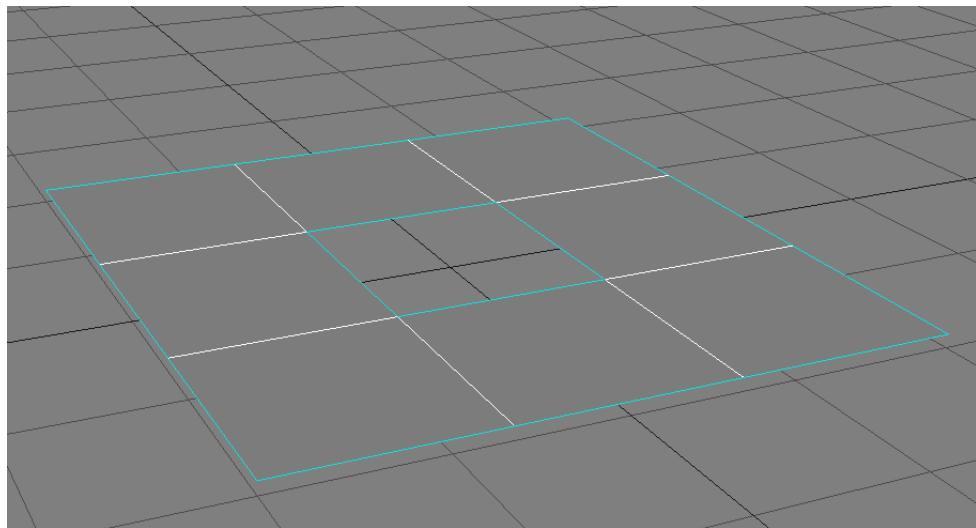
У объекта могут быть жёсткие ребра (по аглицки hard-edges), но даже при этом результат на рендере весьма корректный.



Ещё в настройках есть два параметра:

`Vertex Boundary Interpolation` и `Triangle Subdivision Weights`.

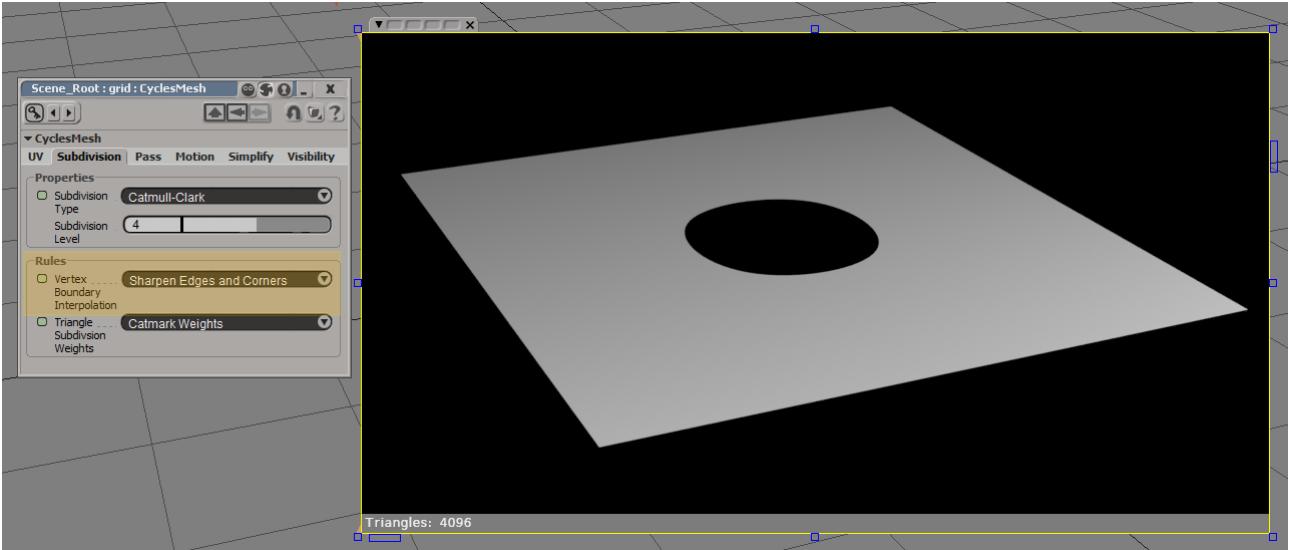
Первый параметр `Vertex Boundary Interpolation` говорит о том, как сглаживать граничные рёбра и вершины. Удобнее всего это увидеть на примере. Возьмём плоскость с дыркой посередине.



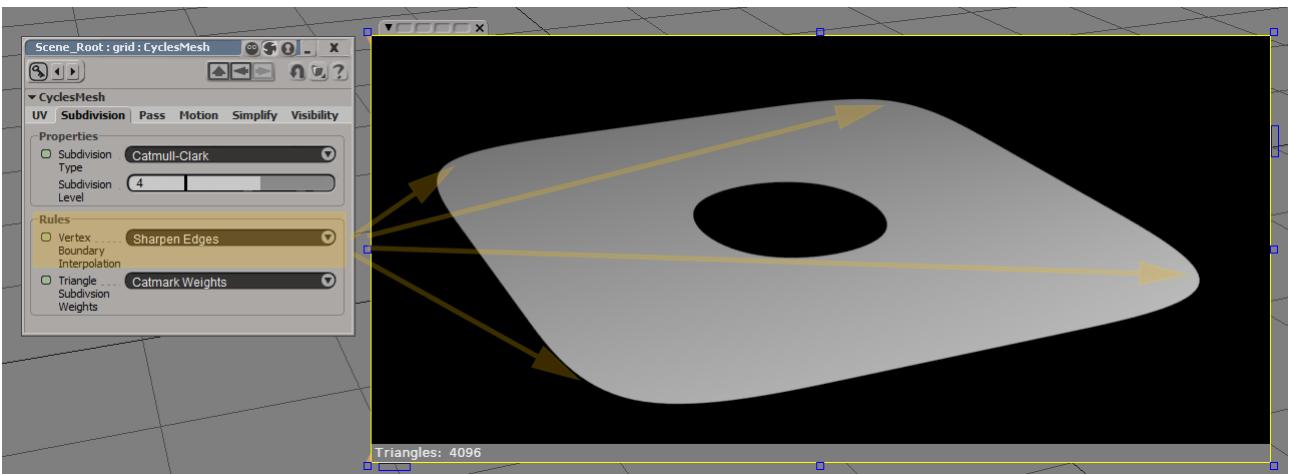
Если значение параметра `Vertex Boundary Interpolation` указано

`Sharpen Edges and Corners`,

то и вершины, и рёбра будут сохранять своё местоположение при сглаживании.



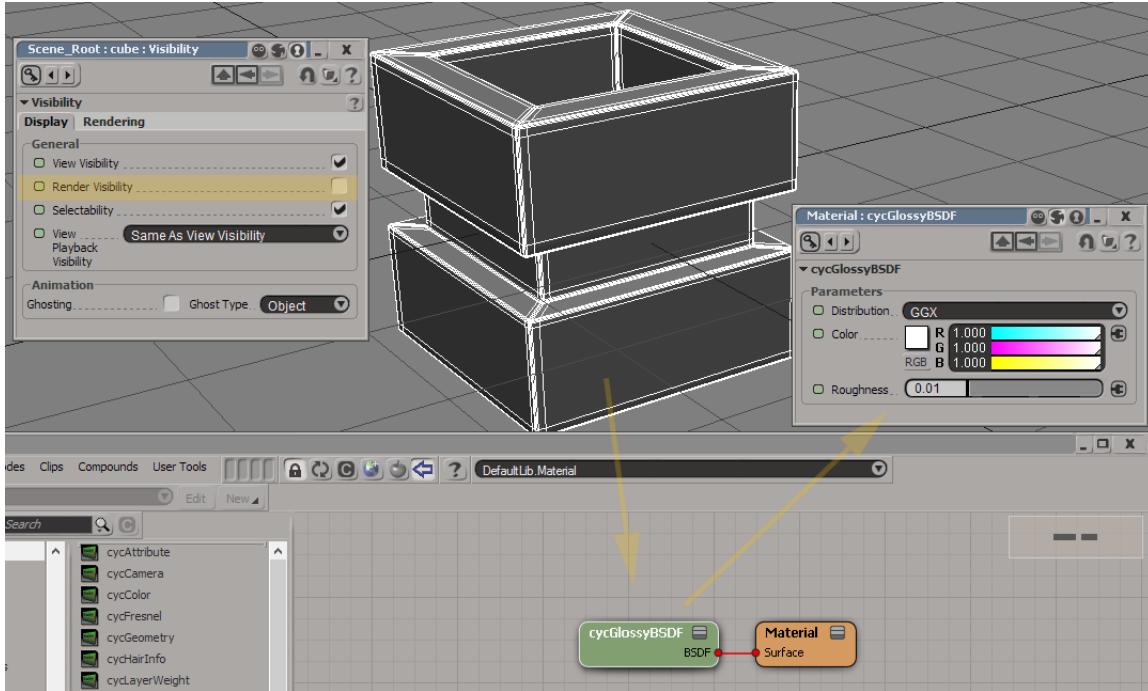
Если же значение параметра выбрано **Sharpen Edges**, то углы модели будут сглаживаться.



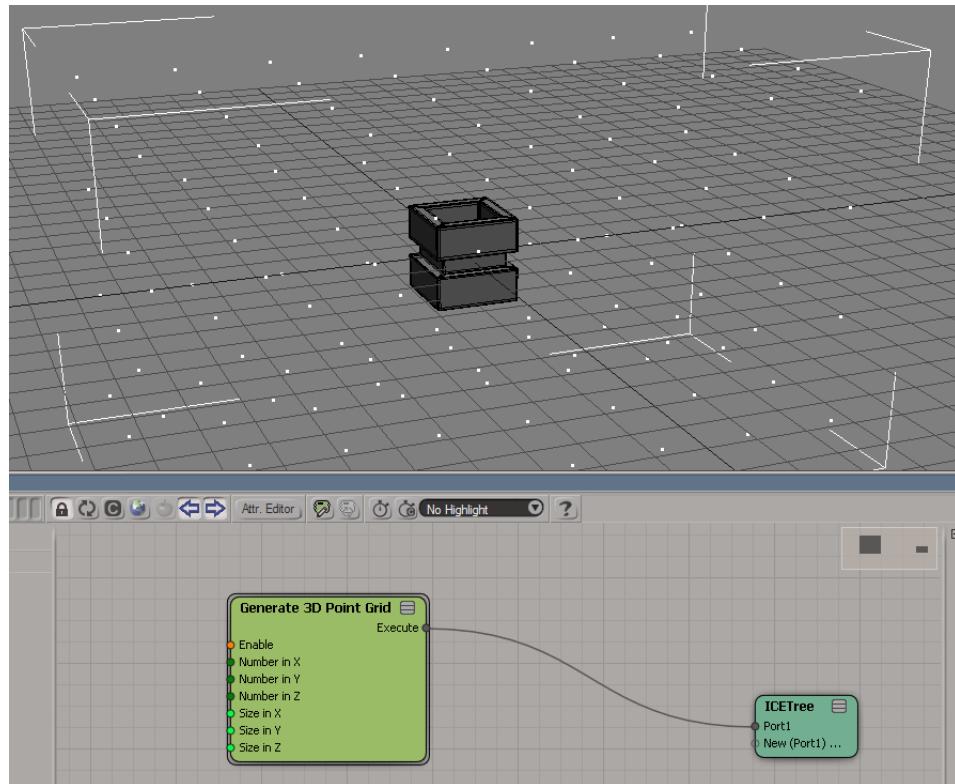
Параметр **Triangle Subdivision Weights** влияет на способ подразбиения треугольных полигонов модели. Невооружённым глазом разница между режимами **Catmark Weights** и **Smooth Triangle Weights** почти всегда незаметна.

13 Какрендерить ICE-инстансы

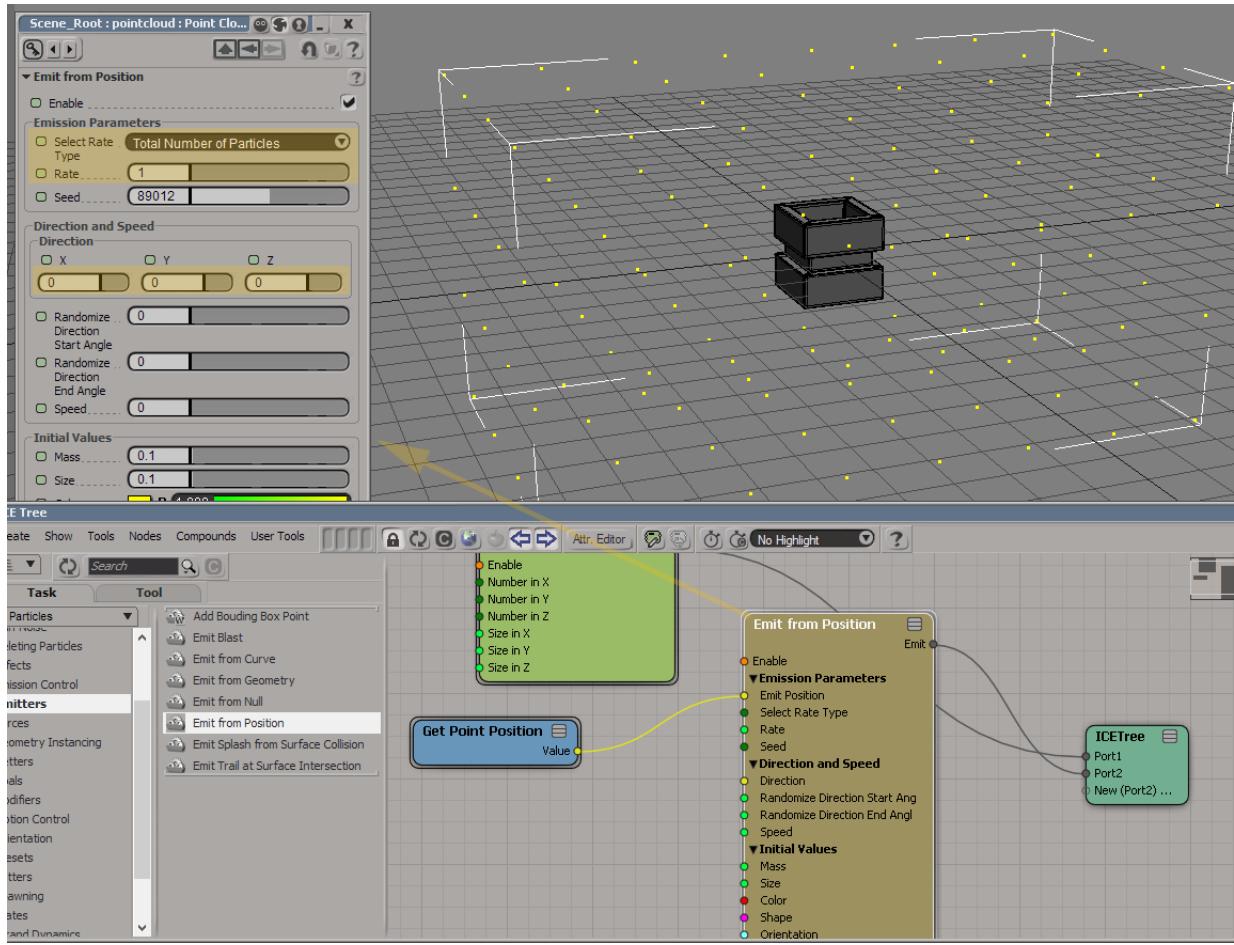
Ниаких особенностей здесь нет. Всё работает так же, как в остальных рендерах. Предположим у нас есть какой-то объект, который мы хотим размножить. На него назначен некий шейдер и выключено отображение при рендре.



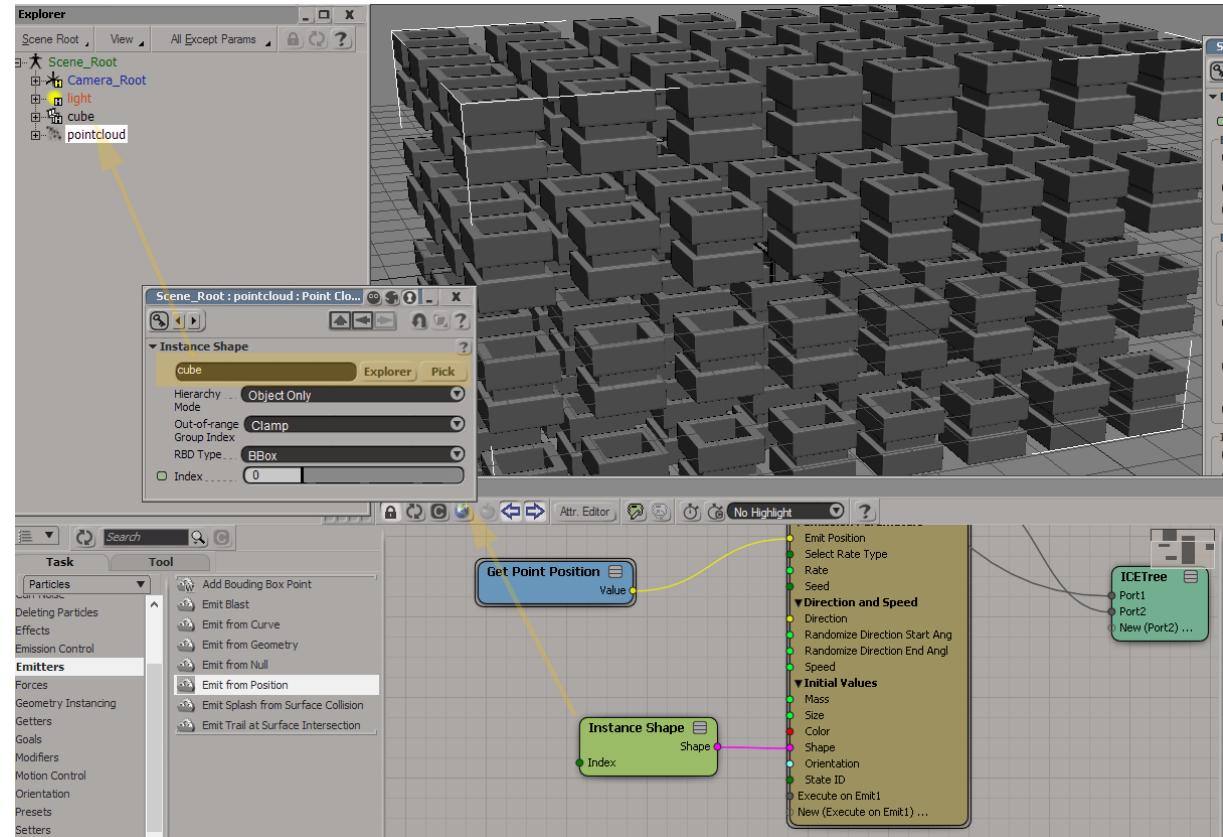
Создаём теперь пустой Point Cloud, в нём обычное ICE-дерево и генерируем облако точек.



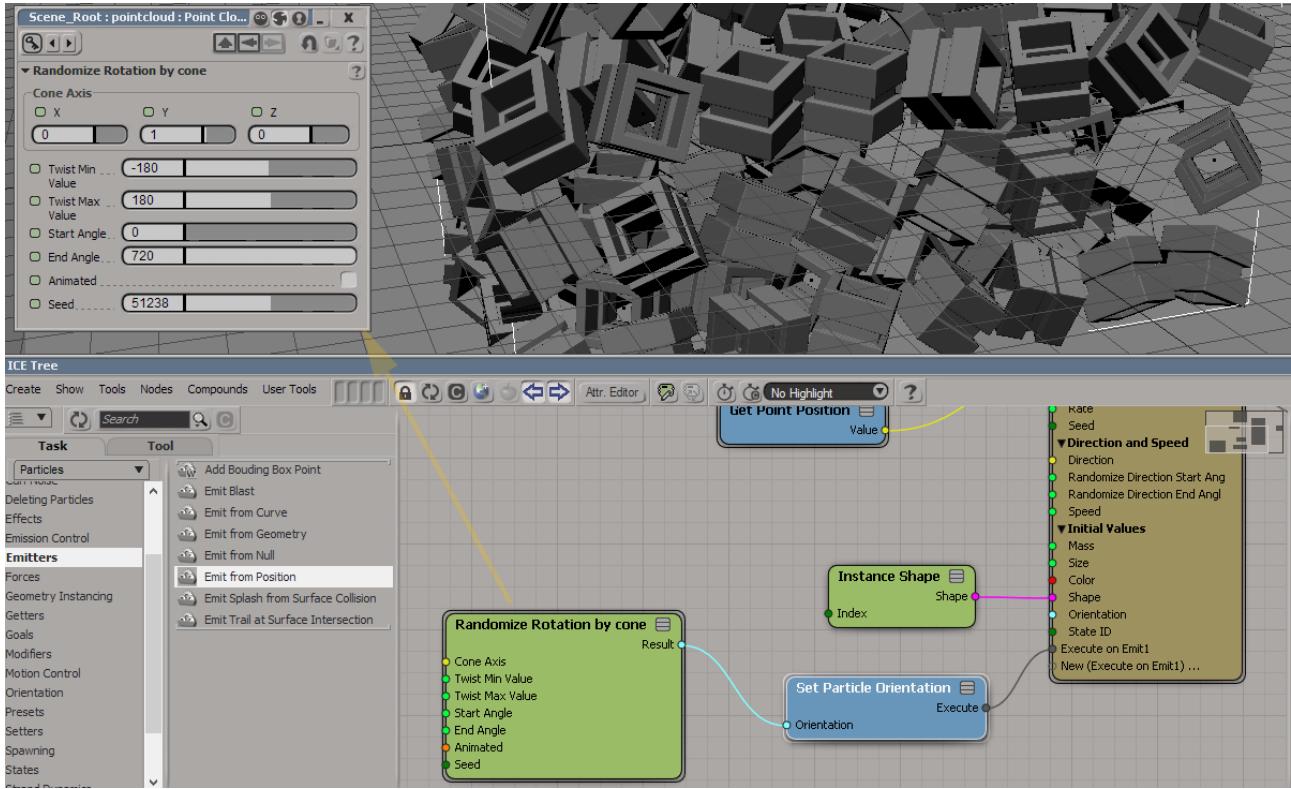
Дальше в ICE-дерево кидаем ноду `Emit from Position`. В ней выставляем параметры как на картинке и подключаем ноду `Get Point Position` к порту `Emit Position`. Это сгенерирует частицы как раз в тех местах, где были точки сетки.



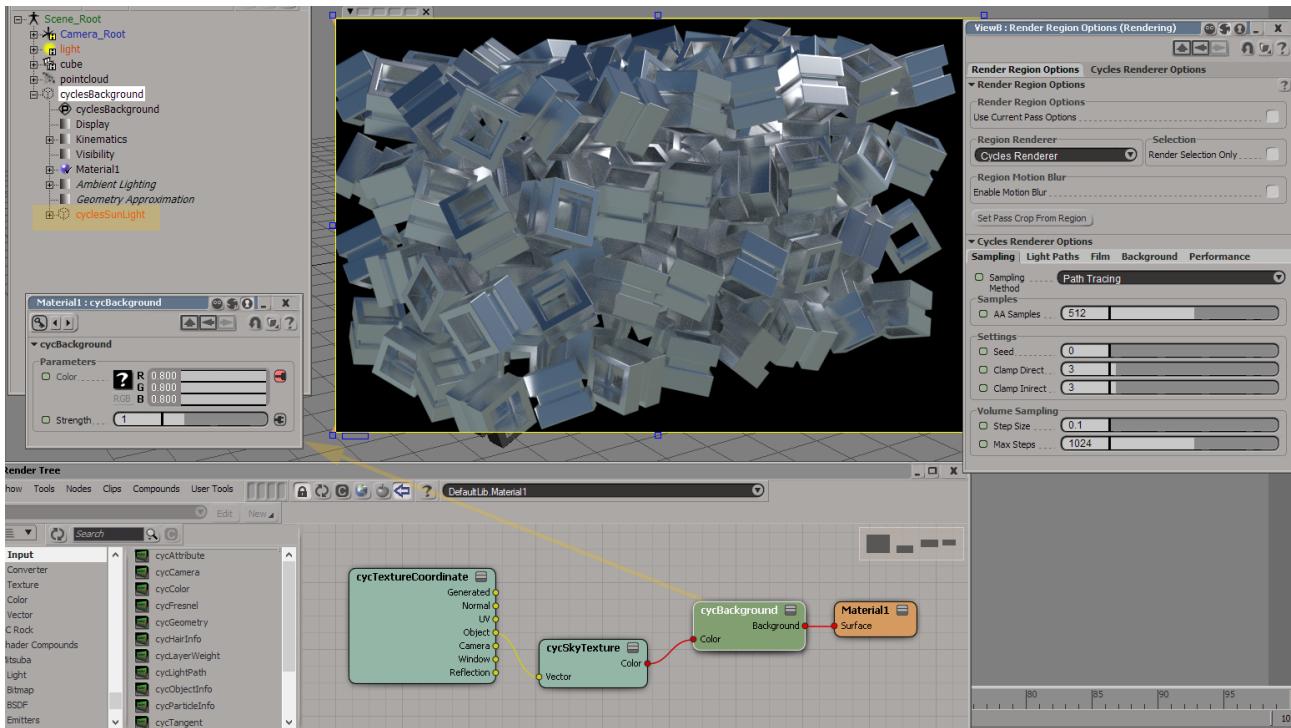
Вместо частиц можно использовать такие примитивы как Disc, Rectangle, Sphere, Box, Cylinder, Cone и Capsule (выглядят так же, как Sphere). Но можно и любой полигональный объект. С помощью ноды Instance Shape указываем наш master-объект.



Добавляем немного хаотичности в ориентацию инстансов.

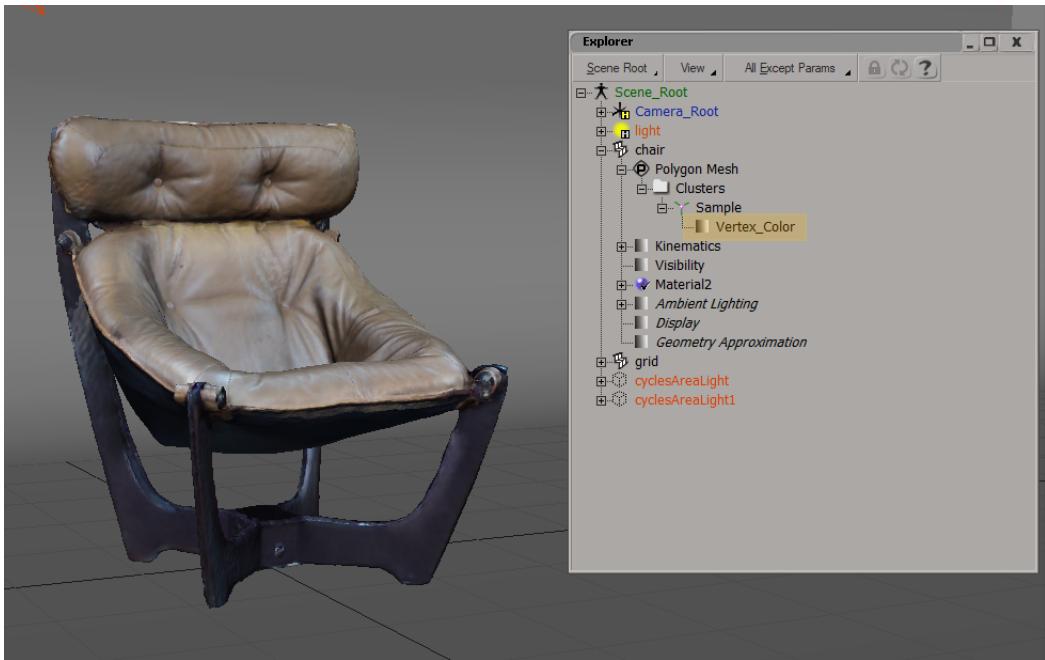


Наконец создаём источник освещения типа **Background**, ему в качестве дочернего объекта добавляем источник света типа **Sun**. В шейдер фона добавляем узел **SkyTexture**. Рендерим.

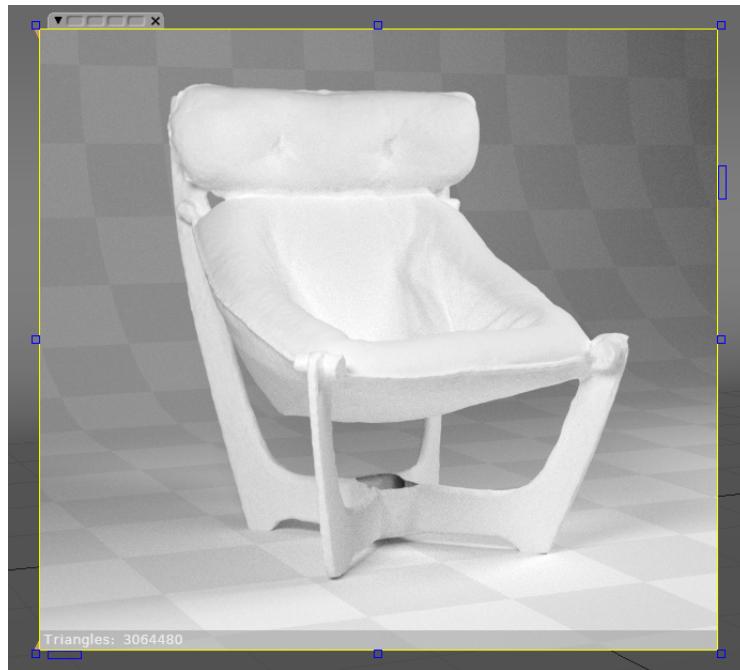


14 Как рендерить Vertex Color

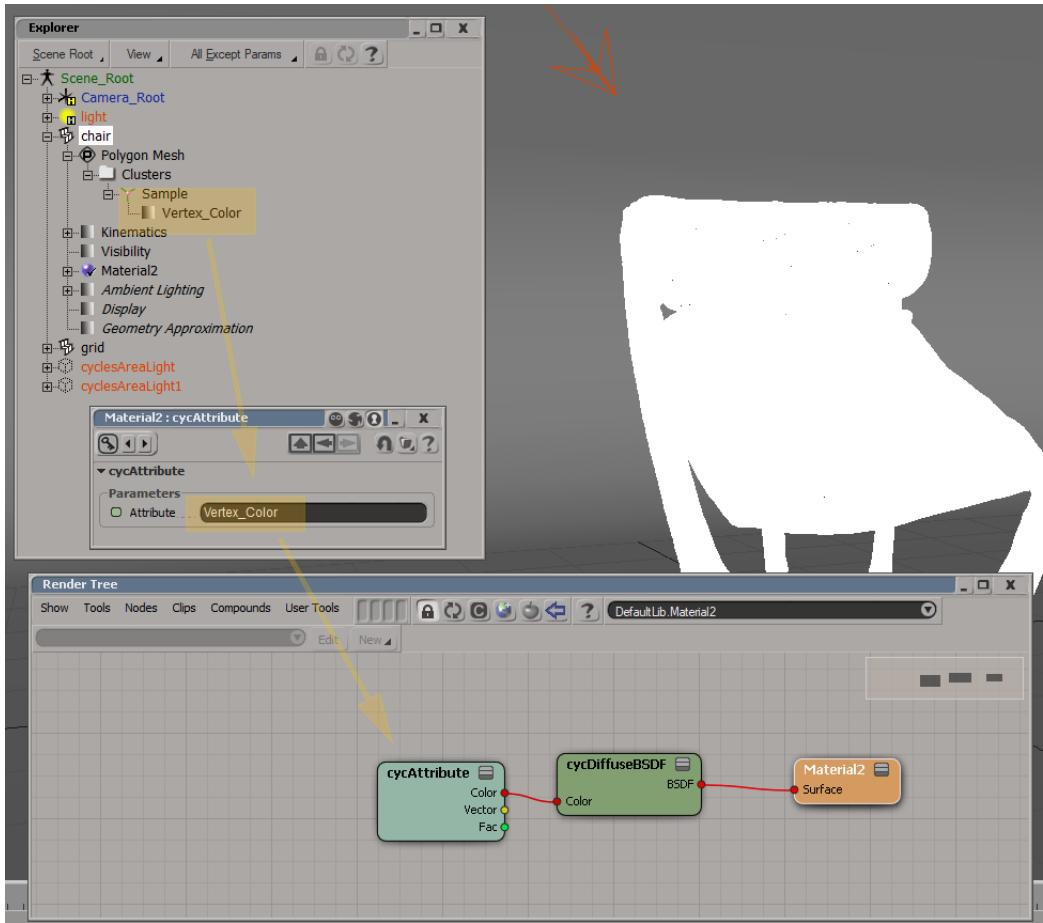
Предположим у нас есть сцена с объектом, который содержит Vertex color.



На рендре этот объект имеет нейтральный цвет, так как **Cycles** не знает, как и куда ему надо использовать цвета вершин.



Добавляем в материал ноду **Attribute** и указываем в ней имя кластера, содержащего цвета вершин (**Vertex_Color** в нашем случае).

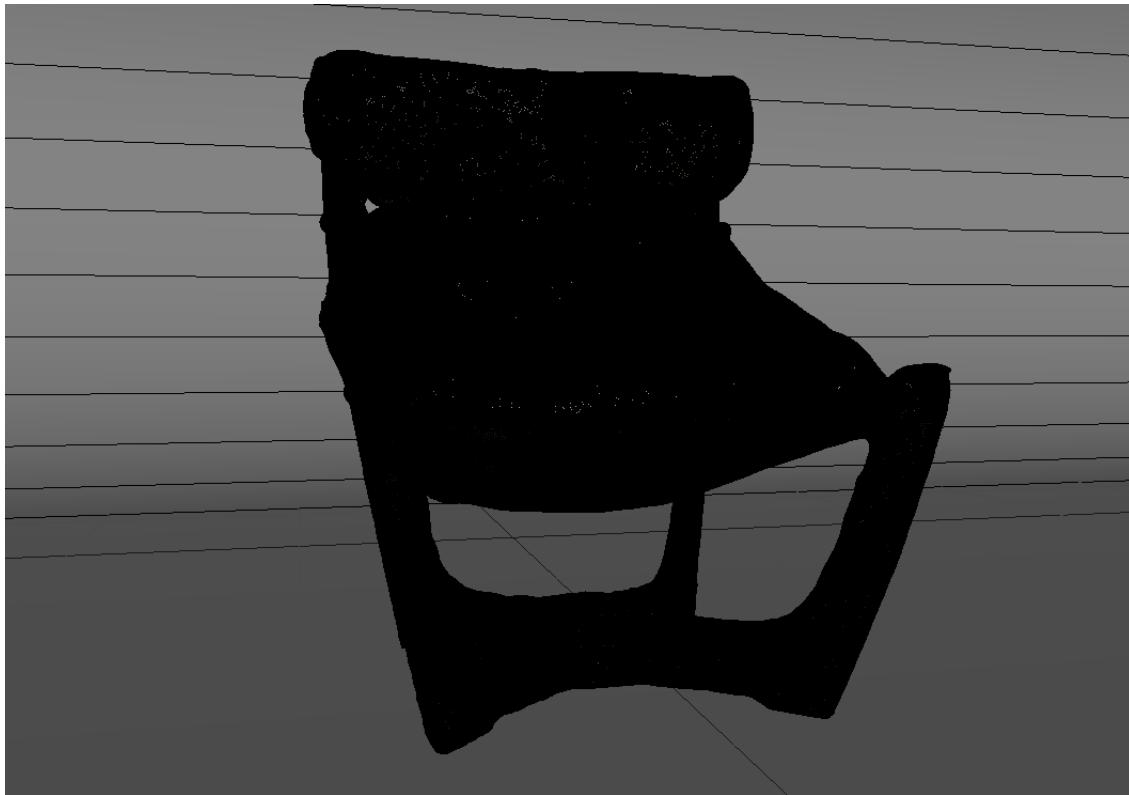


Рендерим.

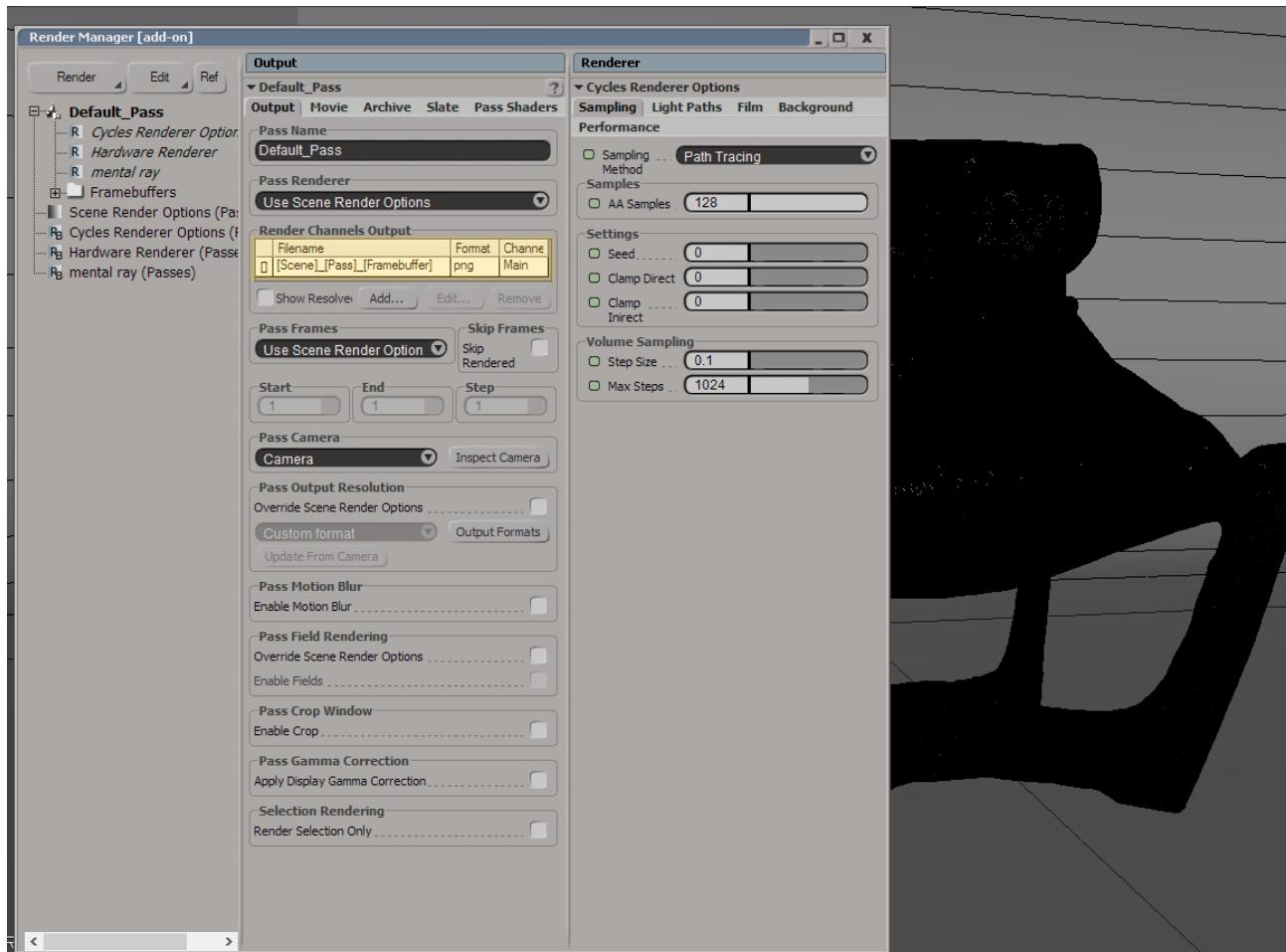


15 Как сохранить результат в Multi-Layer EXR

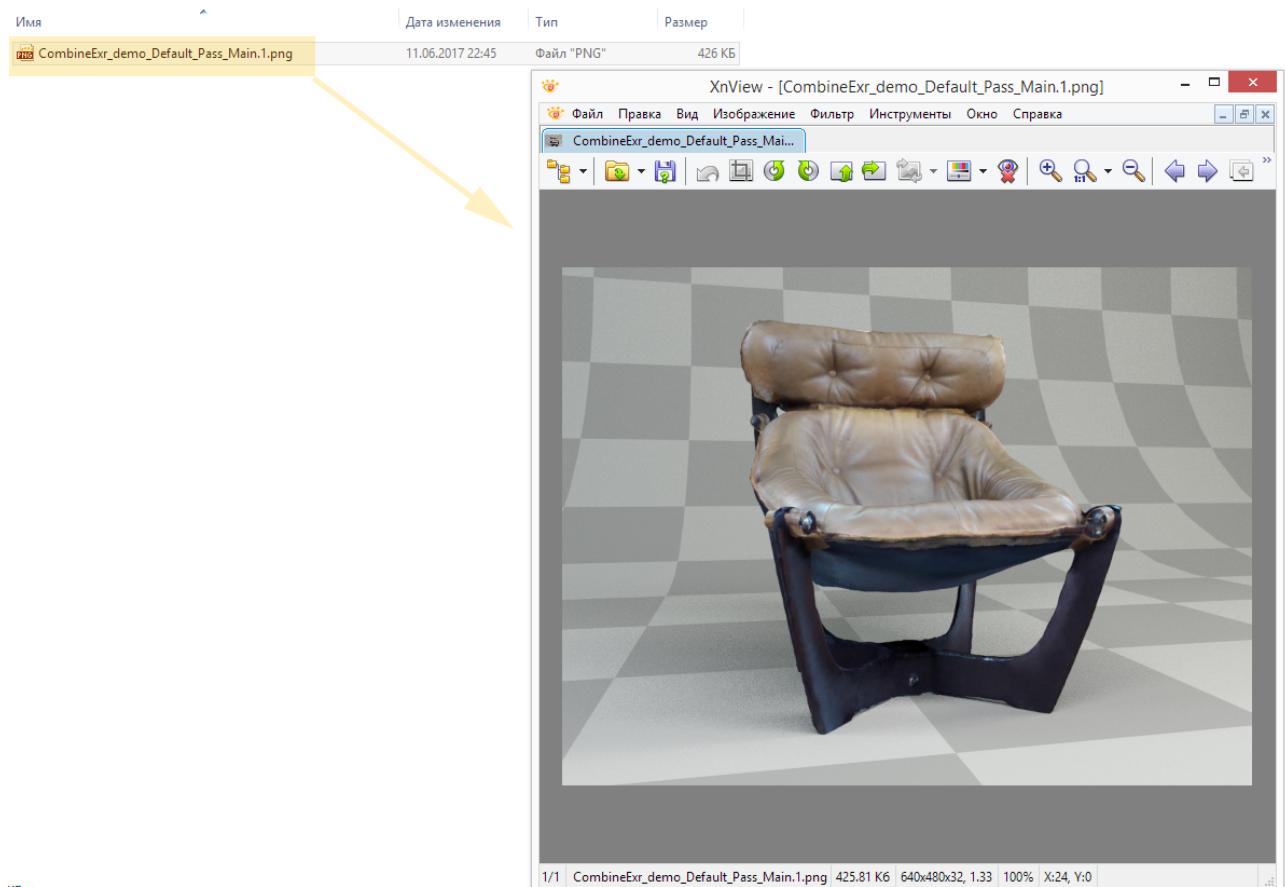
Предположим у нас есть сцена: кресло в простой студии.



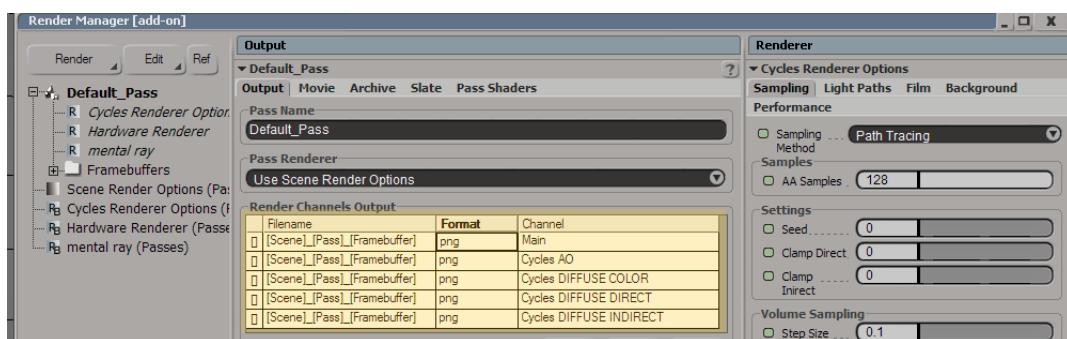
Выбираем в Render Manager-е один выходной канал, и сохраняем его в формате *.png.



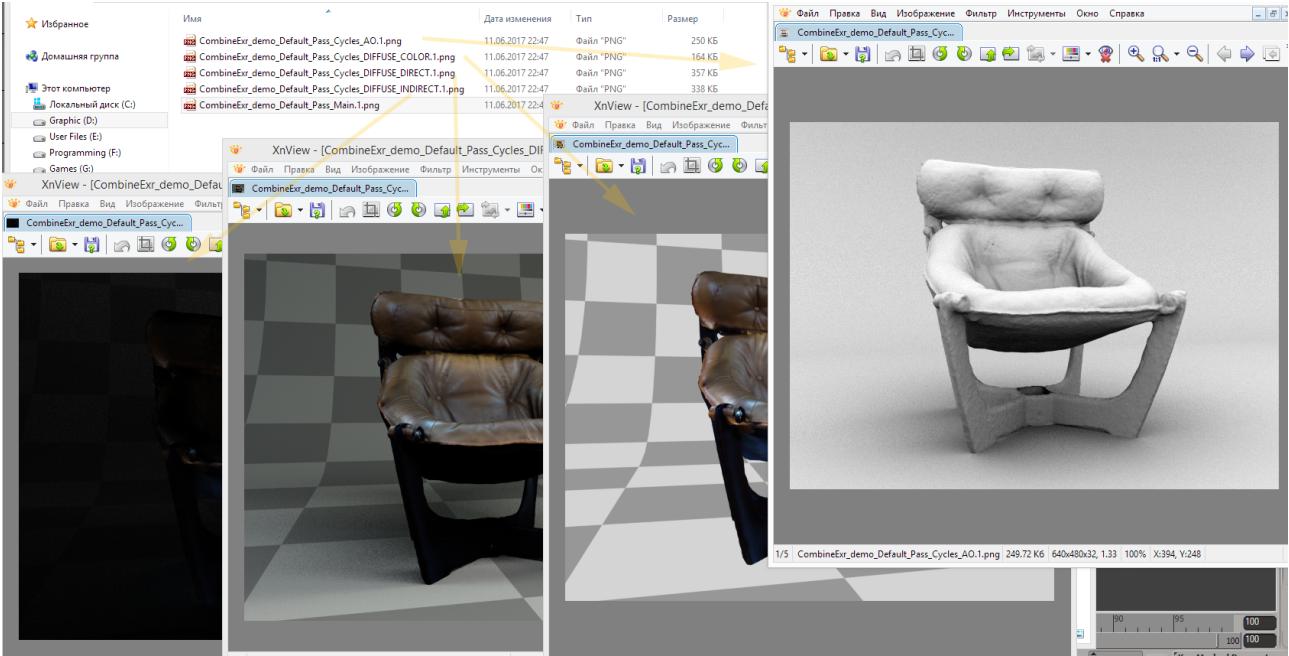
В результате рендера получается один png-файл.



Теперь добавим в рендер дополнительные каналы. Их тоже будем сохранять в формате *.png.

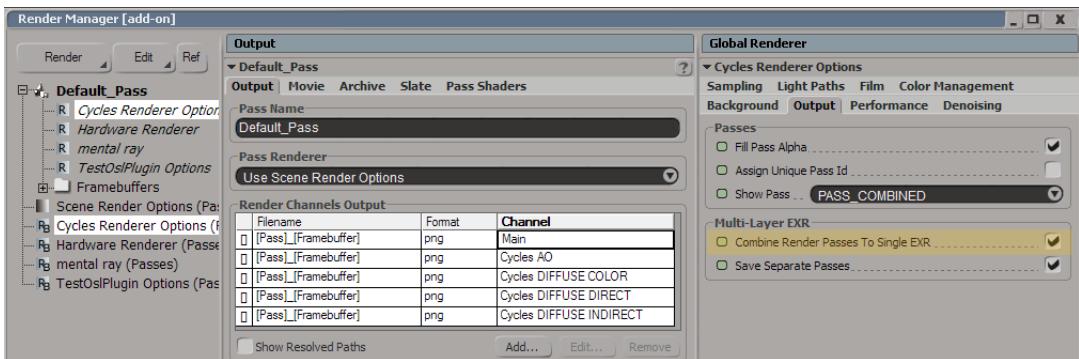


В результате рендера получаем уже набор файлов, каждый со своими данными.

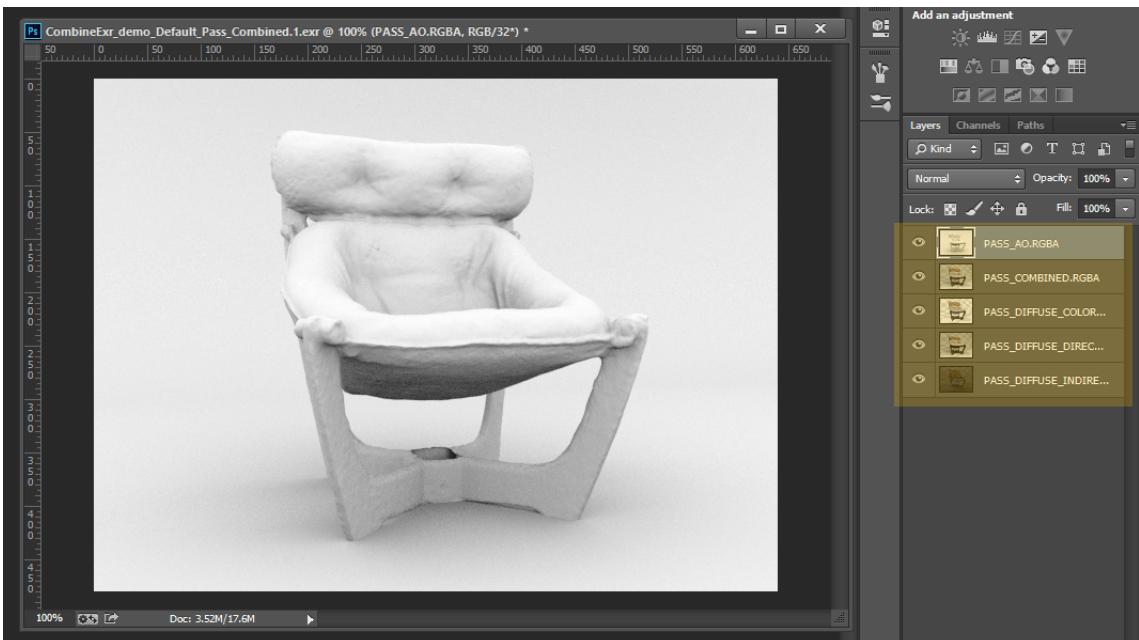


Все каналы, выбранные для рендера, можно сохранить в один многослойный 32-битный exr-файл. Для этого в разделе Output – Multi-Layer EXR включаем параметр

Combine Render Passes To Single EXR.

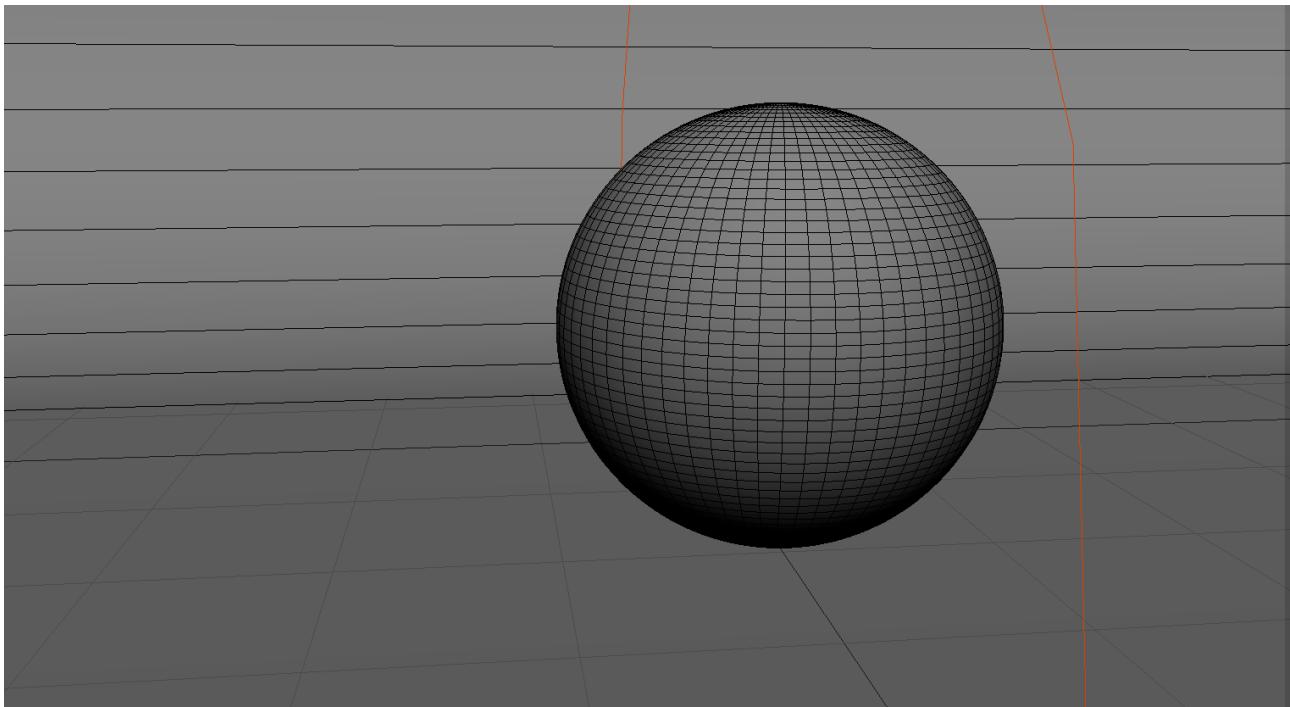


В результате рендера дополнительно создается exr-файл со всеми выбранными каналами. Если выключить параметр Save Separate Passes, то никаких других файлов, кроме этого многослойного, сохраняться не будет.



16 Как использовать Stamp Output

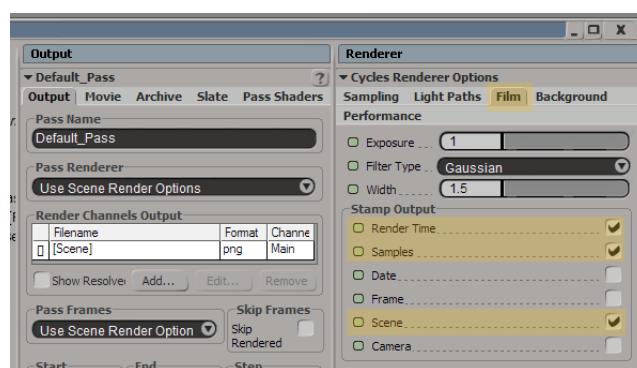
Предположим у нас есть сцена: стеклянная сфера в простой студии.



В результате рендера получается такая картинка:



В свойствах рендера в разделе **Film – Stamp Output** включим параметры **Render Time**, **Samples** и **Scene**.



В результате на отрендерённом изображении будет добавлена полоса с информацией о выбранных параметрах.



Смысл параметров следующий:

Render Time – время рендеринга всего изображения;

Samples – число AA-сэмплов в настройках рендера;

Date – дата рендеринга (число, месяц, год и время);

Frame – номер отрендеренного кадра;

Scene – имя сцены;

Camera – имя камеры, из которой происходит рендер;

Triangles – общее число треугольников во всех полигональных объектах сцены;

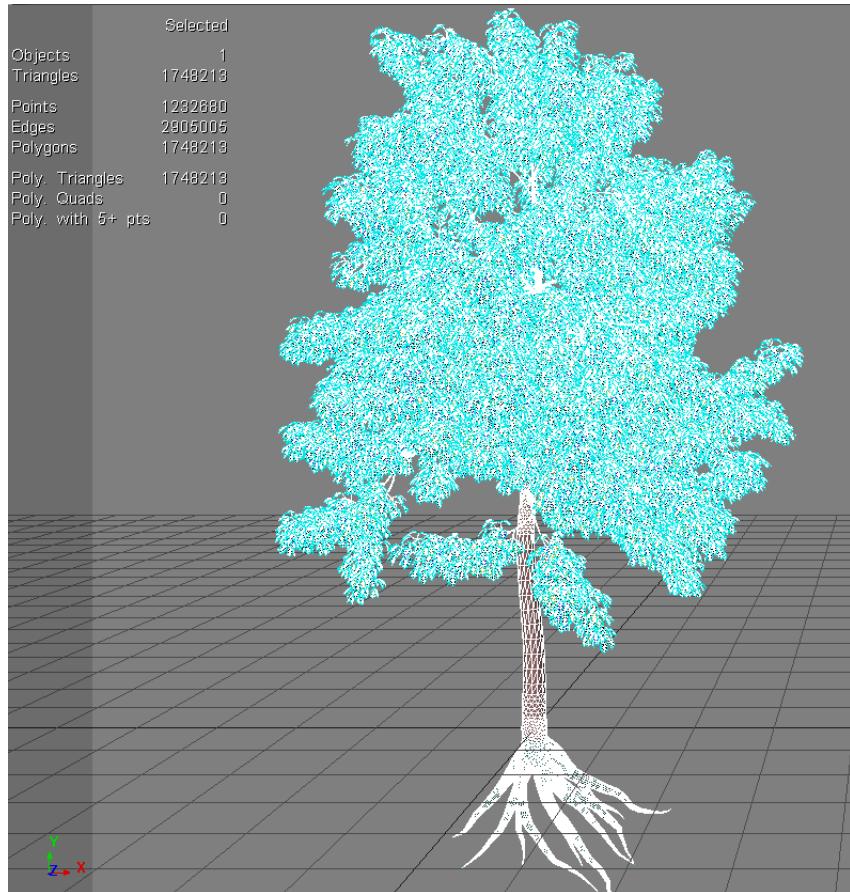
Curves – общее число кривых для всех волос в сцене;

Objects – число объектов на сцене;

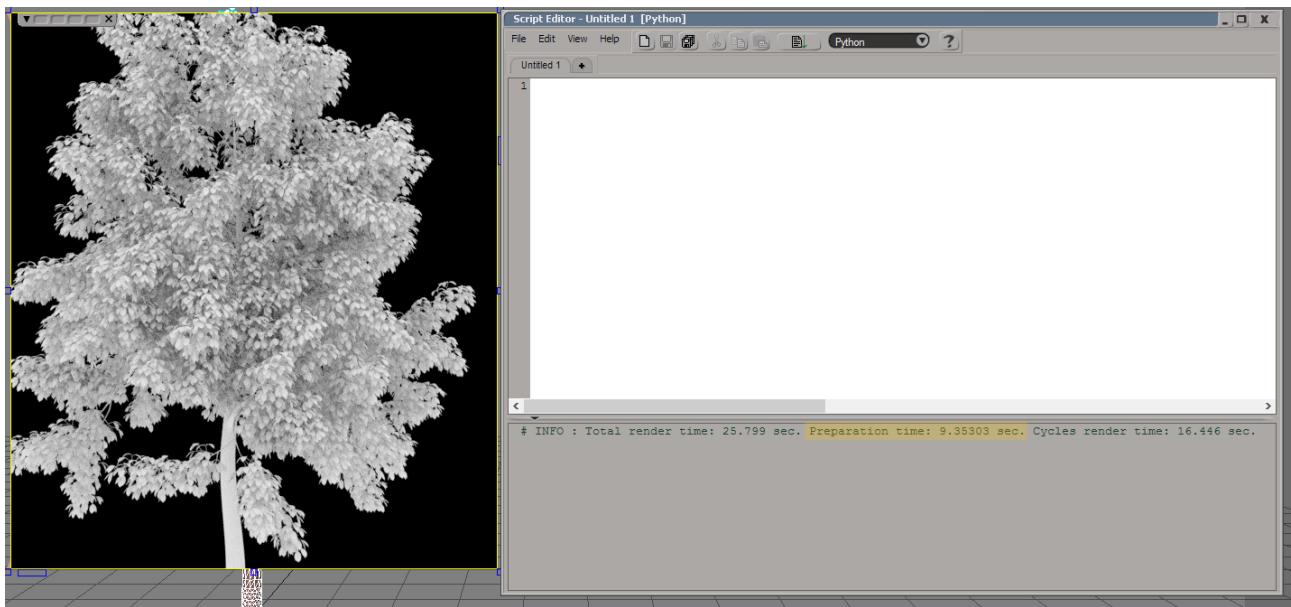
Lights – число источников света.

17 Как использовать Cache

Предположим у нас есть какой-нибудь сложный объект. Например, дерево.

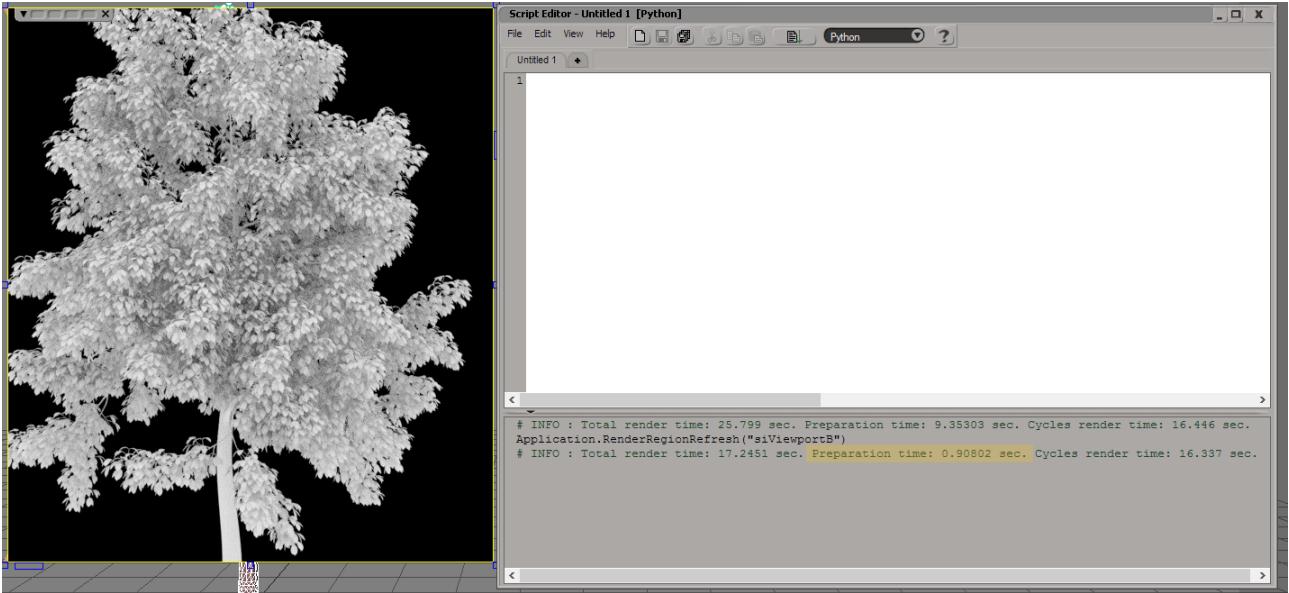


Отрендерим его и посмотрим в лог.



Там написано, что общее время рендера 25 секунд, из них 9 секунд заняла подготовка рендера. Подготовка включает в себя сбор данных о сцене, какие на ней располагаются объекты, какие шейдеры на них назначены и, самое длительное по времени, экспорт всей геометрии сцены в формат, понятный движку рендера. По сути все 9 секунд и ушли на то, чтобы экспортировать геометрию дерева.

Ничего не будем трогать и просто нажмём обновить рендер.



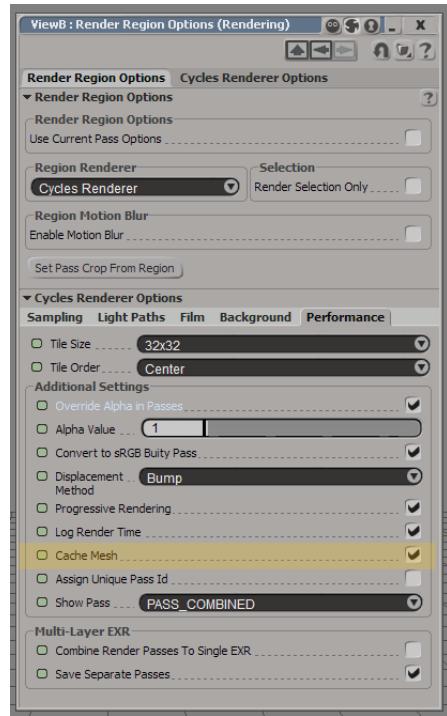
Теперь время подготовки сцены заняло меньше секунды. Для этого и предназначен кэш. После каждого рендера сохраняются все данные об экспортированных в движок объектах, и если в следующий раз требуется отрендерить предыдущий объект, то используются сохранённые данные, а не создаются новые. Кэш не используется в следующих случаях:

1. У объекта меняется число вершин или полигонов;
2. Меняется число subdiv-ов или тип subdivision-a (с linear на Catmull-Clark или наоборот);
3. Меняется индекс используемой uv-развёртки;
4. Шейдер объекта требует атрибут, который не был сохранён в кэше;
5. Происходит рендер полного кадра.

В связи с этим необходимо всё время помнить о следующем:

1. Если у объекта создали полигональный кластер с новым материалом, то рендер это заметит только при обновлении кэша;
2. Если объект деформируется (костями, например), то рендер не будет обновлять местоположение вершин;
3. Если шейдер требует атрибут, который не может быть сгенерирован (например нужны текстурные координаты, а на объекте нет развёртки), то экспорт геометрии будет происходить каждый раз заново.

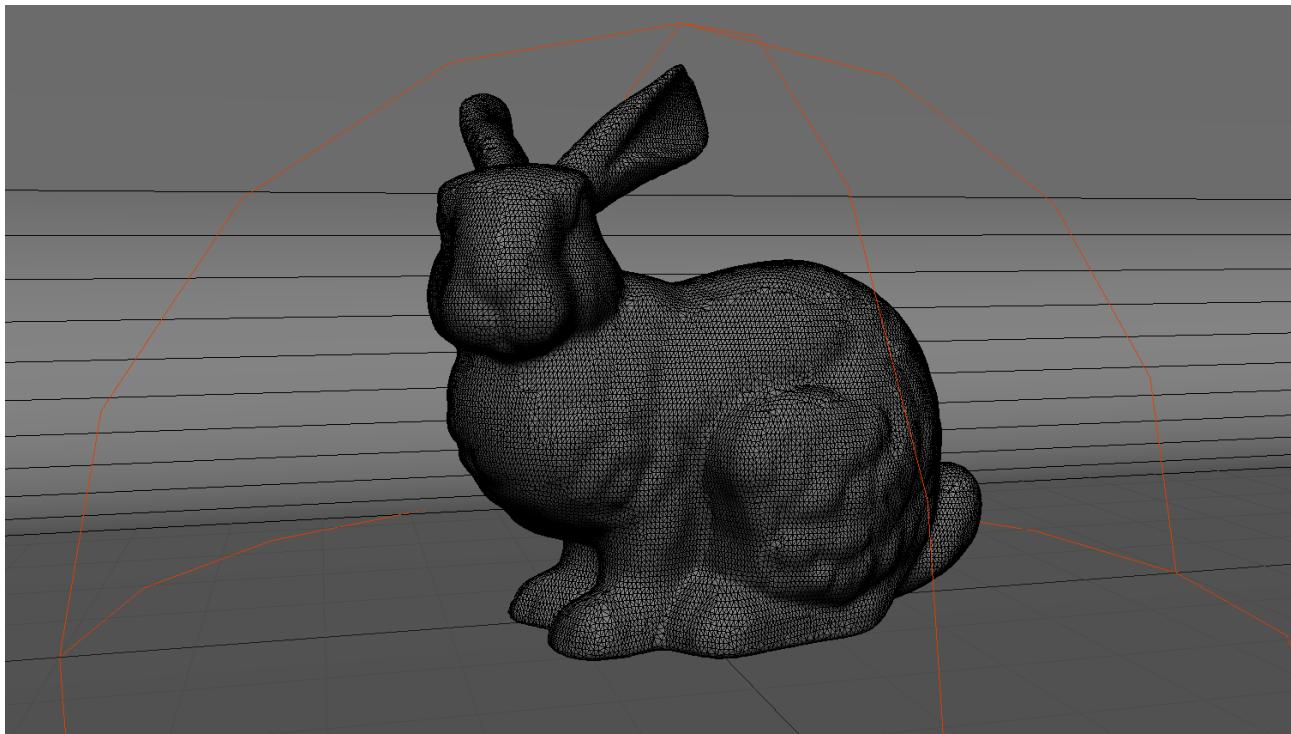
Чтобы заставить экспортировать всю геометрию каждый раз заново, достаточно выключить параметр **Cache Mesh** во вкладке **Performance** настроек рендера.



Так как пересоздание кэша происходит каждый раз в конце рендера, то самый простой способ обновить кэш одного конкретного объекта состоит в том, чтобы исключить этот объект из одного сеанса рендера, а потом включить его обратно.

18 Как использовать цветовые профили

Предположим у нас есть сцена: стеклянный заяц в простой студии.



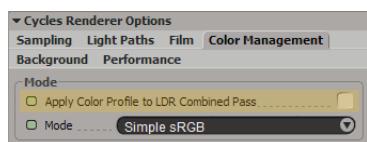
Отрендерим его.



По умолчанию, если содержимое **Combined** пасса выводится в **Region preview** или сохраняется в *.png формат, то к нему применяется преобразование цветового пространства в sRGB. Эту опцию можно отключить, выключив параметр

Apply Color Profile to LDR Combined Pass

во вкладке **Color Management** настроек рендера.



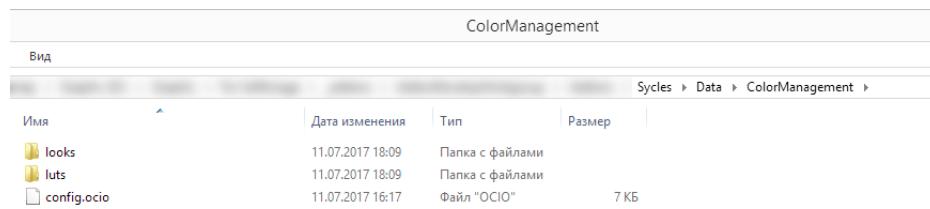
После отключения результат рендера сохраняется в линейном пространстве.



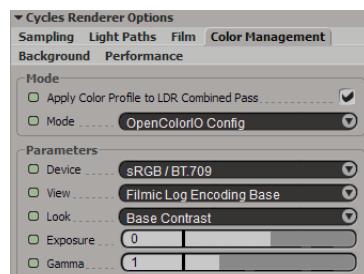
Есть возможность использовать цветовые профили OpenColorIO. Для этого в папку

...\\Sycles\\Data\\ColorManagement\\

надо положить файл config.ocio и все другие связанные с ним файлы.



Эти профили можно скачать с сайта <http://opencolorio.org/>. В аддон уже включены профили из дополнения “Filmic Blender” для Blender. Чтобы их использовать выбираем значение параметра Mode – OpenColorIO Config.



Результат рендера теперь получается таким:



Другой пример. Одна и та же сцен с одинаковыми параметрами освещения. В первом случае использовано цветовое пространство sRGB:



Render time: 3 m. 13.6 s. | Date: Thu Jul 13 11:58:23 2017 | Samples: 1024

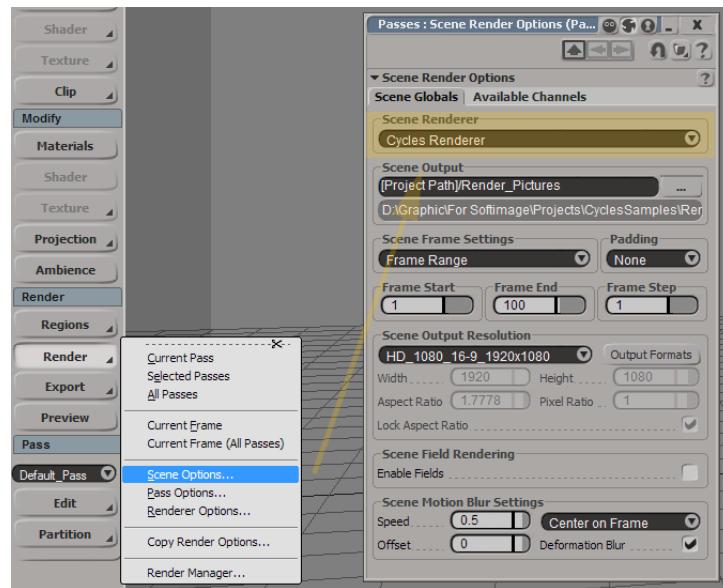
Во втором случае из конфига OpenColorIO:



Render time: 3 m. 27.3 s. | Date: Thu Jul 13 11:53:43 2017 | Samples: 1024

19 Как использовать shaderball-ы

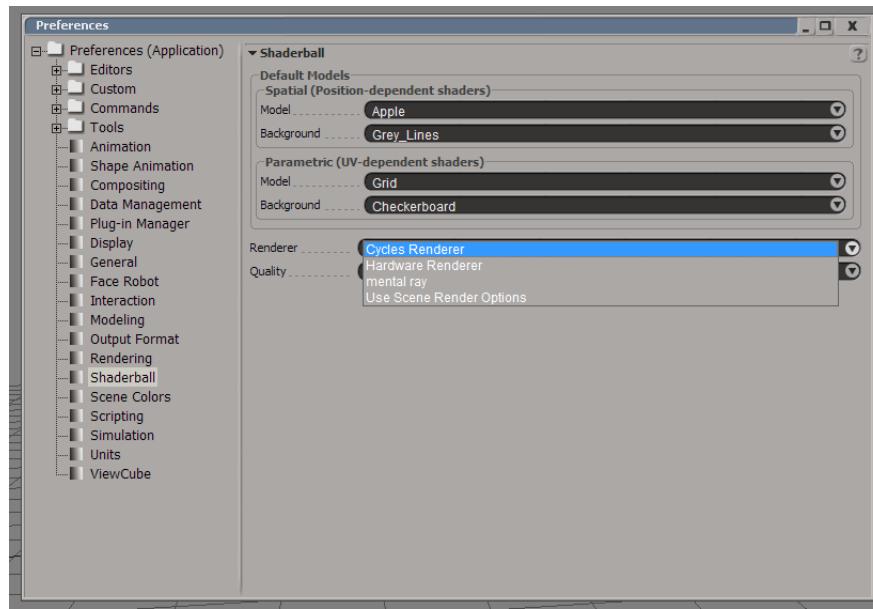
Чтобы shaderball-ы заработали, надо убедиться, что Softimage понимает какой рендер использовать при рендре. Сделать это можно, указав значение **Cycles Renderer** параметра **Scene Renderer** в окне **Render – Scene Options**....



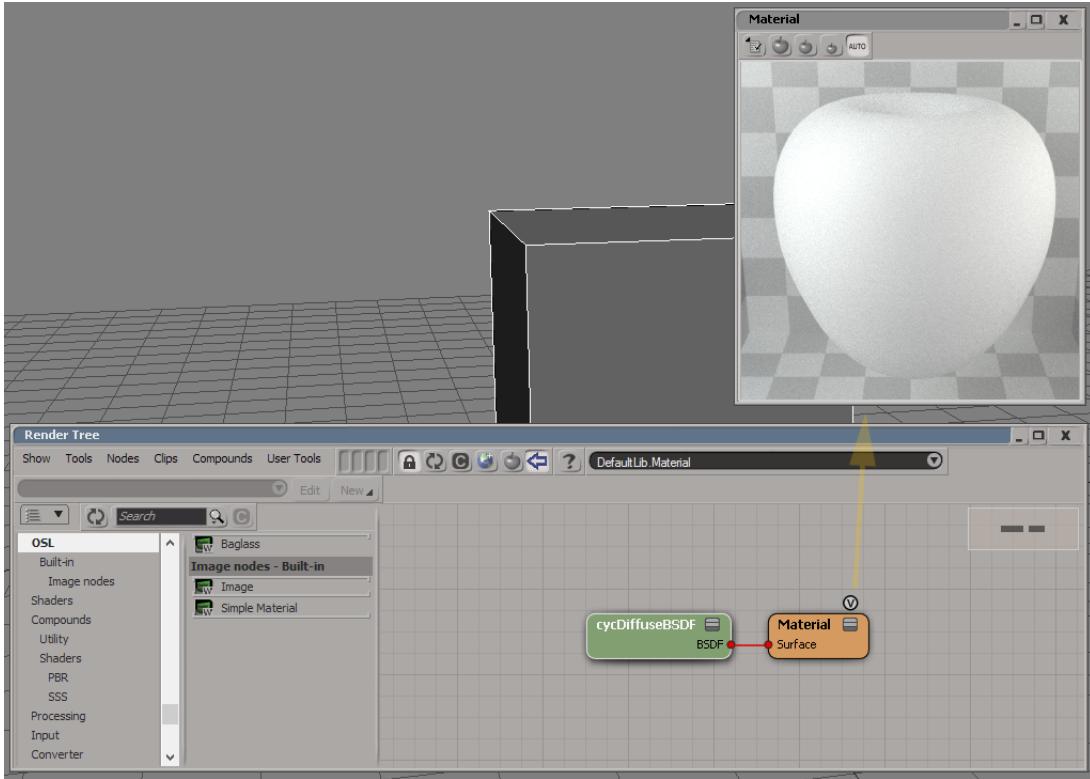
Либо можно в настройках

File – Preferences – Shaderball

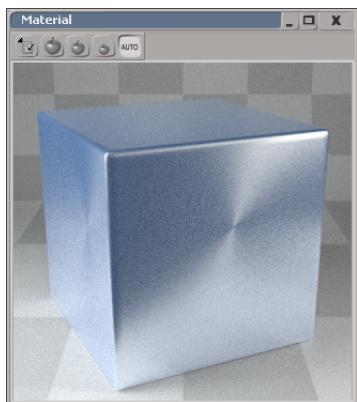
указать значение **Cycles Renderer** параметра **Renderer**.



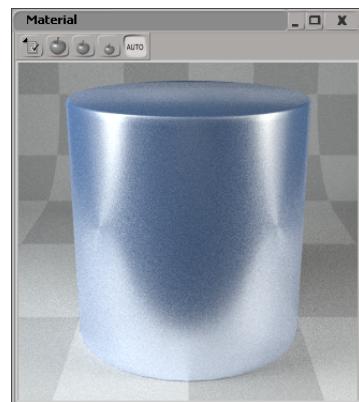
При нажатии на предпросмотр материала появляется окно с shaderball-ом



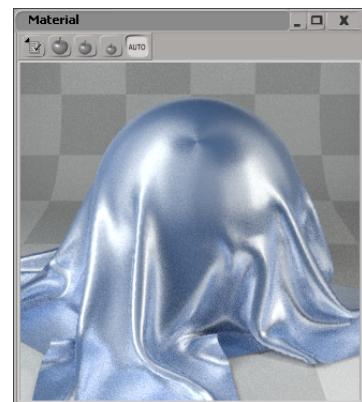
Кроме стандартного яблока можно использовать несколько других объектов.



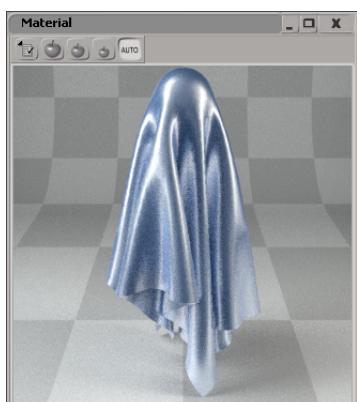
Sycles_Cube



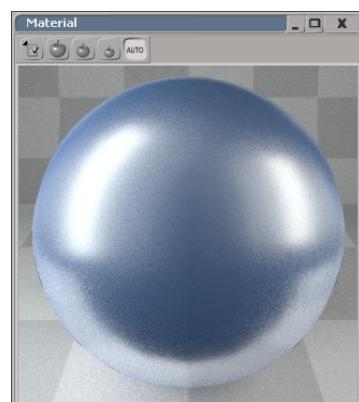
Sycles_Cylinder



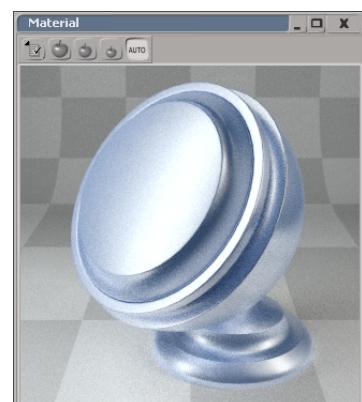
Sycles_Fabric01



Sycles_Fabric02

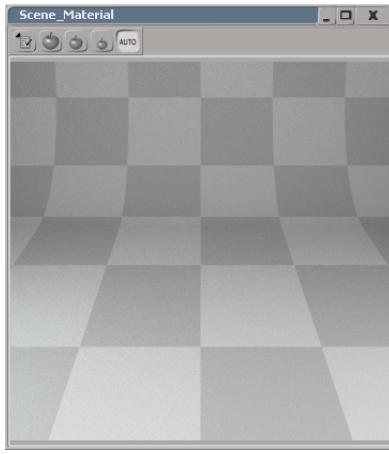


Sycles_Sphere



Sycles_Substance-like

Также есть новый объект для фона:



Sycles_Back

Чтобы добавить новые модели shaderball-ов надо скопировать содержимое папки

...\\Sycles\\Application\\Shaderballs

в папку

Path-to-Softimage\\Application\\Shaderballs

Настройки рендера shaderball-ов, положение камеры, источников света и шейдер фонового объекта захардкожены. Источников света всегда 3: слева, справа и позади камеры. На фоновый объект наложена шахматная текстура. Но всё-таки кое-какие параметры можно менять, редактируя файл

...\\Sycles\\Application\\Plugins\\config.ini

Параметры, доступные для редактирования:

`checker_light_color` – цвет в интервале (0, 1) светлой клетки шахматной текстуры;

`checker_dark_color` – цвет в интервале (0, 1) тёмной клетки шахматной текстуры;

`checker_scale` – число клеток в шахматной текстуре;

`render_mode` – значение 1 переключает режим рендеринга на `progressive`;

`render_samples` – число сэмплов;

`film_exposure` – аналог параметра `Film – Exposure` настроек рендеринга;

`max_bounces` – аналог параметра `Light Paths – Bounces – Max` настроек рендеринга;

`diffuse_bounces` – аналог параметра `Light Paths – Bounces – Diffuse` настроек рендеринга;

`glossy_bounces` – аналог параметра `Light Paths – Bounces – Glossy` настроек рендеринга;

`transmission_bounces` – аналог параметра `Light Paths – Bounces – Transmission` настроек рендеринга;

`clamp_direct` – аналог параметра `Sampling – Settings – Clamp Direct` настроек рендеринга;

`clamp_indirect` – аналог параметра `Sampling – Settings – Clamp Indirect` настроек рендера;

`shading_system` – значение 1 включает поддержку osl-шейдеров;

`left_light_color_r, g, b` – компоненты цвета в интервале (0, 1) для левого источника света;

`left_light_strength` – интенсивность левого источника света;

`left_light_ignore_glossy` – значение 1 делает левый источник света невидимым в отражениях;

`right_light_color_r, g, b` – компоненты цвета в интервале (0, 1) для правого источника света;

`right_light_strength` – интенсивность правого источника света;

`right_light_ignore_glossy` – значение 1 делает правый источник света невидимым в отражениях;

`center_light_color_r, g, b` – компоненты цвета в интервале (0, 1) для центрального источника света;

`center_light_strength` – интенсивность центрального источника света;

`center_light_ignore_glossy` – значение 1 делает центральный источник света невидимым в отражениях;

20 Как использовать OSL-шейдеры

Давайте создадим простейший OSL-шейдер. Для этого в любом текстовом редакторе пишем такой код:

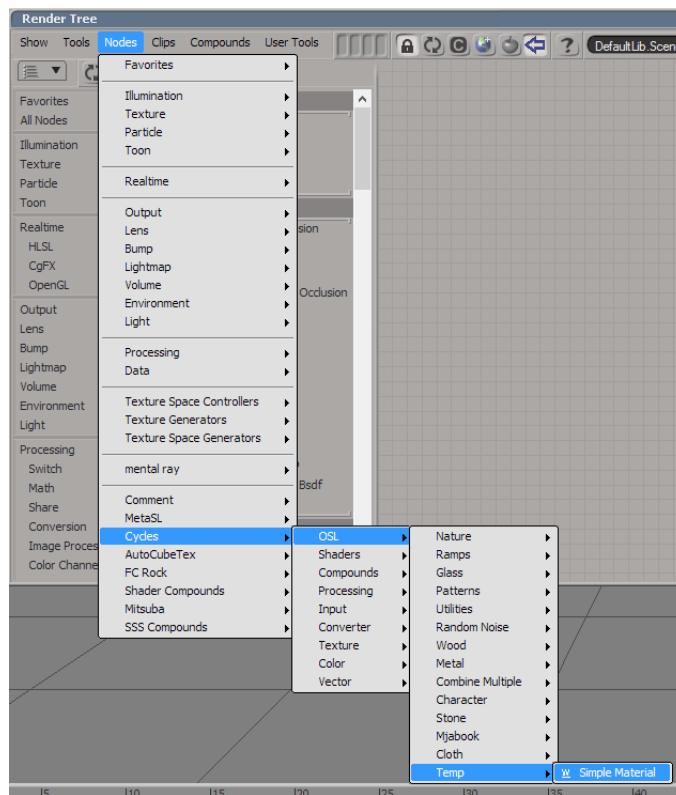
```
//XSICategory: Temp
shader simple_material(
    color Diffuse_Color = color(0.6, 0.8, 0.6),
    float Noise_Factor = 0.5,
    output closure color BSDF = diffuse(N))
{
    color material_color = Diffuse_Color*mix(1.0, noise(P*10.0), Noise_Factor);
    BSDF = material_color * diffuse(N);
}
```

Сохраняем в файл с любым названием и расширением *.osl. Помещаем этот файл в папку

\Application\osl\

любой воркгруппы. Теперь запускаем Softimage и видим, что наш шейдер Simple Material появился в категории

Cycles – OSL – Temp.



В какую категорию помещается шейдер зависит от того, что написано в первой закомментированной строке шейдера после слов `XSICategory`. Если ничего не написано или такой строки вообще нет, то шейдер будет помещён в категорию

Cycles – OSL.

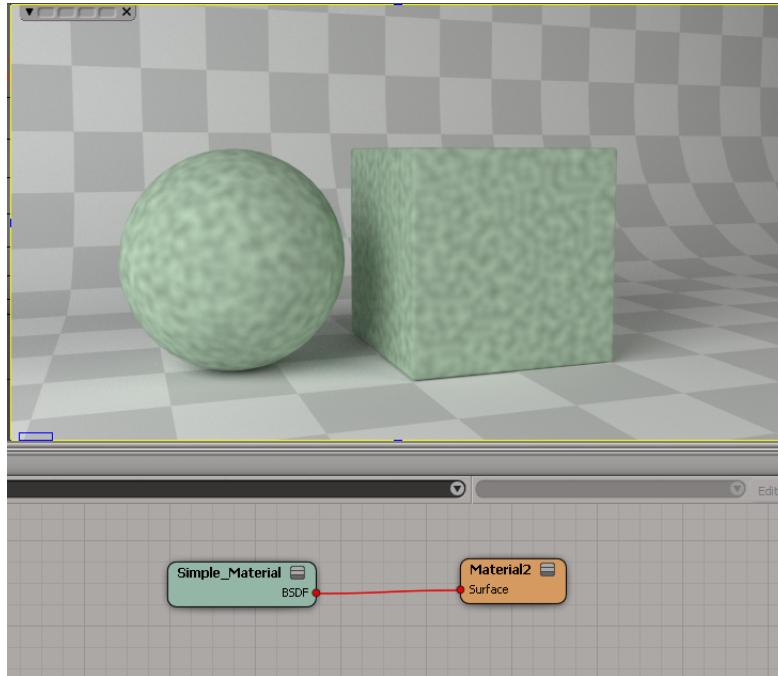
Если написано что-то вроде

Category/Subcategory/Subsubcategory

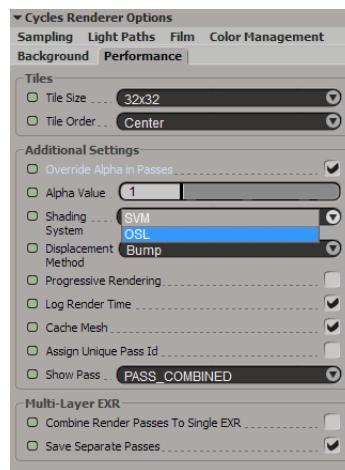
то шейдер будет помещён в категорию

Cycles – OSL – Category – Subcategory – Subsubcategory.

Подключаем выходной порт нашего шейдера Simple Material к порту Surface материала и видим результат.



Если результата не видно, то это потому что рендер не понимает, что надо рендерить OSL-шейдеры. Объяснить это ему можно, выбрав во вкладке **Performance** настроек рендера значение параметра **Shading System** равное **OSL**.



Использовать режим рендера OSL-шейдеров надо очень осторожно и только при крайней необходимости, так как при нём сильно увеличивается время рендера. Вот пример: простая сцена со стеклянным шариком. Шейдер стекла – это обычный GlassBSDF. Без поддержки OSL-шейдеров:



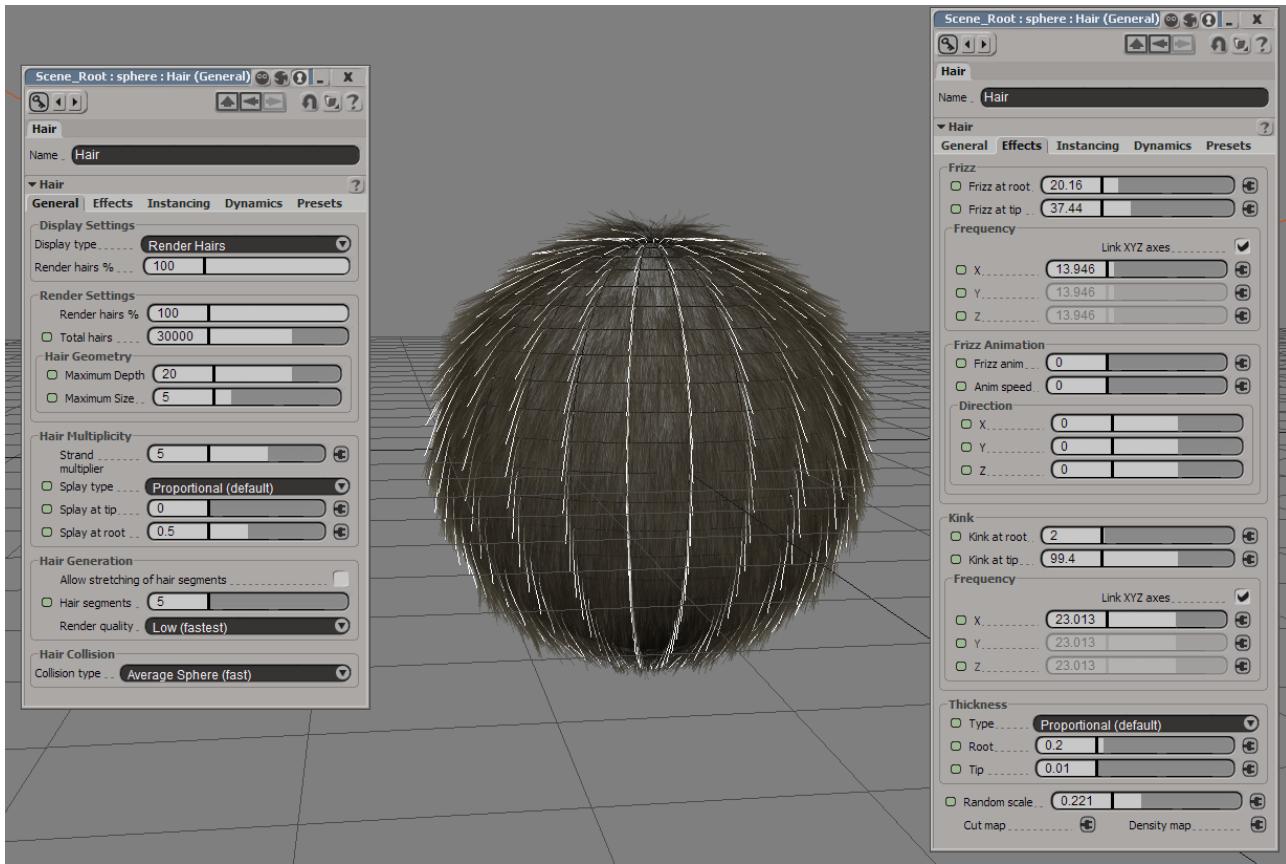
С поддержкой OSL-шейдеров:



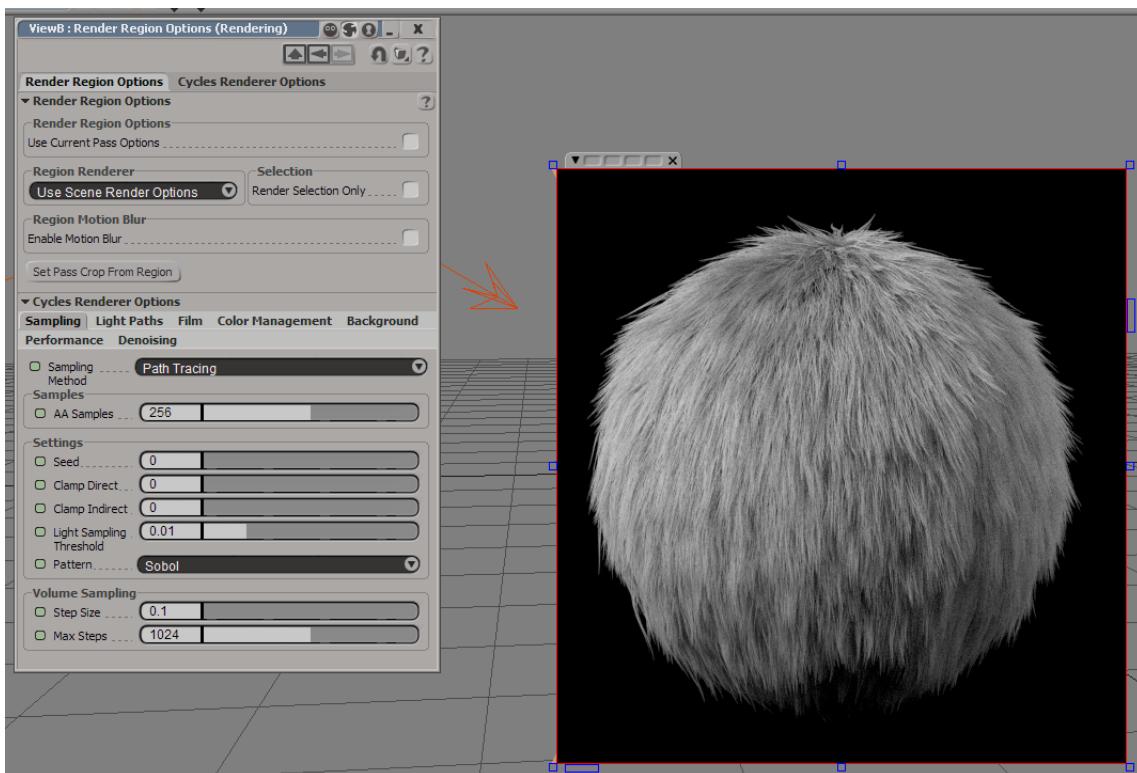
Результаты идентичны, но время рендера отличается почти в два раза.

21 Как рендерить стандартные XSI-шные волосы

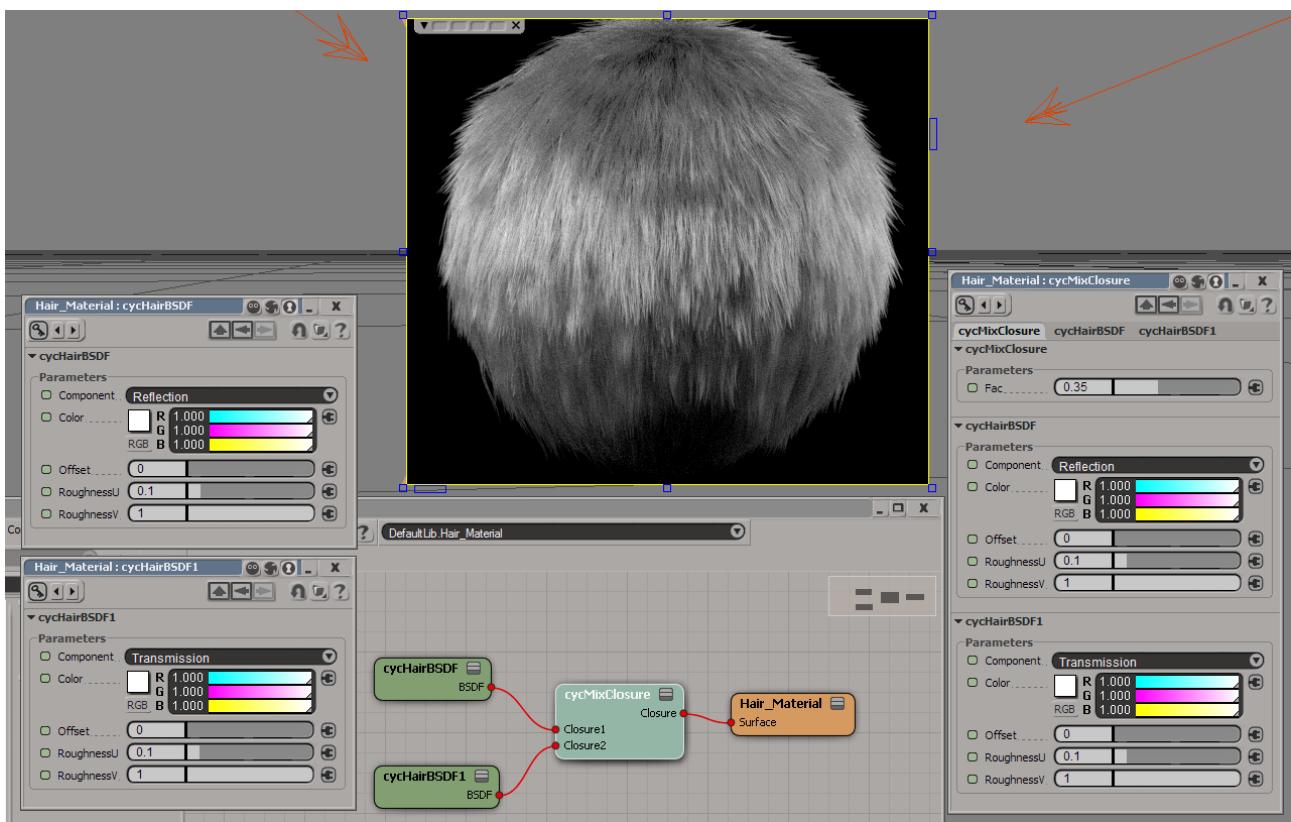
Предположим у нас есть сцена: шарик с волосами, сделанными стандартными средствами Softimage.



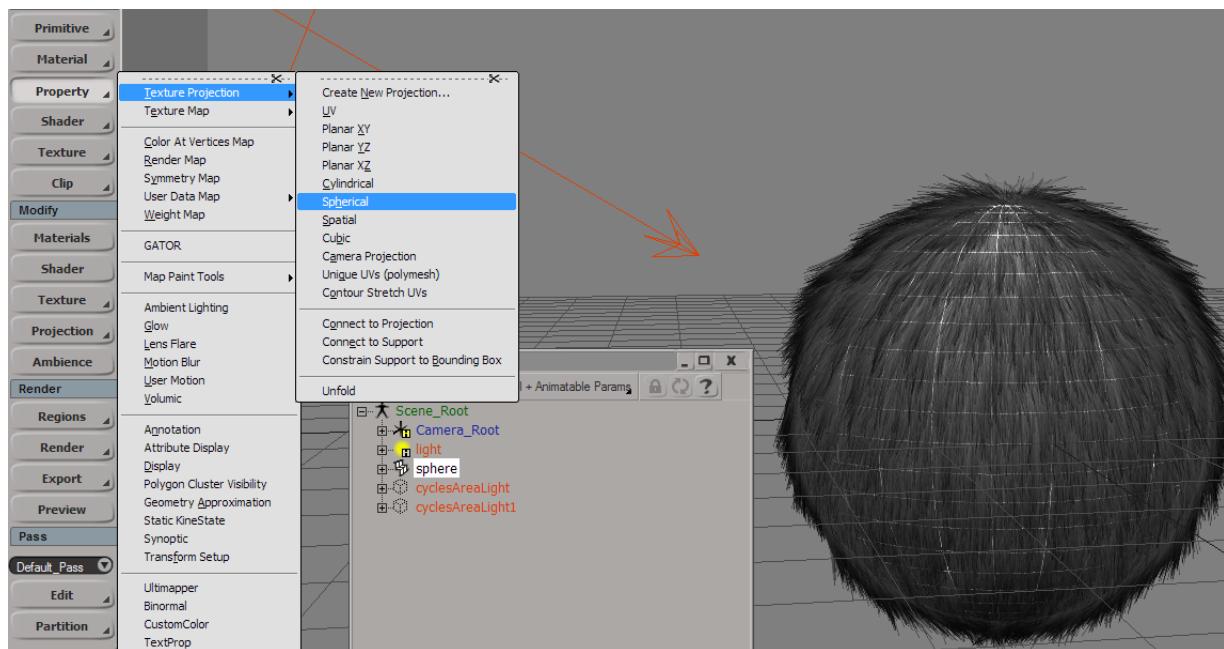
Отрендерим его. Здесь два параметра влияют на толщину волосков: толщина у корней берётся из **Thickness – Root**, а толщина у концов из **Thickness – Tip**.



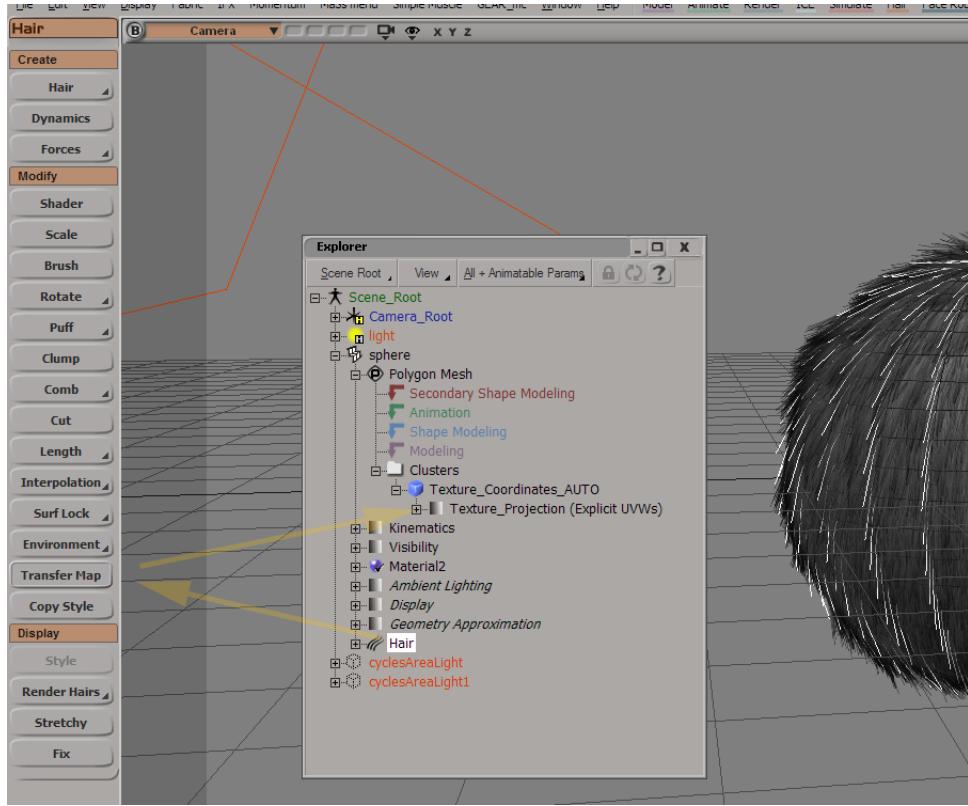
Выделяем волосы и назначаем на них новый материал. В нём смешиваем два экземпляра ноды HairBSDF с коэффициентом 0.35. Первая нода имеет режим Reflection, вторая – Transmission. Рендерим.



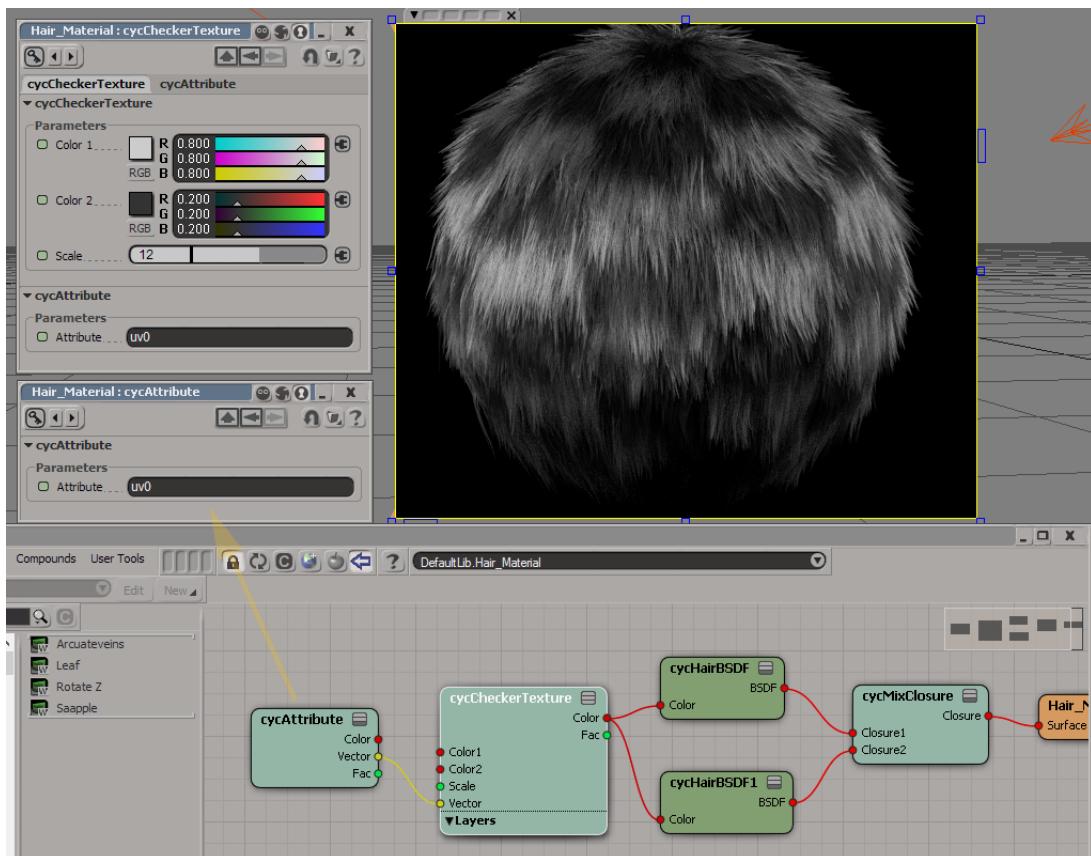
Единственный способ передать в рендер цвет каждого волоска состоит в использовании текстурных координат. Добавляем сферические текстурные координаты.



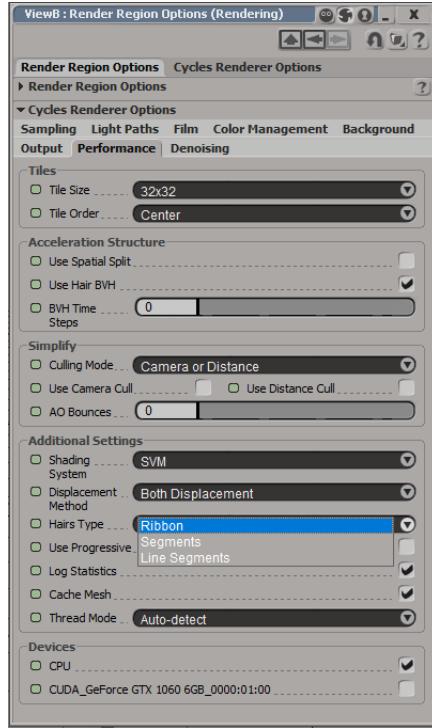
Переносим их на волосы.



При рендре волос все текстурные координаты на объекте записываются в векторные атрибуты с именами `uv0`, `uv1` и т.д. В материал волос добавляем ноду `Attribute` со значением `uv0` и подключаем выход `Vector` ко входу `Vector` ноды `CheckerTexture`, у которой выход `Color` подсоединяется к портам `Color` нод `HairBSDF`.



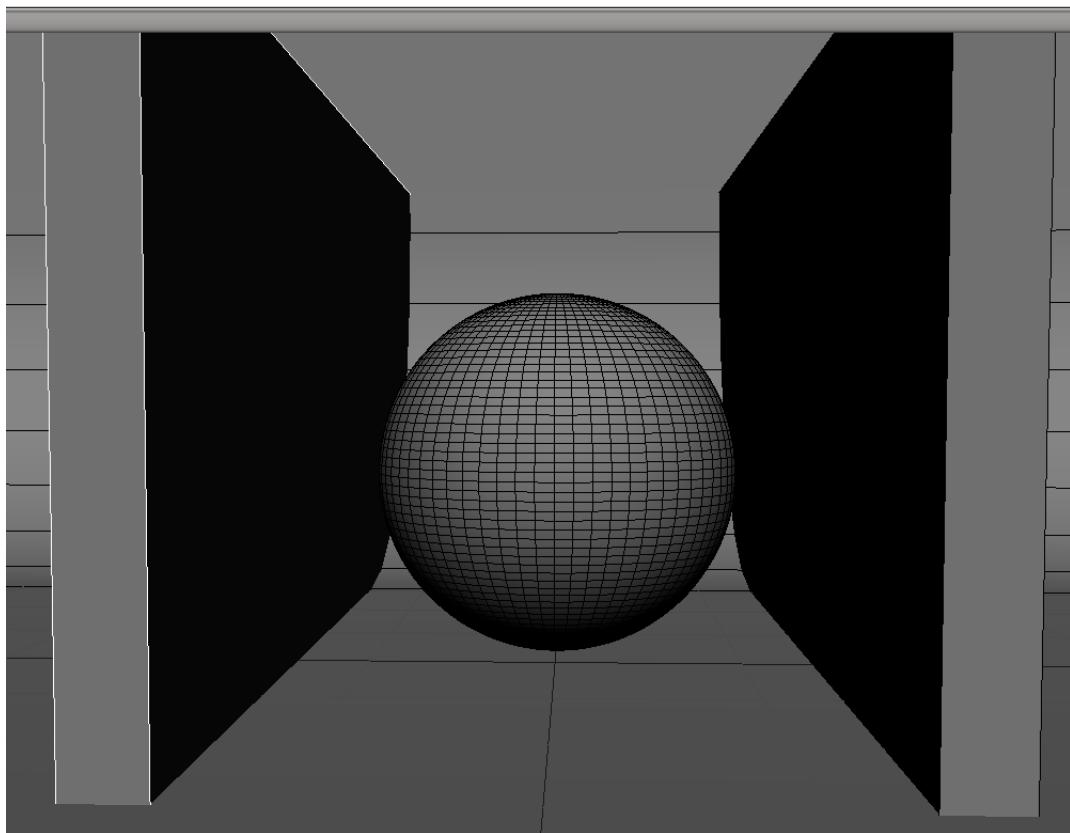
Теперь видно, что волоски имеют шахматную раскраску. Качество геометрии волос можно указать в настройках рендера `Performance – Hair Type`.



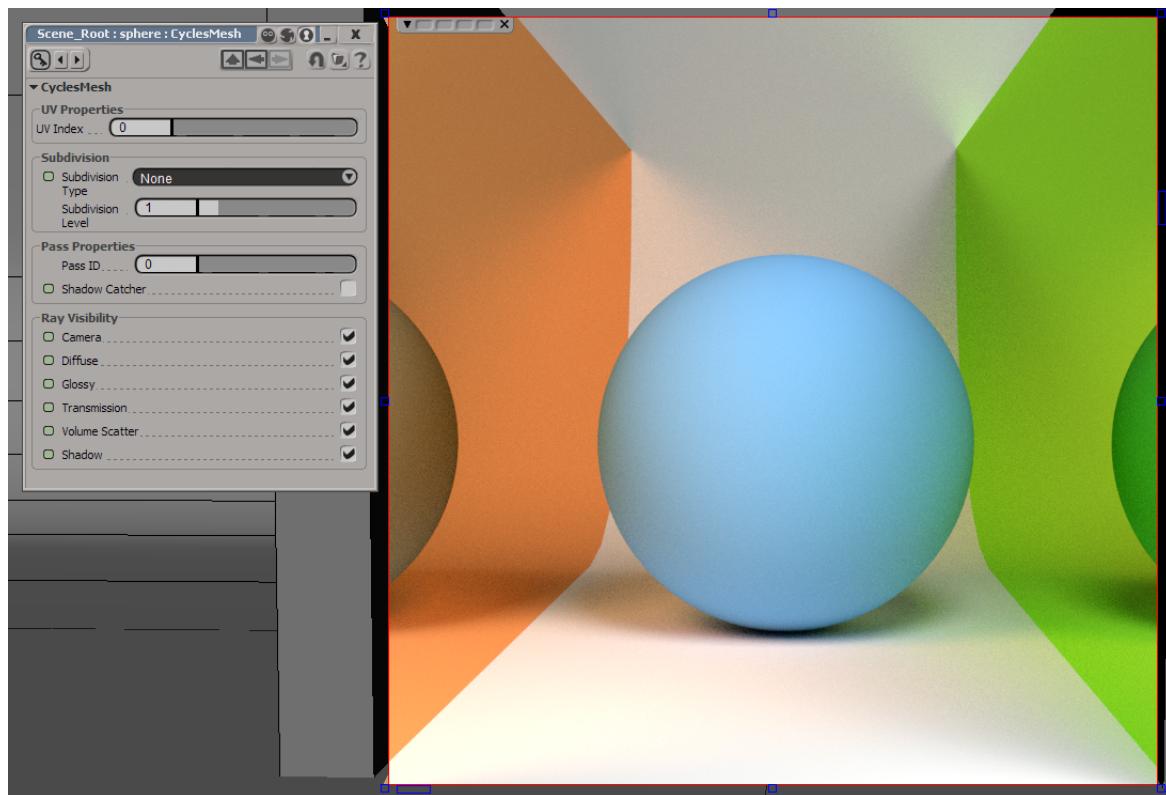
Если выбран режим **Ribbon**, то волоски будут рендериться плоскими плашками, ориентированными в сторону камеры. В режиме **Segments** волосы имеют цилиндрическую форму, а в режиме **Line Segments** волосы рендерятся в виде изолированных цилиндрических сегментов с разрывами на сгибах.

22 Как использовать видимость объектов

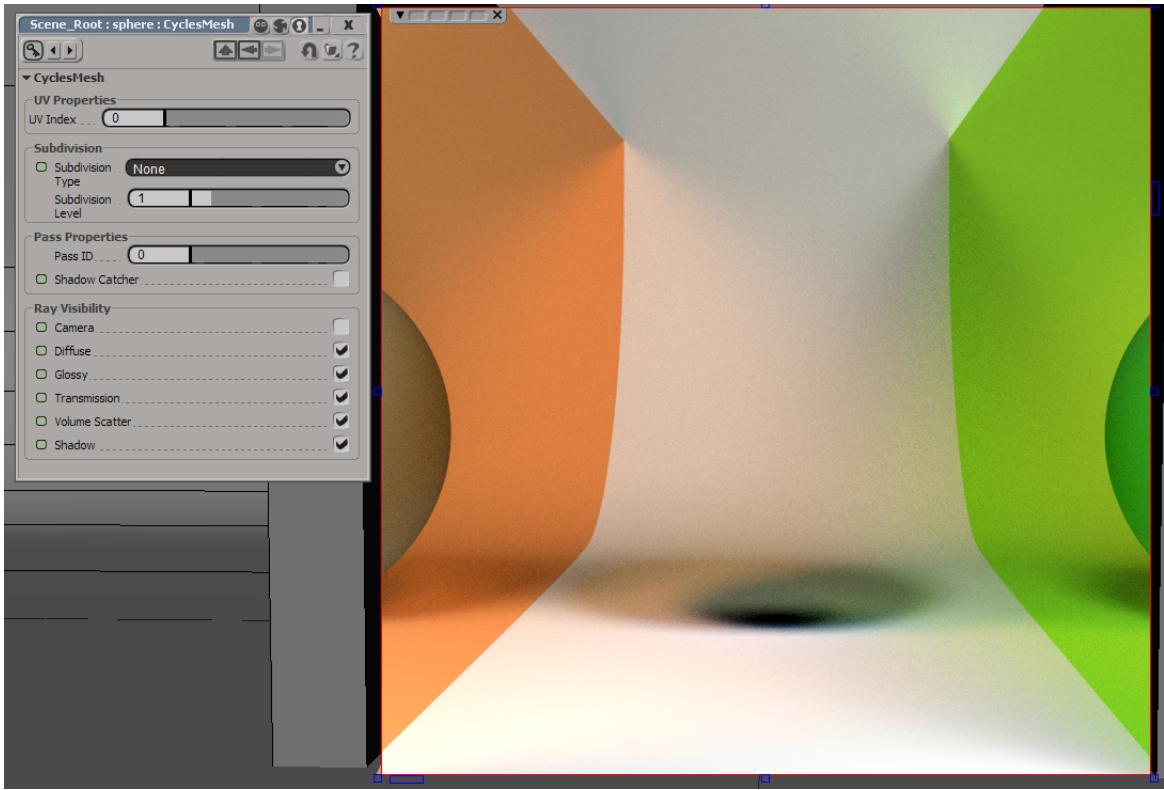
Предположим у нас есть сцена с шариком и кубиками.



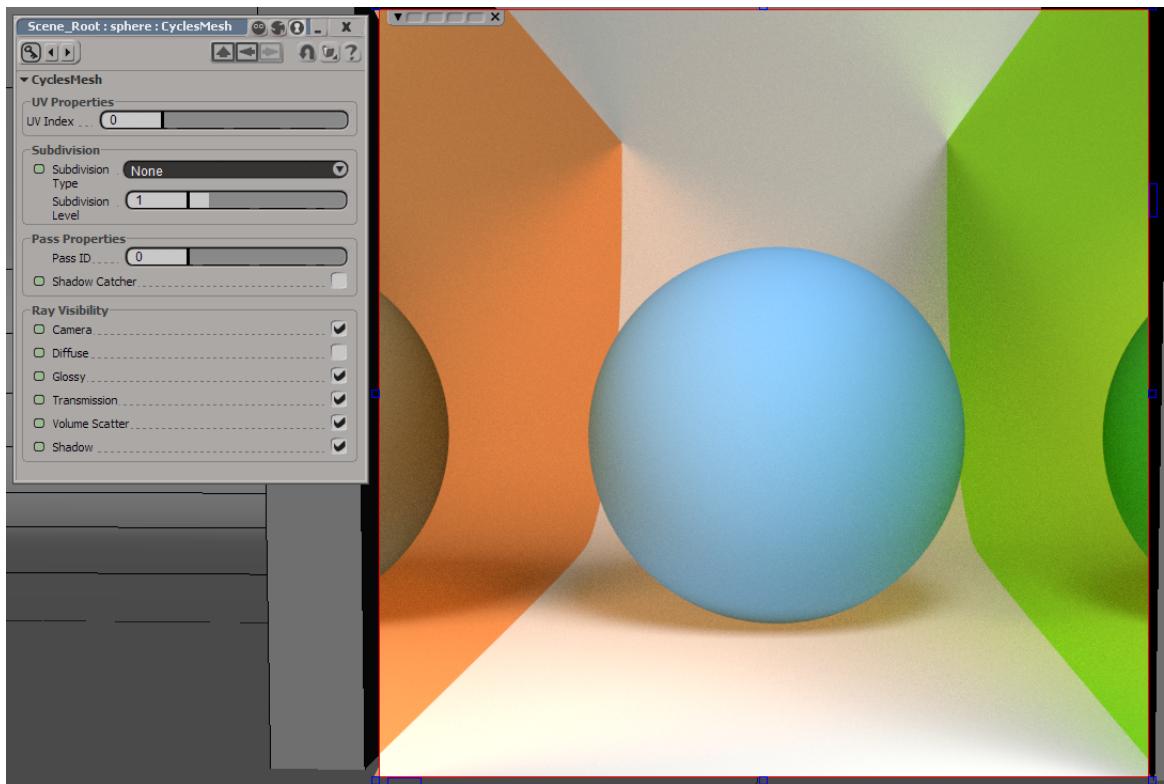
Добавляем на сферу свойство CyclesMesh и рендерим.



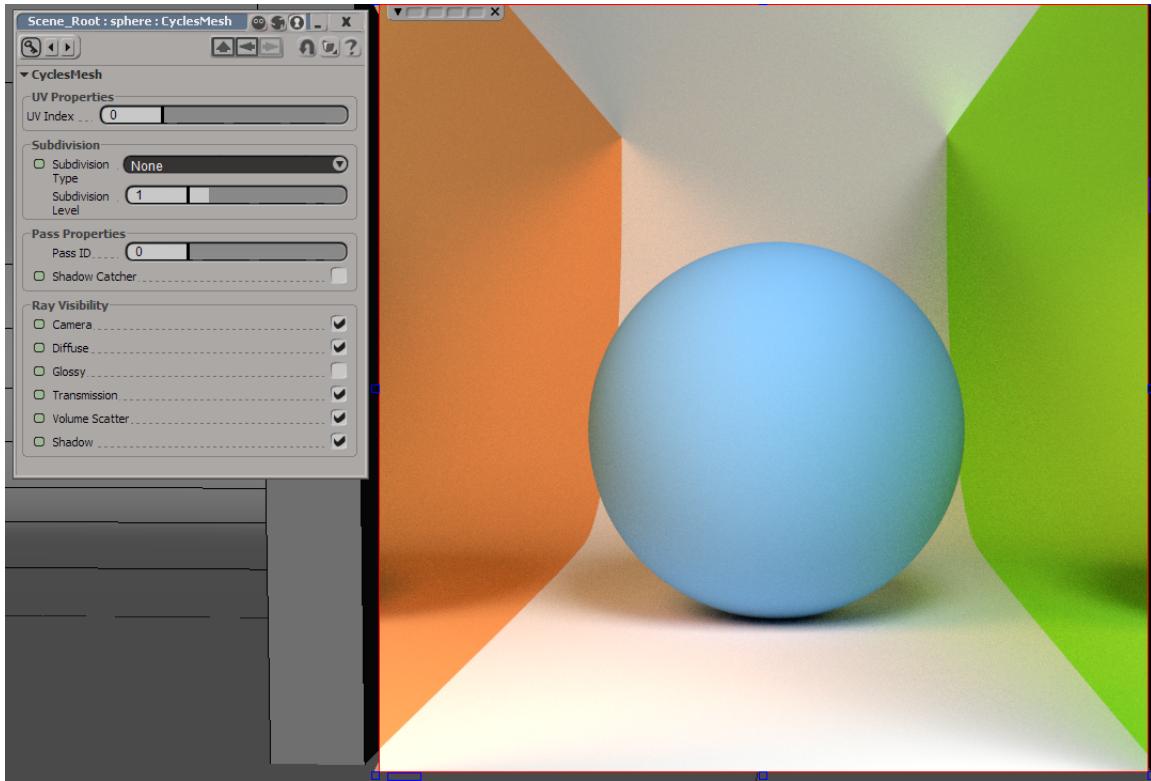
При выключенном параметре Ray Visibility – Camera объект не рендерится напрямую, но его видно в отражениях и он участвует в расчёте GI.



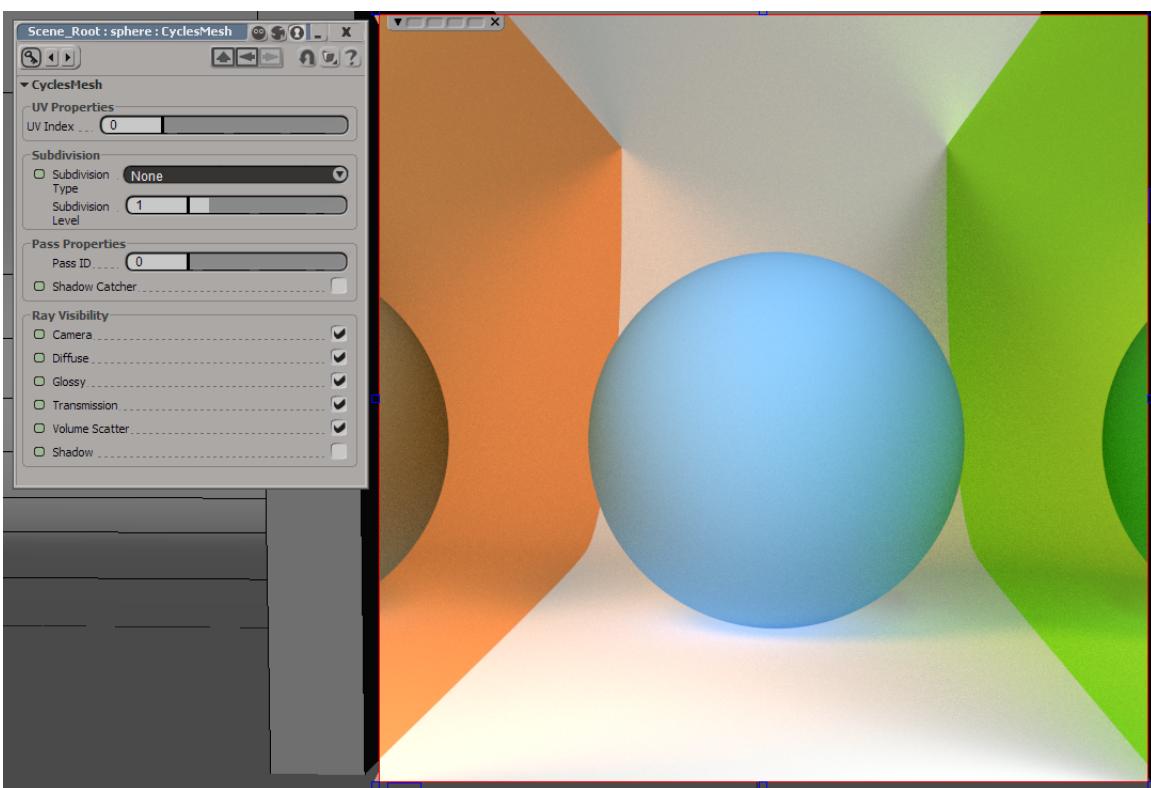
При выключенном параметре Ray Visibility – Diffuse объект не участвует в расчёте GI.



При выключенном параметре Ray Visibility – Glossy объект становится не виден в отражениях.

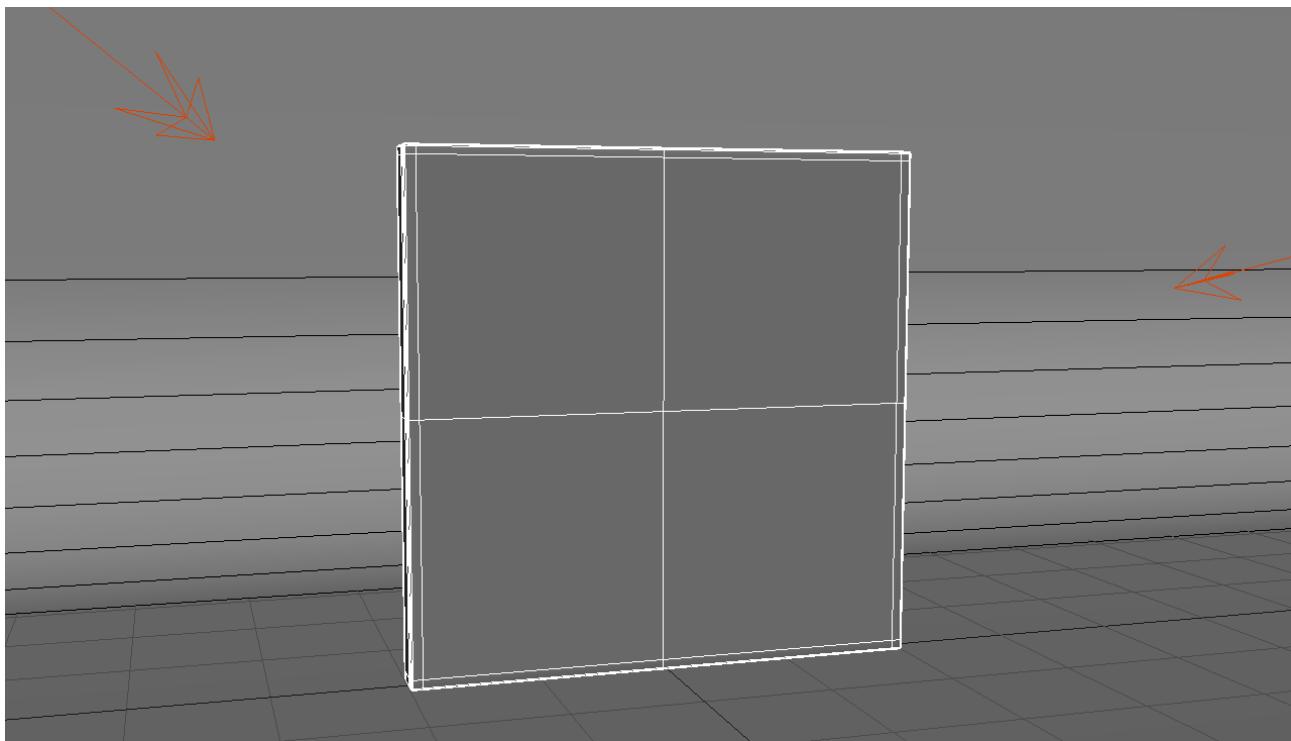


При выключенном параметре Ray Visibility – Shadow для объекта не считаются тени.



23 Как использовать несколько uv-координат

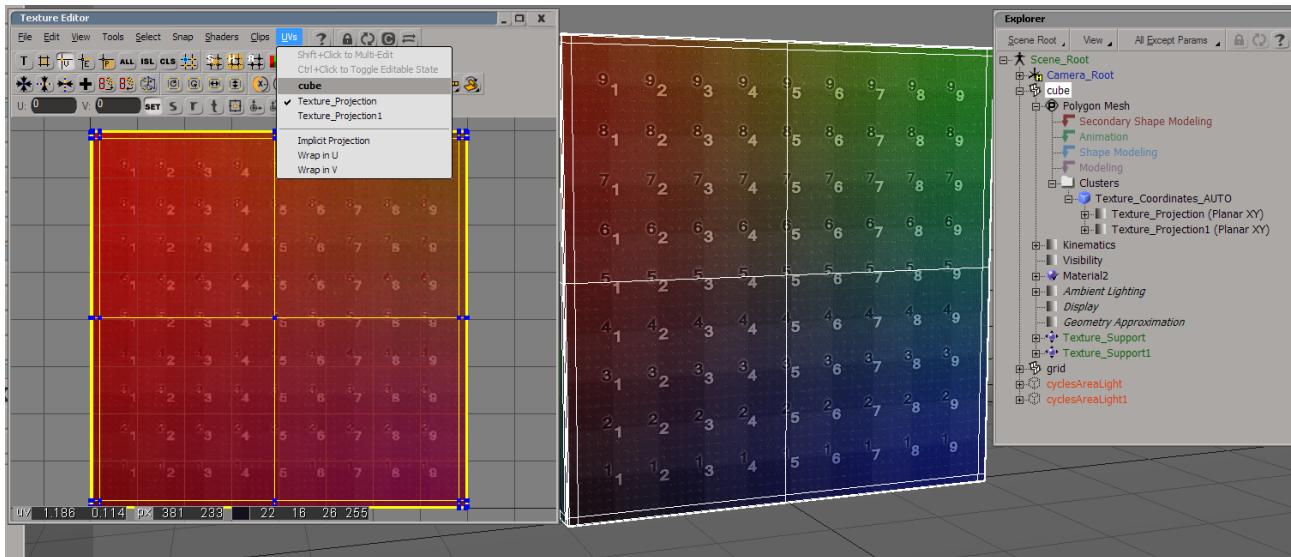
Предположим у нас есть простая сцена – квадрат с парой источников света.



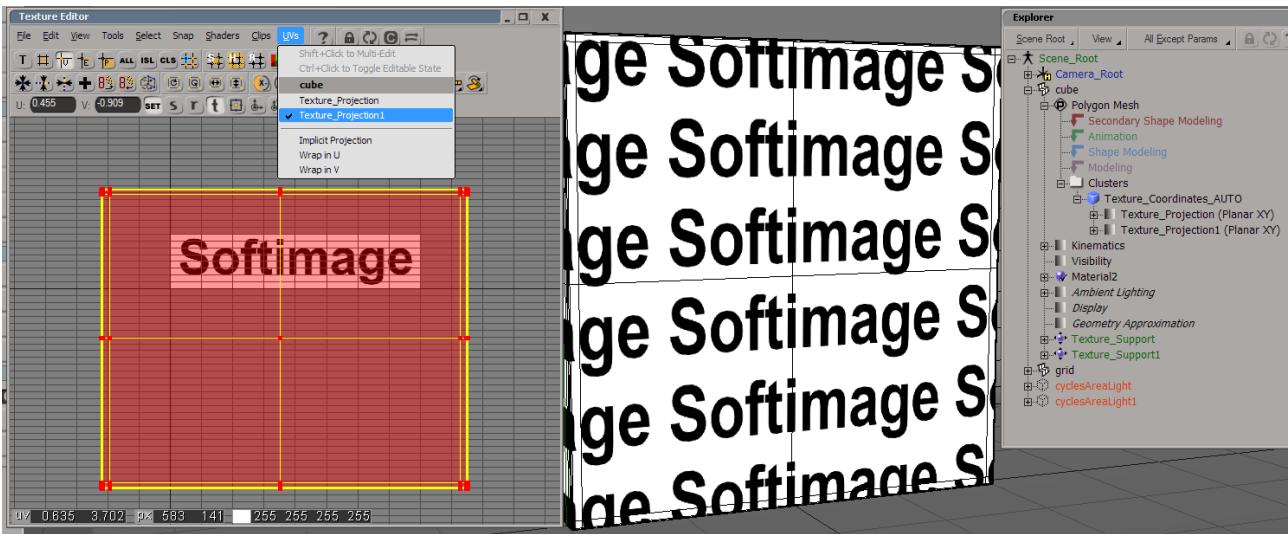
Хочется сделать так, чтобы это была как будто кирпичная стена и на ней что-то написано. Например такое:

Softimage

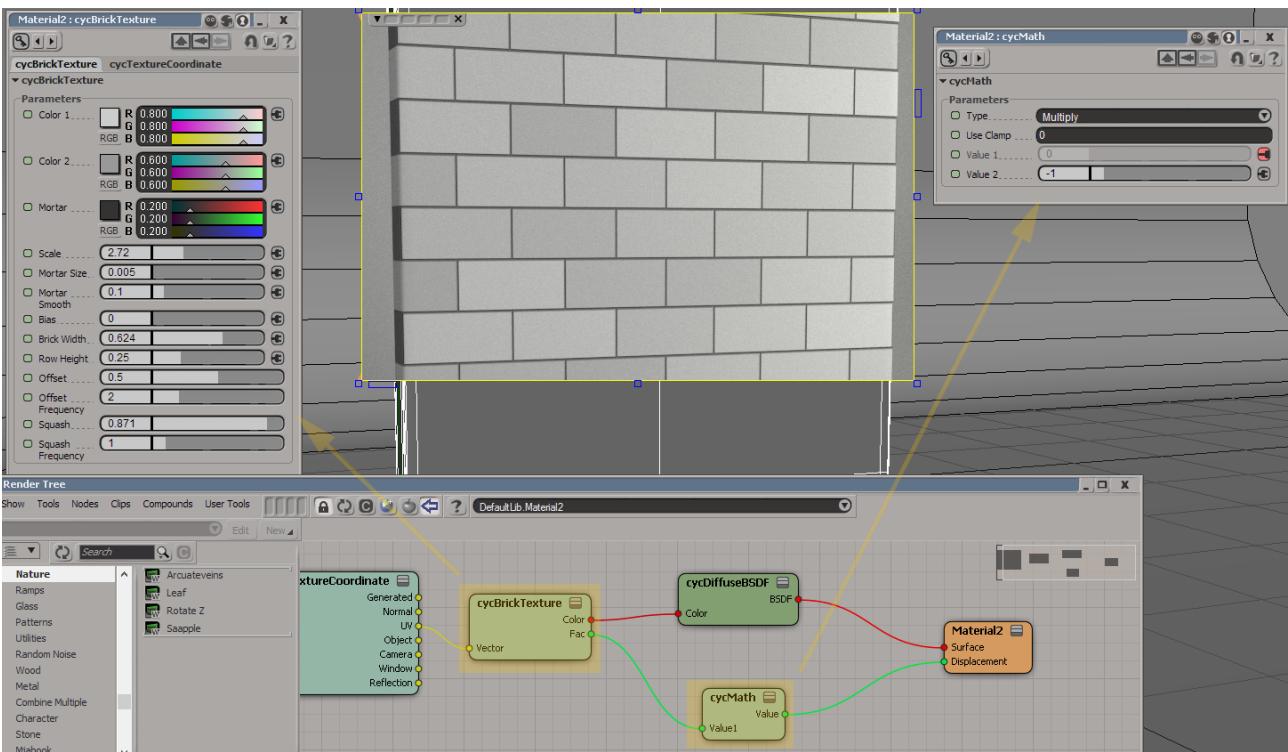
Добавляем на наш квадрат две текстурные проекции на плоскость XY . Одну оставляем неизменной.



Вторую проекцию масштабируем так, чтобы надпись располагалась в желаемом месте.

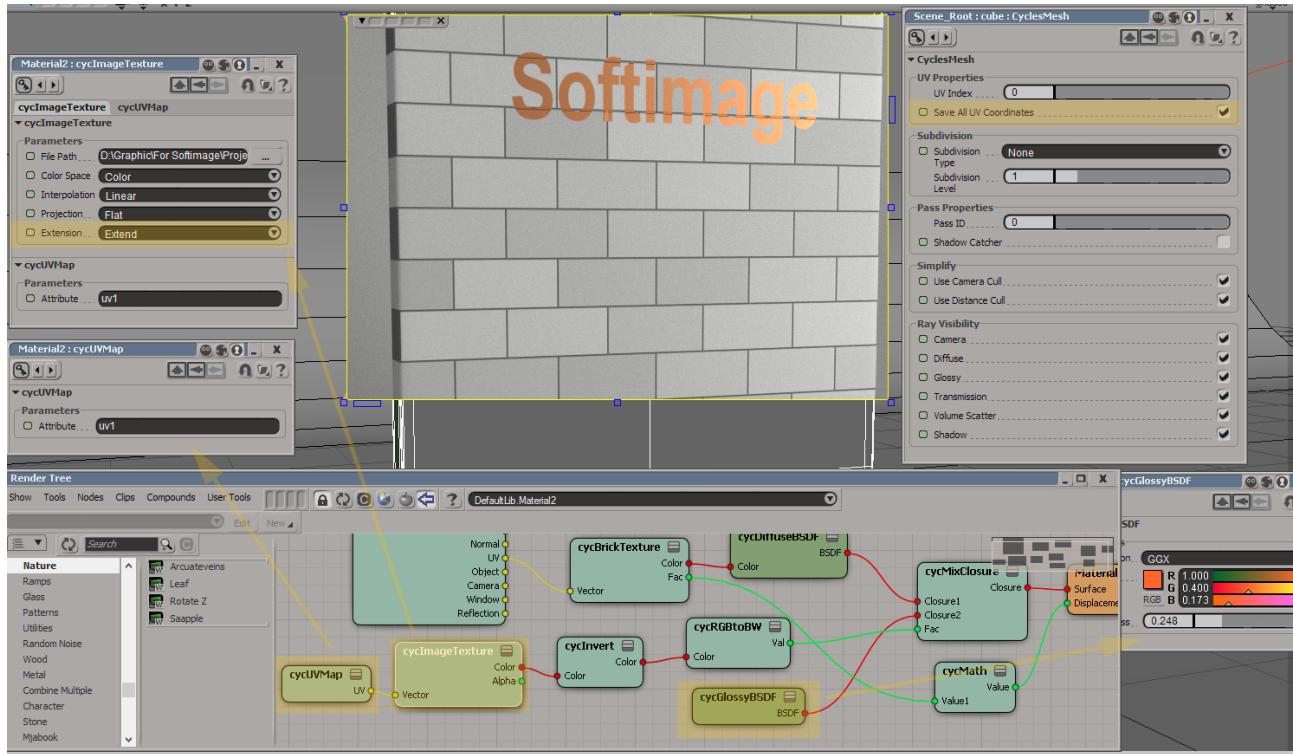


На квадрат назначаем материал, изображающий кирпичную стену. Для задания текстурных координат используем ноду **TextureCoordinate**. Она содержит в себе данные о первой (неизменённой) текстурной проекции.



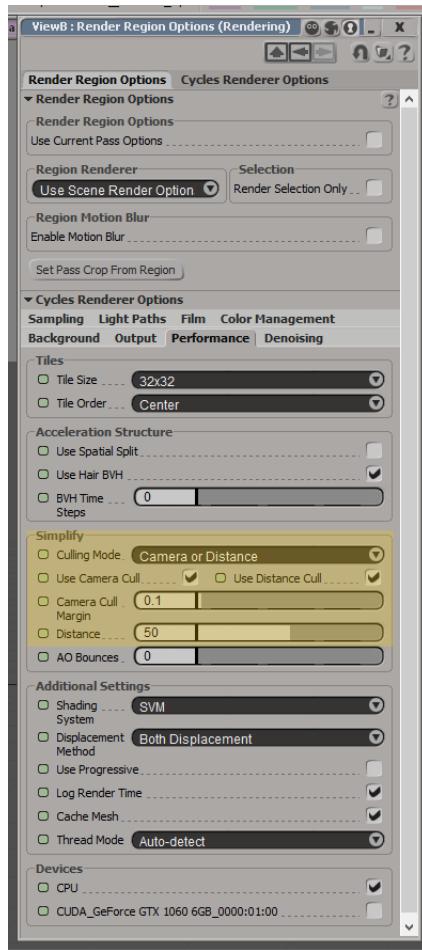
Теперь надо сказать рендеру, чтобы он использовал также и вторые текстурные координаты. Для этого добавляем на квадрат свойство **Cycles Mesh** и включаем в нём параметр **Save All UV Coordinates**. Теперь с помощью ноды **UVMap** можно выбрать данные о любых текстурных координатах на объекте. Все они имеют имена **uv0**, **uv1**, ...

Добавляем в материал картинку с надписью. Указываем в ноде **ImageTexture** значение параметра **Extension** равное **Extend**. Это надо для того, чтобы текстура не тайллась. Задаём в качестве текстурных координат результат ноды **UVMap** с атрибутом, равным **uv1**. Ну и наконец используем эту текстуру в качестве маски.



24 Что такое Camera Cull и Distance Cull

В разделе Performance — Symplyfy настроек рендерера можно включить параметры Use Camera Cull и Use Distance Cull.

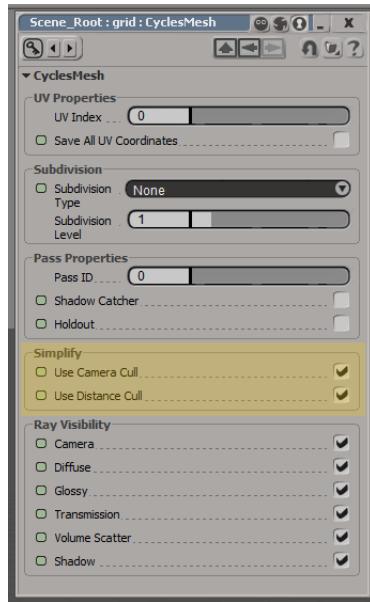


При включенном параметре **Use Camera Cull** каждый объект на сцене, прежде чем добавляться в движок для рендера проверяется, видим ли он из камеры. Если объект находится вне конуса видимости камеры, то значение **Camera Cull** для него становится **True** и он не отображается на рендере. Параметр **Camera Cull Margin** показывает, насколько далеко объект должен быть от границы видимости, чтобы всё-таки участвовать в рендере. Чем больше этот параметр, тем дальше объект может отстоять от конуса видимости и при этом принимать значение параметра **Camera Cull = False**.

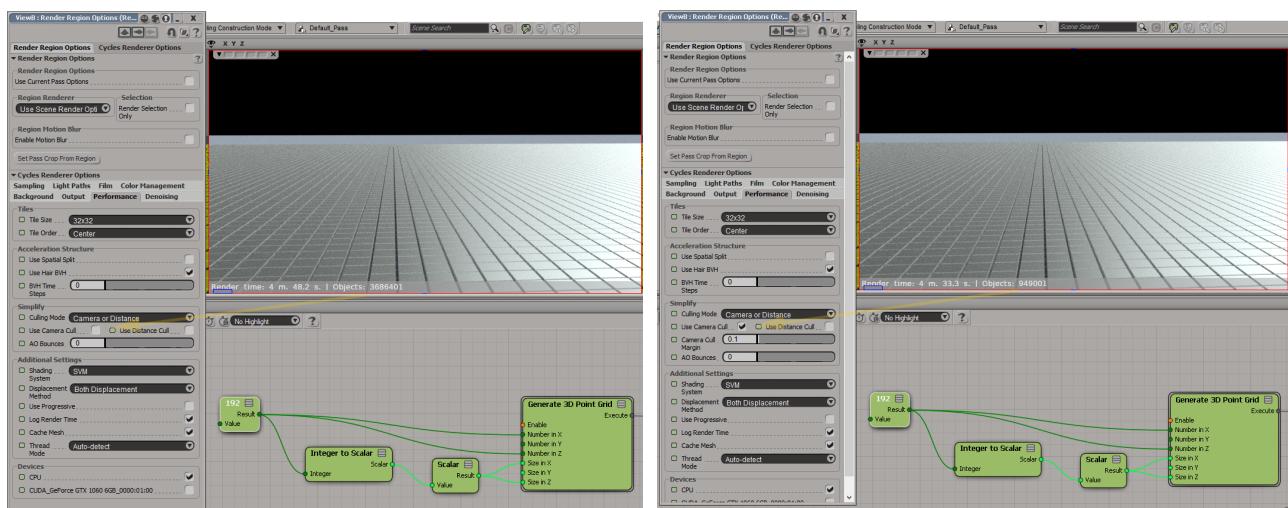
Параметр **Use Distance Cull** включает проверку местоположения объекта на удалённость от камеры. Если объект удалён на расстояние, большее чем значение параметра **Distance**, то для этого объекта значение параметра **Distance Cull** считается равным **True** и объект исключается из рендера.

Значение параметра **Culling Mode** может принимать два значения: **Camera or Distance** и **Camera and Distance**. Значение этого параметра не имеет значения, если включен только один метод исключения объектов из рендера. Но если включены оба метода, то для режима **Camera or Distance** объект исключается в случае, когда он должен быть исключён хотя бы по одному из методов, а для режима **Camera and Distance** исключается в случае, когда он должен быть исключён по обоим методам.

Каждый объект можно принудительно исключить из проверок. Для этого в свойствах **CyclesMesh** и **CyclesHairs** надо выключить параметры **Use Camera Cull** и **Use Distance Cull**. Для объектов, на которых нет этих свойств, параметры считаются включенными.



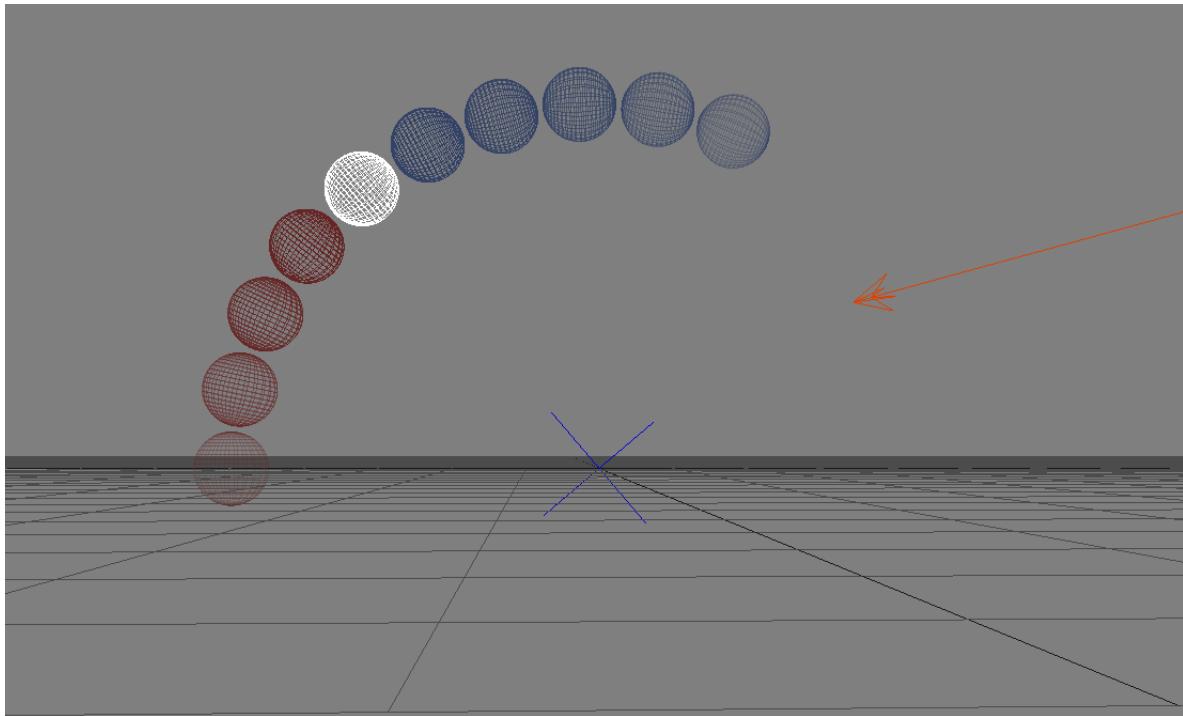
Проведём эксперимент. Сгенерируем много кубиков и отрендерим в двух режимах: сначала будем учитывать все объекты, а потом отрендерим с отсечением невидимых из камеры. Результаты:



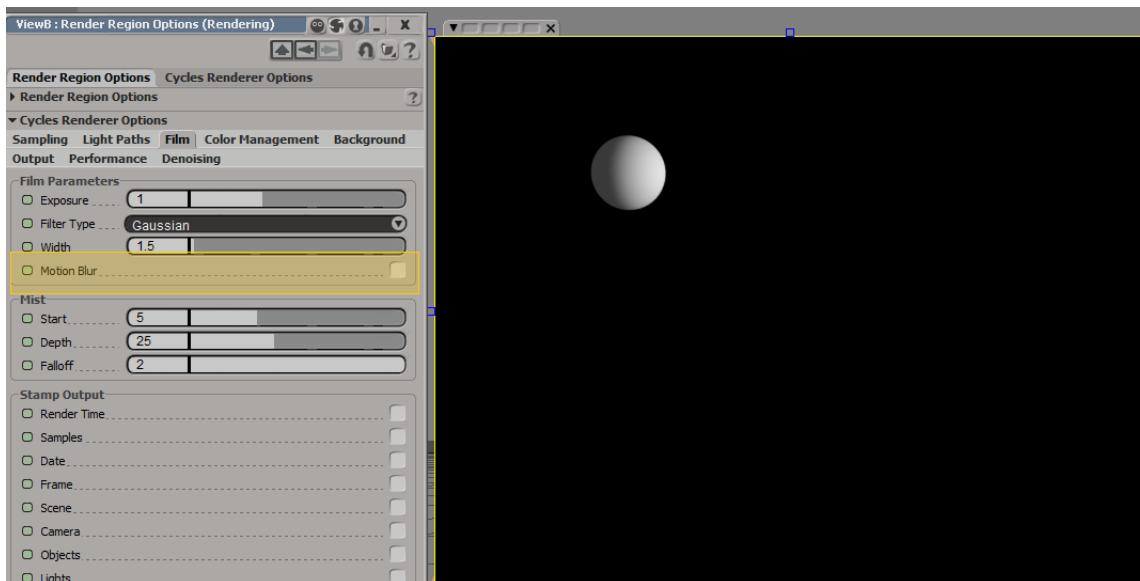
В первом случае число объектов 3 686 401, время рендера 4 мин. 48 сек. Во втором случае объектов 949 001, а время рендера 4 мин. 33 сек. Чуть быстрее, но не на сильно много, так как при включенных проверках их надо делать, что само по себе увеличивает время рендера.

25 Какрендерить motion blur

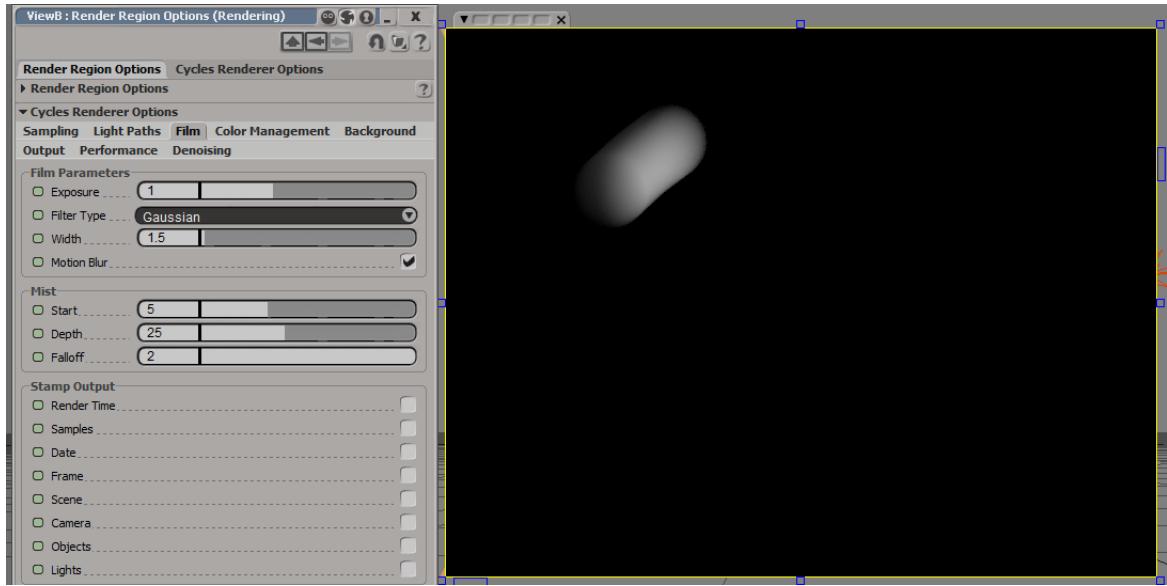
Предположим у нас есть сцена, в которой шарик крутится по кругу. Реально он сделан дочерним к null-y, расположенному в центре сцены, а тот уже крутится по оси z.



Чтобы включить эффект motion blur надо активировать параметр Film – Motion Blur в настройках рендерера.

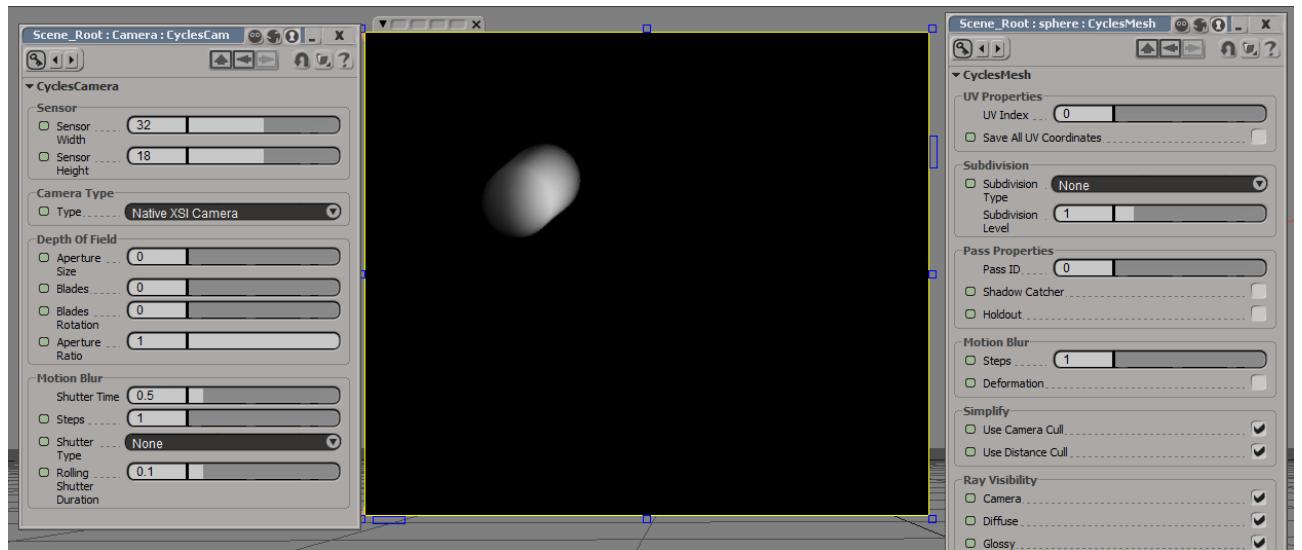


Результат.

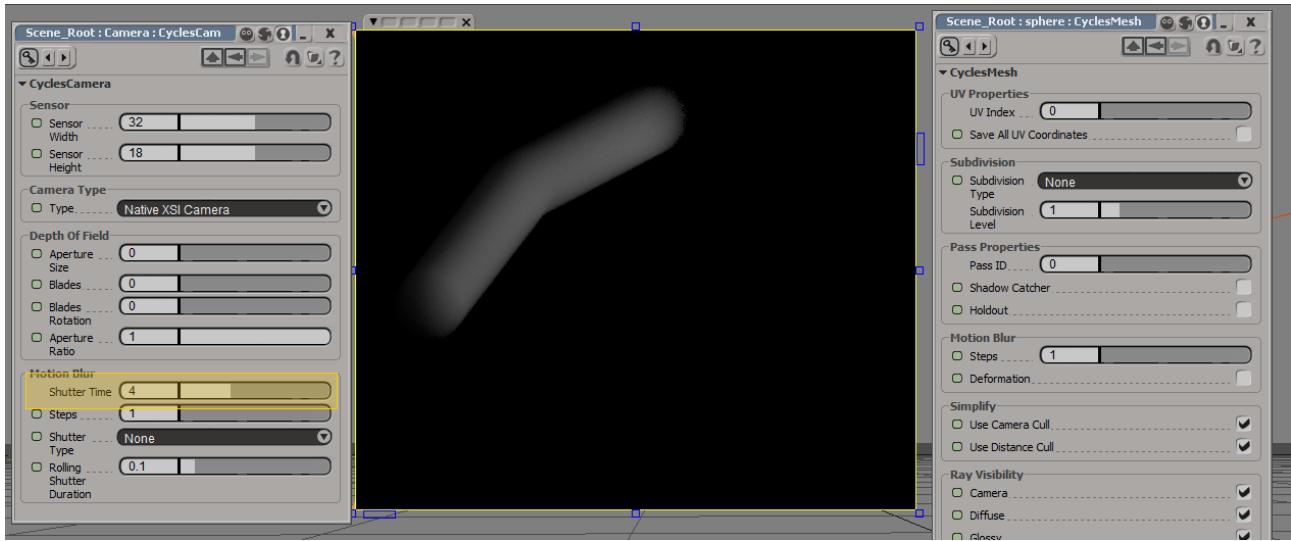


Для более детальной настройки эффекта надо на камеру (которая используется для рендера) и на шарик добавить свойства командами

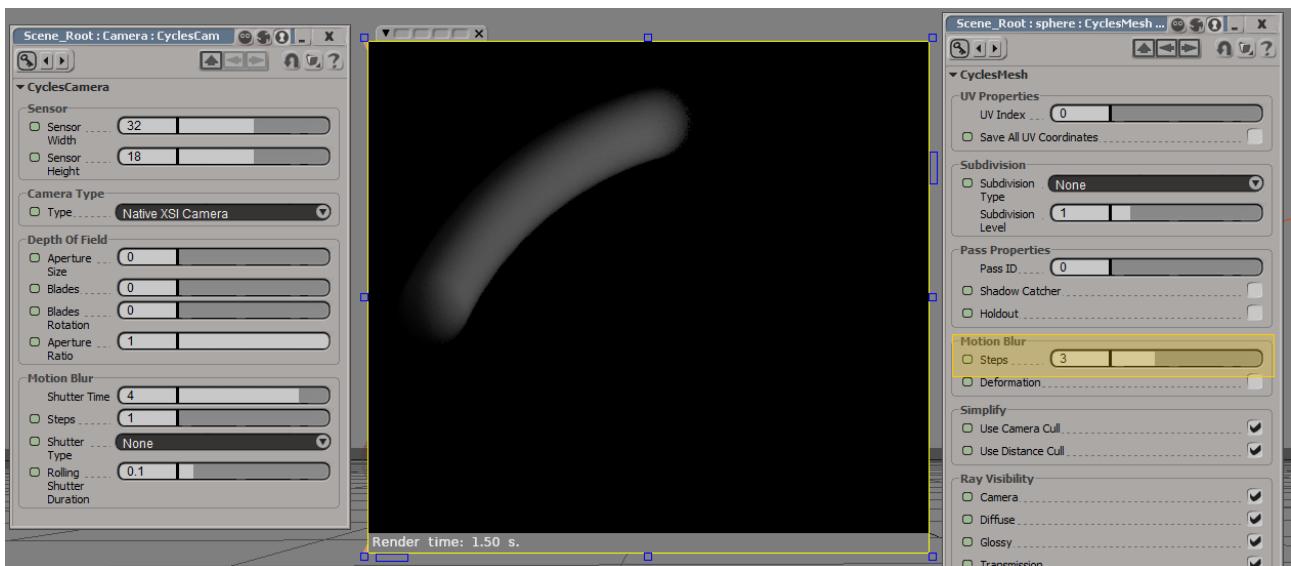
Property – Cycles Properties – Add Camera Property и
Property – Cycles Properties – Add Mesh Property.



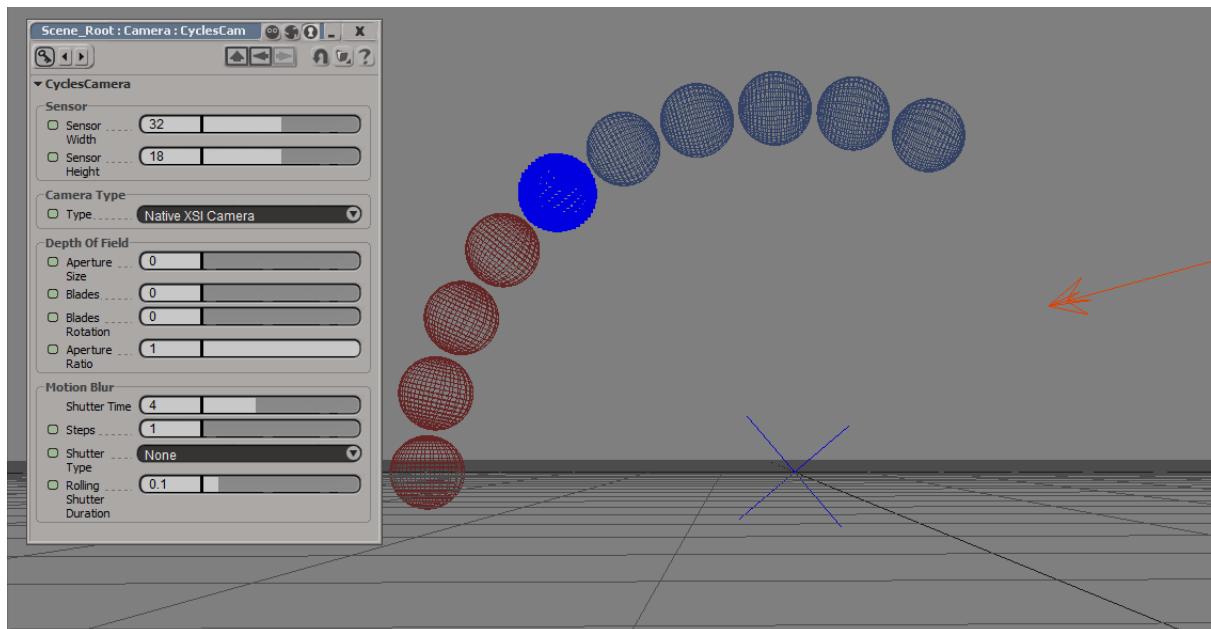
Меняем значение Shutter Time у камеры на какое-нибудь число побольше (4, к примеру). Видим, что траектория размытия какая-то слишком линейная.



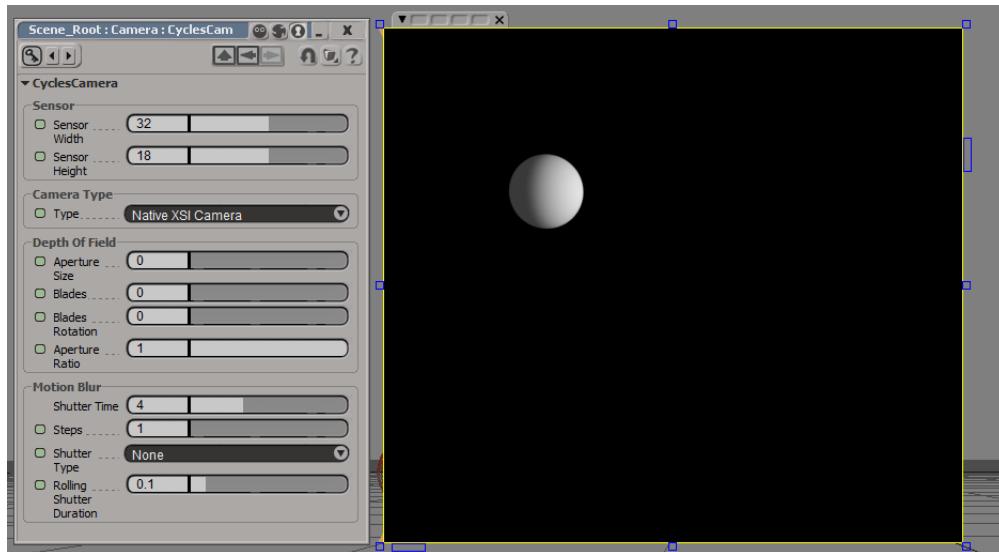
Чтобы это исправить, надо у шарика повысить значение параметра **Steps** (до, например, 3). Теперь всё выглядит корректно.



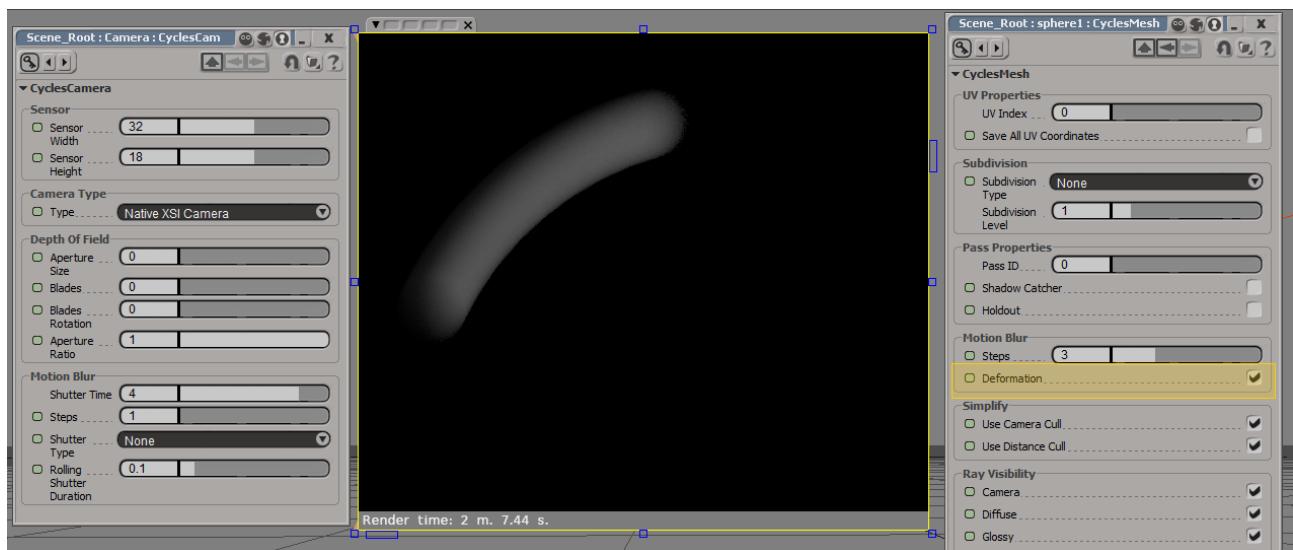
Пусть теперь у нас сцена такая же, только шарик движется за счёт того, что он прискинен к null-y.



Рендерим и видим, что никакого эффекта размытия нет.



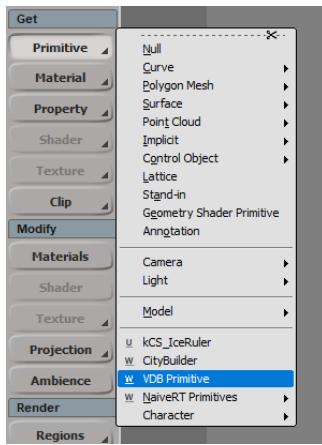
Дело в том, что физически сам объект никуда не движется, а двигаются его вершины. Чтобы рендер понимал такое движение, надо как и раньше добавить шарику свойство Cycles Mesh и включить параметр Motion Blur – Deformation.



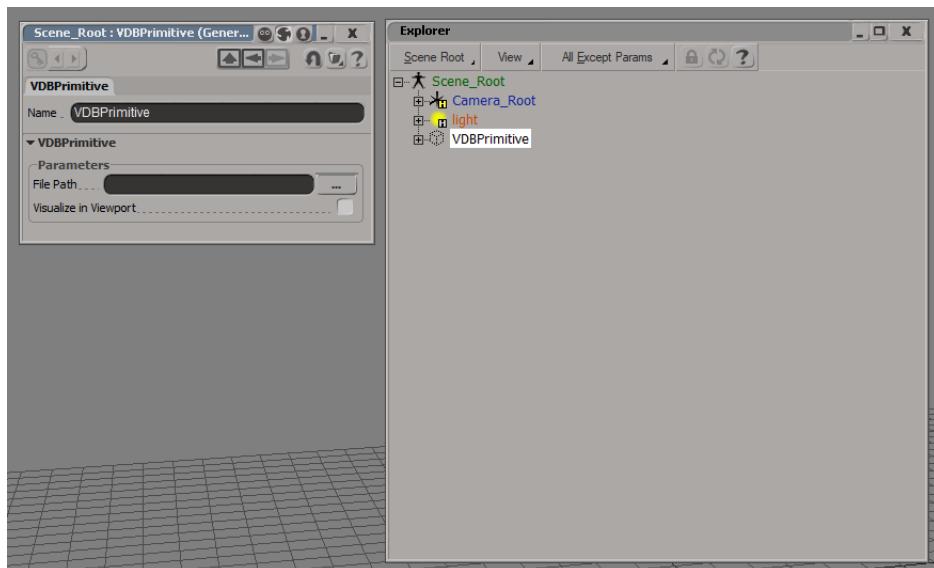
По умолчанию этот параметр считается выключенным, так как в некоторых случаях очень сильно возрастает время рендера. Вот как в этом случае, когда шарик перемещался сам, рендер занимал полторы секунды, а когда с помощью скрининга, то аж 2 минуты.

26 Как использовать VDB Primitive

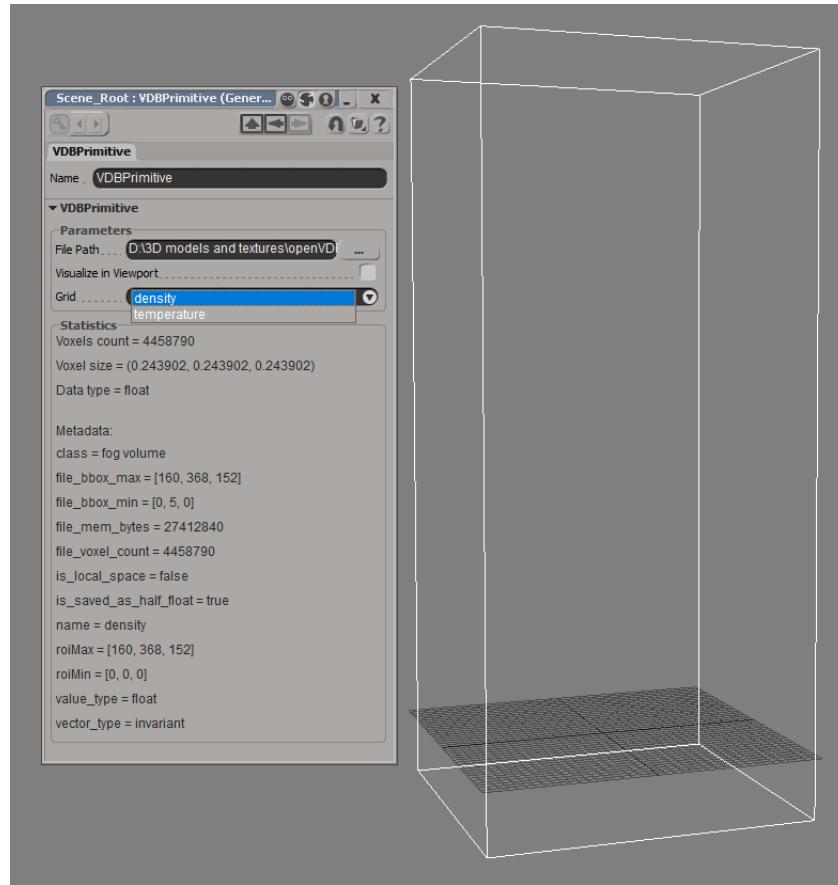
Рендер **Sycles** привносит новый тип примитива — **VDBPrimitive**. Этот примитив позволяет импортировать в сцену статичные файлы *.vdb. Чтобы добавить такой примитив в сцену — выбираем **Get — Primitive — VDBPrimitive**.



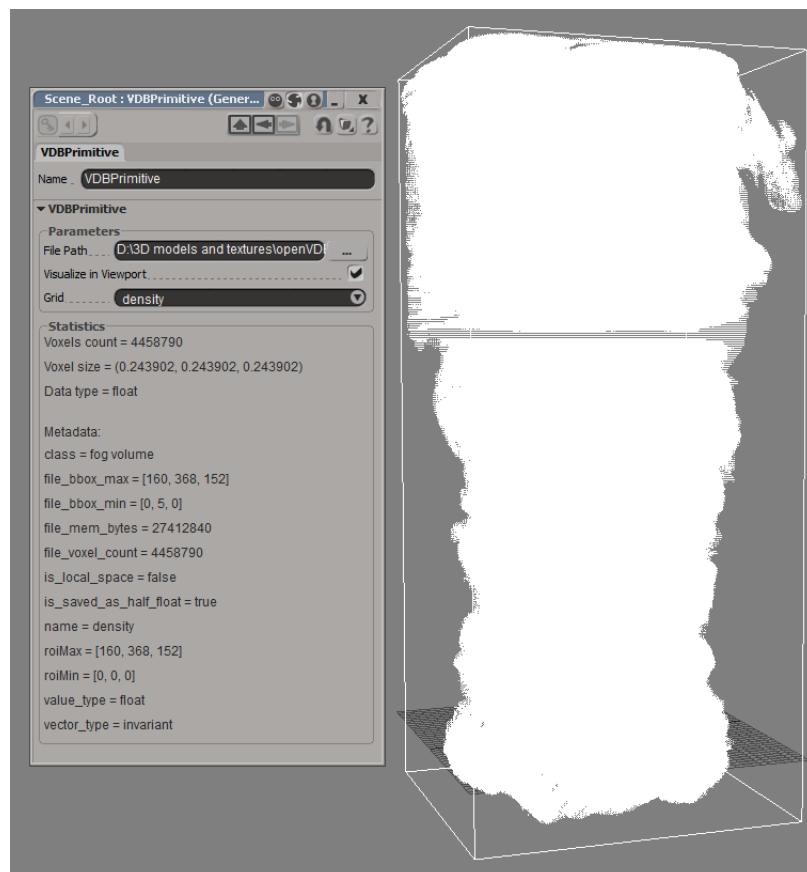
Будет добавлен пустой объект, для которого единственное, что можно — это указать путь до *.vdb файла.



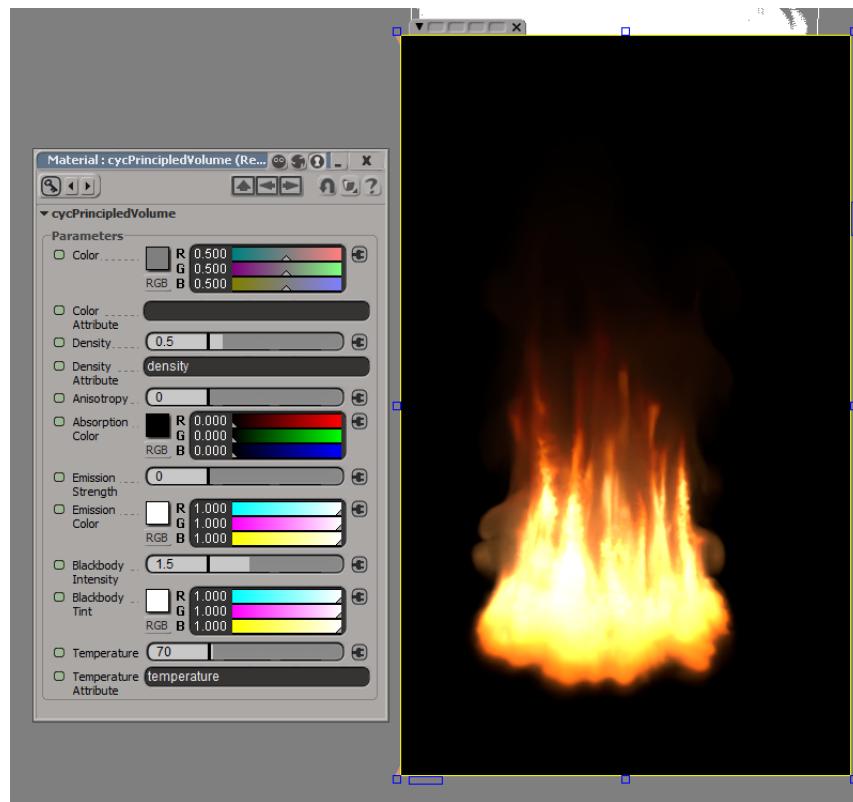
Указываем в параметре **File Path** полный путь до какого-нибудь файла с расширением *.vdb. После этого для объекта будет обозначен его габаритный контейнер, и в окне с параметрами появится некоторая статистическая информация о выбранном файле. Самое важно для нас в ней — это имена и типы сеток (которые по агицки называются grids), содержащиеся в файле. В нашем файле сетки две — **density** и **temperature**, обе содержат float значения.



Можно отобразить в окне просмотра сцены точки, составляющие содержимое выбранной сетки. Для этого надо включить параметр *Visualize in Viewport*. По умолчанию он выключен, так как для больших файлов это отображение сильно тормозит отрисовку сцены.



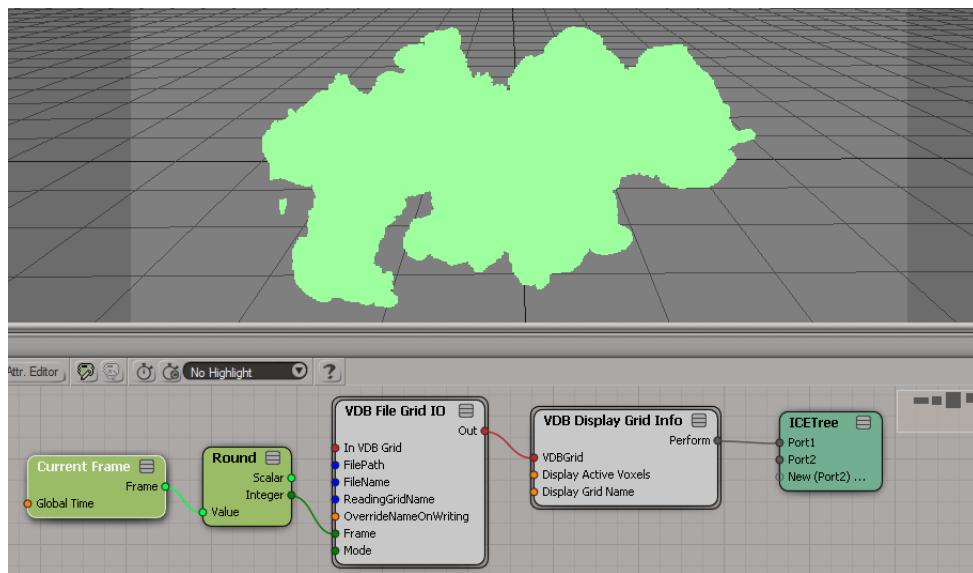
Теперь назначаем на объект материал с шейдером Principled Volume. Указываем Density = 0.5, Blackbody Intensity = 1.5 и Temperature = 70. Рендерим.



Стоит отметить, что шейдеру для рендера необходимо указание двух атрибутов — для плотности и температуры. Имена этих атрибутов должны совпадать с именами сे�ток, содержащихся в файле. В нашем случае они называются стандартно — **density** и **temperature** соответственно.

27 КакрендеритьOpenVDB

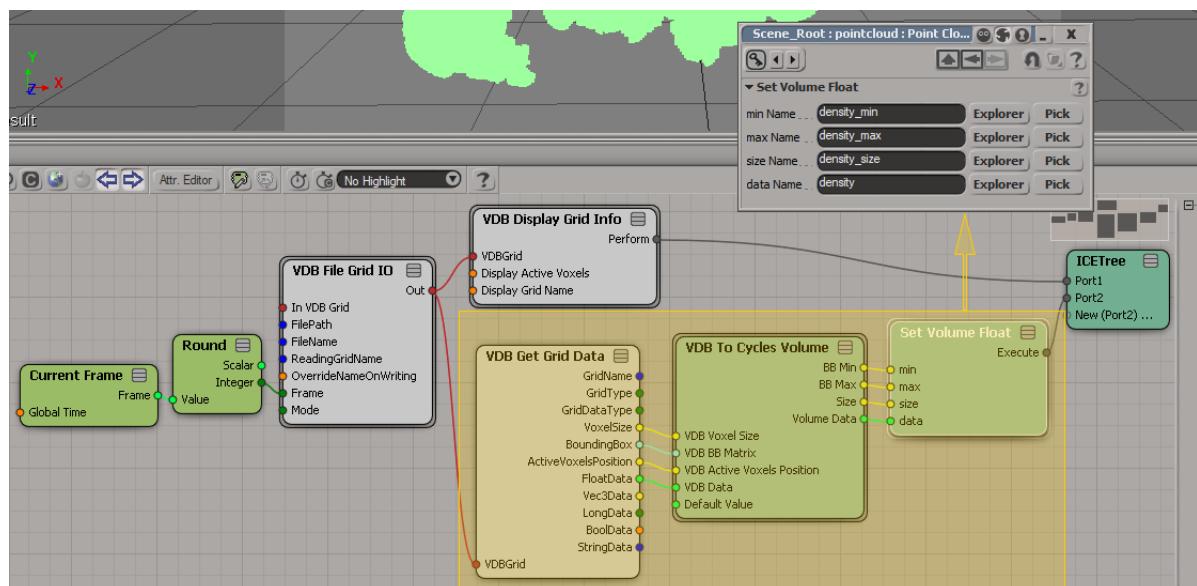
Предположим у нас есть сцена, на которую мы загрузили *.vdb файл с помощью OpenVDB for Softimage.



В нашем случае файл содержит всего одну сетку с именем `density`. Чтобы передать данные об этой сетке в рендер, надо задать четыре атрибута:

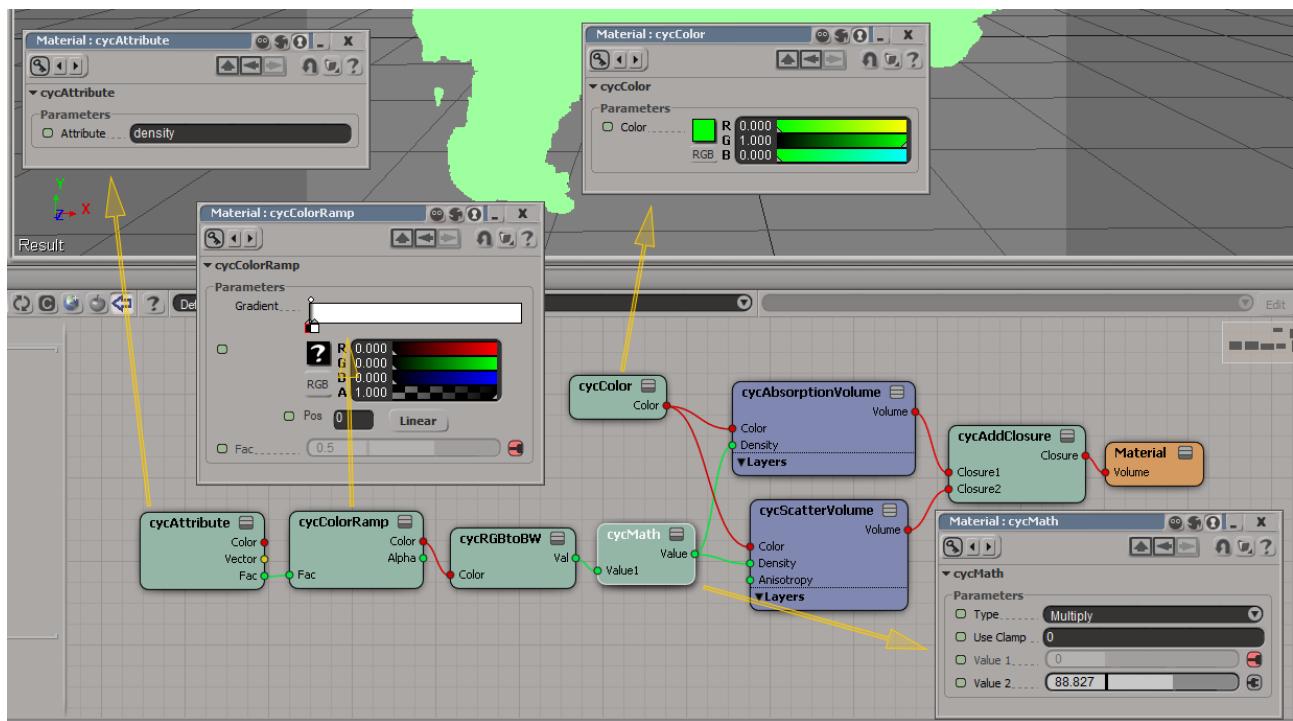
1. `density`, в который передать массив из float-чисел, задающих плотность вокселей;
2. `density_size`, в который передать один вектор, содержащий число вокселей по каждой из трёх осей. Этот вектор всегда целочисленный;
3. `density_min`, в который передать вектор, содержащий координаты левого нижнего угла воксельного куба. То есть это координаты вершины куба, имеющей наименьшие координаты по каждой из трёх осей;
4. `density_max`, в который передать координаты правой верхней вершины воксельного куба, то есть вершины, имеющей наибольшие координаты среди всех вершин куба.

Имя атрибута с данными может быть любым и не зависеть от имени в *.vdb-файле, но только другие три атрибуты должны начинаться с этого же имени и заканчиваться `_size`, `_min` и `_max`.

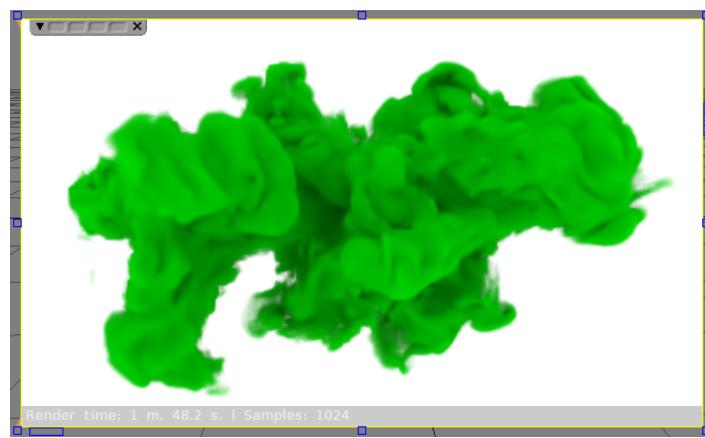


Получаем данные о сетке с помощью ноды VDB Get Grid Data. Выходной порт этой ноды **FloatData** содержит все нужные данные, однако они расположены не по порядку, а также содержат пропуски, в которых значение равно нулю. Чтобы полностью восстановить все данные пропускаем их через ноду VDB To Cycles Volume, а потом создаем все четыре нужных атрибута с помощью ноды Set Volume Float.

Назначаем нашему объекту материал примерно такой, как изображено на картинке. Нода **Attribute** там самая главная, так как с помощью неё рендер получает данные о плотности вокселей.

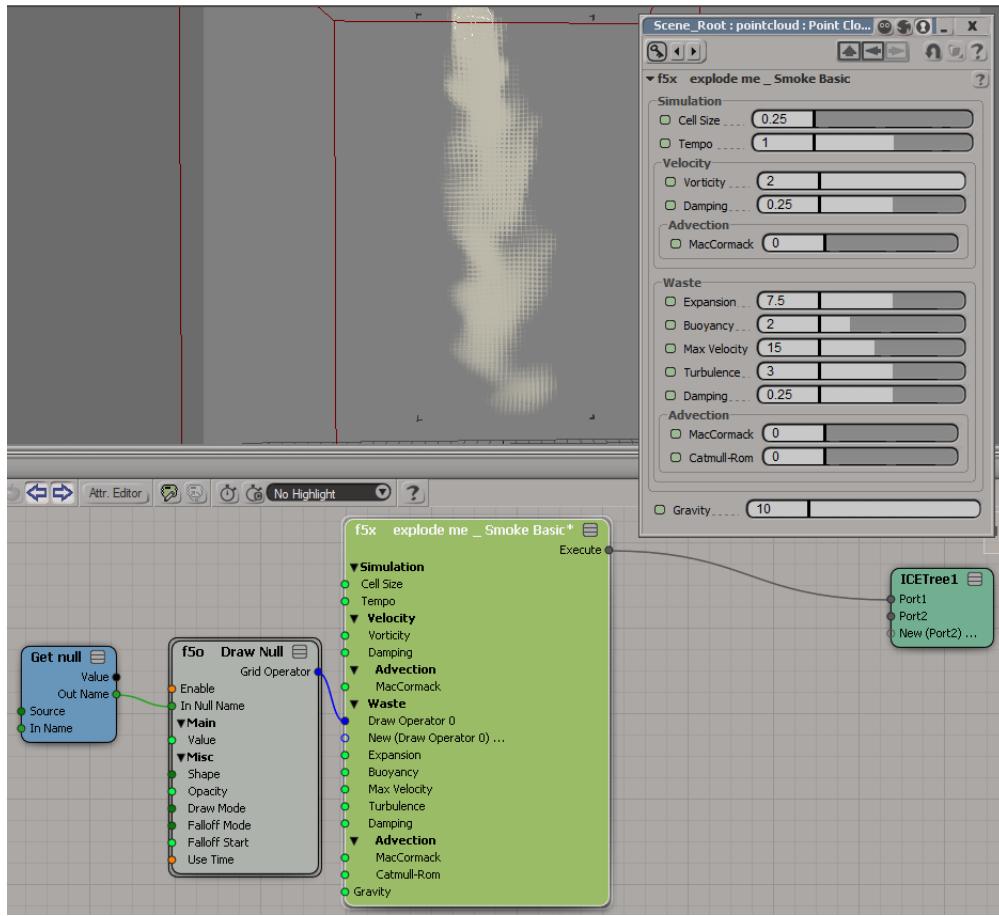


Рендерим.

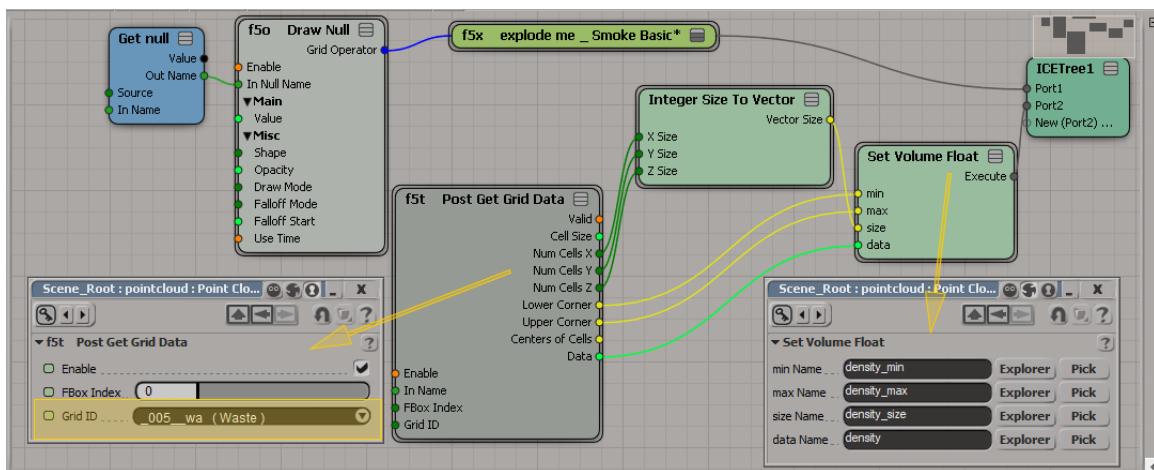


28 Какрендерить emFluid

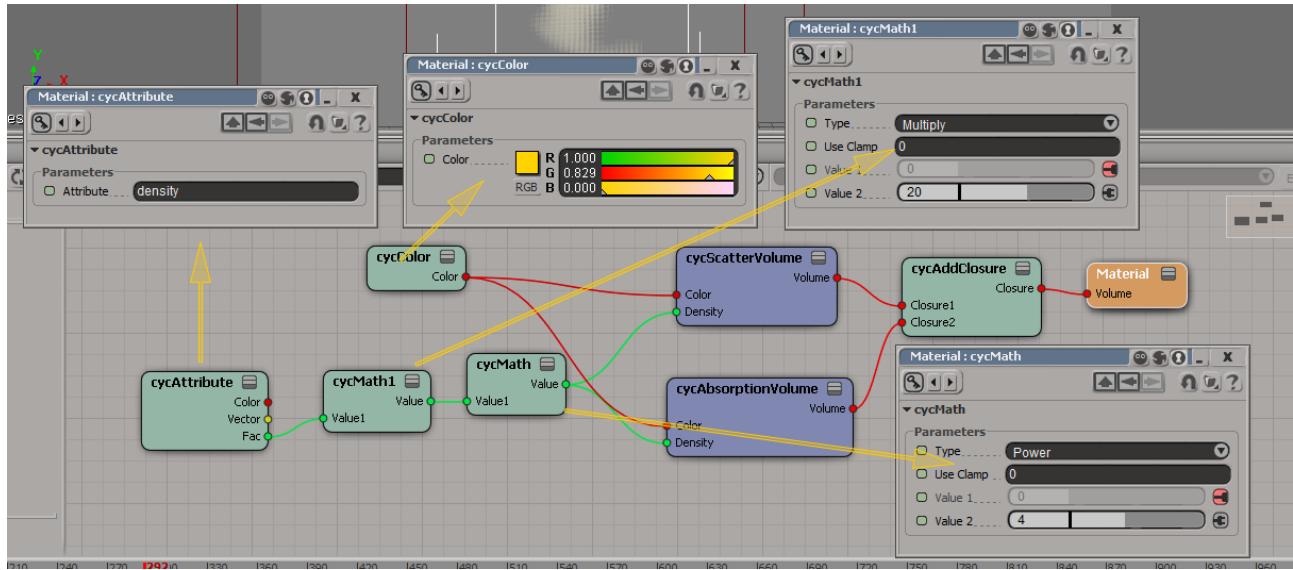
Предположим у нас есть сцена с дылом, сгенерированным с помощью emFluid.



Доступ к данным осуществляется с помощью ноды Post Get Grid Data. В качестве идентификатора сетки надо выбрать Waste, так как в нашем случае именно в неё пишется результат симуляции. Записываем четырёхку атрибутов точно так же, как делали для OpenVDB.



Назначаем дыму материал.

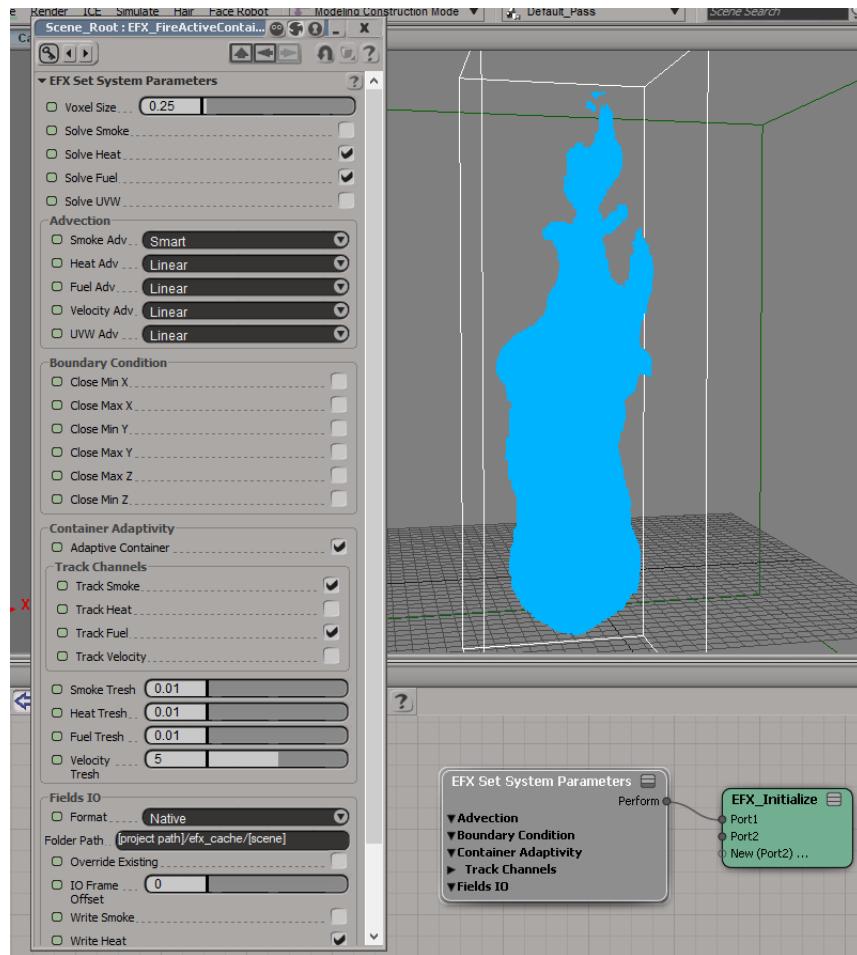


Рендерим.

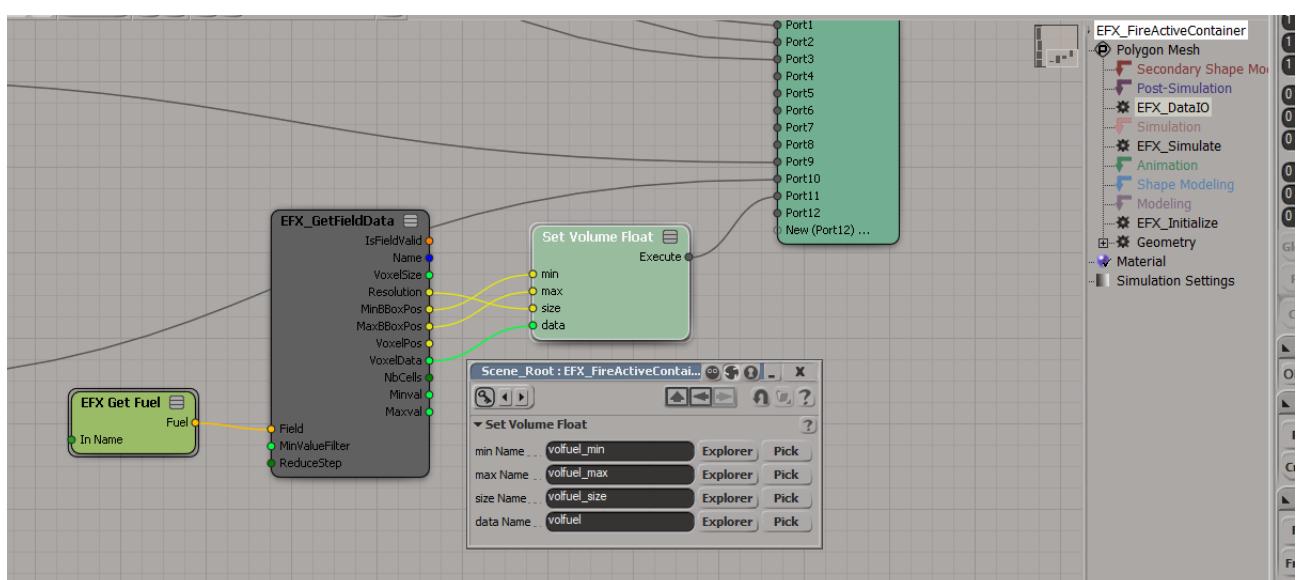


29 Какрендерить Explosia FX

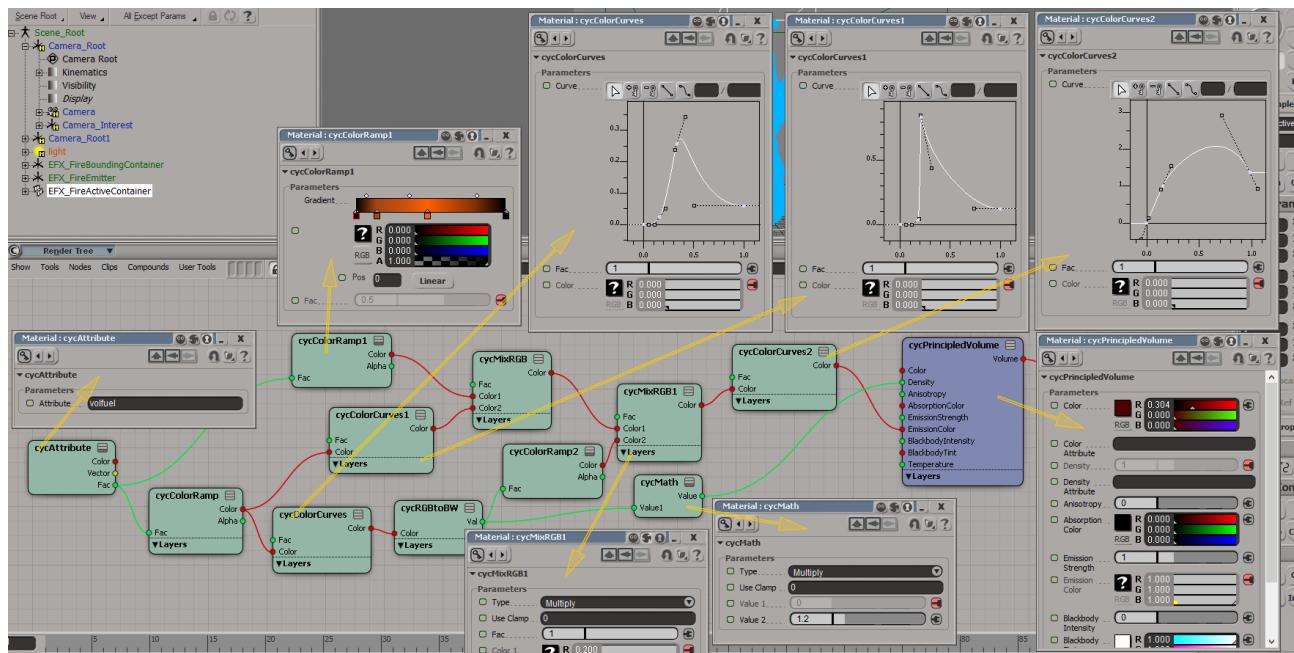
Возьмём для примера стандартную сцену с огоньком.



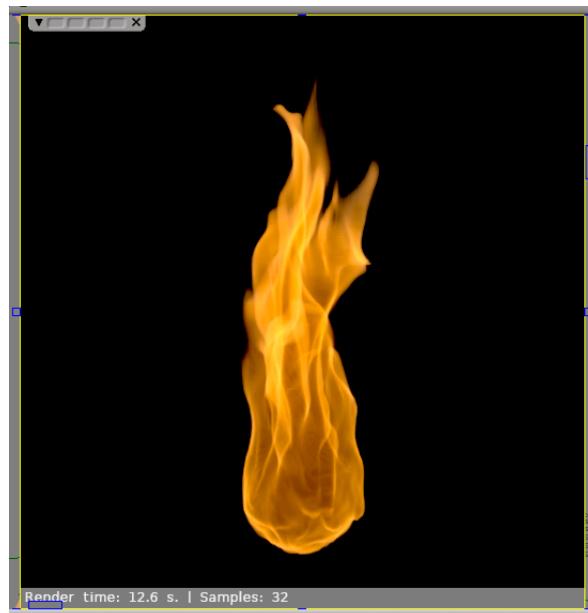
Для огня все нужные значения хранятся в атрибуте fuel. Данные из него получаем после всех симуляций в разделе Post-Simulation ICE-стэка. Для этого используем ноду EFX_GetFieldData, подсоединённую к ноде Set Volume Float с атрибутом volfuel (так как просто fuel уже занят).



Используем материал, изображённый на картинке. Те узлы, что не раскрыты, имеют значения параметров по умолчанию.

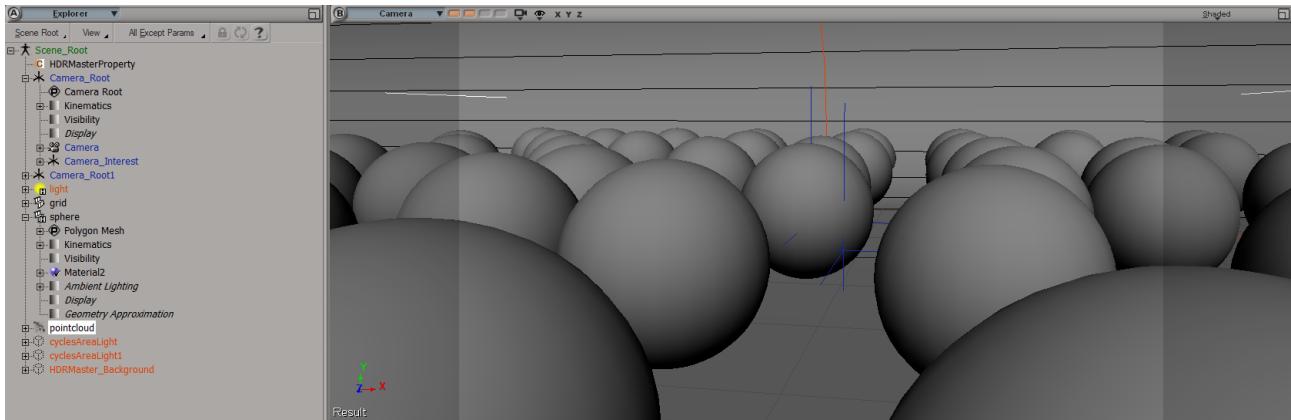


Рендерим.

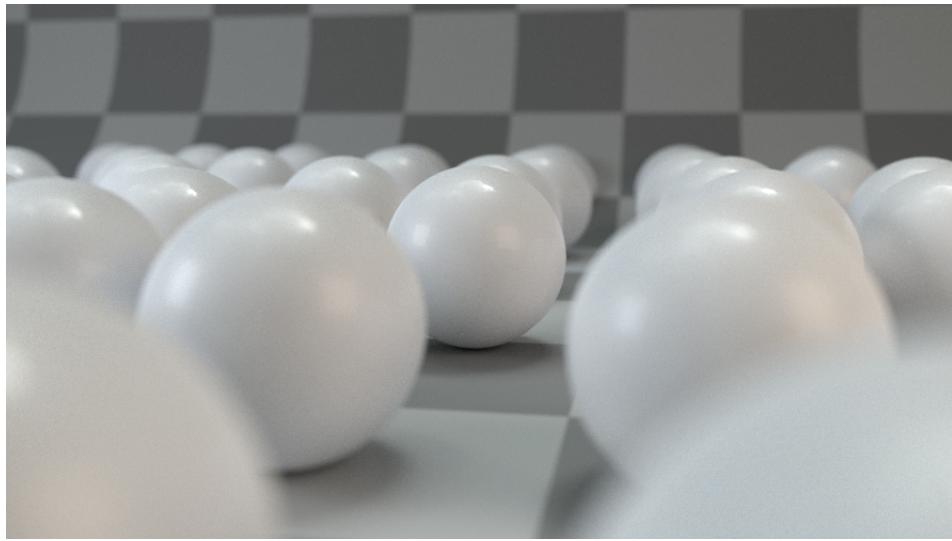


30 Какрендерить и использовать Cryptomatte пассы

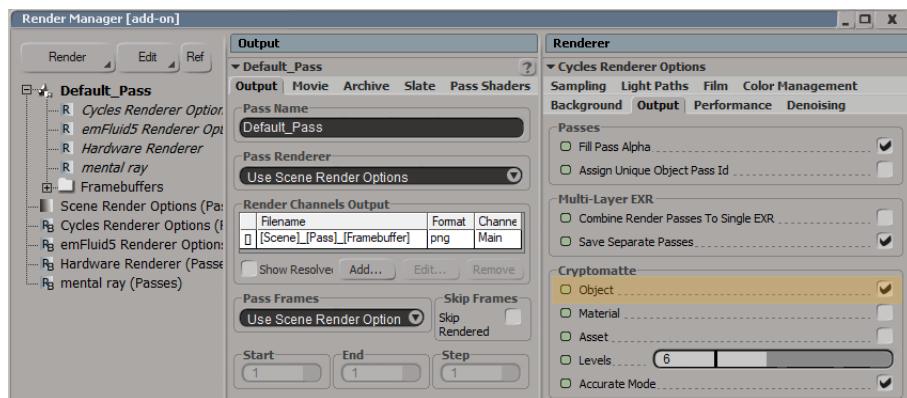
Пусть у нас есть простая сцена с шариками.



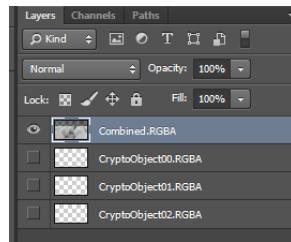
Результат рендера:



Чтобы включить сохранение пассов Cryptomatte, надо во вкладке **Output** настроек рендера отметить пункт **Cryptomatte – Object**. Это означает, что во время рендеринга будет сохраняться информация о том, где какой объект расположен.

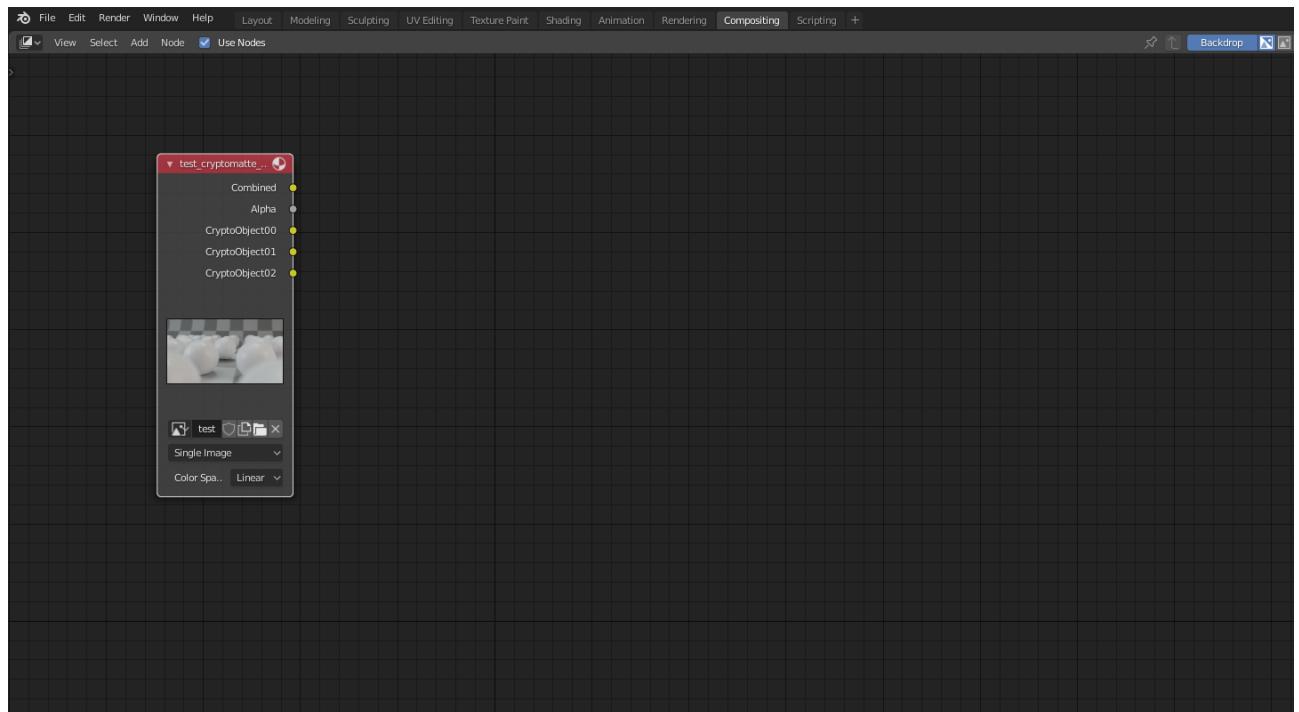


После рендеринга будет создан дополнительный файл с расширением ***.exr**, в конце имени которого есть слово **Cryptomatte**, и который содержит 4 слоя: картинка с итоговым рендером и три слоя с информацией об объектах сцены.

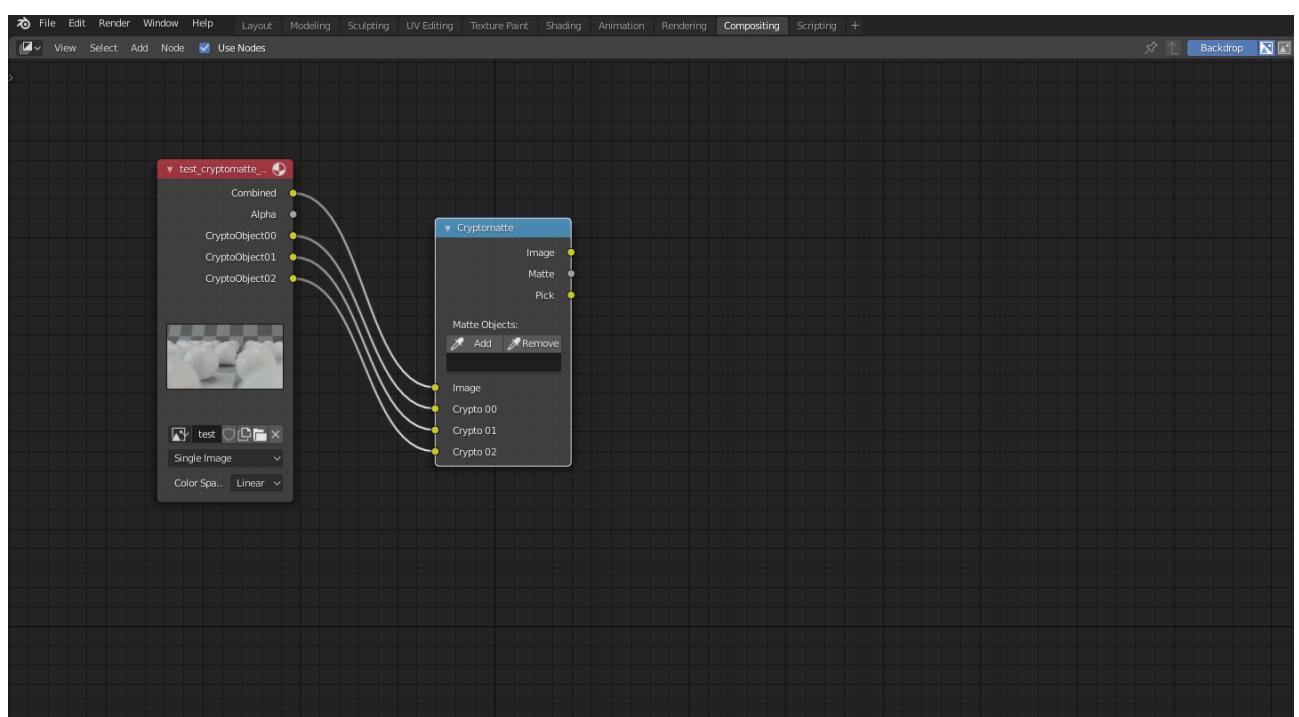


Теперь давайте для примера попробуем с использованием отрендерённых Cryptomatte пассов поменять цвет одного шарика. Будем использовать Blender, хотя можно и любой другой композер.

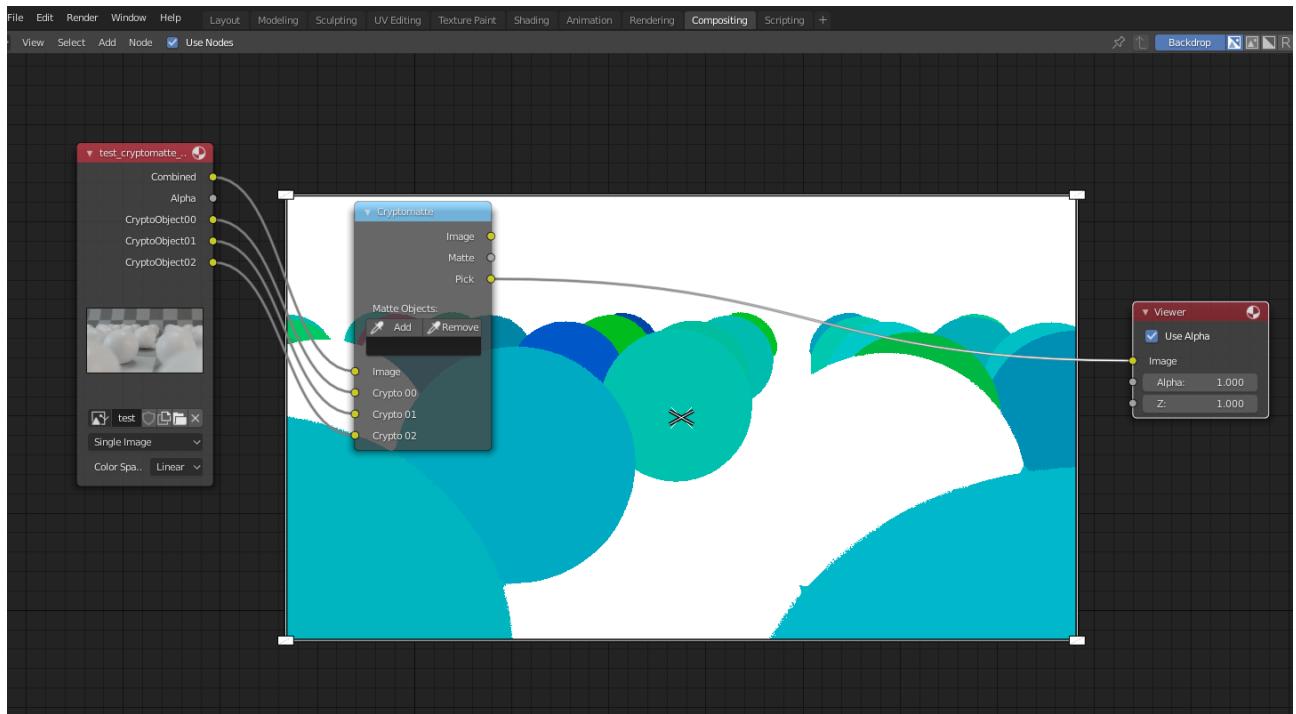
Добавляем на рабочий холст Blender-а ноду **Input – Image** и выбираем в ней наш **exr**-файл:



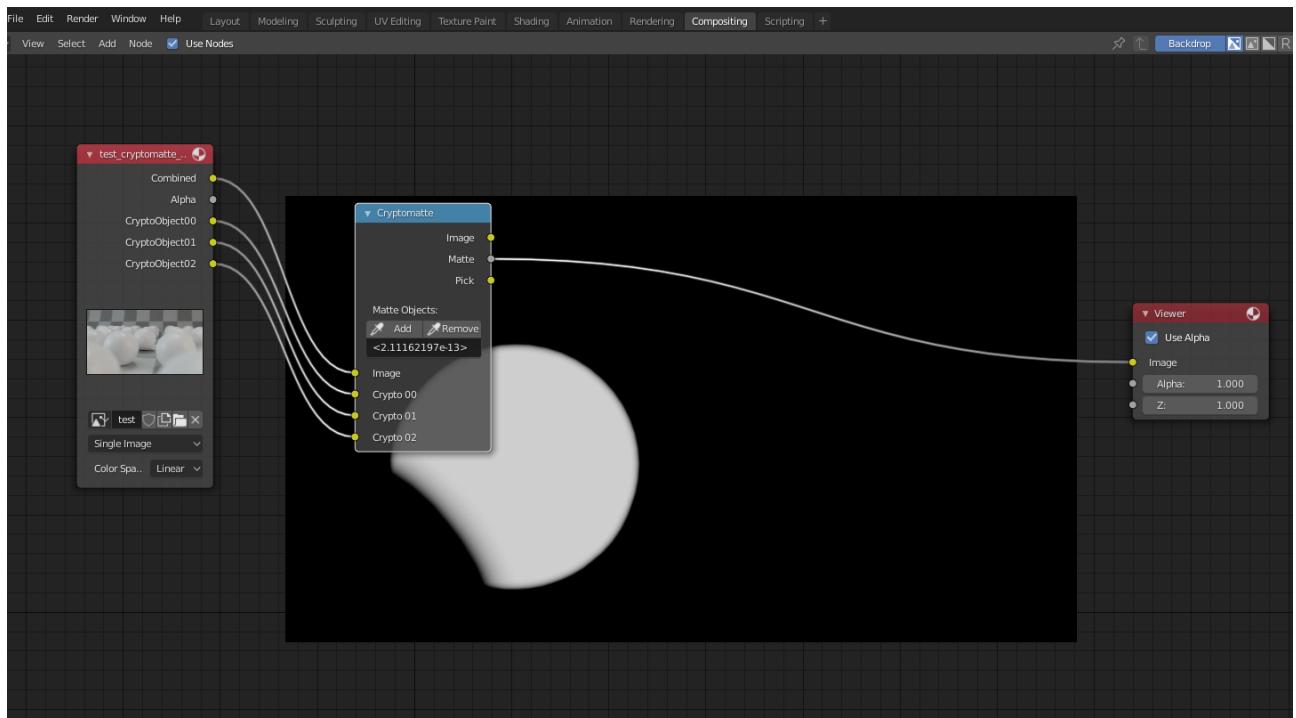
Добавляем ноду **Matte – Cryptomatte** и соединяем порты:



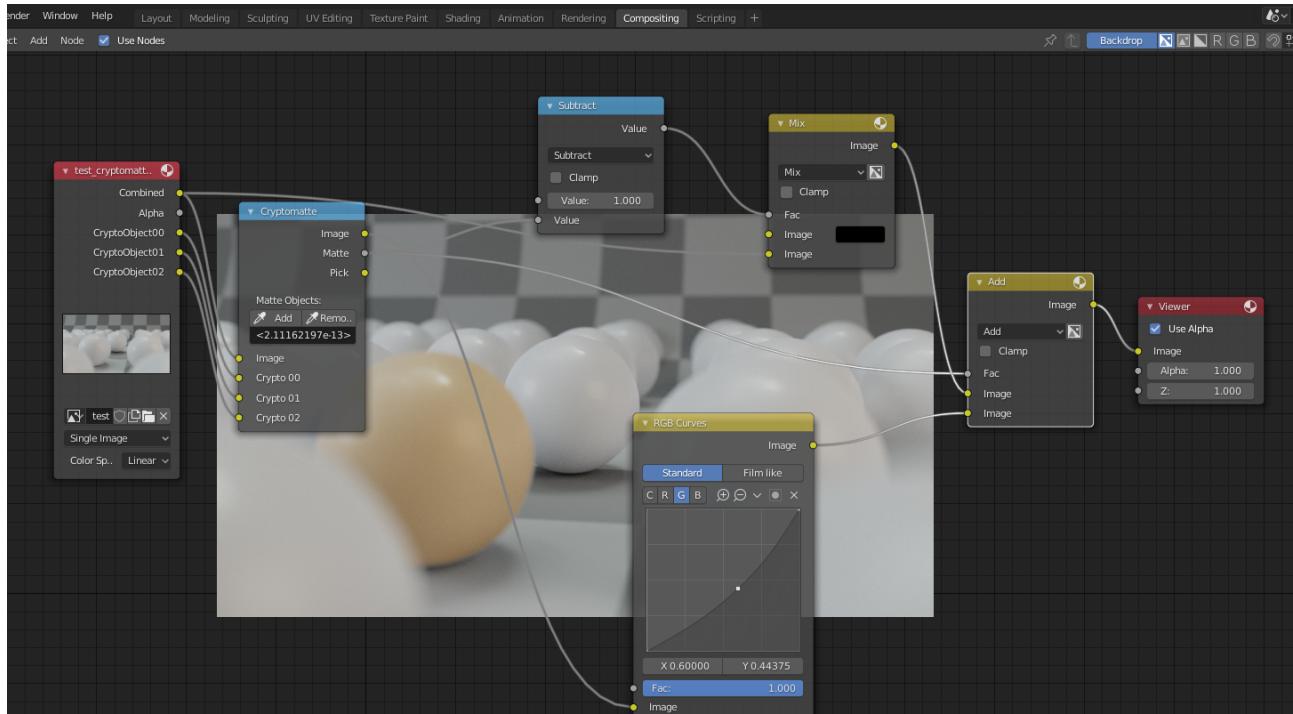
Чтобы посмотреть какую информацию содержит в себе файл, добавляем ноду **Output Viewer** и подсоединяем к ней порт **Pick**:



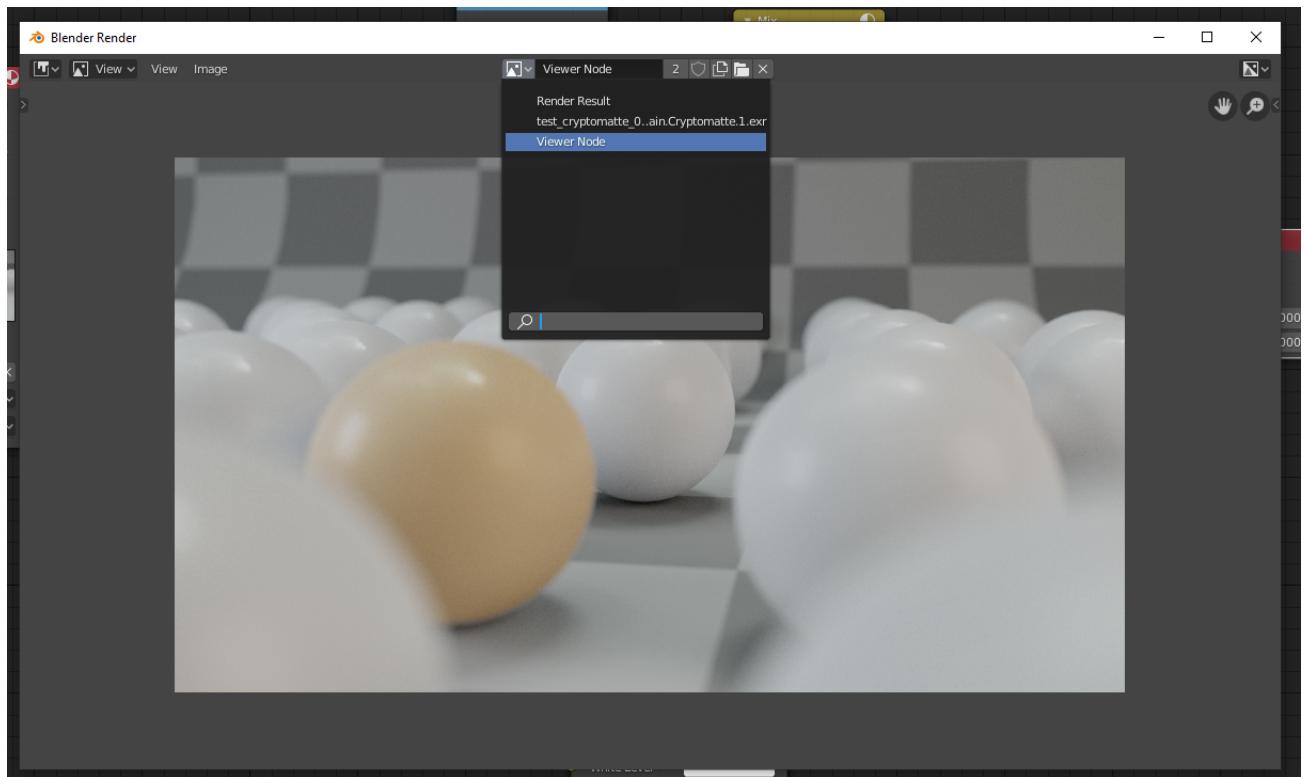
Теперь с помощью пипетки пишем на какой-нибудь шарик, и смотрим сгенерированную маску:



Добавляем немного композитной магии:



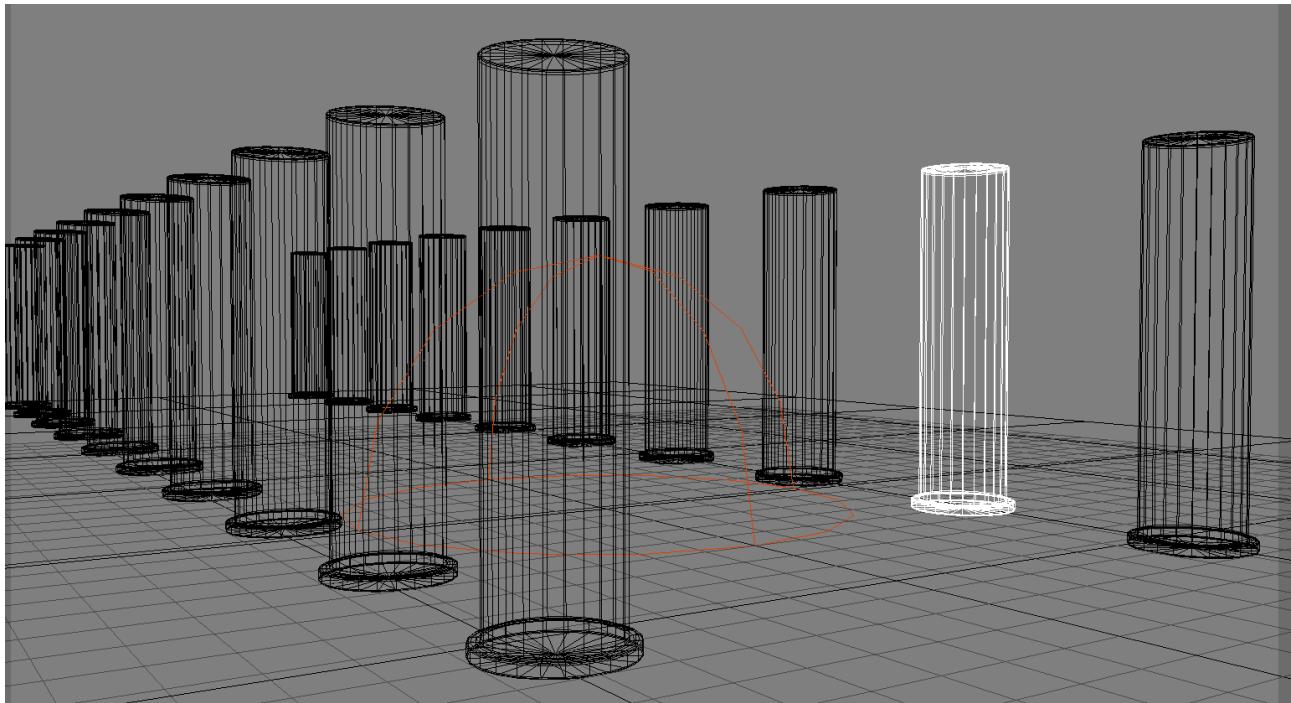
Чтобы сохранить результат композа открываем окно для результатов рендерера, нажав F11 и выбрав в этом окне **ViewerNode**:



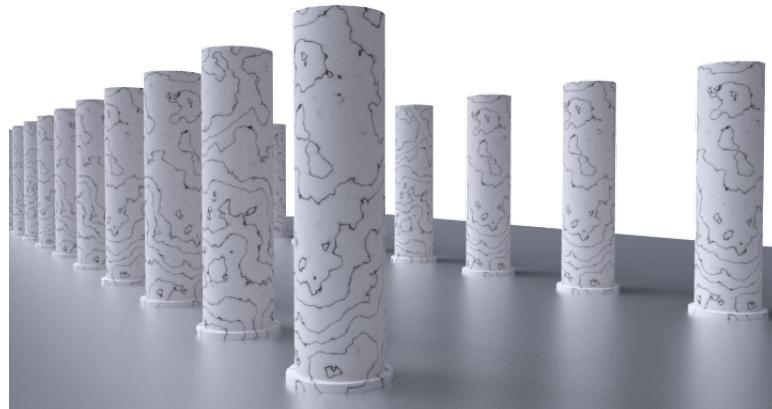
Жмём **Image – Save As...** и сохраним результат.

31 Какрендерить AOV-ы

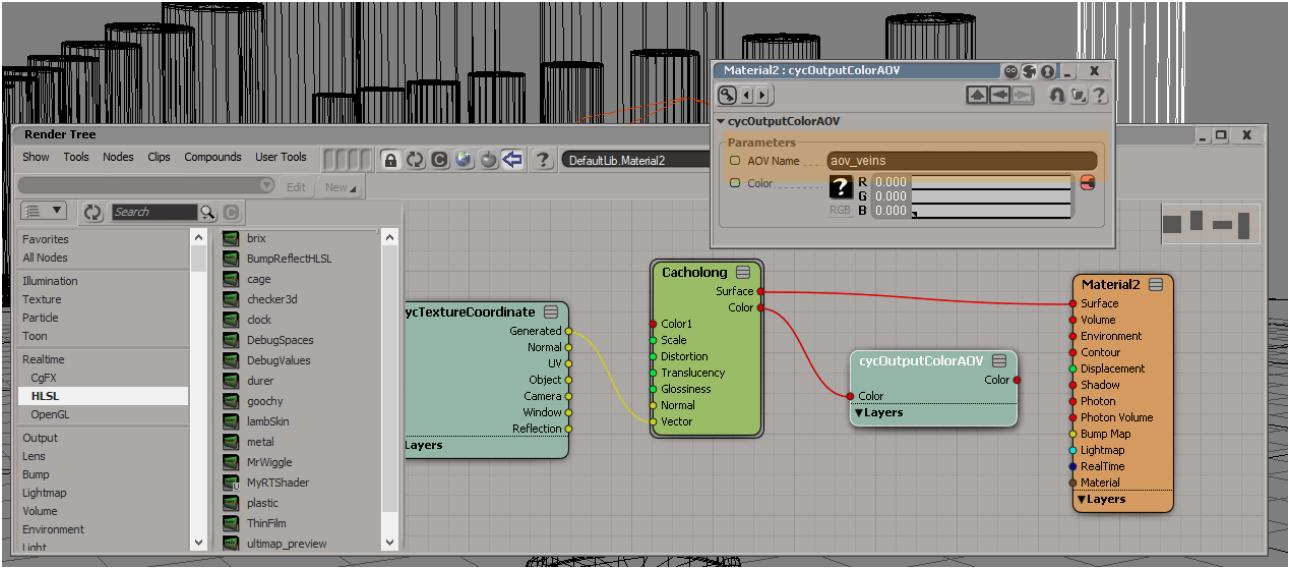
AOV (Arbitrary Output Variables) представляют собой по сути пользовательские пассы, в которые можно складывать практическую любую информацию из шейдеров во время рендеринга. Рассмотрим пример, в котором у нас есть сцена – колонны на плоскости.



На колонны назначен материал типа камня.

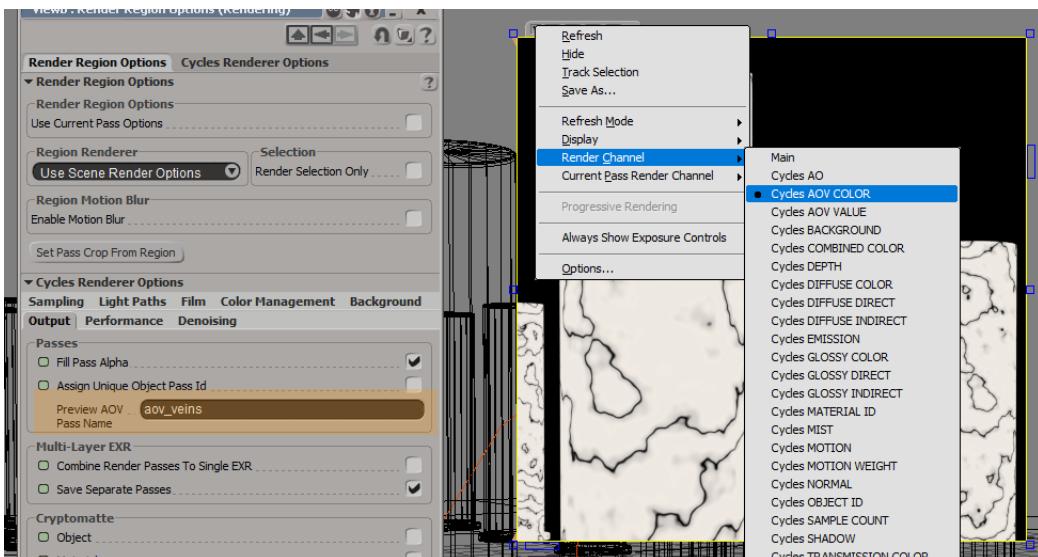


Мы хотим вывести чёрные прожилки из текстуры в отдельный пасс. Для этого добавляем в шейдер колонн ноду `уссOutputColorAOV` и подсоединяя к её входному порту `Color` порт ноды шейдера камня, который содержит в себе маску этих прожилок. Конечно, что подавать на вход ноде `уссOutputColorAOV` должно быть подготовлено заранее. Называем пасс, в который выводить прожилки, например `aov_veins`.

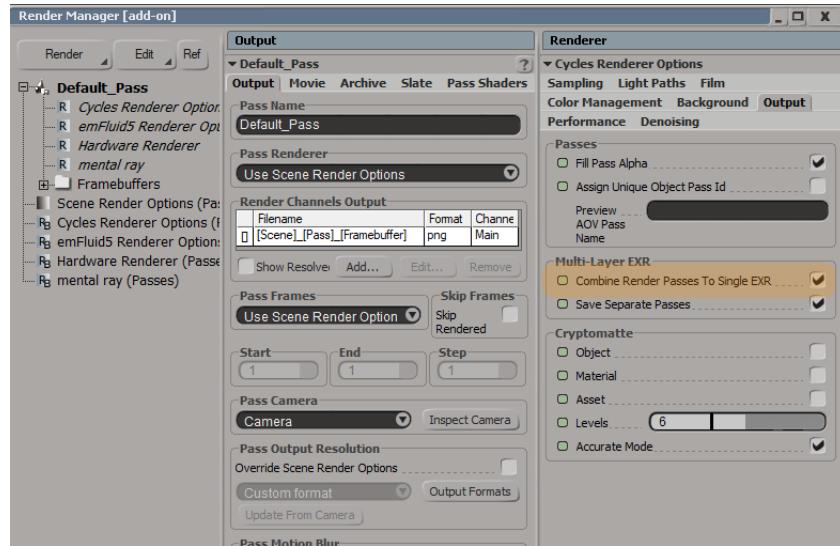


Подсоединять выход ноды `cycOutputColorAOV` куда-либо дальше не обязательно. Но можно и подсоединить туда, куда должен был быть подсоединен выходной порт, используемый нами для записи пасса. В этом случае нода будет использоваться просто как сквозная.

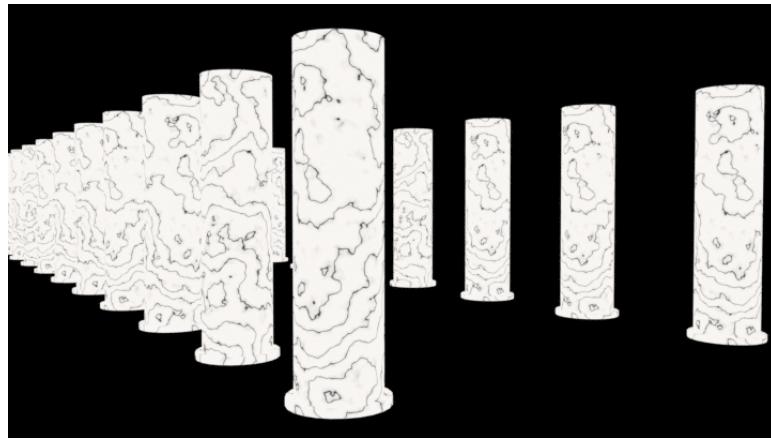
Можно просматривать содержимое AOV-пассов в окне рендера. Для этого надо выбрать показ пасса `Cycles AOV COLOR`. В общем случае мы можем записывать информацию во много разных пассов, поэтому рендер не знает заранее, какой из них показывать. Чтобы ему объяснить, надо в настройках рендера во вкладке `Output` указать в поле `Preview AOV Pass Name` название нужного нам пасса.



Единственный способ сохранить все AOV пассы в виде картинки – это использовать функцию сохранения в многослойный exr-файл. Для этого ставим птицу для параметра `Combine Render Passes To Single EXR` во вкладке `Output`.



Результат:



Всё то же самое работает и для случая, когда мы хотим сохранить в пасс не цвет, а числовое значение. Отличие только в том, что для этого надо использовать ноду `уссOutputValueAOV`.