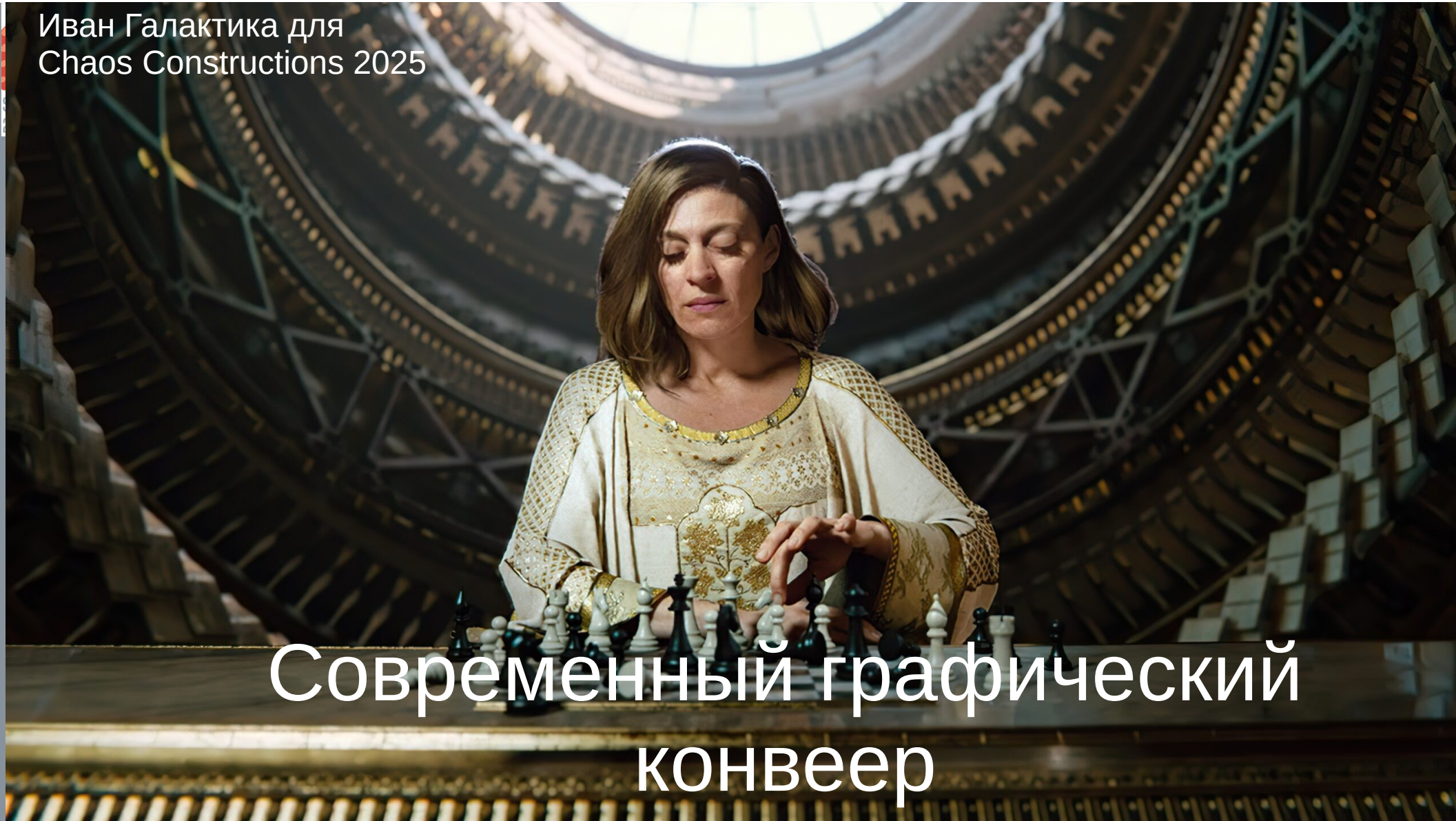
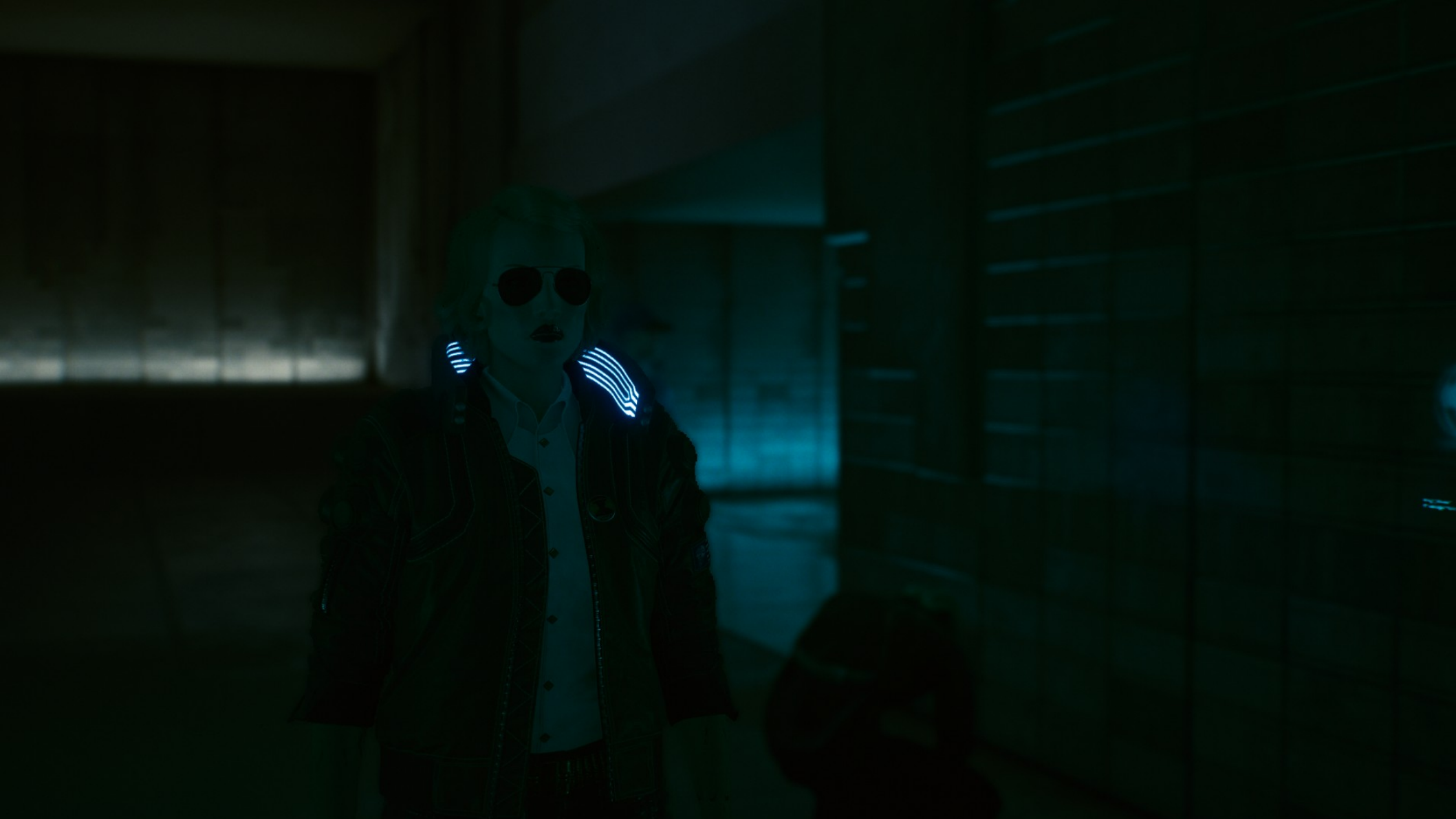
Иван Галактика для
Chaos Constructions 2025



Современный графический конвеер

CHAOS CONSTRUCTIONS 2025
ST.PETERSBURG
23-24 AUG 2025

Картинки для привлечения внимания)




02
His life is in your hands


NW
HEALING SUITE
Press to restore your health to maximum by using the Surgery Machine.
MEDICAL
HEALING SUITE
WEAPONS
EXPLOSIVES
CONSUMABLES
LEAD PIPE
MEDIPATCH

О чем поговорим?

- А как вообще получается вкусная картинка в игре?
- Как к этому пришли? (эволюция)
- Кто это придумывает?
- Как устроены рендереры?

  С инженерно-архитектурной точки зрения

Рендерер

- Это лишь часть игрового движка
- Они есть много где помимо игр:
  - САПР/редакторы (самые разные)
  - Web-контент
  - Симуляторы
  - AR/VR
  - ...

Рендерер

Самые известные:

- Unreal Engine (Epic Games)
- Unity (Unity Technologies)
- CryEngine (CryTek)
- Godot (The Godot Foundation)

Рендерер

Известные:

- Frostbite Engine (Battlefield, Star Wars Battlefront / DICE = Electronic Arts)
- Unigine / Unigine
- idTech (Doom, Quake / id Software)
- RED Engine (Witcher, Cyberpunk 2077 / CD Project RED)
- RAGE (RDR2, GTA / Rockstar)

Рендерер

Веб:

- Three.js (mrdoob)
- Babylon.js (David Catuhe, David Rousset, Pierre Lagarde, Michel Rousseau)
- gLTF-viewer (Khronos)
- Sketchfab
- <modelviewer> (Google)

Рендерер

Ещё:

- Blender (Eevee, Cycles)
- Filament (Google)
- Panda3D
- Ogre3D (Steve Streeting)

Рендерер

И ещё:

- Amazon Lumberyard
- Asobo Studio (Microsoft Flight Simulator)
- Dassault (3DEXPERIENCE)
- …
- тысячи их!
- *Пс.. Я тоже пишу свой движок) называется Party Engine*

Рендерер: задачи

Задачи простые:

- Нарисовать красиво
- Нарисовать много
- Нарисовать быстро

Мы поговорим о красоте, и немножко о скорости

Рендерер: типы

Принципиально различаются подходы:

- Растеризация: аналитическое решение
- Трассировка лучей: симуляция физики

Все движки реализуют растеризацию, и большинство также реализует элементы трассировки лучей.

Полная замена на трассировку желательна, но возможна только после достижения «1 луч на 1 пиксель»

Рендерер: растеризация

Аналитическое решение уравнения освещения

$$L_o(\omega_o) = \int_\Omega L_i(\omega_i)\ \cos\theta_i\ f_r(\omega_i, \omega_o)\ d\omega_i$$

- На самом деле уравнений несколько (точнее оно видоизменяется), в зависимости от вида материала и условий сцены
- Просто подставляем параметры материала и сцены в формулу?
- В крации тут: https://habr.com/ru/articles/923220/

  и тут https://dl.acm.org/doi/pdf/10.1145/15886.15902

Cook-Torrance model

$$k_{\text{spec}} = \frac{\exp\left(-\tan^2(\alpha)/m^2\right)}{\pi m^2 \cos^4(\alpha)}, \quad \alpha = \arccos(N \cdot H)$$

Beckmann Distribution of specular

$$R(\theta) = R_0 + (1 - R_0)(1 - \cos\theta)^5$$
$$R_0 = \left(\frac{n_1 - n_2}{n_1 + n_2}\right)^2$$

Schlick's Fresnel approximation

$$G = \min\left(1, \frac{2(H \cdot N)(V \cdot N)}{V \cdot H}, \frac{2(H \cdot N)(L \cdot N)}{V \cdot H}\right).$$

Geometric attenuation term (self-shadowing)

$$k_{\text{spec}} = \frac{DFG}{\pi(V \cdot N)(N \cdot L)}.$$

Cook-Torrance model

Normalization term

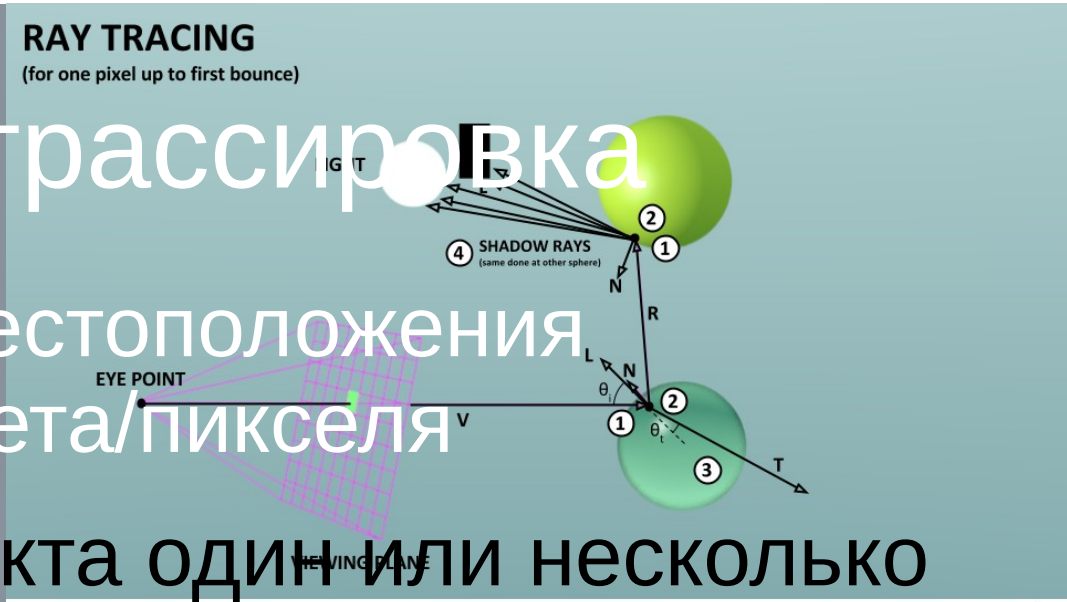
Рендерер: растеризация

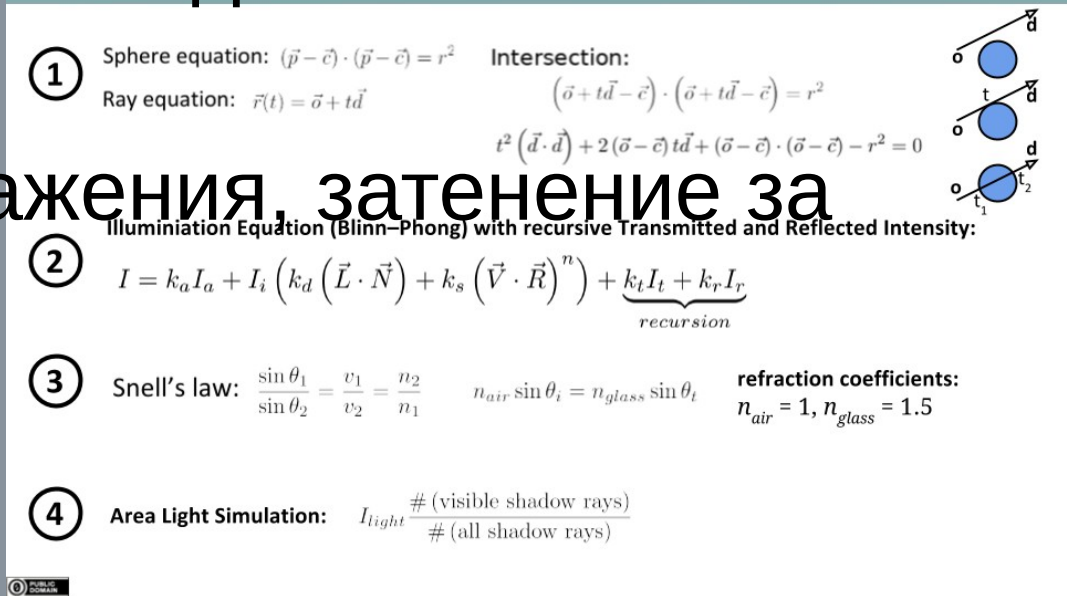
Ограничения:

- Необходимость множества подготовительных проходов (о них и доклад)
- Небольшой бюджет по количеству теней и отражений
- Общая громоздкость, из-за частных случаев и компромиссов

Рендерер: трассировка

- Испускаем лучи из местоположения камеры/источника света/пикселя
- Отражаем их от объекта один или несколько раз
- Получаем блики, отражения, затенение за один проход!





Рендерер: трассировка

Все это прекрасно, не особо громоздко.

Главная проблема: нужен один луч на один пиксель.

Пока этого нет, но есть демо Zorah

https://www.youtube.com/watch?v=eGA4fMTq9Po

Модели освещения

1995-2010 — модель Фонга, Блинна-Фонга

- "Illumination for Computer Generated Pictures" (01.06.1975) Bui Thong Phong
- "Models of light reflection for computer synthesized pictures" James F. Blinn (1977)
- Разделение света на фоновую, диффузную и зеркальную компоненты
- Простые уравнения
- Неплохой визуал
- Физическая недостоверность: нарушается закон сохранения энергии
- Трудности работы с материалами (трудно подобрать параметры под фотореалистичность, но легко создавать фантастические сцены)

Модели освещения

PBR = Physicall Based Rendering: 2010 — наши дни

- Физическая достоверность (закон сохранения энергии выполняется)
- Визуальная достоверность (очень близко к реальности)
- Легко подбирать материалы (есть техники сканирования, есть атласы материалов)
- Значительно более сложные формулы
- Может рассчитываться в реальном времени — сейчас

Модели освещения

- "A Reflectance Model for Computer Graphics" (01.01.1982) Robert L. Cook and Kenneth E. Torrance
- "The Rendering Equation" (1986) James Kajiya / ACM SIGGRAPH Computer Graphics, Volume 20, Issue 4
- "Realistic Image Synthesis Using Photon Mapping" (2001) Henrik Wann Jensen / ISBN: 1568811470
- Physically Based Rendering (2004) Matt Pharr, Wenzel Jakob, and Greg Humphreys
- "Physically Based Shading at Disney" (2012) Brent Burley

Модели освещения от Фонга к PBR

- The Last of Us (2013)
- Bioshock Infinite (2013)
- GTA V (2013)
- Assassin's Creed IV: Black Flag (2013)
- Far Cry 4 (2014)

Модели освещения от Фонга к PBR

- Dragon Age: Inquisition (2014)
- The Witcher 3: Wild Hunt (2015)
- Metal Gear Solid V: The Phantom Pain (2015)
- Final Fantasy XV (2016)

  Микс из Фонга и PBR: уже частично используются продвинутые техники вместе с элементами Фонга (например такими как фоновая компонента) — в разной пропорции

Материалы

Материалы — это визуальные свойства объектов, задаваемые как константы либо как текстурные карты.

Материалы состоят из слоёв.

Материалы напрямую связаны с моделью освещения.

Материалы для PBR в разных движках имеют общую основу, и имеют небольшие различия в редко используемых и «тяжелых» свойствах.

Концепция материала развивалась 1980-е и 1990-е

Материалы

Khronos стандартизирует:

https://www.khronos.org/gltf/pbr

Base Color (Albedo)

Материалы

Металличность (metalness)

Материалы

Шероховатость (roughness)

Материалы

Нормаль (карта нормалей)

Материалы

Рассеяное затенение (ambient occlusion)

Материалы

Излучающий слой (emissive)

Материалы

Альфа-канал и прозрачность

CHAOS CONSTRUCTIONS 2025
ST.PETERSBURG
23-24 AUG 2025

Материалы

Индекс преломления (IOR, index of refraction)

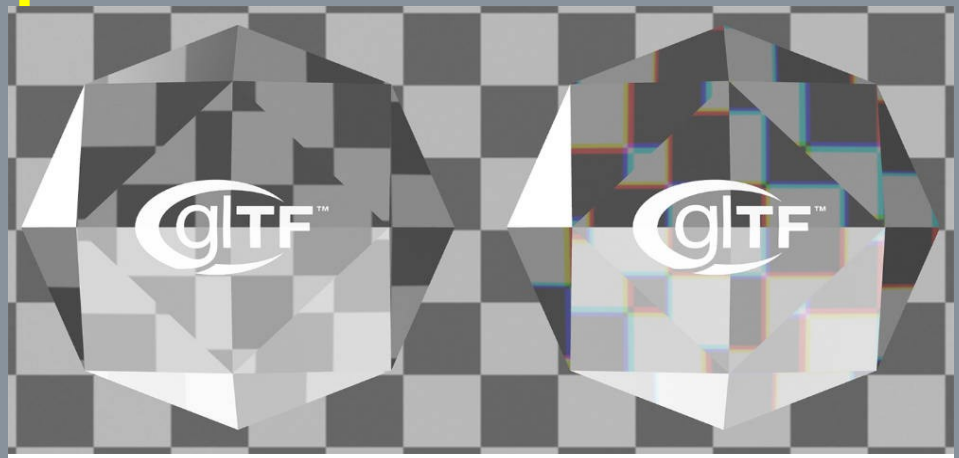
Материалы

Передача света (transmission)

Объем (volume)

Материалы

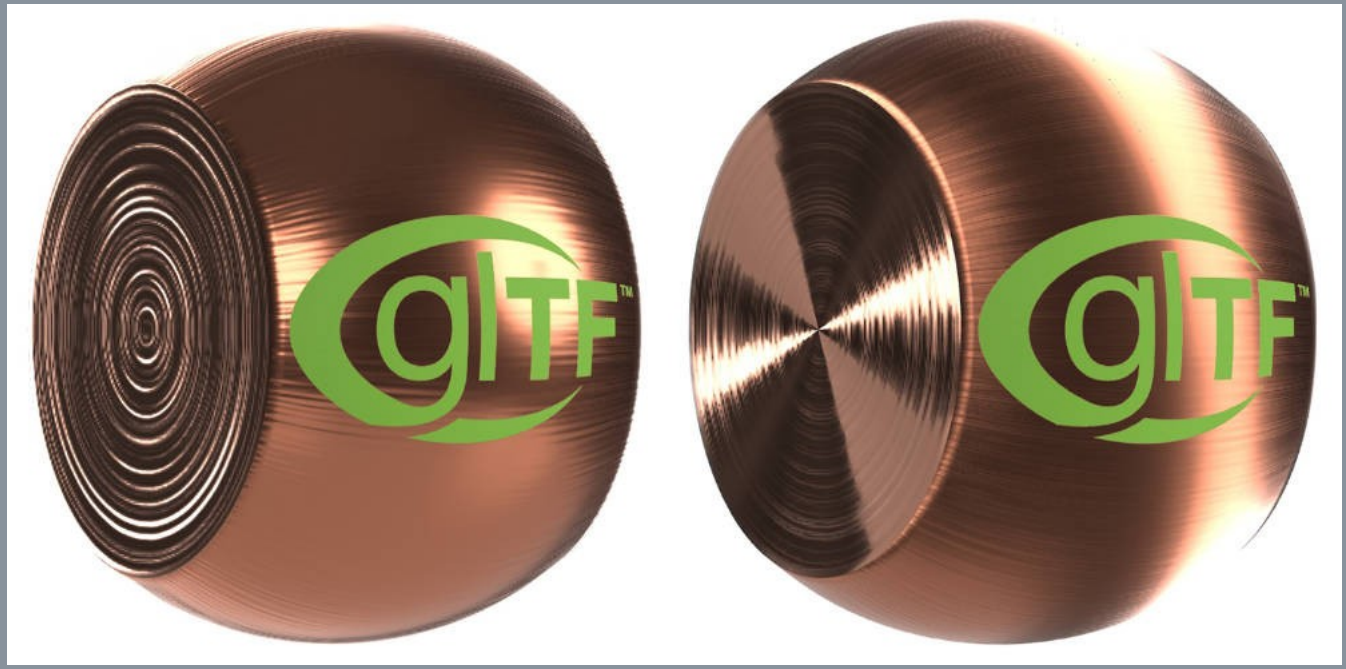
Дисперсия



Переливчатость (iridescence)

Материалы

Анизотропия



Материалы

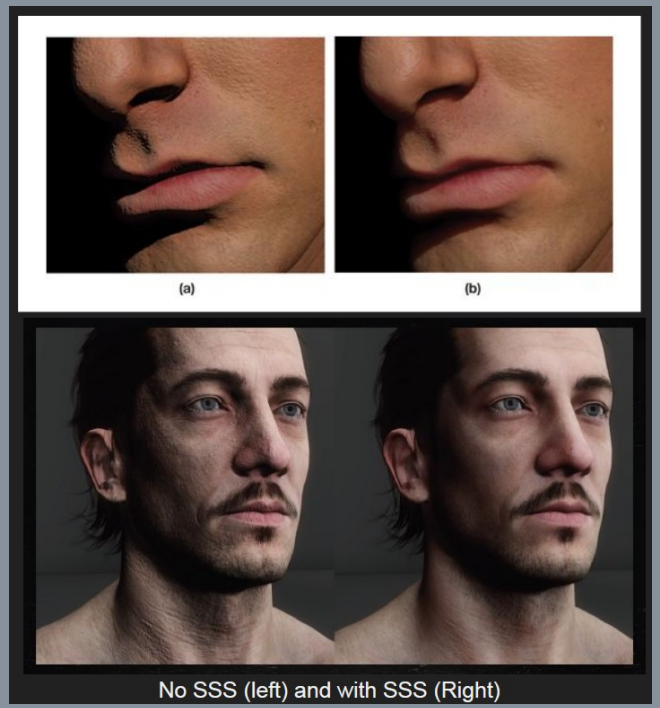
Clearcoat

(верхний слой)

Sheen (отблеск)

Материалы

Это не всё! Есть редкие и трудные в рендере свойства, например подповерхностное рассеяние, полупрозрачность (transclucency), и другие!



No SSS (left) and with SSS (Right)

Материалы: итог

Чтобы нарисовать крутую картинку, движок должен эти материалы понимать. Это не так просто само по себе — проимпортировать их и как-то хранить, и применять к объектам.

Каждое из свойств материалов участвует в расчетах.

Примеры — тут *(много матана и зауми)*:

- https://google.github.io/filament/Filament.md.html
- https://google.github.io/filament/Materials.md.html

Конвеер

Многозначный термин

- Шейдерный конвеер видеокарты: вершинный, геометрический, фрагментный шейдеры итд
- Архитектурный конвеер: организация проходов рендера, и данных для рендера

Мы поговорим об архитектурном конвеере растеризатора

Конвеер

Базовым понятием является проход рендера

- Итогом прохода рендера является выходной буфер: текстура
- Во время прохода в буфер можно только записывать
- Во время прохода рендера из буфера **нельзя читать!**

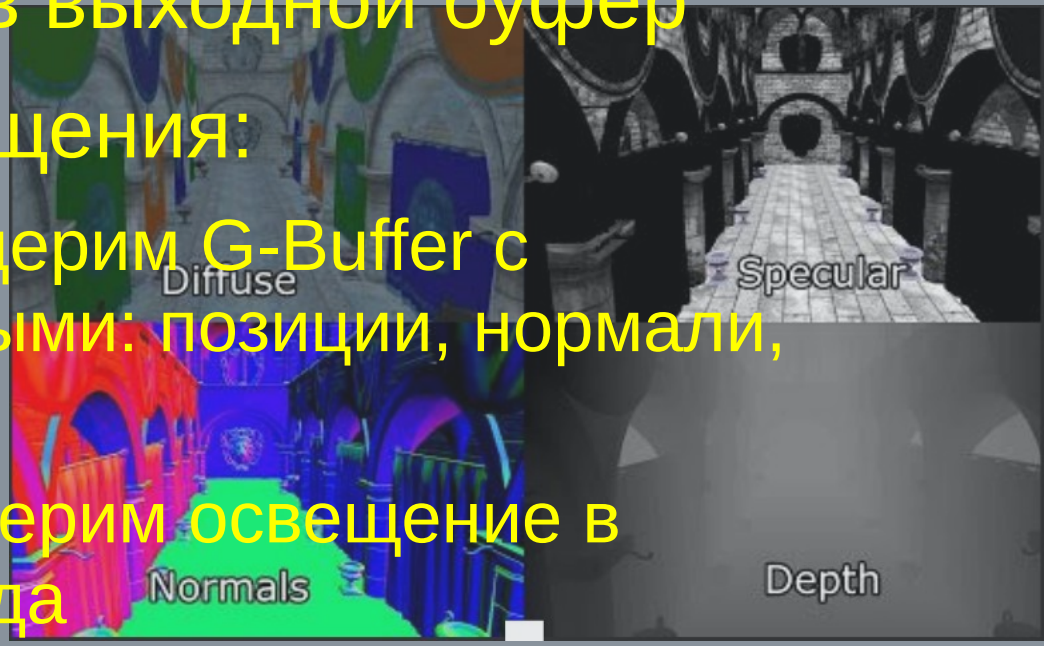
Конвеер

Освещение

- Прямой рендер (forward)
- Отложенный рендер (deferred), придуман Michael Deering в 1988 году, применяется в играх с 2001 года (игра Shrek для Xbox), в профессиональной графике в с 2007 года (3ds Max 2008)

1. Проход освещения

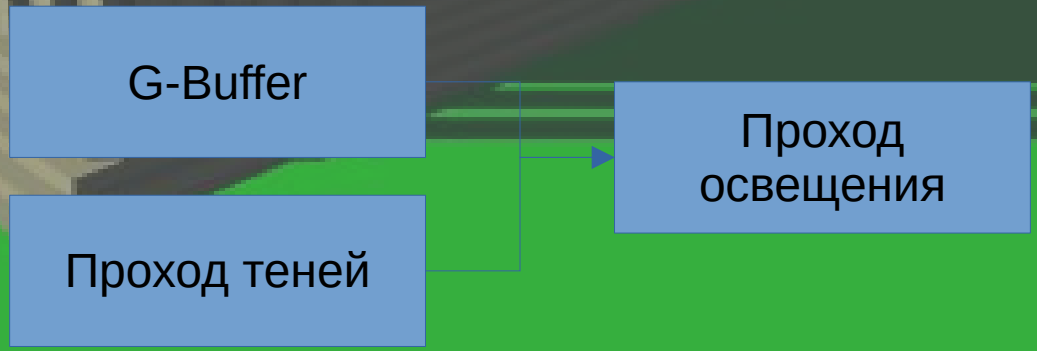
- Для прямого освещения: мы просто складываем картинку в выходной буфер
- Для отложенного освещения:
  - Первым проходом рендерим G-Buffer с промежуточными данными: позиции, нормали, материалы
  - Вторым проходом рендерим освещение в выходной буфер прохода



0. Проход карт теней

Lance Williams «CASTING CURVED SHADOWS ON CURVED SURFACES», 1978

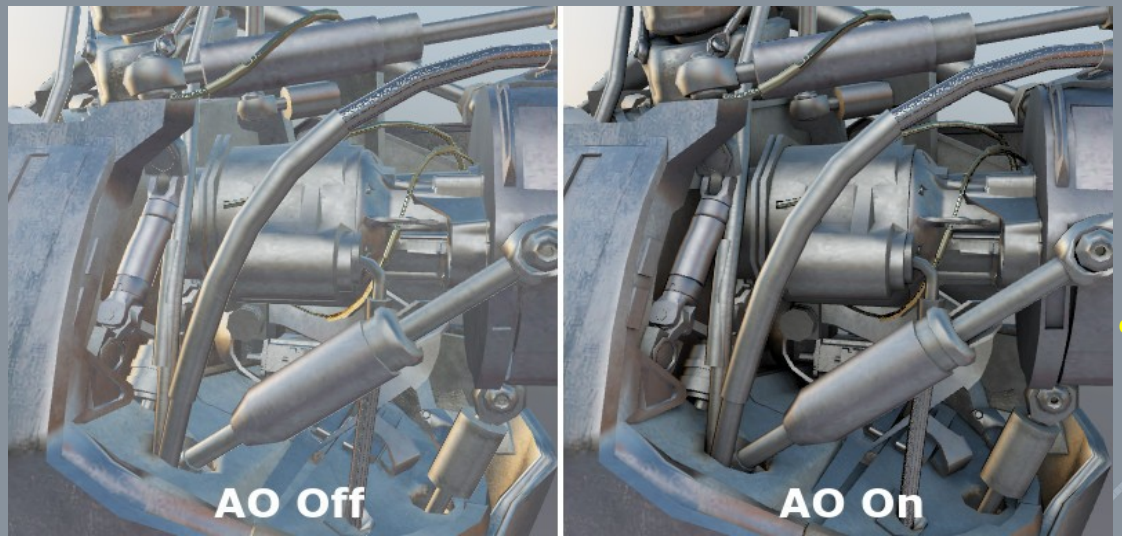
- Для техники карты теней нужен отдельный проход теней
- Для него нужен свой выходной буфер
- Проход теней должен предшествовать проходу освещения



2. Рассеяное затенение

Развитие алгоритмов AO: 1990-е годы

- *BUNNELL, M. 2005. Dynamic Ambient Occlusion and Indirect Lighting. In GPU Gems 2, M. Pharr, Ed. Addison Wesley, March. 2, 223–233.*

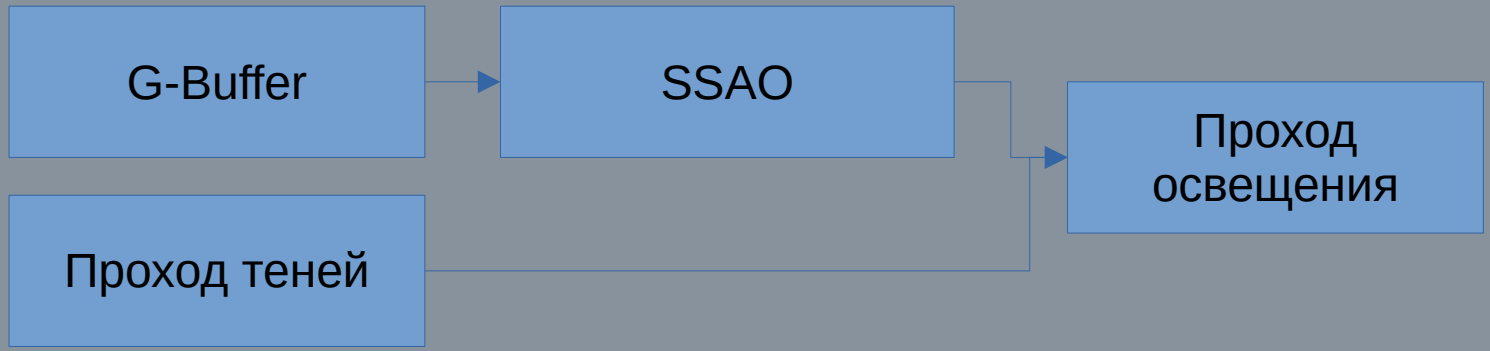


- Approximating Dynamic Global Illumination in Image SpaceTobias Ritschel, Thorsten Grosch, Hans-Peter Seidel, 2009?
- Реализация: Crysis, 2007 / CryEngine2

2. Рассеяное затенение

Техника относится к глобальному освещению

- Самая популярная — Screen Space Ambient Occlusion (SSAO)
- Для прямого освещения нужен дополнительный проход глубины (depth pre-pass)
- Для отложенного освещения данные глубины уже есть в G-Buffer
- Мы сравниваем глубину каждого пикселя с окружением. Чем больше разница, тем значительней затенение



Отражения

Множество техник:

- Планарные отражения (от плоскости): рендерим сцену ещё раз из нужной точки
- Планарные отражения для нищих: клонируем сцену зеркально и делаем зеркало полупрозрачным
- Пробы окружения (скайбокс и куб.карты)

  Texture and Reflection in Computer Generated Images / James F. Blinn and Martin E. Newel / University of Utah / Communications of the ACM, Volume 19, Issue 10 Pages 542 — 547 / 01.10.1976 — это сферическая (не кубическая!) проба окружения!
- Screen Space Reflections
- Ещё много техник

CHAOS CONSTRUCTIONS 2025
ST.PETERSBURG
23-24 AUG 2025

3. Пробы окружения

G-Buffer → SSAO

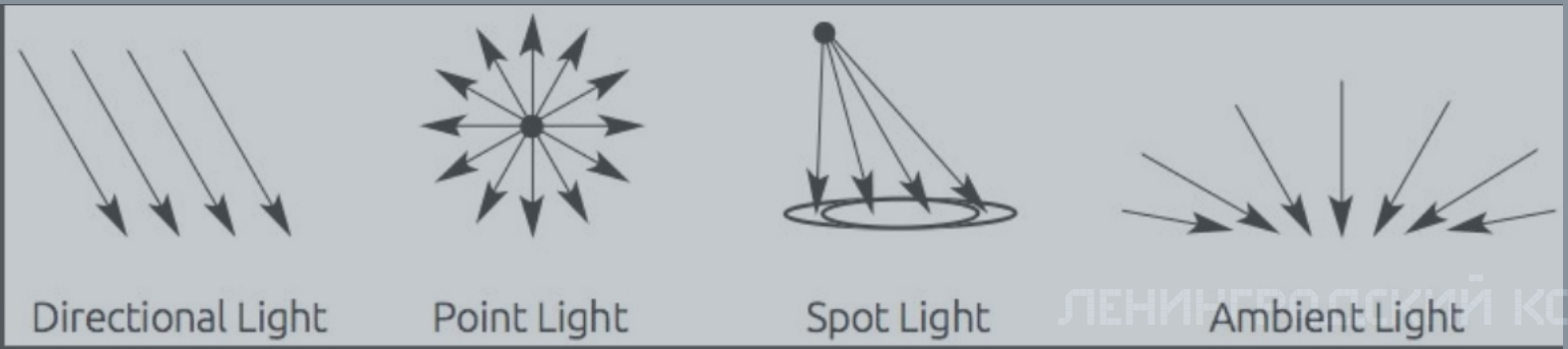
Проход теней → Пробы окружения

Проход освещения

Глобальное освещение

С Фонгом было просто. Мы брали константу ambient и прибавляли её к цвету пикселя.

- С PBR такой фокус больше не прокатывает. Фоновое освещение нужно **вычислять**
- Для вычисления фоновой компоненты существуют техники глобального освещения (уже рассмотрели минорную технику Ambient Occlusion)



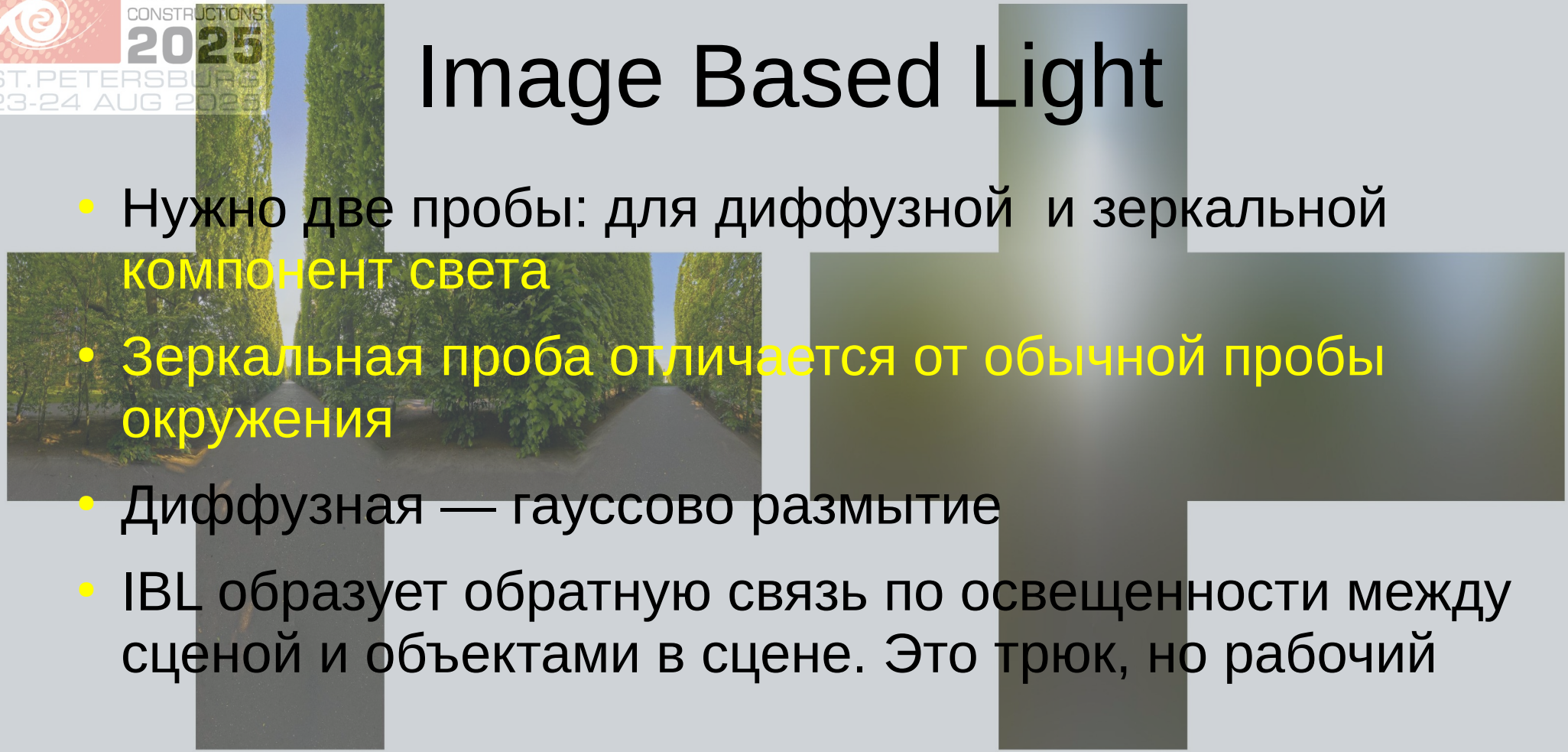
Глобальное освещение

- Ambient Cube (HalfLife 2, 2006)
- Image Based Light

  Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography / Paul Debevec / SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques
- Surfel Irradiance
- Screenspace Irradiance / Irradiance Cascades
- Probabilistic Radiance Transfer Probes

  Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments / Peter-Pike Sloan, Jan Kautz, John Snyder / ACM Transactions on Graphics (TOG), Volume 21, Issue 3 / Pages 527 — 536 / 01.07.2002

Image Based Light



- Нужно две пробы: для диффузной и зеркальной компонент света
- Зеркальная проба отличается от обычной пробы окружения
- Диффузная — гауссово размытие
- IBL образует обратную связь по освещенности между сценой и объектами в сцене. Это трюк, но рабочий

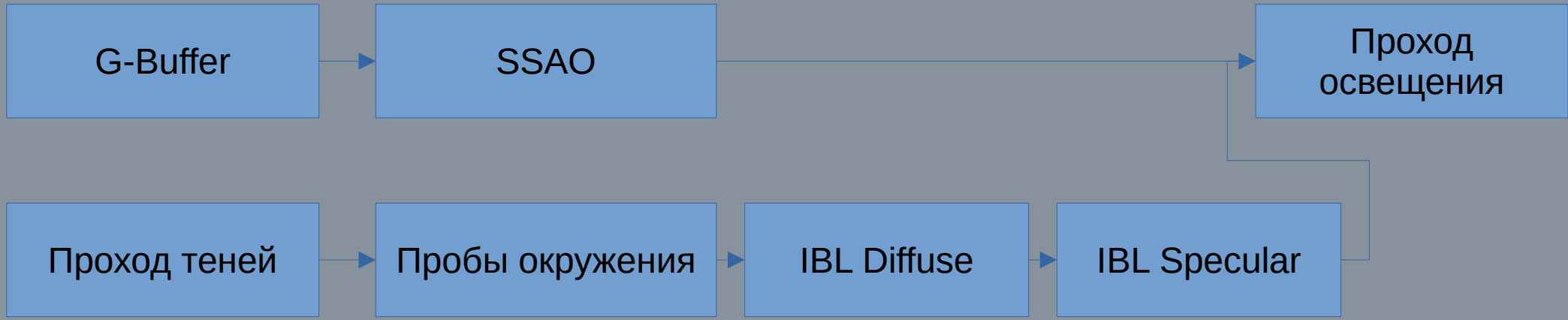
Standard cubemap

Irradiance cubemap

Image Based Light

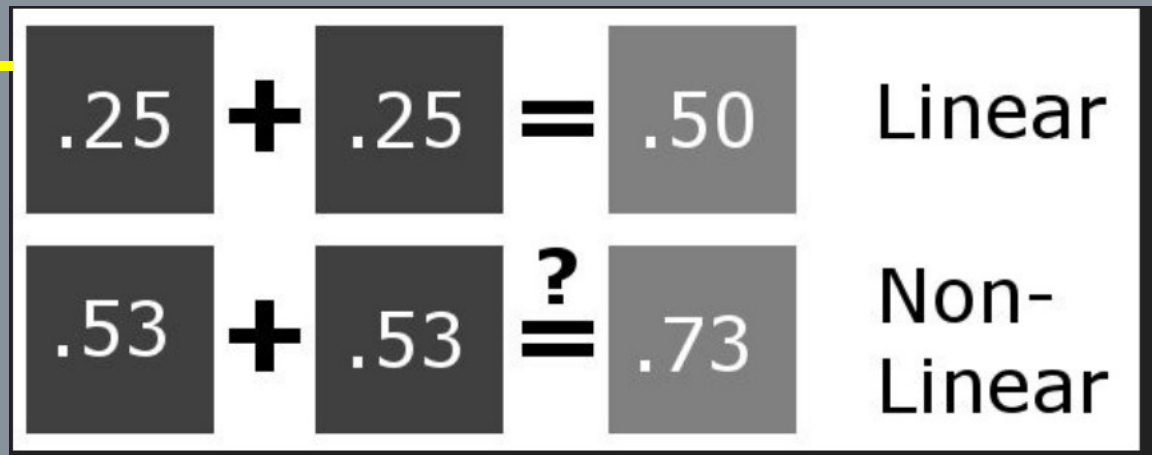
- Пробы помещают в определенных местах сцены
- Для простых сцен достаточно 1-2 проб на комнату
- Важно подчеркнуть разницу между внутренними помещениями и улицей (например)
- На фоновой картинке примеры проб

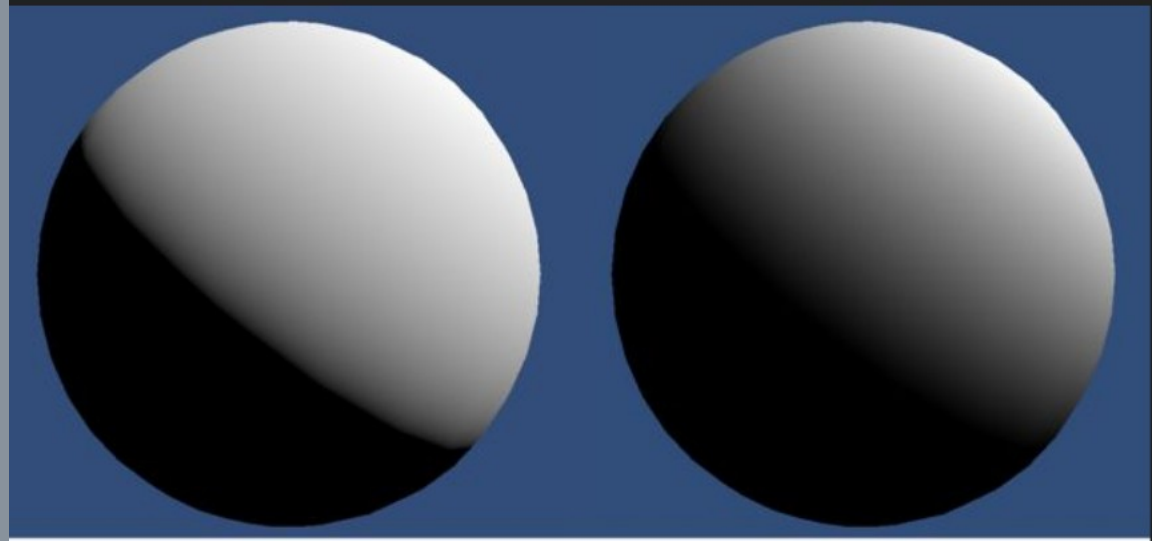
Image Based Light



HDR и тон-коррекция

- Человеческий глаз воспринимает цвета нелинейно
- Есть два цветовых пространства: линейное, и гамма-пространство
- Цветовые текстуры обычно идут в гамма-пространстве
- Текстуры не воспринимаемых непосредственно свойств (нормали, металличность итд) идут в линейном пространстве
- Приходится постоянно следить за единой размерностью значений
- А также приводить итоговую картинку к ожидаемой гамме
- Это и есть тон-коррекция





HDR и тон-коррекция

- HDR = High Dynamic Range, высокий динамический диапазон
- Речь об информации в очень темных и очень светлых участках
- Чтобы не терять информацию, нужно делать расчеты в full precision / double precision

HDR и тон-коррекция

- Тон-коррекция должна производиться после этапа освещения
- Конверсия HDR → SDR также производится после освещения
- Нам нужен проход пост-обработки :)
- Но постойте… А что ещё мы можем сделать на этапе пост-обработки?

Пост-обработка

Эффекты:

- Bloom (свечение) — нужно два прохода (вертикальное и горизонтальное сэмплирование)
- Размытие движения (motion blur) — технически похожий эффект
- Хроматическая аберрация
- Этих эффектов может быть много, они могут быть многопроходными, и их стоит организовывать в «стопки»

Пост-обработка

Что-нибудь ещё?..

Ну вообще-то да...

- Мы ещё забыли, что нужно решить что именно рендерить. Какие именно меши видны из позиции камеры
- И это решение — это здоровенный кусок работы

Выборка объектов

Выборка объектов

Оптимизация рендеринга — это отдельная сложная тема. Мы лишь вспомним:

- BSP = Binary Space Partitioning (Lucas Arts → Doom)
- Octrees
- Сортировку объектов по расстоянию
- LOD = level of details
- Поддержка в кремнии: face culling, depth test, frustum test, stencil test
- Mesh shaders

Запекание

Запекание (baking) — это использование предрассчитанных значений. Обычно они представлены в виде текстур. Наиболее ходовые карты:

- Ambient Occlusion
- Глобальное освещение (карты облученности IBL, фоновой компоненты, и подобное, в зависимости от типа глобального освещения)
- Пробы окружения

Запекание

Это типичный размен ЦП<→память.

То что мы не можем быстро посчитать, мы считаем заранее.

Но им нельзя злоупотреблять: бюджет как памяти, так и процессора/GPU ограничен!

Поиск баланса между вычислительной сложностью и ограниченностью ресурсов — основной вызов инженера.

Домашнее задание

- Объемные источники света (area lights)
- Объемный туман (volumetric fog, god lights)
- Анимация: skinning, morphing
- Рендер человека: волосы, lipsync, transclucency мягких тканей, ...
- Рендер воды и волн
- Процедурная генерация: облака, ландшафты, текстуры, …
- Физика: коллизии, поведение воды, занавесок, верёвок, травы, листьев, разрушаемость
- Трассировка лучей (весь класс алгоритмов); начните с знаковых полей расстояний
- Mesh shaders и буфер видимости
- Не-треугольные движки: воксельные, ray marching
- AR/VR
- ...

SIGGRAPH

- ACM SIGGRAPH = Association for Computing Machinery Special Interest Group on Computer Graphics and Interactive Techniques (1969 год)
- Научная организация, базируется в США

  Несколько периодических изданий, включая:
- ACM Transactions on Graphics (TOG)

  Это одно из самых престижных изданий в области компьютерной графики. Оно публикует статьи, посвященные теоретическим и практическим аспектам графики, включая алгоритмы, технологии и приложения
- SIGGRAPH Conference Proceedings

  Материалы конференции SIGGRAPH содержат статьи, доклады и постеры, представленные на ежегодной конференции. Это важный источник новейших исследований и разработок в области графики.

We could allow artists/developers to choose the Disney diffuse BRDF depending on the quality they desire and the performance of the target device. It is important to note however that the Disney diffuse BRDF is not energy conserving as expressed here.

4.6 Standard model summary

Specular term: a Cook-Torrance specular microfacet model, with a GGX normal distribution function, a Smith-GGX height-correlated visibility function, and a Schlick Fresnel function.

Diffuse term: a Lambertian diffuse model.

The full GLSL implementation of the standard model is shown in listing 9.

```
float D_GGX(float NoH, float a) {
    float a2 = a * a;
    float f = (NoH * a2 - NoH) * NoH + 1.0;
    return a2 / (PI * f * f);
}

vec3 F_Schlick(float u, vec3 f0) {
    return f0 + (vec3(1.0) - f0) * pow(1.0 - u, 5.0);
}

float V_SmithGGXCorrelated(float NoV, float NoL, float a) {
    float a2 = a * a;
    float GGXL = NoV * sqrt((-NoL * a2 + NoL) * NoL + a2);
    float GGXV = NoL * sqrt((-NoV * a2 + NoV) * NoV + a2);
    return 0.5 / (GGXV + GGXL);
}

float Fd_Lambert() {
    return 1.0 / PI;
```

```
float F_Schlick(float u, float f0, float f90) {
    return f0 + (f90 - f0) * pow(1.0 - u, 5.0);
}

float Fd_Burley(float NoV, float NoL, float LoH, float roughness) {
    float f90 = 0.5 + 2.0 * roughness * LoH * LoH;
    float lightScatter = F_Schlick(NoL, 1.0, f90);
    float viewScatter = F_Schlick(NoV, 1.0, f90);
    return lightScatter * viewScatter * (1.0 / PI);
}
```

Listing 8: *Implementation of the diffuse Disney BRDF in GLSL*

Figure 11 shows a comparison between a simple Lambertian diffuse BRDF and the high-quality Disney diffuse BRDF, using a fully rough dielectric material. For comparison purposes, the right sphere was mirrored. The surface response is very similar with both BRDFs but the Disney one exhibits some nice retro-reflections at grazing angles (look closely at the left edge of the spheres).

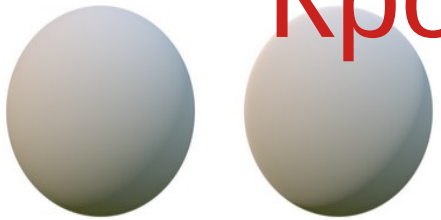


Figure 11: *Comparison between the Lambertian diffuse BRDF (left) and the Disney diffuse BRDF (right)*

We could allow artists/developers to choose the Disney diffuse BRDF depending on the quality they desire and the performance of the target device. It is important to note however that the Disney diffuse BRDF is not energy conserving as expressed here.

$$F_{Schlick}(v,h,f_0,f_{90}) = f_0 + (f_{90} - f_0)(1 - v \cdot h)^5 \tag{18}$$

The constant f_0 represents the specular reflectance at normal incidence and is achromatic for dielectrics, and chromatic for metals. The actual value depends on the index of refraction of the interface. The GLSL implementation of the Fresnel term requires the use of a `pow`, as shown in listing 5, which can be replaced by a few multiplications.

```
vec3 F_Schlick(float u, vec3 f0, float f90) {
    return f0 + (vec3(f90) - f0) * pow(1.0 - u, 5.0);
}
```

Listing 5: *Implementation of the specular F term in GLSL*

This Fresnel function can be seen as interpolating between the incident specular reflectance and the reflectance at grazing angles, represented here by f_{90}. Observation of real world materials show that both dielectrics and conductors exhibit achromatic specular reflectance at grazing angles and that the Fresnel reflectance is 1.0 at 90°. A more correct f_{90} is discussed in section 5.2.

Using f_{90} for f_0, the Schlick approximation for the Fresnel term can be optimized for scalar operations by refactoring the f_0 slightly. The result is shown in listing 6.

```
vec3 F_Schlick(float u, vec3 f0) {
    float f = pow(1.0 - u, 5.0);
    return f + f0 * (1.0 - f);
}
```

Listing 6: *Scalar optimization of the specular F term in GLSL*

4.5 Diffuse BRDF

In the diffuse term, f_m is a Lambertian function and the diffuse term of the BRDF becomes:

$$f_d(v,l) = \frac{\sigma}{\pi} \frac{1}{|n \cdot v||n \cdot l|} \int_\Omega D(m,\alpha) G(v,l,m)(v \cdot m)(l \cdot m) dm \tag{19}$$

Our implementation will instead use a simple Lambertian BRDF that assumes a uniform diffuse response over the microfacets hemisphere:

$$f_d(v,l) = \frac{\sigma}{\pi} \tag{20}$$

In practice, the diffuse reflectance σ is multiplied later, as shown in listing 8.

```
float Fd_Lambert() {
    return 1.0 / PI;
}

vec3 Fd = diffuseColor * Fd_Lambert();
```

Listing 7: *Implementation of the diffuse Lambertian BRDF in GLSL*

The Lambertian BRDF is obviously extremely efficient and delivers results close enough to more complex models.

However, the diffuse part would ideally be coherent with the specular term and take into account the surface roughness. Both the Disney diffuse BRDF [Burley12] and Oren-Nayar model [Oren94] take the roughness into account and create some retro-reflection at grazing angles. Given our constraints we decided that the extra runtime cost does not justify the slight increase in quality. This sophisticated diffuse model also renders image-based and spherical harmonics more difficult to express and implement.

For completeness, the Disney diffuse BRDF expressed in [Burley12] is the following:

$$f_d(v,l) = \frac{\sigma}{\pi} F_{Schlick}(n,l,1,f_{90}) F_{Schlick}(n,v,1,f_{90}) \tag{21}$$

Where:

$$f_{90} = 0.5 + 2 \cdot \alpha cos^2(\theta_d) \tag{22}$$

```
float F_Schlick(float u, float f0, float f90) {
```

Eric Heitz showed in [Heitz14] that the Smith geometric shadowing function is the correct and exact G term to use. The Smith formulation is the following:

$$G(v,l,\alpha) = G_1(l,\alpha) G_1(v,\alpha) \tag{6}$$

G_1 can in turn follow several models, and is commonly set to the GGX formulation:

$$G_1(v,\alpha) = G_{GGX}(v,\alpha) = \frac{2(n \cdot v)}{n \cdot v + \sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot v)^2}} \tag{7}$$

The full Smith-GGX formulation thus becomes:

$$G(v,l,\alpha) = \frac{2(n \cdot l)}{n \cdot l + \sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot l)^2}} \frac{2(n \cdot v)}{n \cdot v + \sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot v)^2}} \tag{8}$$

We can observe that the dividends $2(n \cdot l)$ and $2(n \cdot v)$ allow us to simplify the original function f_r by introducing a visibility function V:

$$f_r(v,l) = D(h,\alpha) V(v,l,\alpha) F(v,h,f_0) \tag{9}$$

Where:

$$V(v,l,\alpha) = \frac{G(v,l,\alpha)}{4(n \cdot v)(n \cdot l)} = V_1(l,\alpha) V_1(v,\alpha) \tag{10}$$

And:

$$V_1(v,\alpha) = \frac{1}{n \cdot v + \sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot v)^2}} \tag{11}$$

Heitz notes however that taking the height of the microfacets into account to correlate masking and shadowing leads to more accurate results. He defines the height-correlated Smith function thusly:

$$G(v,l,h,\alpha) = \frac{\chi^+(v \cdot h)\chi^+(l \cdot h)}{1 + \Lambda(v) + \Lambda(l)} \tag{12}$$

$$\Lambda(m) = \frac{-1 + \sqrt{1 + \alpha^2 tan^2(\theta_m)}}{2} = \frac{-1 + \sqrt{1 + \alpha^2 \frac{(1 - cos^2(\theta_m))}{cos^2(\theta_m)}}}{2} \tag{13}$$

Replacing $cos(\theta_m)$ by $n \cdot v$, we obtain:

$$\Lambda(v) = \frac{1}{2}\left(\frac{\sqrt{\alpha^2 + (1 - \alpha^2)(n \cdot v)^2}}{n \cdot v} - 1\right) \tag{14}$$

From which we can derive the visibility function:

$$V(v,l,\alpha) = \frac{0.5}{n \cdot l\sqrt{(n \cdot v)^2(1 - \alpha^2) + \alpha^2} + n \cdot v\sqrt{(n \cdot l)^2(1 - \alpha^2) + \alpha^2}} \tag{15}$$

The GLSL implementation of the visibility term, shown in listing 3, is a bit more expensive than we would like since it requires two `sqrt` operations.

```
float V_SmithGGXCorrelated(float NoV, float NoL, float roughness) {
    float a2 = roughness * roughness;
    float GGXV = NoL * sqrt(NoV * NoV * (1.0 - a2) + a2);
    float GGXL = NoV * sqrt(NoL * NoL * (1.0 - a2) + a2);
    return 0.5 / (GGXV + GGXL);
}
```

Listing 3: *Implementation of the specular V term in GLSL*

We can optimize this visibility function by using an approximation after noticing that all the terms under the square roots are squares and that all the terms are in the $[0..1]$ range:

$$V(v,l,\alpha) = \frac{0.5}{n \cdot l(n \cdot v(1 - \alpha) + \alpha) + n \cdot v(n \cdot l(1 - \alpha) + \alpha)} \tag{16}$$

This approximation is mathematically wrong but saves two square root operations and is good enough for real-time mobile applications, as shown in listing 4.

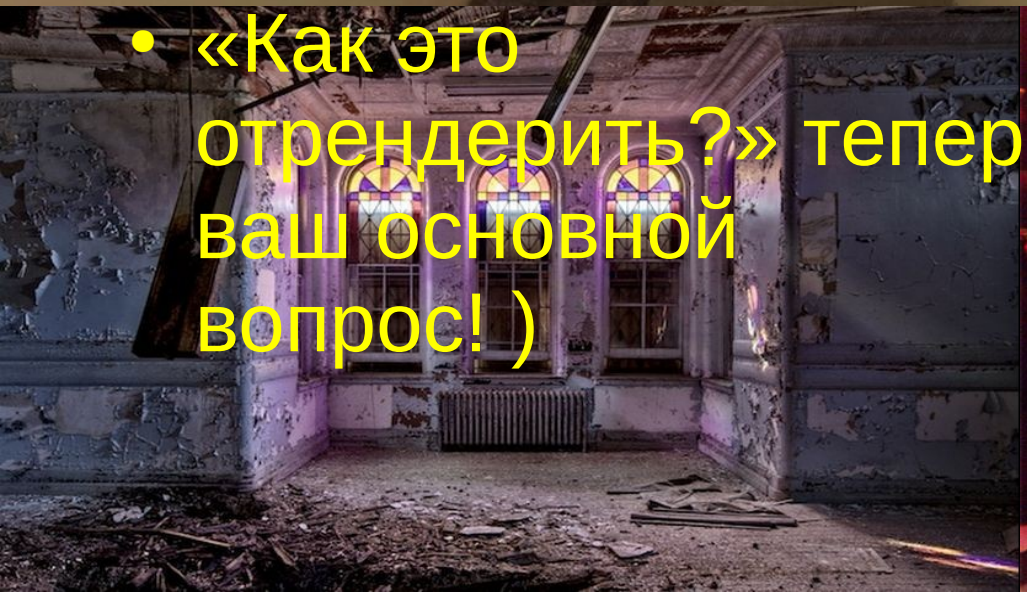

FPS /// x,y,z=0,0,0 /// direction=0,
FPS /// x,y,z=-3,
1,0 /// direction=3,2,0 ///
- прямой рендеринг освещения
- модель освещения PBR
- KHR_materials_transmission
- KHR_materials_volume
- KHR_materials_ior
Реклама: Party Engine

Party Engine

- пространственные источники света: световые трубки, лайтбоксы
- объемный туман
- рендеринг ttf-шрифтов
- субтитры
- динамические текстуры и видео на текстурах
- И другое)

Поздравляю!

- Проклятие разработчика графики разблокировано!
- «Как это отрендерить?» теперь ваш основной вопрос!)



Дополнительные материалы

- https://learnopengl.com/
- Курс Javi Agenjo: Real Time Graphics (2021-2022)
- Курс Javi Agenjo: Computer Graphics (2018)

PartyEngine

/// ETA 19.294 ms 306 FPS /// x,y,z=33,38,6 /// direction=-15,-40,7 /// flight mode=on

Спасибо за внимание!