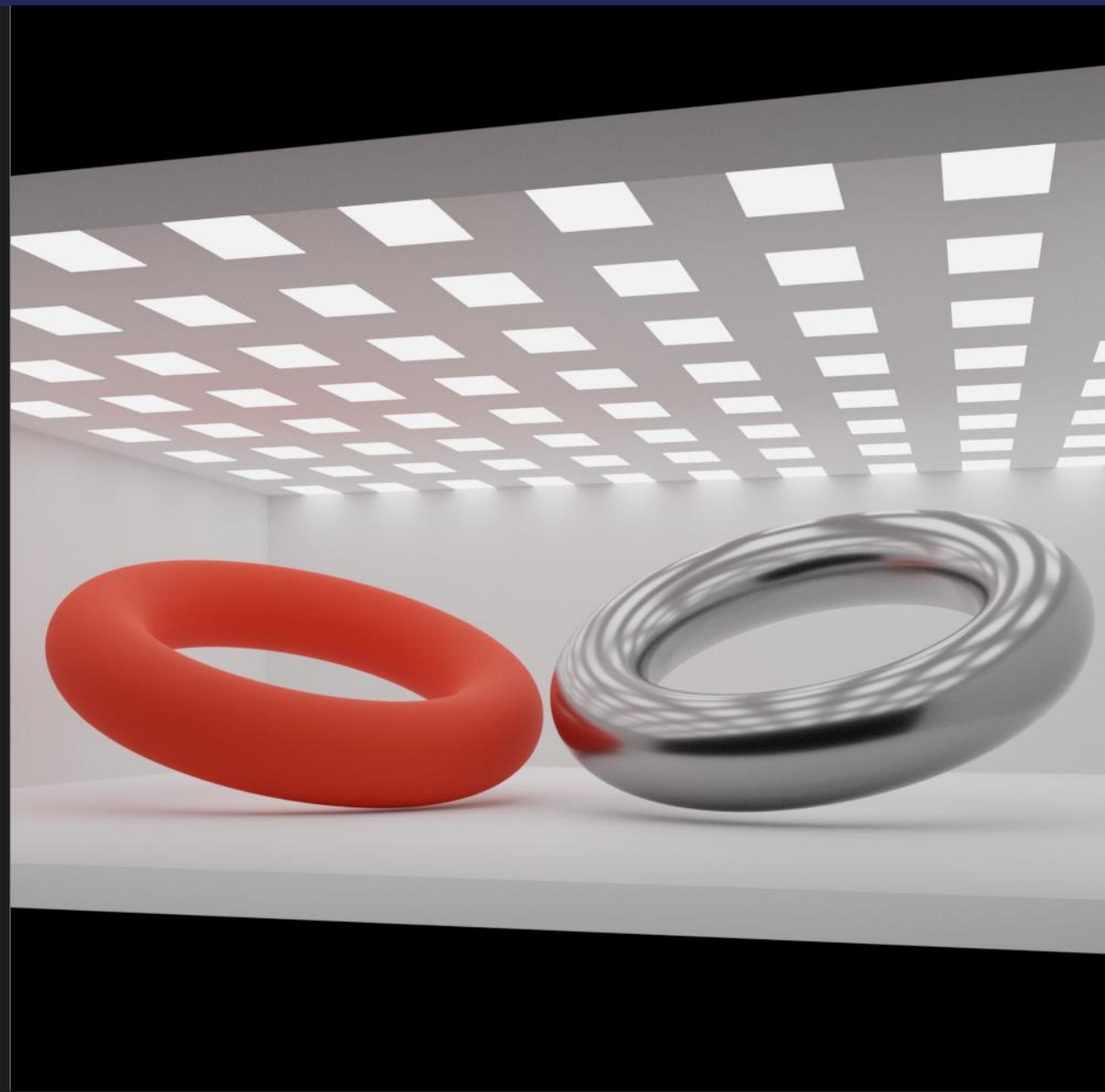
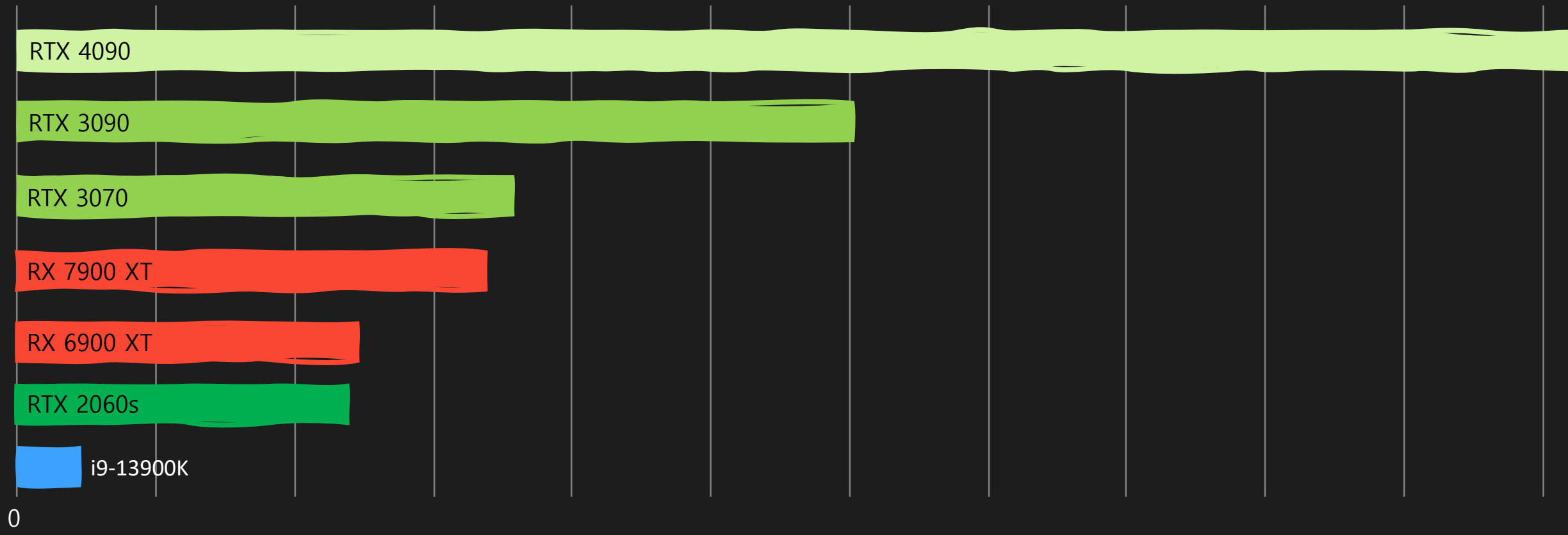
100강 Special Edition 3 : Cycles 렌더 엔진 설정

Cycles 렌더 엔진 설정



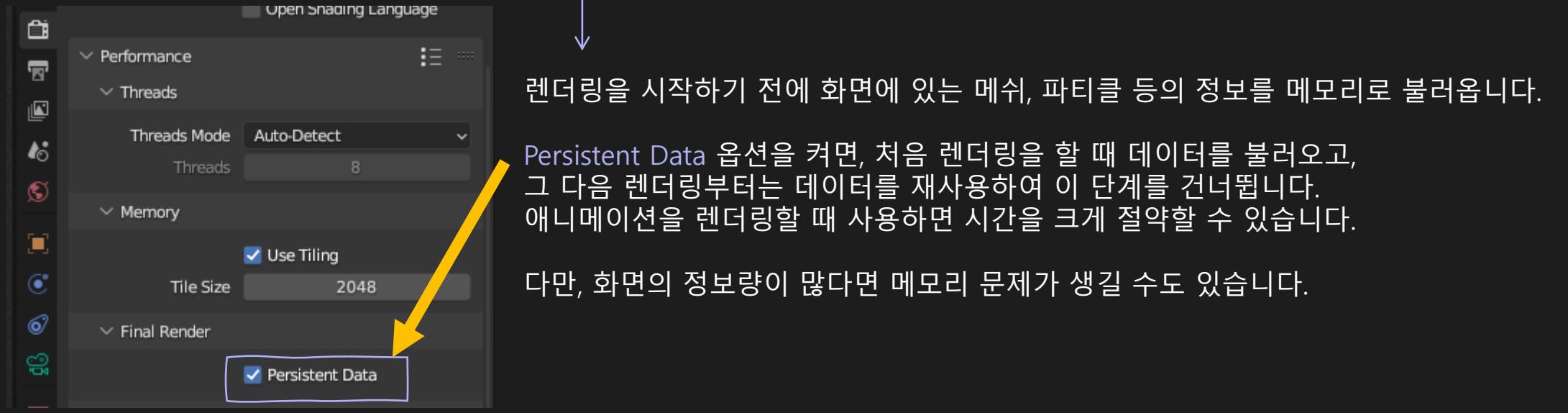
Blender Opendata

https://opendata.blender.org/ 에서 블렌더 벤치마크를 확인할 수 있습니다.
렌더속도는 GPU가 압도적이므로, 사이클 렌더링을 위해서는 그래픽카드에 최우선적으로 투자해야 합니다.



렌더링의 시작과 끝

렌더 버튼을 눌렀을 때, 렌더링을 위한 자료를 수집하고 렌더링을 시작합니다. 그 이후 디노이즈와 같은 후처리까지 포함한 것이 총 렌더 시간입니다.
이 강의에서는 실제 렌더링 시간만을 생각합니다.



렌더링을 시작하기 전에 화면에 있는 메쉬, 파티클 등의 정보를 메모리로 불러옵니다.

Persistent Data 옵션을 켜면, 처음 렌더링을 할 때 데이터를 불러오고,
그 다음 렌더링부터는 데이터를 재사용하여 이 단계를 건너뜁니다.
애니메이션을 렌더링할 때 사용하면 시간을 크게 절약할 수 있습니다.

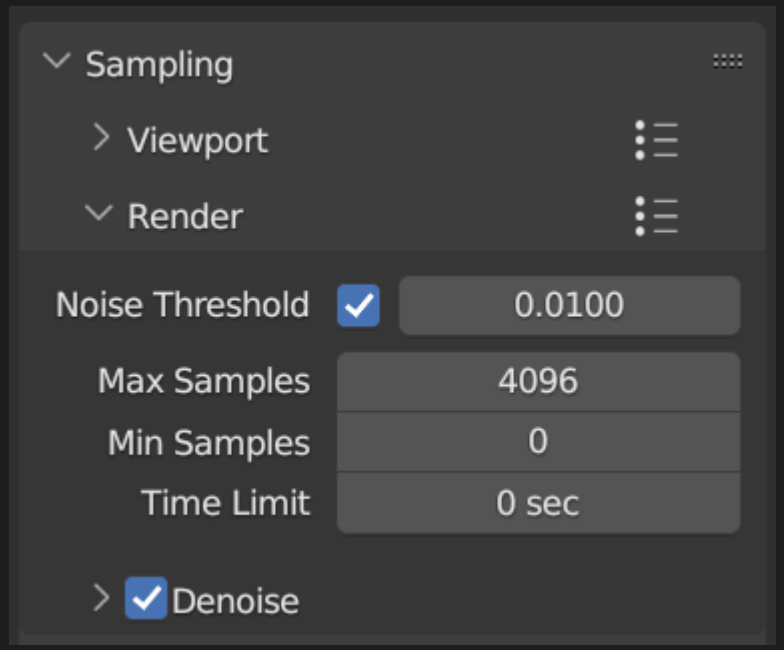
다만, 화면의 정보량이 많다면 메모리 문제가 생길 수도 있습니다.

0번 규칙 : 최신 버전을 사용합니다

블렌더가 업데이트될 때마다 렌더링 속도도 빨라집니다.
애드온 호환등의 문제로 불가피하게 이전 버전을 사용하더라도,
가능하다면 렌더링만큼은 최신버전을 사용하는 것이 좋습니다.

샘플수

사이클 렌더 엔진은 샘플 수만큼 광선을 쏘아서 렌더링을 구현합니다.
따라서 기본적으로 렌더링 시간은 샘플 수에 비례합니다.



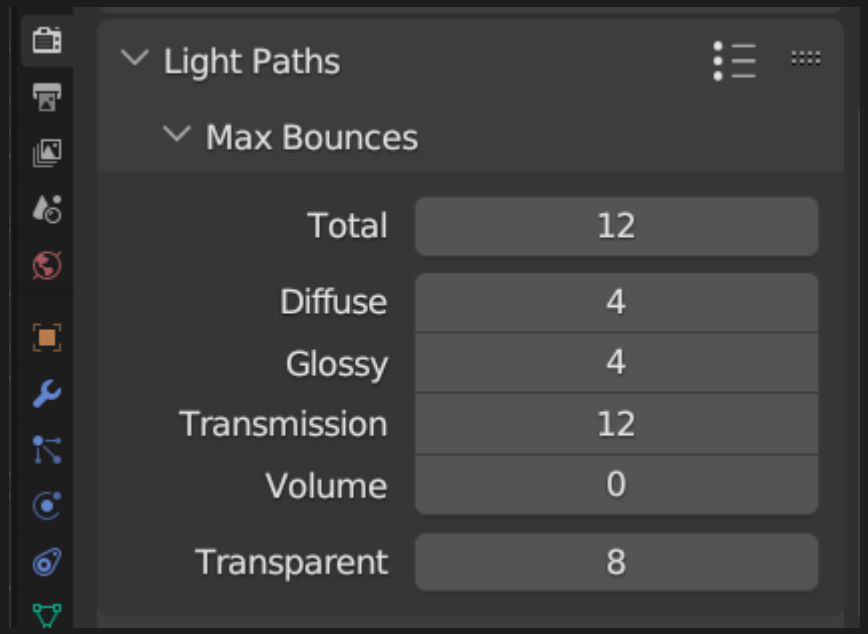
Noise Threshold : 화면상의 노이즈가 충분히 적다고 판단하면,
최대 샘플에 도달하지 않아도 해당 영역의 연산을 끝마칩니다.

0부터 1 사이의 값을 가지며, 0에 가까울 수록 엄격하게 판단하고,
1에 가까울 수록 느슨하게 판단합니다.
대체로 0.01은 아주 엄격한 판단으로 여겨집니다.

Noise Threshold는 렌더 시간을 크게 절약하지만,
화면의 각 영역마다 샘플 수가 달라지므로,
드물게 디노이즈가 비정상적으로 작동할 수 있습니다.

Light Bounces

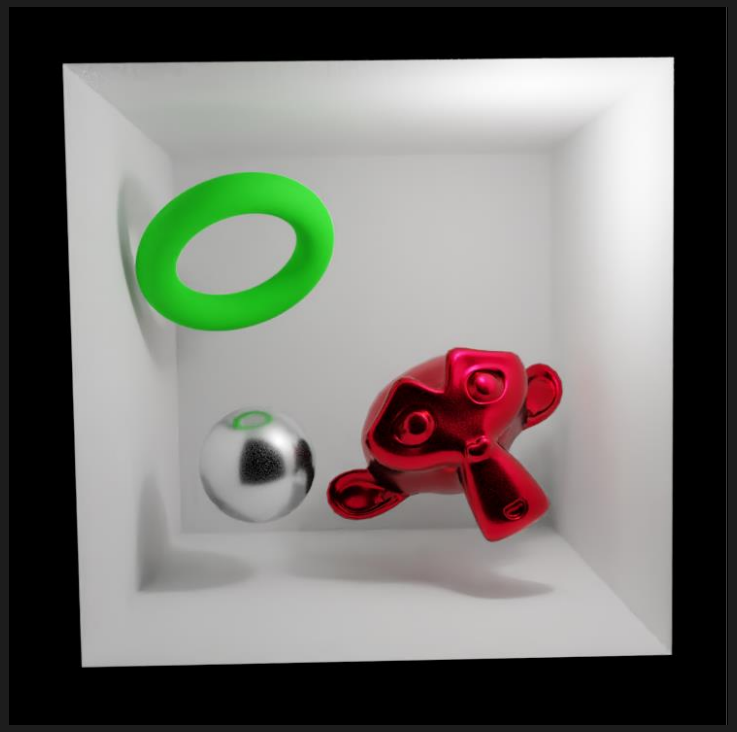
최대 바운스 수는 광선의 반사/굴절이 일어나는 횟수입니다. 숫자가 작을수록 렌더 시간이 빨라집니다.



최대 바운스는 극단적으로 줄여도 생각보다 괜찮습니다.

기본값 (Diffuse 4, Glossy 4)

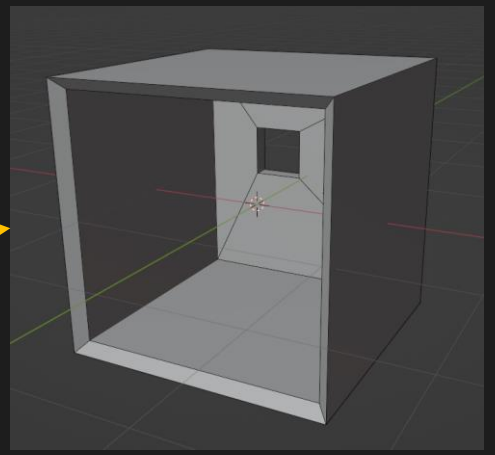
Diffuse 2, Glossy 1

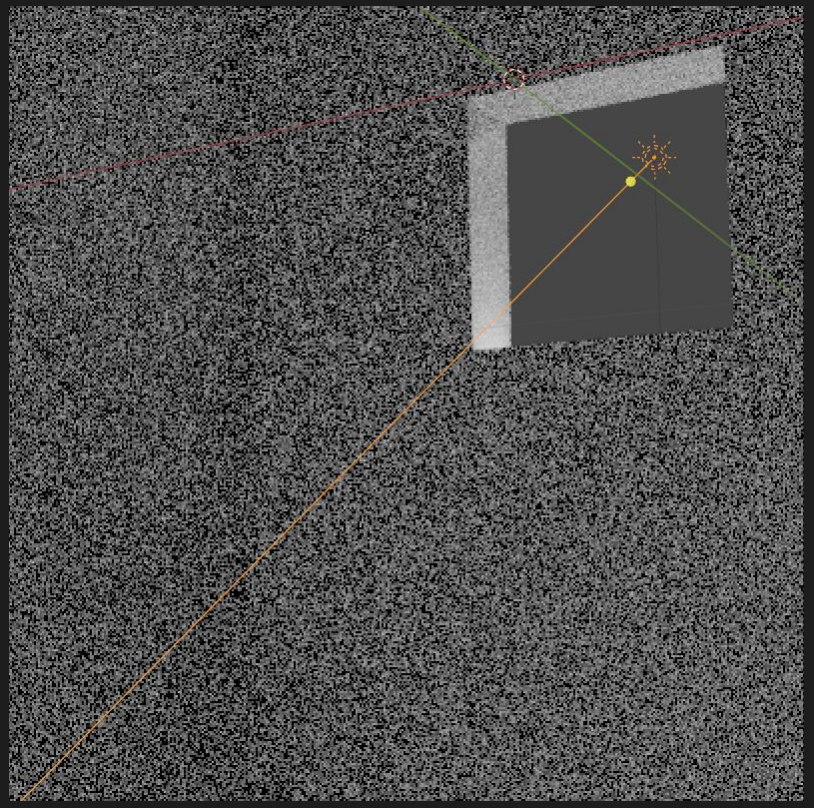


라이팅 문제 (1) : 간접 조명

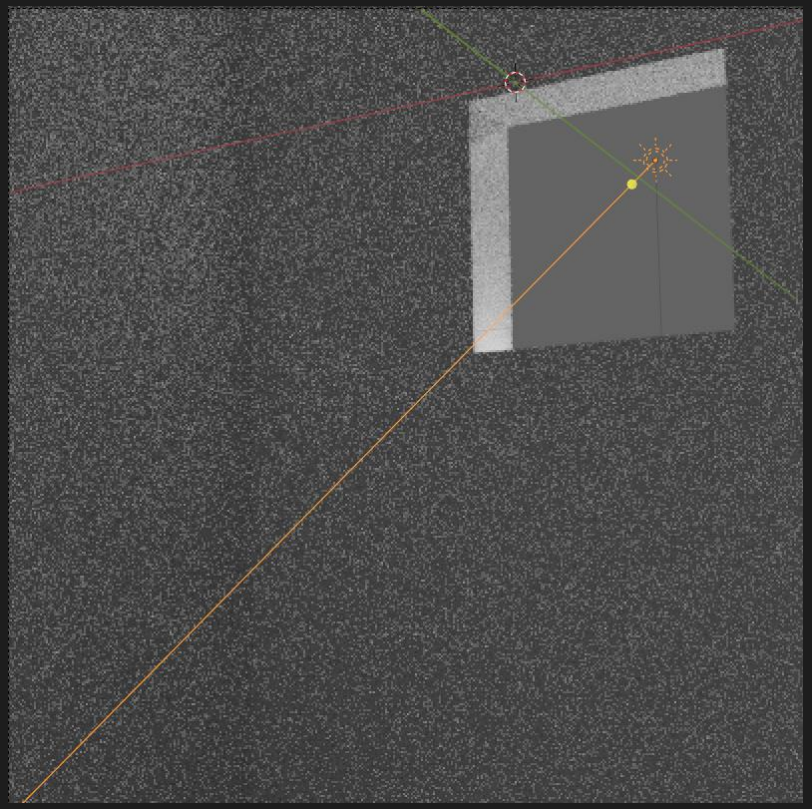
Cycles 렌더 엔진은 Eevee와 다르게 간접광을 계산할 수 있을 뿐, 계산이 빠르다고는 할 수 없습니다.
여러 번 빛이 반사되는 간접조명은 렌더시간에 악영향을 줍니다.

실내 장면을 만들어야 한다면, 카메라에서 보이지 않는 뒤쪽을 완전히 터서
구멍을 뚫는다면 큰 도움이 됩니다.
(광선을 최대한 빨리 배경으로 내보내 연산을 끝내려는 목적입니다.)





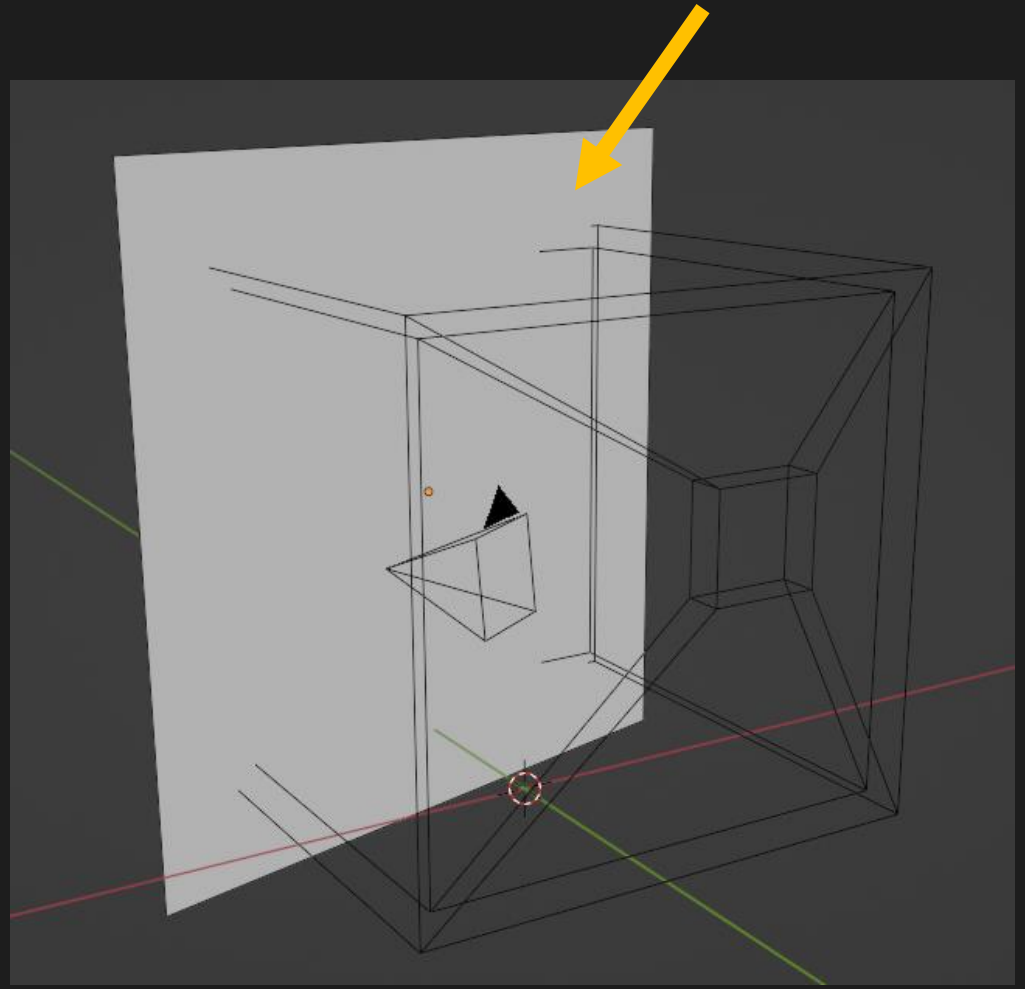
기본 상태

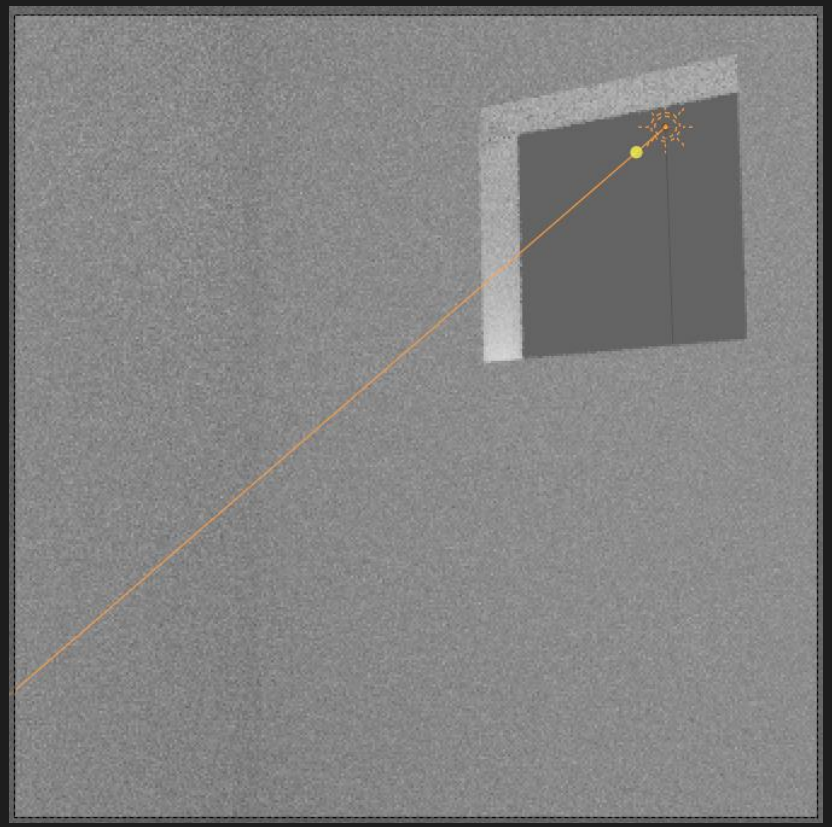


건물의 뒤쪽을 뚫은 경우

라이팅 문제 (1) : 간접 조명

건물을 물리적으로 뚫는 게 곤란하다면,
카메라가 보지 않는 뒤쪽에 거대한 Emission 평면을 갖다 두는 것도 벽이 뚫린 것과 같은 효과입니다.

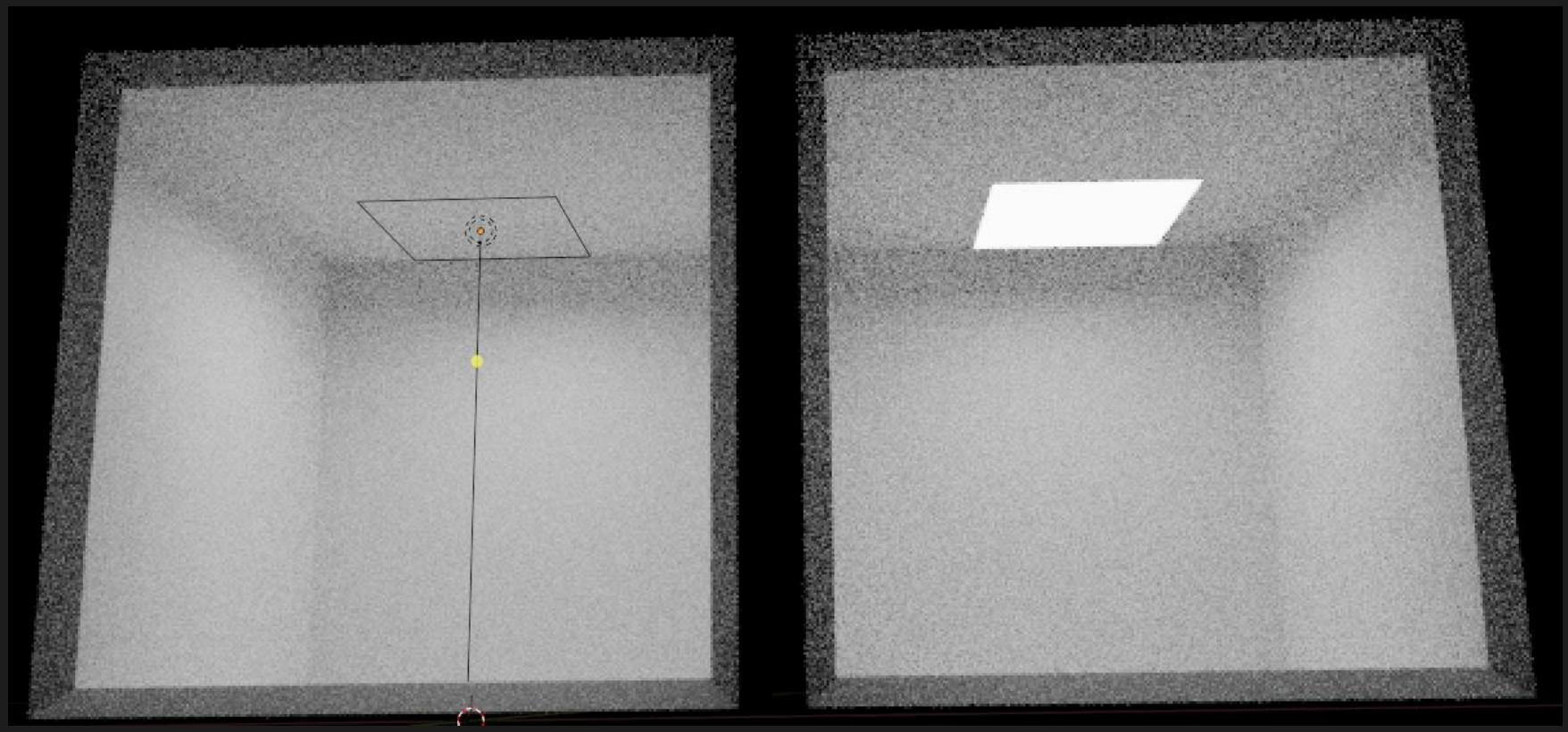




※단색이 아니라 Hdri 배경을 사용하면 더 효과적입니다.

라이팅 문제 (2) : 조명이 아닌 광원

Cycles는 조명 오브젝트가 아닌 Emission재질도 빛으로써 상호작용할 수 있습니다.
하지만 Emission을 조명으로 사용하는 것은 연산 효율이 낮아 추천되지 않**았**습니다.
그러나 현재는 Emission도 충분히 조명의 역할을 할 수 있도록 업데이트되었습니다.

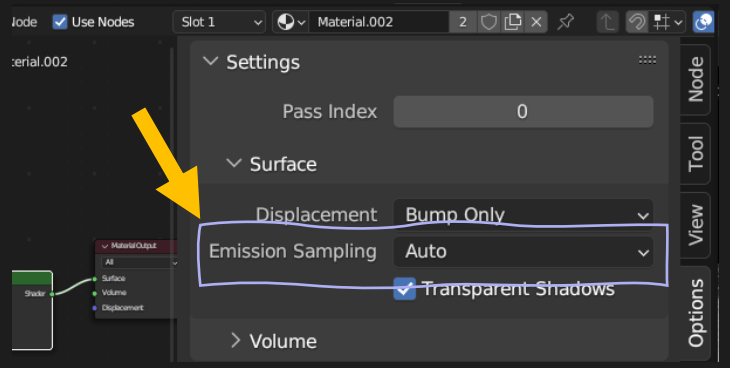


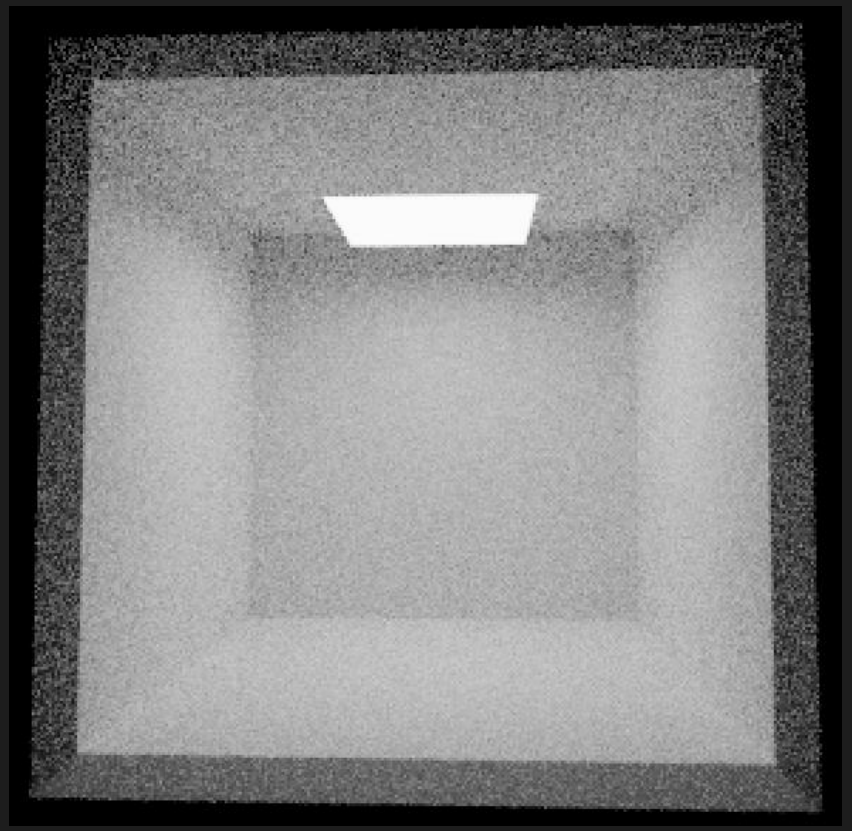
Area Light

Emission Plane

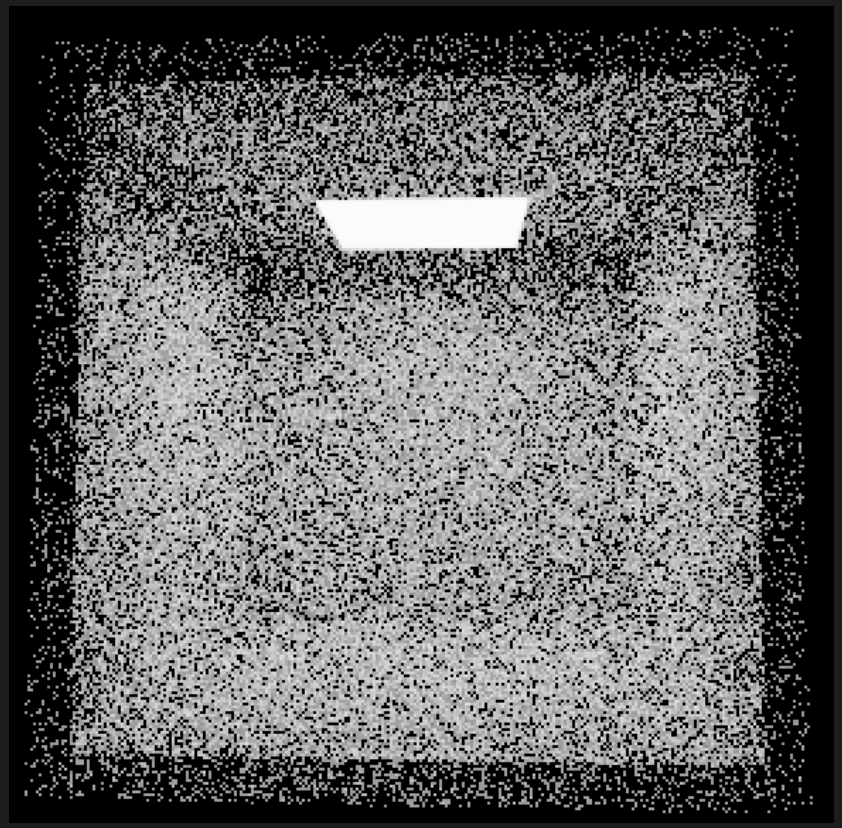
라이팅 문제 (2) : 조명이 아닌 광원

Emission을 조명으로 사용하는 Material 설정에서, Emission Sampling을 확인합니다.
기본값은 Auto이며, 밝기가 충분히 밝을 때 조명으로 간주하여 광선이 빠르게 찾아가도록 합니다.
이것을 끄면 연산 효율이 극도로 떨어집니다.





Emission Sampling Auto(기본값)

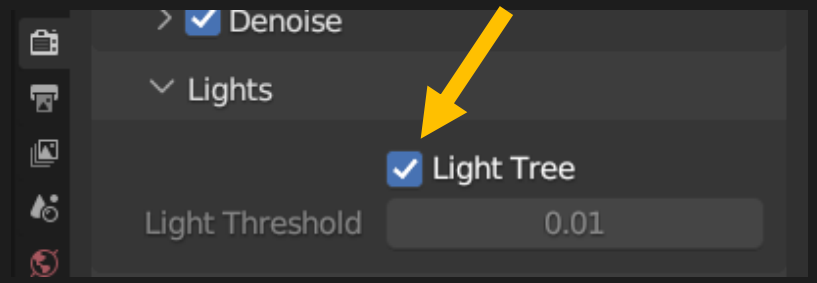


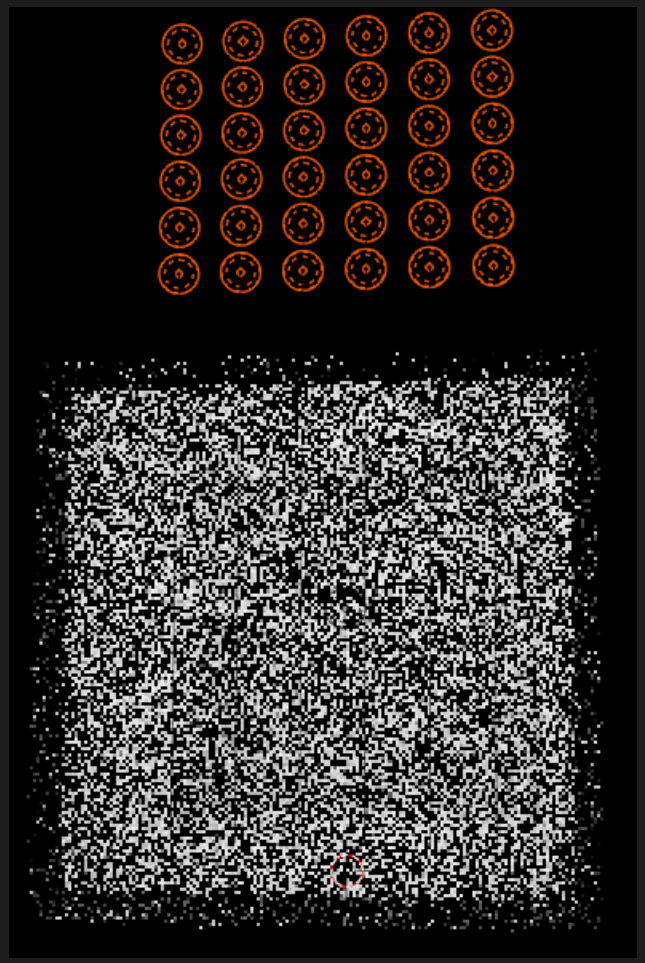
Emission Sampling 을 껐을 때

라이팅 문제 (3) : 너무 많은 조명

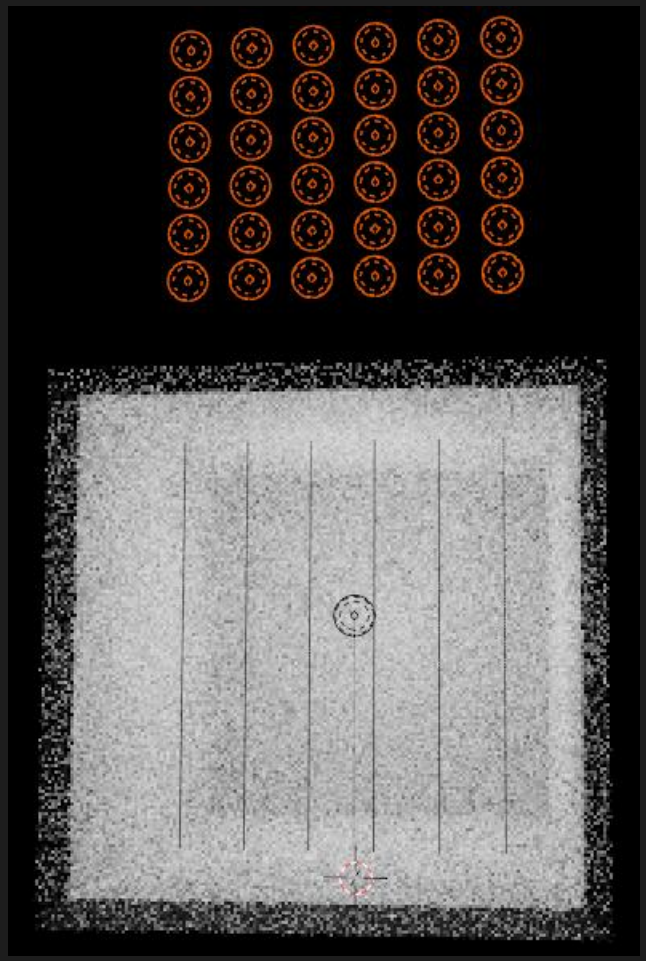
Cycles 렌더 엔진은 효율을 높이기 위해 매 연산마다 조명의 위치를 확인합니다.
매 연산마다 모든 광선은 모든 조명의 위치를 확인하므로,
조명이 많을수록 연산량이 늘어납니다.

이로 인해 발생하는 문제 중 하나는,
화면 밖에 조명이 많으면, 보이지도 않는 조명을 계산하느라
연산에 악영향을 준다는 것이**었**습니다.
하지만 3.5버전에서 생긴 Light Tree 기능을 사용하면
닿지 않는 조명에 의한 연산량이 크게 줄어듭니다.





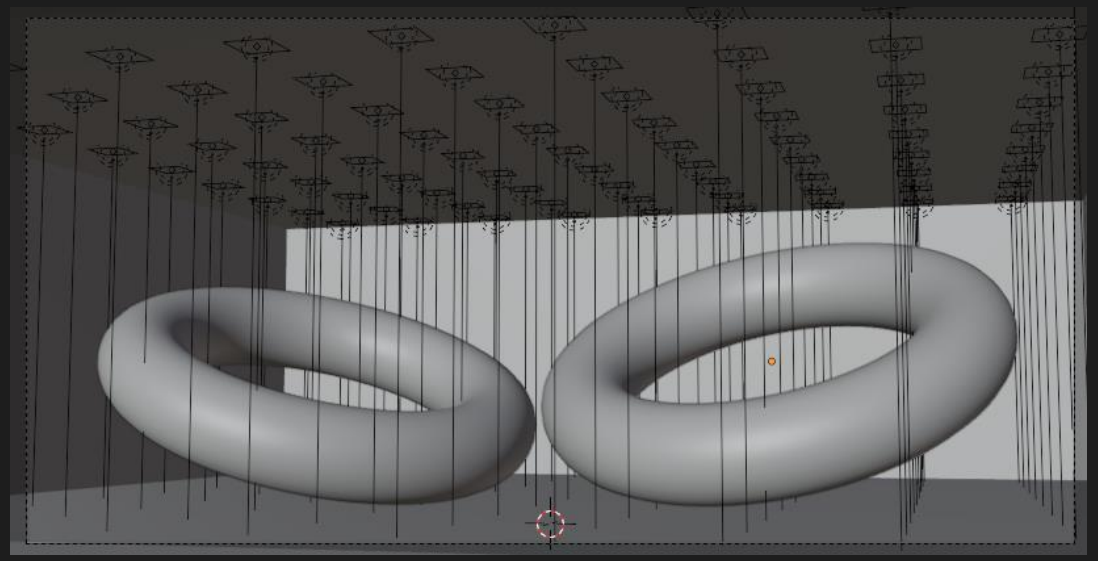
Light Tree Off

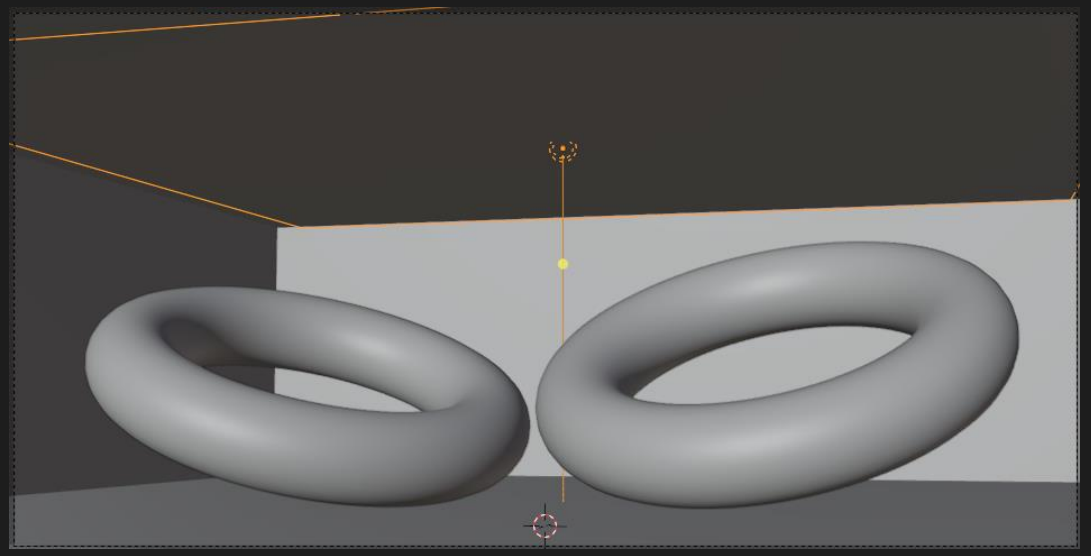


Light Tree On (기본값)

라이팅 문제 (3) : 너무 많은 조명

조명이 여러 개일 때 보다 한 개일 때 더 빠릅니다.
아래의 렌더 시간은 3번 렌더링한 평균입니다.

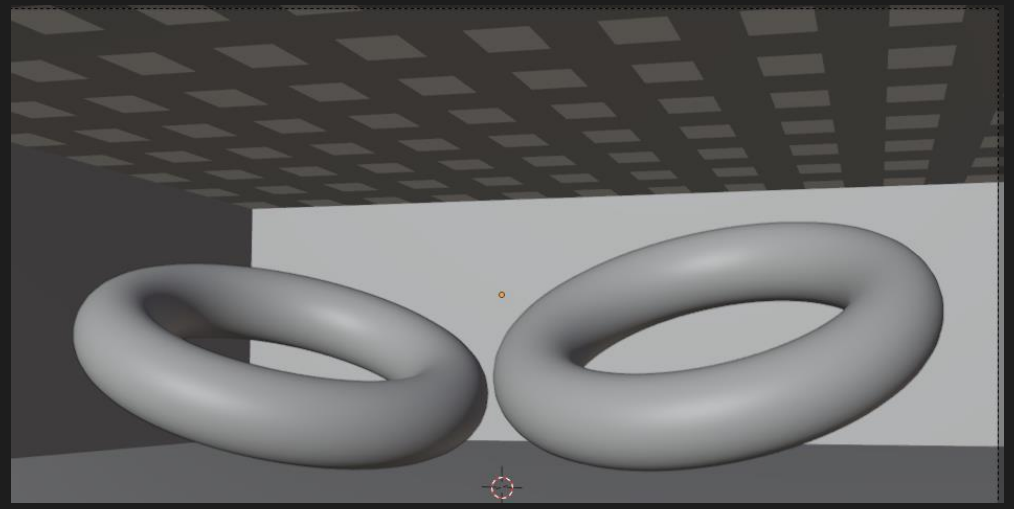




라이팅 문제 (3) : 너무 많은 조명

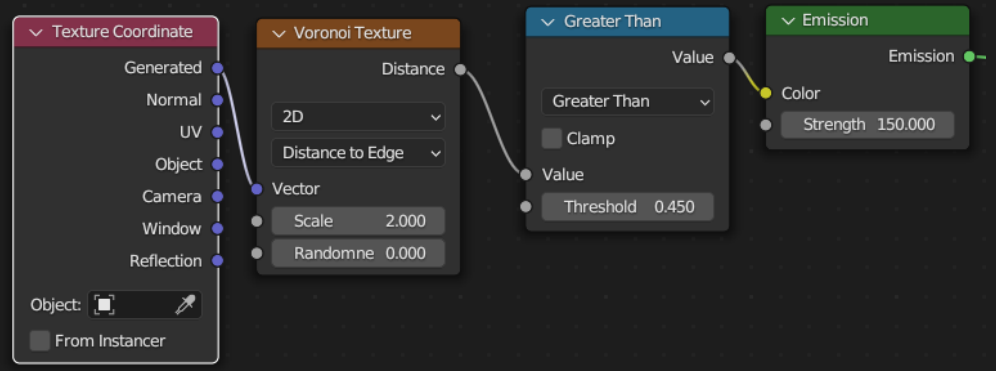
천장을 Emission으로 교체 : 1분 40초

Emission Sampling 덕분에 여러 개의 Area Light와 비슷한 성능을 가집니다.



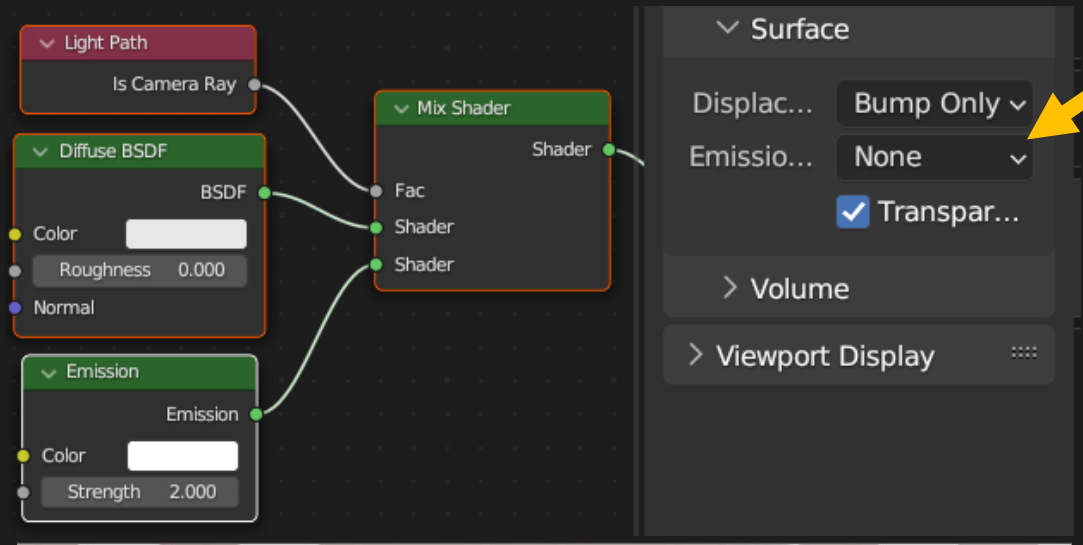
하나의 Area Light에 텍스쳐 사용 : 5분 18초

조명 개수가 하나여도 텍스쳐를 사용하면
오히려 여러개일 때보다 느려질 수 있습니다!



라이팅 문제 (3) : 너무 많은 조명

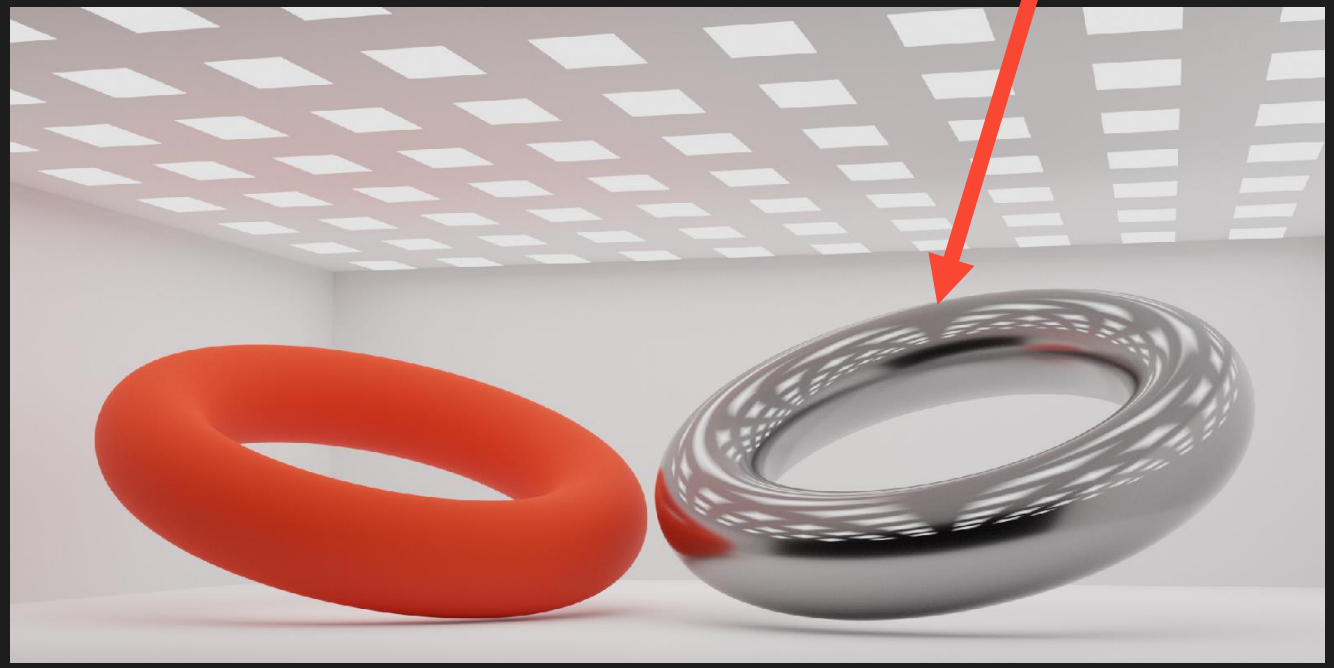
Area Light를 사용하고,
Emission은 카메라에만 보이게 : 59초



Emission이 상호작용하지 않는 경우, (혹은 않아야 하는 경우)
Emission Sampling을 끄는 쪽이 더 빠릅니다!

Area Light는 Glossy에 안보이게
+ Emission은 카메라에만 보이게
+ 가짜 광택 : 57초

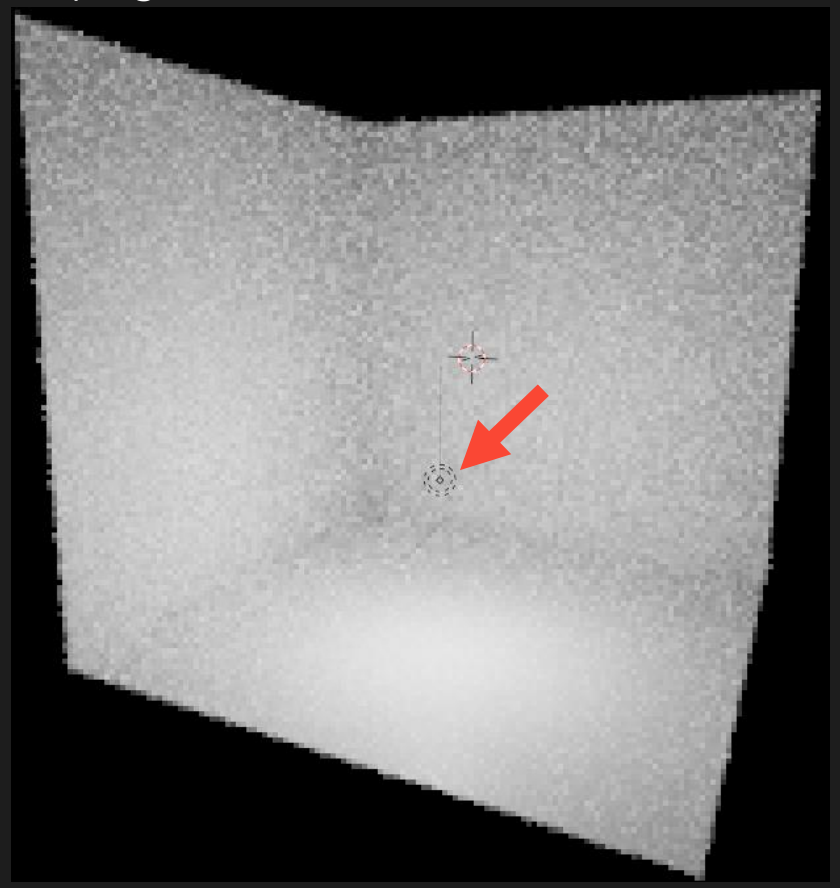
Reflection 좌표를 사용하여
가짜 반사광을 넣은 것.



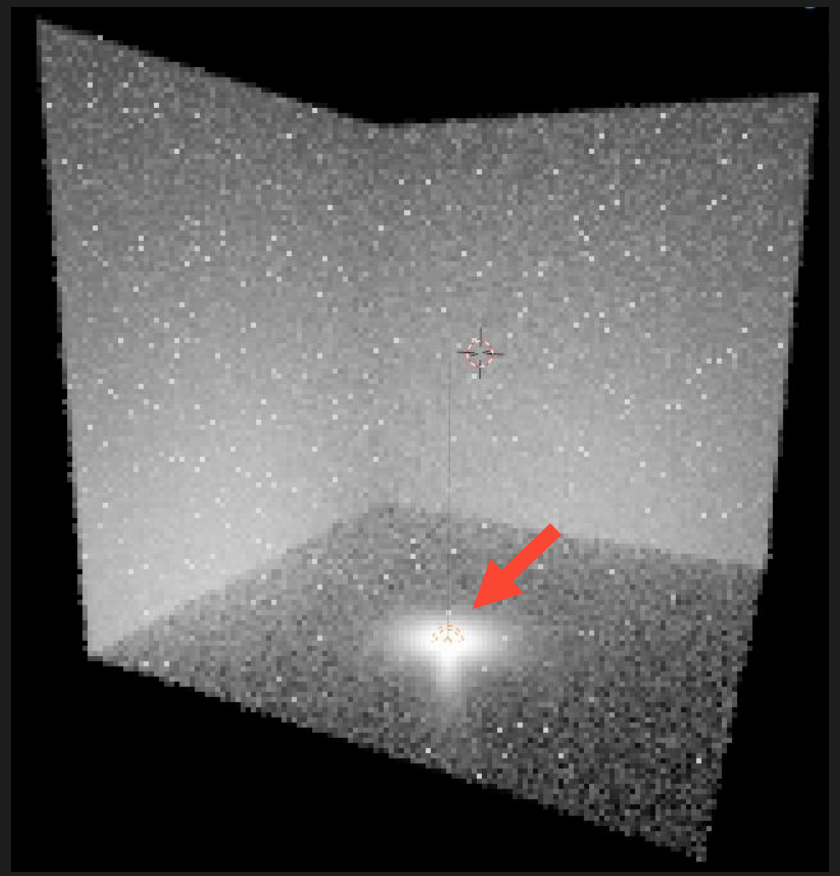
그 외

조명의 위치

포인트라이트를 표면에 가까이 두는 것은 좋지 않습니다.
빛의 세기는 거리의 제곱에 반비례하므로, 가까이 붙은 조명은 표면을 아주 강하게 비춥니다.
그러면, 밝아진 표면이 **조명이 아닌 광원**의 역할을 해서,
Emission Sampling을 사용하지 않는 Emission 같은 역할을 하여, 렌더링에 악영향을 끼칩니다.



조명이 바닥과 멀리 떨어진 경우

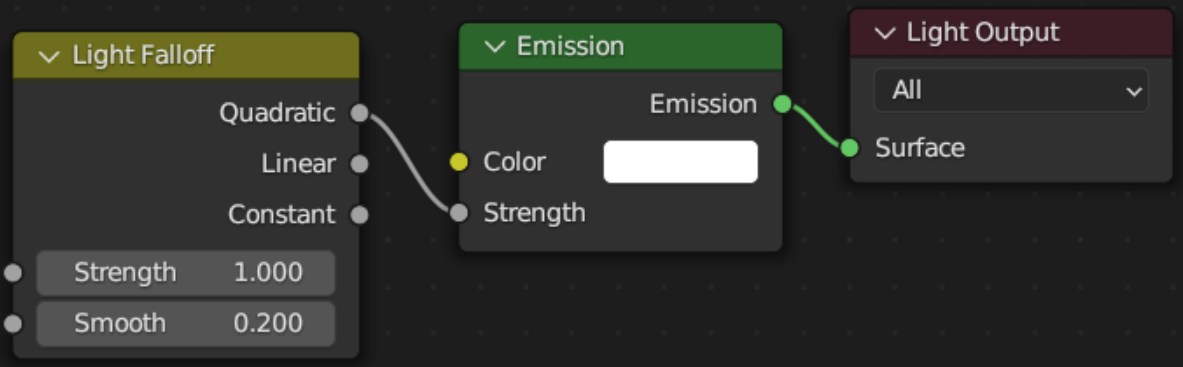


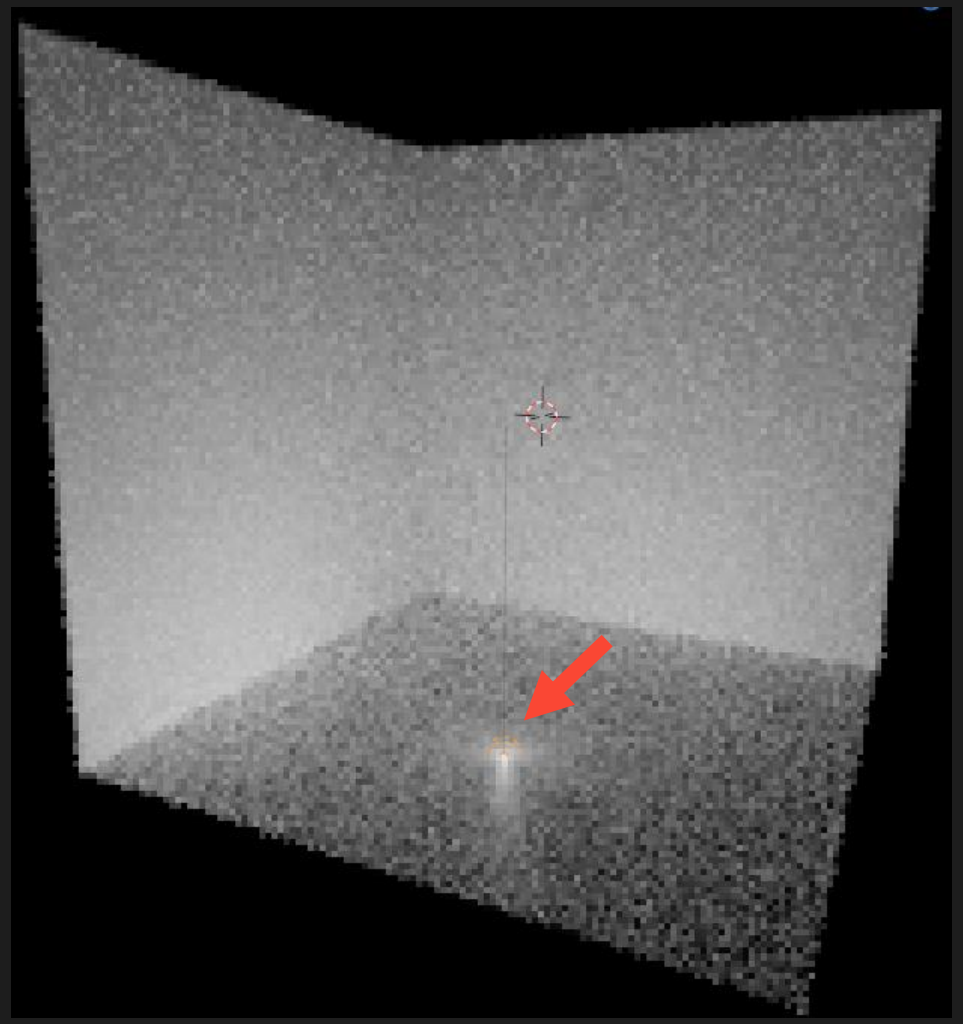
조명이 바닥에 가까이 있는 경우

그 외

Light Falloff 의 Smooth를 사용하면 조명이 가까울 때의 밝기를 낮춰

이와 같은 현상을 줄일 수 있습니다.





그 외

타일사이즈는 한번에 렌더링하는 화면의 크기를 지정합니다.
이전에는 타일 사이즈를 조절하여 렌더 시간을 줄이기도 했**었**습니다.
대체로 CPU는 타일 사이즈가 작을 때, GPU는 타일 사이즈가 클 때 빠르다고 알려져 있**었**습니다.

하지만 3.0 버전 이상에서는, 타일 사이즈를 조절하여 얻는 속도 변화가 거의 없습니다.
보통은 기본값에서 바꿀 일이 없으며, 메모리 문제가 있을 때만 작게 조절합니다.