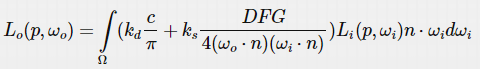
IBL / Specular IBL

-이전 장에서 반사 방정식의 indirect diffuse 부(kd)분인 irradiance map을 만들었음

-이번에는 specular 부분(ks)을 다룰 예정

-반사는 적분에 대해서 일정하지 않고, 들어오는 빛의 방향과 바라보는 방향에 의존함을 알 수 있음

-두 방향에 대해서 모두 적분하는 방법은 과부하가 커지고, 실시간으로 계산하기에는 너무 무거운 작업임

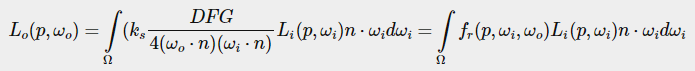
-Epic games에서 Split Sum Approximation(분할 합계 근사치)이라고 알려진 몇 가지 방법으로 특정 부분을 실시간으로 convolute할 수 있는 방법을 제안하였음

-분할 합계 근사치는 반사 방정식의 특정 부분을 2개의 부분으로 분할하여 나눔

-이후 PBR shader에 specular indirect image based lighting을 합쳐서 개별적으로

convolution함

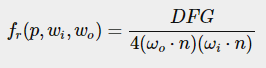
-irradiance map을 미리 계산한 것과 비슷한 방법이고, convolution 입력으로 HDR 환경 맵이 필요로 함

-성능 때문에 실시간으로 specular 부분을 계산하기가 어렵기 때문에 미리 계산해서 Specular IBL map을 얻어서 fragement의 normal로 샘플링을 수행할 것임

-irradianced는 ωi만 영향을 받았기 때문에 쉬웠고, albedo항을 적분에서 빼내서 계산하였기 때문에 쉬웠음

-이번 specular항은 ωi뿐 아니라 다른 것에도 영향을 받음

-BRDF

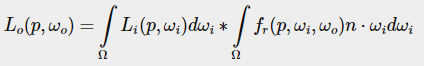
-ωi뿐 아니라 ωo에도 영향을 받는 모습을 볼 수 있음

-두 방향 벡터를 사용해서 미리 계산되어진 큐브 맵을 샘플링 할 수 없음, 두 벡터에 대해 조합이 가능한 모든 내용을 미리 계산하는 것은 실용적이지 않음

-여기서 p는 이전 장에서 설명한 것과는 관련이 없는 다른 점임

-분할 합계 근사치를 사용해서 미리 계산할 부분을 2개의 파트로 나눠서 해결하였음

-나눠진 specular 적분 부분

-첫 번째 부분(convoluted)은 사전 필터링된 환경 맵이고, irradiance map과 유사하게 사전 계산되어진 환경 convolution 맵이지만, roughness가 포함되어 있음

-거칠기 레벨을 높이기 위해서 더 많은 샘플 벡터로 복잡하게 해서 흐릿한 반사를 만듬

-순차적으로 흐릿한 결과를 mipmap에 저장

-normal과 view 방향을 입력으로 받는 Cook-Torrance BRDF의 NDF를 사용해서 샘플 벡터와 그 산란정도를 생성함

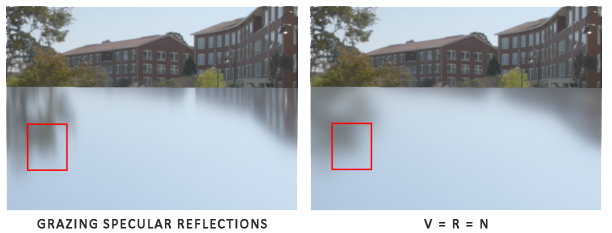
-환경 맵을 만들 때(convoluting할 때), view방향을 모르기 때문에 출력 샘플 방향 ωo와 같다고 가정하여 근사치를 만듬

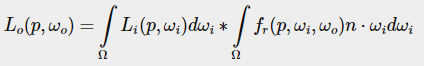
|  |
| --- |
| vec3 N = normalize(w\_o);  vec3 R = N;  vec3 V = R; |

-이 방법은 미리 필터링 되어진 환경맵에서 view방향을 인식할 필요가 없음

-그러나 specular surface reflection(정계 표면 반사)에서 좋은 grazing specular reflection

(방목 반사)을 얻을 수 없음



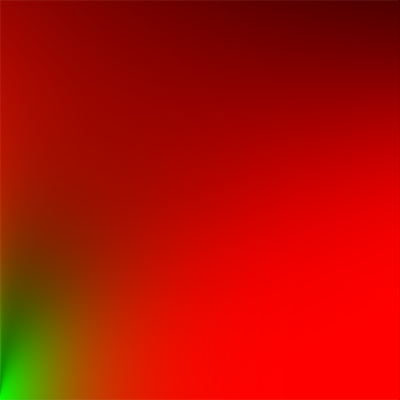
-방정식의 두 번째 부분은 BRDF의 Specular 부분과 동일함

-들어오는 빛이 완전히 흰색이라고 가정(L(p,x)=1)하면 normal n이랑 빛의 방향 ωi또는

n⋅ωi사이각과 거칠기 값이 주어지면 BRDF의 값을 계산할 수 있음

-Epic games는 2D lookup texture라는 다양한 조도 값에 대한 각 normal 및 light 방향 조합에 대한 사전 계산된 값인 BRDF 통합 맵을 저장함

-2D lookup texture은 scale(빨강), bias value(초록)을 출력해서 적분의 두 번째 부분을 제공함

-수평 텍스처 좌표를 BRDF의 input n⋅ωi

-수직 텍스처 좌표를 입력 조도 값으로 처리

|  |
| --- |
| float lod = getMipLevelFromRoughness(roughness);  vec3 prefilteredColor = textureCubeLod(PrefilteredEnvMap, refVec, lod);  vec2 envBRDF = texture2D(BRDFIntegrationMap, vec2(NdotV, roughness)).xy;  vec3 indirectSpecular = prefilteredColor \* (F \* envBRDF.x + envBRDF.y) |

Pre-Filtering an HDR environment map

-HDR 환경맵을 사전 필터링하는 것은 irradiance map을 convolut랑 비슷함

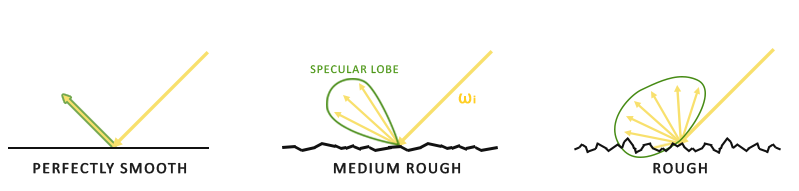
-다른 점은 거칠기가 있고, 사전 필터링된 map의 mip level에서 순차적으로 거칠기 반사를 저장함

-환경 맵 데이터를 저장하기 위한 큐브맵을 생성하고, 밉 레벨에 메모리를 할당하기 위해 필요한 양의 메모리를 glGenerateMipamp을 호출

|  |
| --- |
| unsigned int prefilterMap;  glGenTextures(1, &prefilterMap);  glBindTexture(GL\_TEXTURE\_CUBE\_MAP, prefilterMap);  for (unsigned int i = 0; i < 6; ++i)  {  glTexImage2D(GL\_TEXTURE\_CUBE\_MAP\_POSITIVE\_X + i, 0, GL\_RGB16F,  128, 128, 0, GL\_RGB, GL\_FLOAT, nullptr);  }  glTexParameteri(GL\_TEXTURE\_CUBE\_MAP, GL\_TEXTURE\_WRAP\_S, GL\_CLAMP\_TO\_EDGE);  glTexParameteri(GL\_TEXTURE\_CUBE\_MAP, GL\_TEXTURE\_WRAP\_T, GL\_CLAMP\_TO\_EDGE);  glTexParameteri(GL\_TEXTURE\_CUBE\_MAP, GL\_TEXTURE\_WRAP\_R, GL\_CLAMP\_TO\_EDGE);  glTexParameteri(GL\_TEXTURE\_CUBE\_MAP, GL\_TEXTURE\_MIN\_FILTER,  GL\_LINEAR\_MIPMAP\_LINEAR);  glTexParameteri(GL\_TEXTURE\_CUBE\_MAP, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);  glGenerateMipmap(GL\_TEXTURE\_CUBE\_MAP); |

-밉맵을 매핑하기 위해서 MIN\_FILTER가 GL\_LINEAR\_MIPMAP\_LINEAR로 되어있는지 확인해 삼선형 필터링을 사용 가능하게 해야함

-표면의 거칠기에 기초하는 반사는 빛은 반사 벡터r에 대해 normal 벡터 n에 대해 가깝거나, 비슷하게 반사됨, 표면이 극도로 거칠지 않는 이상 반사 벡터 주위에서 반사가 되어짐

-반사되어진 빛의 일반적인 모양은 specular lobe(반사 로브)라고 불림

-거칠기가 증가하면 반사 로브의 크기가 커짐

-들어오는 빛의 방향에 따라 반사 로브의 모양이 전함

-반사 로브는 material에 크게 의존함

-마이크로 표면 모델에 관해서 들어오는 빛의 방향이 주어진다면 미세면 halfway 벡터에 대한 반사 방향으로 반사 로브를 상상할 수 있는데, 대부분이 halfway 벡터가 반사된 반사 로브에서 끝나는 것이 나타나기에 비슷한 방식으로 샘플 벡터를 생성하는 것이 가장 효과적임

Monte Carlo integration and importance sampling

-중요도 샘플링을 이해하기 위해선 몬테카를로 통합을 먼저 알아야 함

-통계와 확률의 조합으로 이루어져 있음

-이것은 모든 정보를 고려하지 않고도, 집합의 어떤 통계치나 값을 알아내는 문제를 해결하는데 도움이 됨

-ex) 한 나라의 모든 시민의 평균 신장 구하기

-모든 시민들의 키를 측정해서 평균을 구하는 것이 가장 정확한 답이지만, 현실적으로 매우 많은 수의 인구를 하나씩 재는것은 문제가 있음

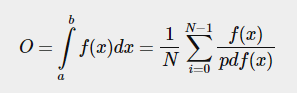
-다른 접근법으로 전체 인구수보다 훨씬 적은 완전한 랜덤의 부분 집합을 선택해서 키를 측정, 평균화 하는 방법이 있음

-이는 완벽하게 정확한 답은 아니지만, 답과 근사치의 값을 만들어냄

-부분 집합의 크기 N이 클수록 실제 값과 비슷해짐

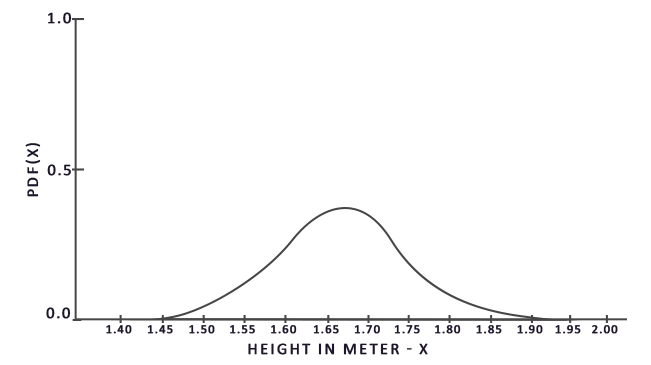
-몬테카를로 통합은 이 방법과 동일한 접근법을 취함

-샘플 가능한 모든 x에 대한 적분을 푸는 것이 아닌, 무작위로 추출한 N개의 샘플 값을 이용함

-a에서 b까지의 n개의 무작위 샘플을 취해서 합산, 평균을 내 값을 만듬

-여기서 pdf는 특정 샘플이 전체 샘플 set에서 발생할 확률을 알려주는 확률 밀도 함수임

-ex)키에 대한 확률 밀도 함수



-몬테카를로 통합과 관련해서 일부 샘플은 다른 샘플보다 뽑힐 확률이 높을 수 있음

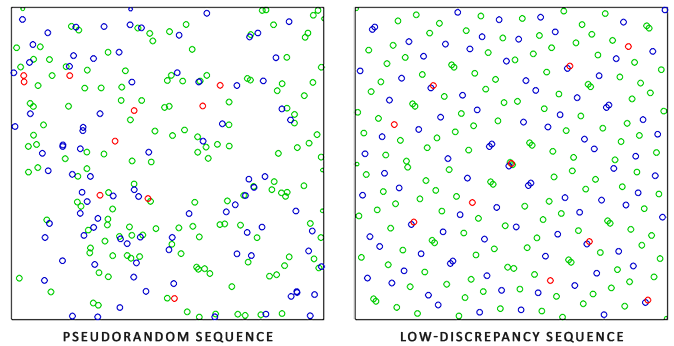
-샘플 값을 pdf에 따라서 샘플 확률로 나누거나 곱함

-일부 몬테카를로 추정량은 편향되어 있어서 생성된 샘플이 완전히 무작위가 아니지만, 특정 값이으로 수렴이 됨

-정확한 값이 나오지는 않지만, 훨씬 더 빠른 속도로 나오는 편향된 특성으로 인해서 정확한 해결책이 중요하지 않은 분야에서 사용이 되고, 컴퓨터 그래픽스도 그 분야중 하나임

-기본적으로 샘플은 이전처럼 완전히 무작위지만 semi-random sequence의 특정 속성을 사용해 여전히 임의이지만 흥미로운 속성을 가지는 샘플 벡터를 생성할 수 있음

-예를 들어서 무작위 샘플을 생성하는 low-discrepancy sequences라고 불리는 것으로 몬테카를로 통합을 사용하면 각 샘플은 보다 균등하게 분산되어짐

-왼쪽이 완전 랜덤

-low-discrepancy sequences를 사용하는 몬테카를로를 Quasi-Monte-Carlo통합 이라고 함

-Quasi-Monte-Carlo 방법은 빠른 속도, 수렴하는 정확도가 높아서 성능이 중요한 응용 프로그램에 사용함

-빛의 반사에 관해서는 반사광 벡터가 표면의 거칠기에 의해서 결정되는 크기를 가지는 반사 로브에 대해서 구속됨

-반사 로브 외부의 임의로 생성된 샘플이 specular integral과 관련이 없기 때문에, 몬테카를로 결합으로 편향되어진 샘플 생성으로 반사 로브에 집중시키는 것이 알맞음

-이 내용이 중요도 샘플링에 대한 본질적인 내용임

-microfacet의 halfway 벡터 주변의 거친 방향으로 구속되어진 일부 영역에서 샘플을 생성

-Quasi-Monte Carlo 샘플링과 Low-Discrepancy Sequence를 결합

-중요도 샘플링을 사용해서 샘플 벡터를 바이어싱해서 높은 속도를 얻음

-따라서 충분한 근사치에 도달하기 위해 적은수의 샘플이 필요하게 됨

-이 방법을 사용해서 응용 프로그램이 화면에 결과를 나타내는데 사전 계산하는 것 보다는 느려지긴 하지만 충분히 빠르게 실시간으로 반사 적분을 해결할 수 있음