GOD RAY

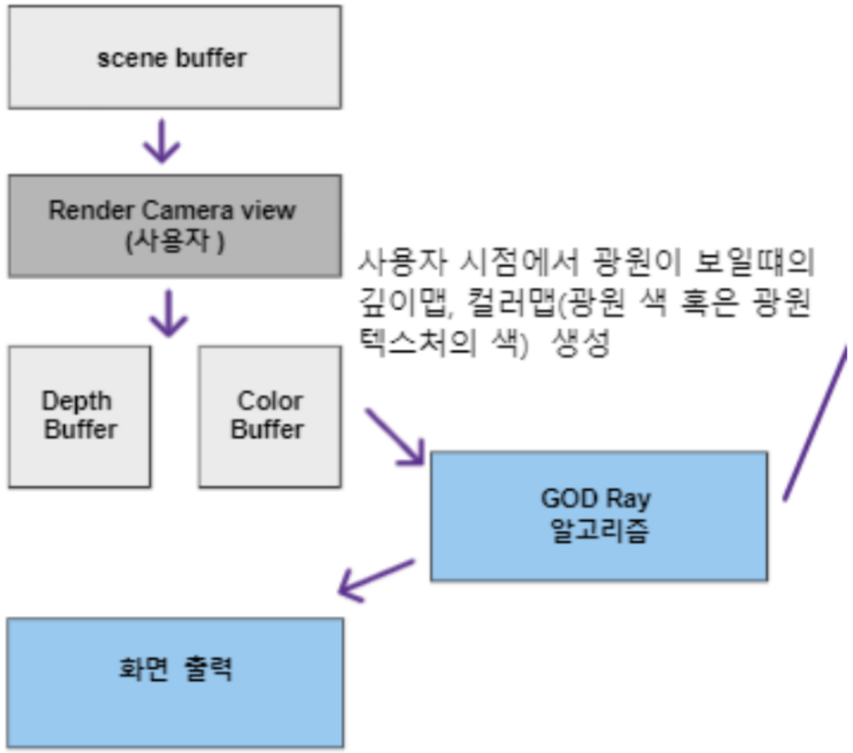
Shader Algorithm

추가적인 세부사항 설명

최유진

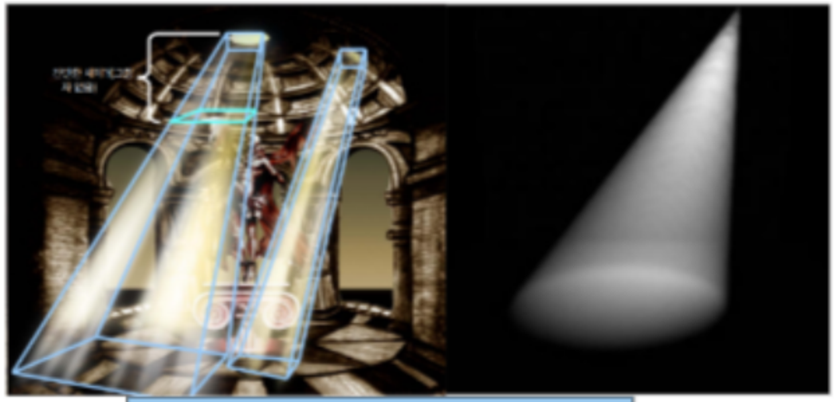
2023.02.15

전체과정 간략화



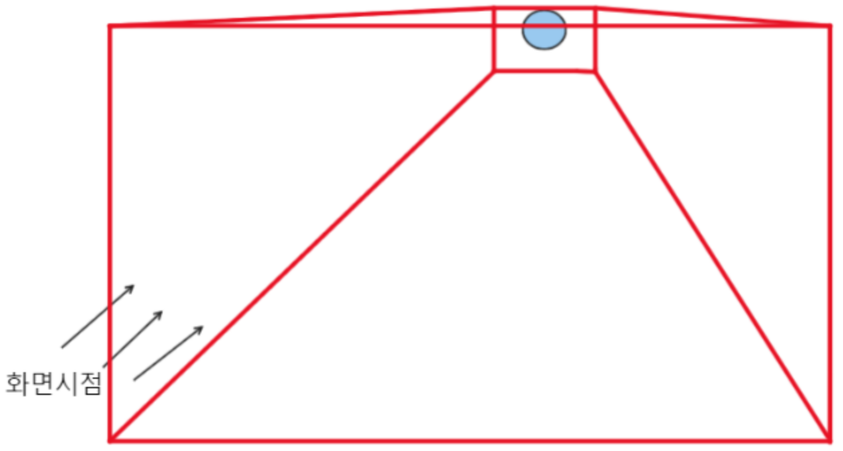
1. 사용자 시점 에서 보는 장면을 FrameBuffer에 낮은 해상도로 가려질 객체들을 랜더링 &깊이 버퍼 설정
2. 정상적으로 원래 화면을 랜더링한 FrameBuffer
3. 깊이 버퍼로 가려질 부분을 구분하여 빛의 퍼짐을 1번째 버퍼에 그린다.
4. 직교투영으로 두 프레임 버퍼를 혼합하여 그린다.

Frustrum설정



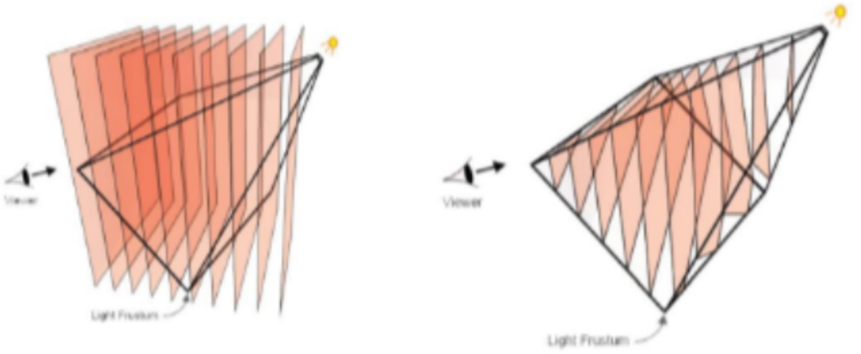
광원을 기점으로 빛이 비칠 원뿔(스펙트럼) 형태의 범위 지정(GULT 기준, 수정필요)

광원



광원을 중심으로 빛의 범위를(Frustrum)설정한다

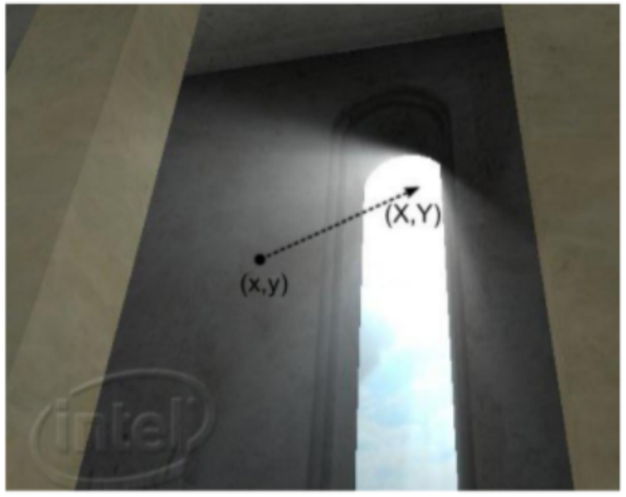
카메라를 광원의 위치에 설정하고, Frustrum의 방향과 위치가 같도록 할 수 있게 한다.



광원 - 카메라 사이를 일정간격으로 나누 샘플링 후 원뿔 내부에 있는지 체크한다.

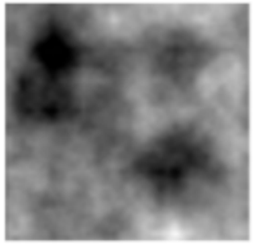
이 범위를 N개의 사각형(샘플)으로 나누고, Frustrum모양으로 체크, 및 설정하여 범위를 조정한다.

1. 같은 크기의 사각형N개를 그린 후 잘라 조정한다.
2. aspect을 이용해 각 사각형 N개의 크기를 각각 조정한다.

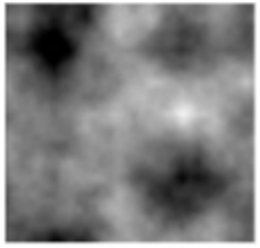


광원의 시작을 (X,Y)
빛의 끝(범위)를 (x,y)라 할때

유사한 두 가지 맵을 섞어쓰면 더 화려한 효과



소스 이미지(노이즈 맵1)



소스 이미지(노이즈 맵2)

(x0,y0) = (x,y) 및 (xN,yN) = (X,Y)

입력 이미지 공간의 세그먼트에 고르게 분포된 N개의 샘플 포인트 (xi,yi)

N개로 클리핑한 면에서**각 샘플 포인트에 대해 소스 이미지 픽셀의 HDR 값을 가져와 가중치 및 감쇠 계수 와 합산!**

$$GodRaysColor_i = weight * \sum_{j=i}^{0} PixelColor(x_j, y_j) * decay^{i-j}$$

weight = 샘플에 대한 강도, decay =[0,1]범위에 따른 샘플의 영향도(거리?)

이 때 또 하나 고려해야 할 점이 있는데 density 라는 추가 변수로 샘플링 할 수의 스케일 값을 지정한다
만약 density 값을 키우면 샘플 사이의 간격을 줄여 더 짧은 범위에 밝은 빛줄기를 만든다

감쇠 계산 - (decay)
광선을 따라 있는 모든 점(샘플링으로 나뉜)에 대해 픽셀과 광원 사이의 거리를 계산합니다. 해당 값을 사용하여 감쇠 계수를 계산합니다.
이 단계는 "번지" 효과를 생성

가중치 계산
산란 - 산란매체의 차폐여부
-일광 산란 분석 모델
(Hoffman and Preetham 2003)

Brezenham의 라인 알고리즘
http://www.cs.helsinki.fi/group/goa/mallinnus/lines/bresenh.html 및
http://www.xlinux.nist.gov/dads/HTML/bresenham.html 을 참조

알고리즘 반복횟수 : Height*width*N
N은 광선을 따른 클리핑 샘플 수
Height 와 Width는 이미지 높이, 너비(픽셀단위)
N에 대해 최대값을 설정할수록 부드러운 결과물을 얻는다.

가장 좋은 변형은 N = max(높이, 너비)
알고리즘의 계산 복잡도는 O(M3)이며 여기서 M = max(높이, 너비)

$$GodRaysColor_i = weight * \left(PixelColor(x_i, y_i) + decay * \sum_{j=i-1}^{0} PixelColor(x_j, y_j) * decay^{i-j-1} \right)$$

$$GodRaysColor_i = weight * PixelColor(x_i, y_i) + decay * GodRaysColor_{i-1}$$

$$GodRaysColor_0 = weight * PixelColor(x_0, y_0)$$

픽셀 (xi-1,yi-1) 에 대해 계산된 합계를 사용 하여 광선 [X,Y; xi,yi] 픽셀 바로 뒤 (xi-1,yi-1).

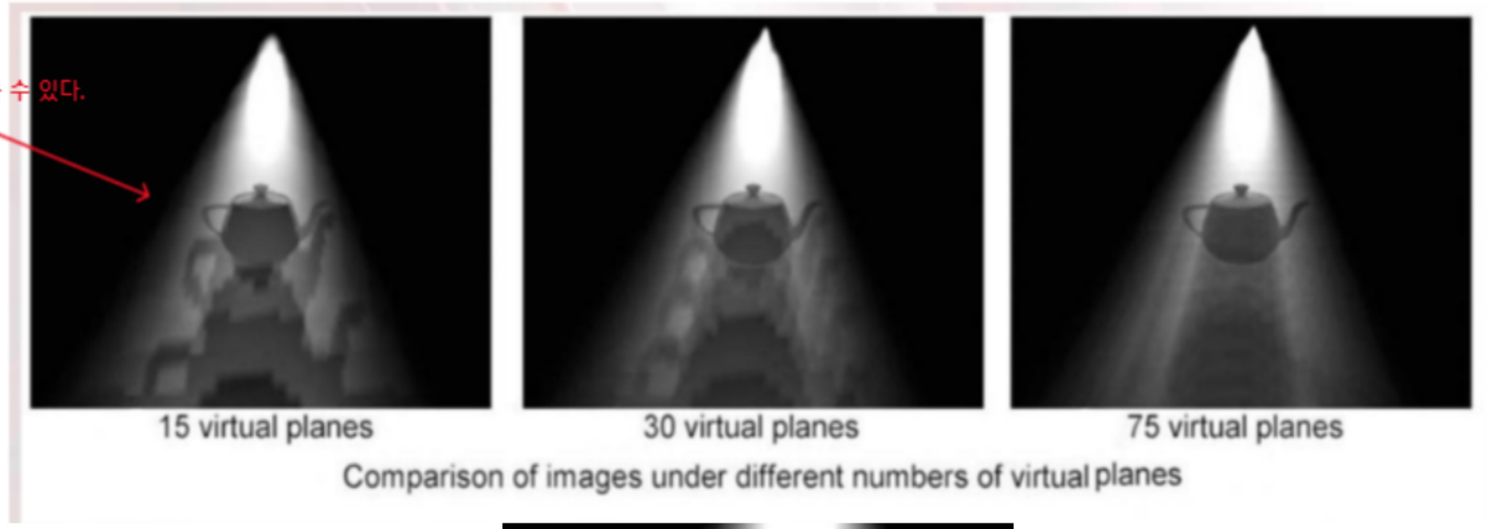
N개의 샘플에 따른 위치 계산과
가중치 + 감쇠계수 + 노이즈맵(텍스처) 를 계산한다.

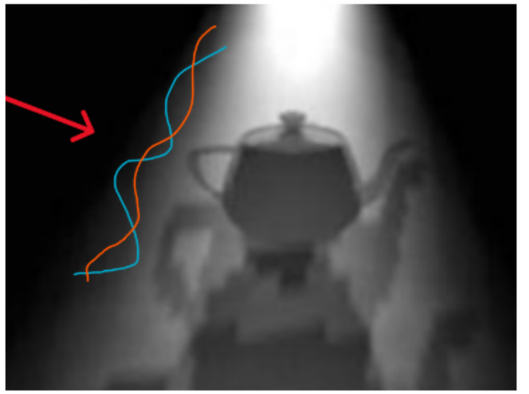
hlsl에서 텍스처를 입히는 작업에서 계산하도록 한다.

N개의 샘플 위치는 정해져있지만 사이사이의 공간속 픽셀들은 왼쪽 공식을 통해 위치를 정한다.
샘플과 샘플 좌표 사이 적분을 통한 곡선으로 이어 그린다.

.

아래는 샘플링 N의 개수에 따른 효과 모습이다





픽셀 사이를 적분처리되어 그려져서 적은 샘플의 경우 구불한 곡선이 보인다.

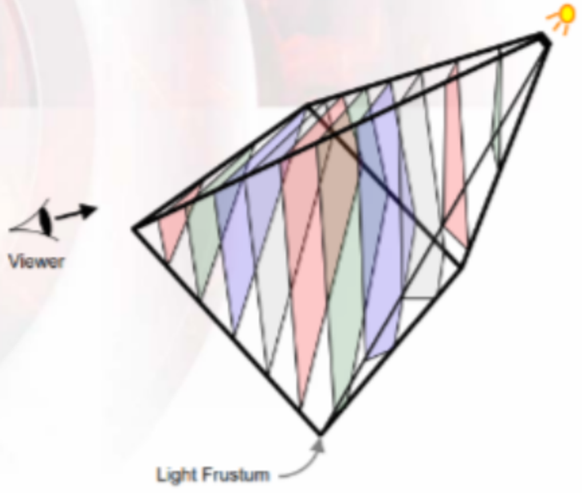
최적 샘플 개수를 찾아 최대한의 효과를 내도록 조정해야한다.

추가 방법

1. 정밀도 증가

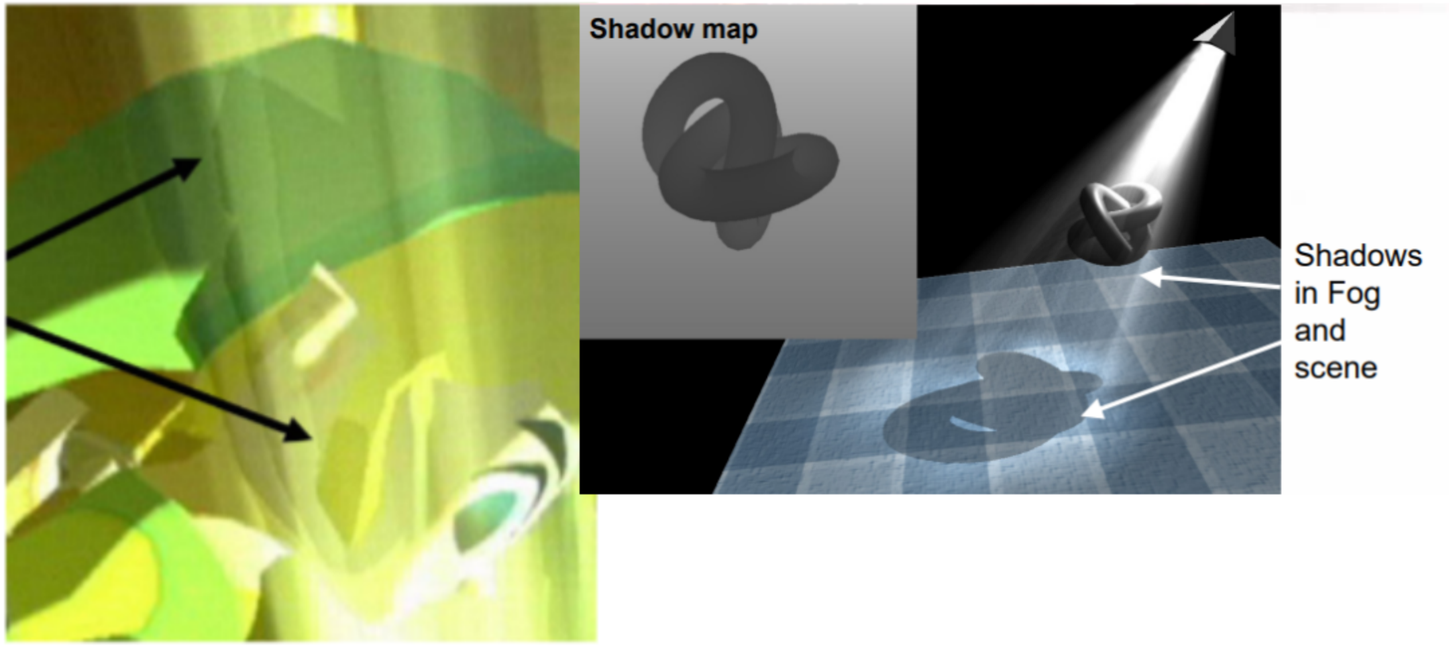
Increasing Destination Precision

- Draw every fourth plane into a different channel of an offscreen RGBA texture
- Clip to light frustum
- When subsequently compositing with back buffer, combine these four channels



GDC 2004 - *Light Shafts: Rendering Shadows in Participating Media*

2. 그림자 맵을 같이 랜더하여 선명도 증가 (동적일 경우 많은 비용소모.)



구체적인 구현 위치 계획

CShader에 구현하여 Scene 생성시 light위에 덮어씌운다.

God Ray 빛 분산 알고리즘 - hlsl 에 텍스처 처리과정에 작성
Frustrum 범위 설정 - GameObject와 다르므로 lightCamera와 함께 Frustrum.cpp, h 사용
깊이 검사 및 혼합 - CShader에서 Texture설정, 깊이검사 버퍼를 설정할 예정.

혼합과정에 대한 설명이 미흡하므로 이해한 뒤 보완하도록 해야한다.