Real-time rendering 4th

Chapter 5

개요

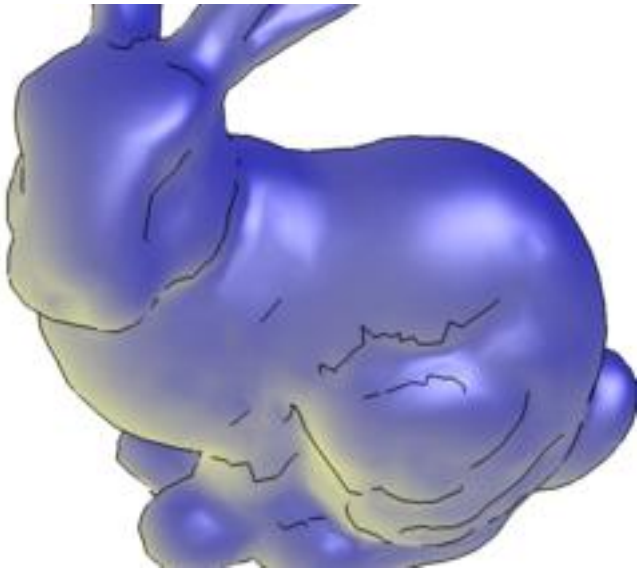
- Shading model
- Light sources
- Implementing shading models
- Aliasing and antialiasing
- Transparency, Alpha, and Compositing
- Display Encoding

Shading model

렌더링된 객체의 외형을 결정하는 1번째 단계

[non-photorealistic]

- Gooch shading (cool to warm)[basic]
- Cel shading

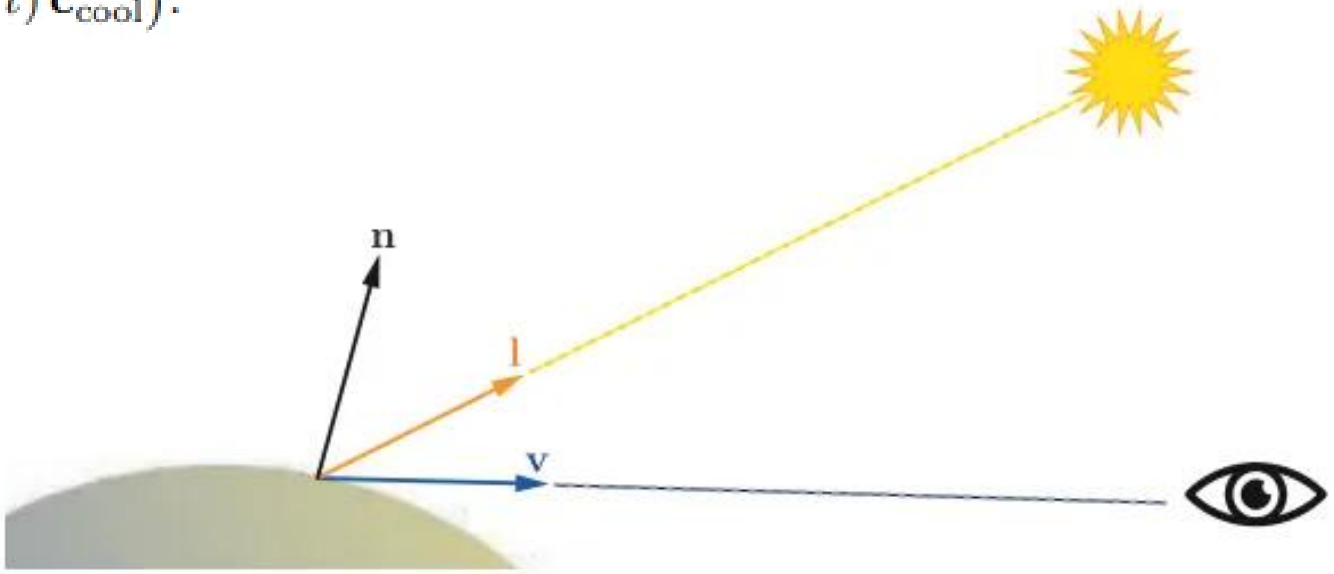


Gooch shading model

- 시점 방향 & 조명 방향이 표면의 지향점과 연관됨
- s : highlight blend factor
- t : c_warm, c_cool 사이의 interpolate factor

$$\mathbf{c}_{\text{shaded}} = s\,\mathbf{c}_{\text{highlight}} + (1-s)\left(t\,\mathbf{c}_{\text{warm}} + (1-t)\,\mathbf{c}_{\text{cool}}\right).$$

$$\begin{aligned}
\mathbf{c}_{\text{cool}} &= (0, 0, 0.55) + 0.25\,\mathbf{c}_{\text{surface}}, \\
\mathbf{c}_{\text{warm}} &= (0.3, 0.3, 0) + 0.25\,\mathbf{c}_{\text{surface}}, \\
\mathbf{c}_{\text{highlight}} &= (1, 1, 1), \\
t &= \frac{(\mathbf{n}\cdot\mathbf{l}) + 1}{2}, \\
\mathbf{r} &= 2\,(\mathbf{n}\cdot\mathbf{l})\mathbf{n} - \mathbf{l}, \\
s &= \left(100\,(\mathbf{r}\cdot\mathbf{v}) - 97\right)^{\overline{+}}.
\end{aligned}$$



Light Sources

- shading의 주요 방향을 제공하는 요소

- 조명의 복잡성은 빛의 존재 여부에 반응하는 정도까지만!
- 광원까지의 거리, 그림자 처리, 광원 & 법선의 관계

Light Sources

- 광원의 이진 상태에서 중간 단계는 선형 보간 (연속적 표현)

$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + k_{\text{light}} f_{\text{lit}}(\mathbf{l}, \mathbf{n}, \mathbf{v}).$$

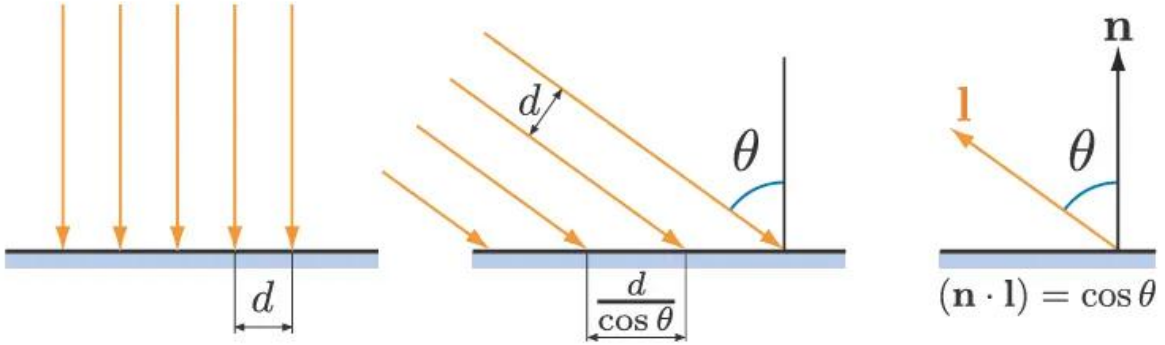
$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \mathbf{c}_{\text{light}} f_{\text{lit}}(\mathbf{l}, \mathbf{n}, \mathbf{v})$$

$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \sum_{i=1}^{n} \mathbf{c}_{\text{light}_i} f_{\text{lit}}(\mathbf{l}_i, \mathbf{n}, \mathbf{v})$$

Light Sources

- 광원의 각도 X

$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \sum_{i=1}^{n} \mathbf{c}_{\text{light}_i} f_{\text{lit}}(\mathbf{l}_i, \mathbf{n}, \mathbf{v})$$

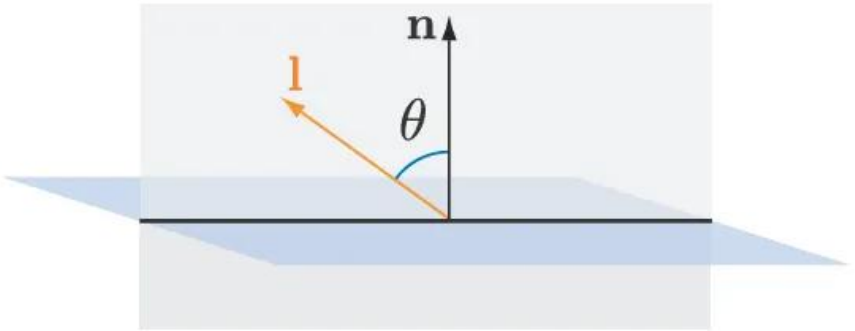


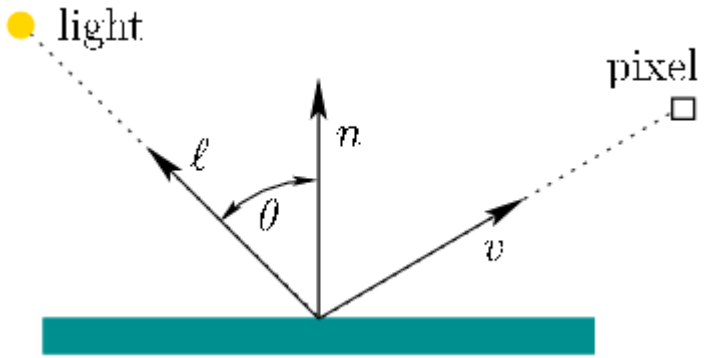
- 광원의 각도 O

$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \sum_{i=1}^{n} (\mathbf{l}_i \cdot \mathbf{n})^{+} \mathbf{c}_{\text{light}_i} f_{\text{lit}}(\mathbf{l}_i, \mathbf{n}, \mathbf{v})$$

$$f_{\text{lit}}() = \mathbf{c}_{\text{surface}}$$

$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \sum_{i=1}^{n} (\mathbf{l}_i \cdot \mathbf{n})^{+} \mathbf{c}_{\text{light}_i} \mathbf{c}_{\text{surface}}$$





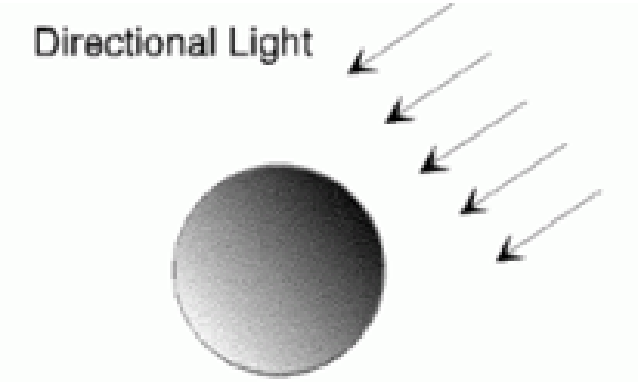
$$R = d_R I_R \max(0, n \cdot \ell)$$
$$G = d_G I_G \max(0, n \cdot \ell)$$
$$B = d_B I_B \max(0, n \cdot \ell)$$

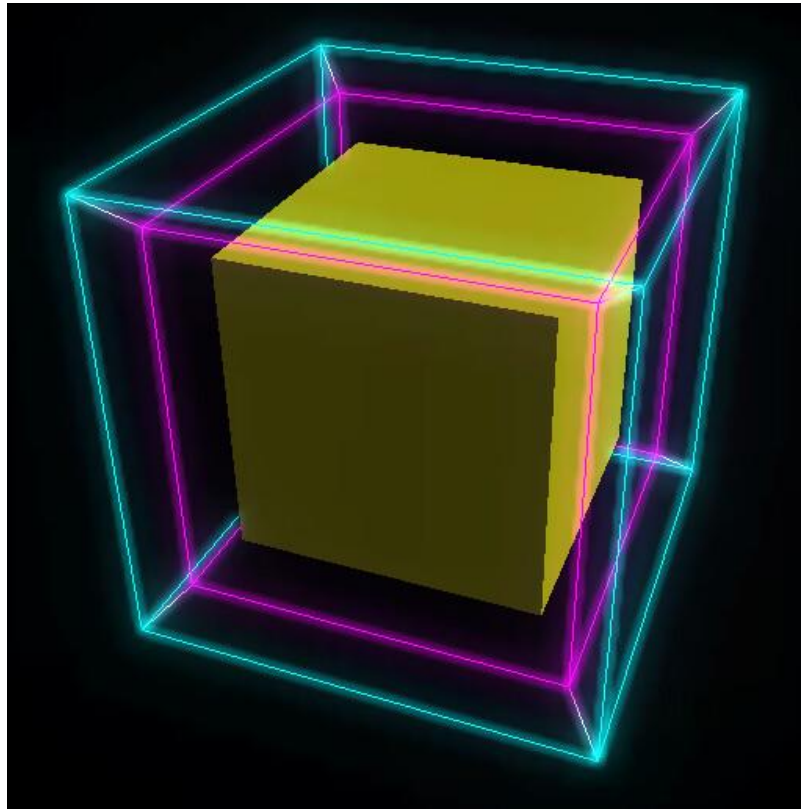
$$L = dI \max(0, n \cdot \ell)$$

Directional Lights

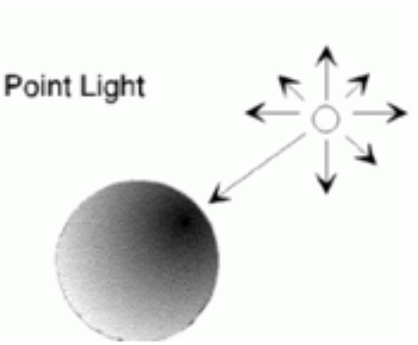
$$\mathbf{c}_{\text{shaded}} = f_{\text{unlit}}(\mathbf{n}, \mathbf{v}) + \sum_{i=1}^{n} (\mathbf{l}_i \cdot \mathbf{n})^+ \mathbf{c}_{\text{light}_i} \mathbf{c}_{\text{surface}}$$

- 광원의 가장 간단한 모델
- l, c_light가 상수화
- 특정한 위치가 없는 형태





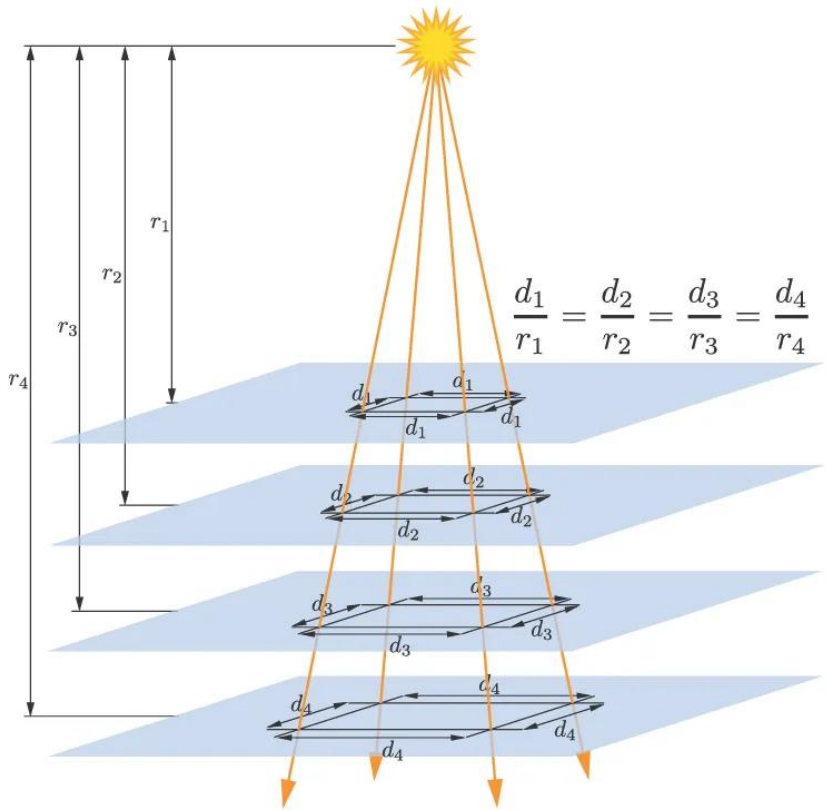
Punctual Lights (point light)



- 부피를 가지지 않는 조명 (point light, spot light)
- 특정한 위치를 가짐
- 균등하게 빛을 발사함

$$\mathbf{l} = \frac{\mathbf{p}_{\text{light}} - \mathbf{p}_0}{\|\mathbf{p}_{\text{light}} - \mathbf{p}_0\|}$$

$$\mathbf{c}_{\text{light}}(r) = \mathbf{c}_{\text{light}_0}\left(\frac{r_0}{r}\right)^2 \qquad \mathbf{c}_{\text{light}}(r) = \mathbf{c}_{\text{light}_0} f_{\text{dist}}(r)$$

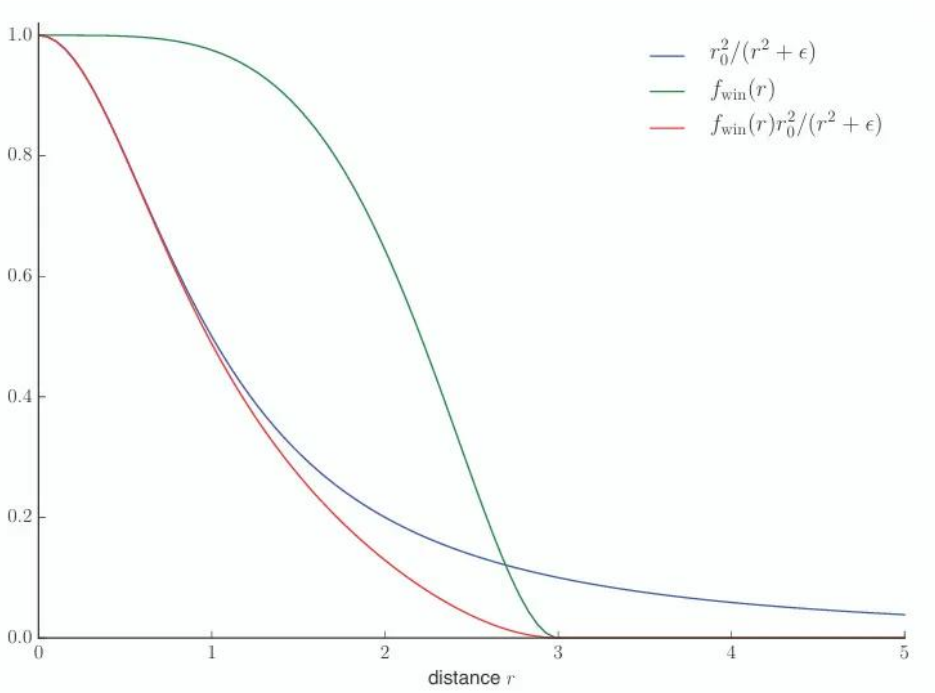


Punctual Lights (point light)

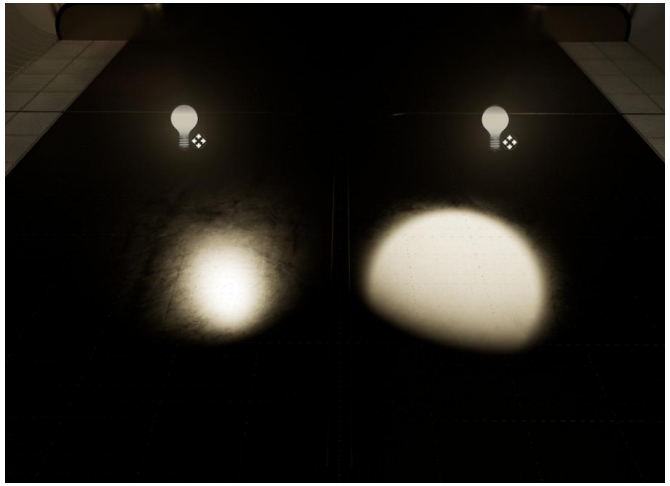
$$\mathbf{c}_{\text{light}}(r) = \mathbf{c}_{\text{light}_0}\left(\frac{r_0}{r}\right)^2$$

- 문제 1. r이 낮은 경우 $\mathbf{c}_{\text{light}}(r) = \mathbf{c}_{\text{light}_0}\frac{r_0^2}{r^2+\epsilon}$ $\mathbf{c}_{\text{light}}(r) = \mathbf{c}_{\text{light}_0}\left(\frac{r_0}{\max(r, r_{\text{min}})}\right)^2$

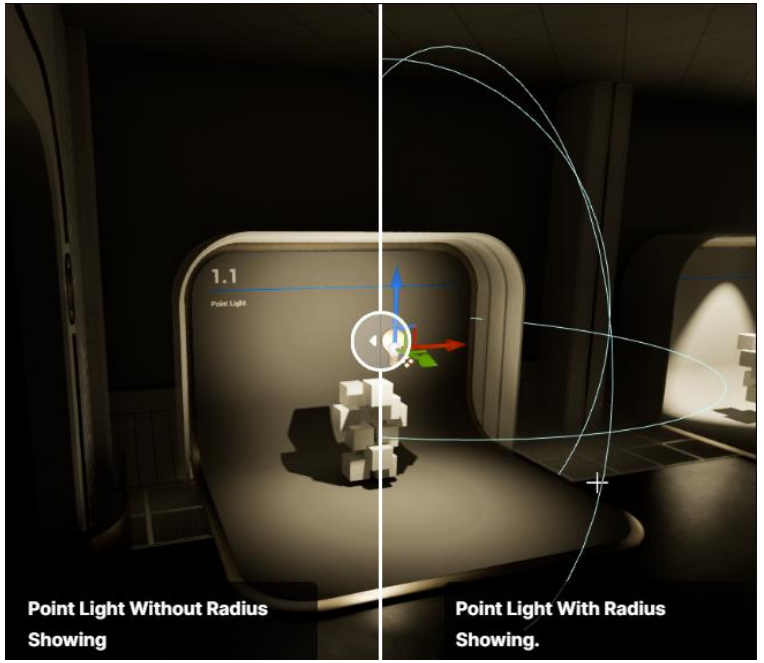
- 문제 2. r이 매우 큰 경우



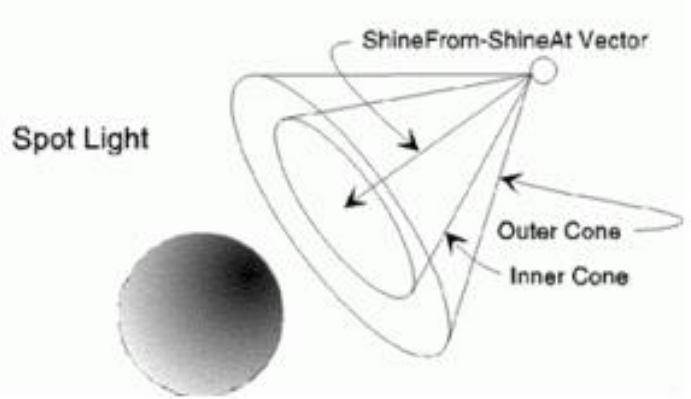
$$f_{\text{win}}(r) = \left(1 - \left(\frac{r}{r_{\text{max}}}\right)^4\right)^{+2}$$



역제곱 x

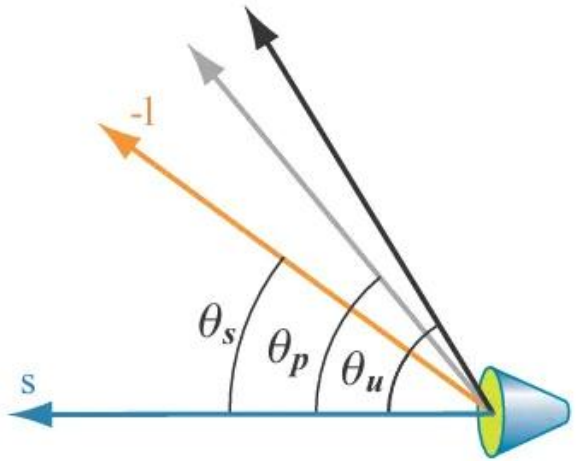


Punctual Lights (spot light)



- 광원의 방향도 중요해짐

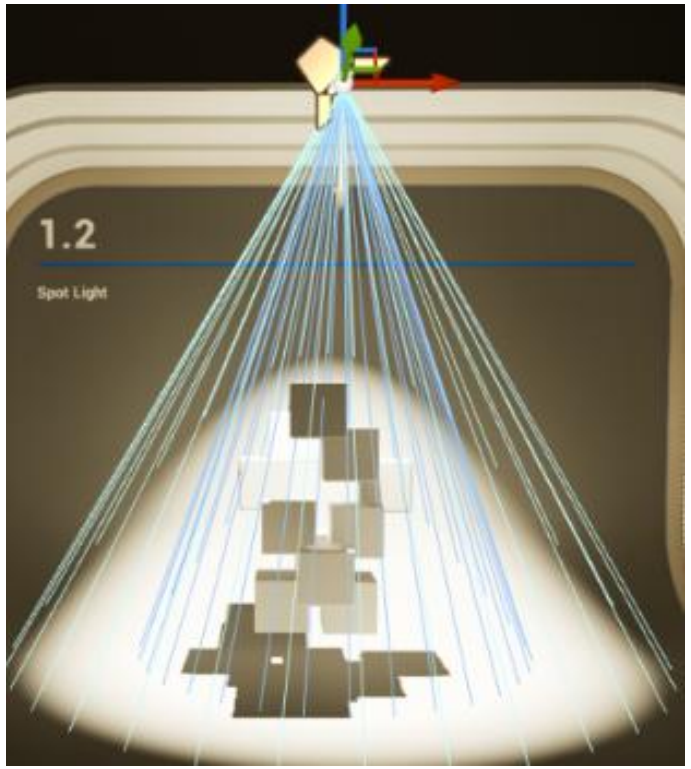
$$\mathbf{c}_{\text{light}} = \mathbf{c}_{\text{light}_0} f_{\text{dist}}(r) f_{\text{dir}}(\mathbf{l})$$



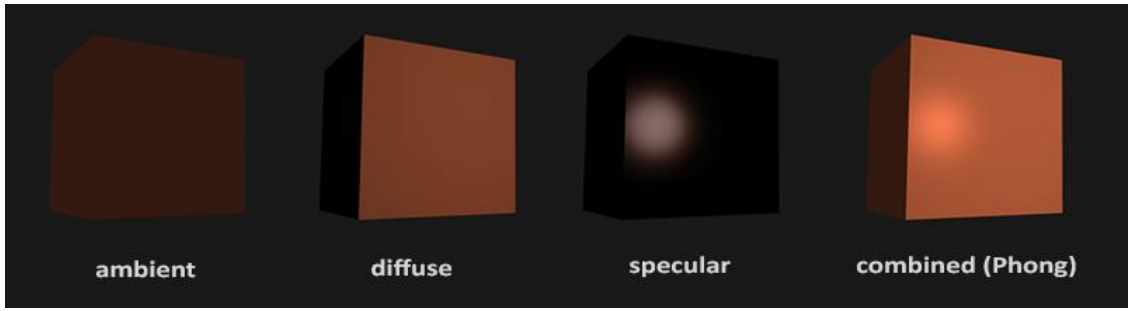
$$t = \left(\frac{\cos\theta_s - \cos\theta_u}{\cos\theta_p - \cos\theta_u} \right)^{\overline{+}},$$

$$f_{\text{dir}_F}(\mathbf{l}) = t^2,$$

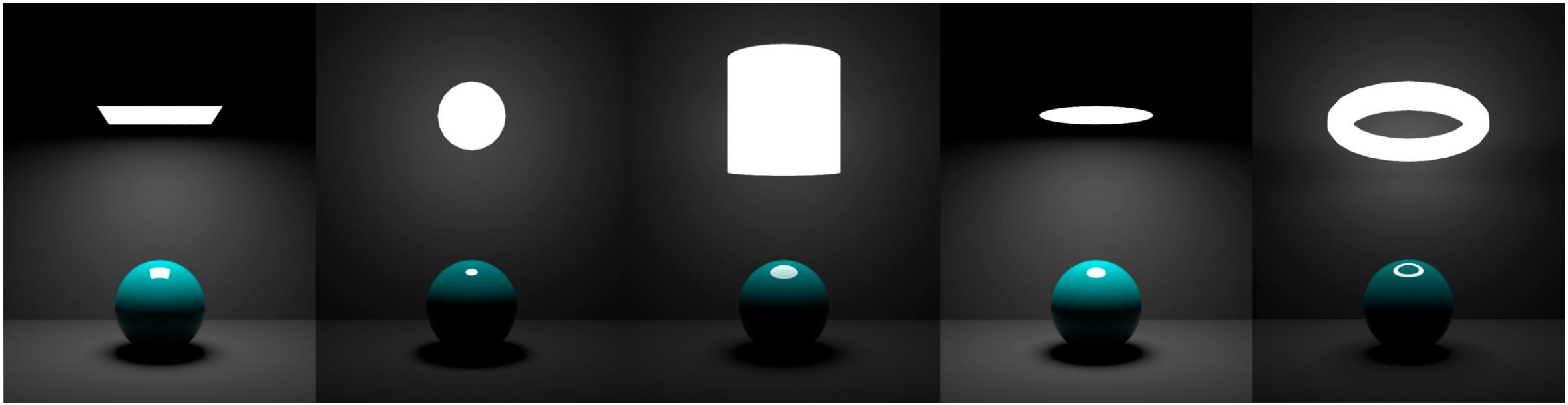
$$f_{\text{dir}_T}(\mathbf{l}) = \text{smoothstep}(t) = t^2(3 - 2t).$$



Other Lights



- Area light : 크기와 형태를 가지는 광원
- Ambient + diffuse + specular = phong



Implementing Shading Models

- Frequency of Evaluation
  - 모든 draw-call에서 일정한 결과

➔계산하는 시점 옮기기

➔여러 프레임에 걸쳐 변경하기

➔…

Frequency of Evaluation

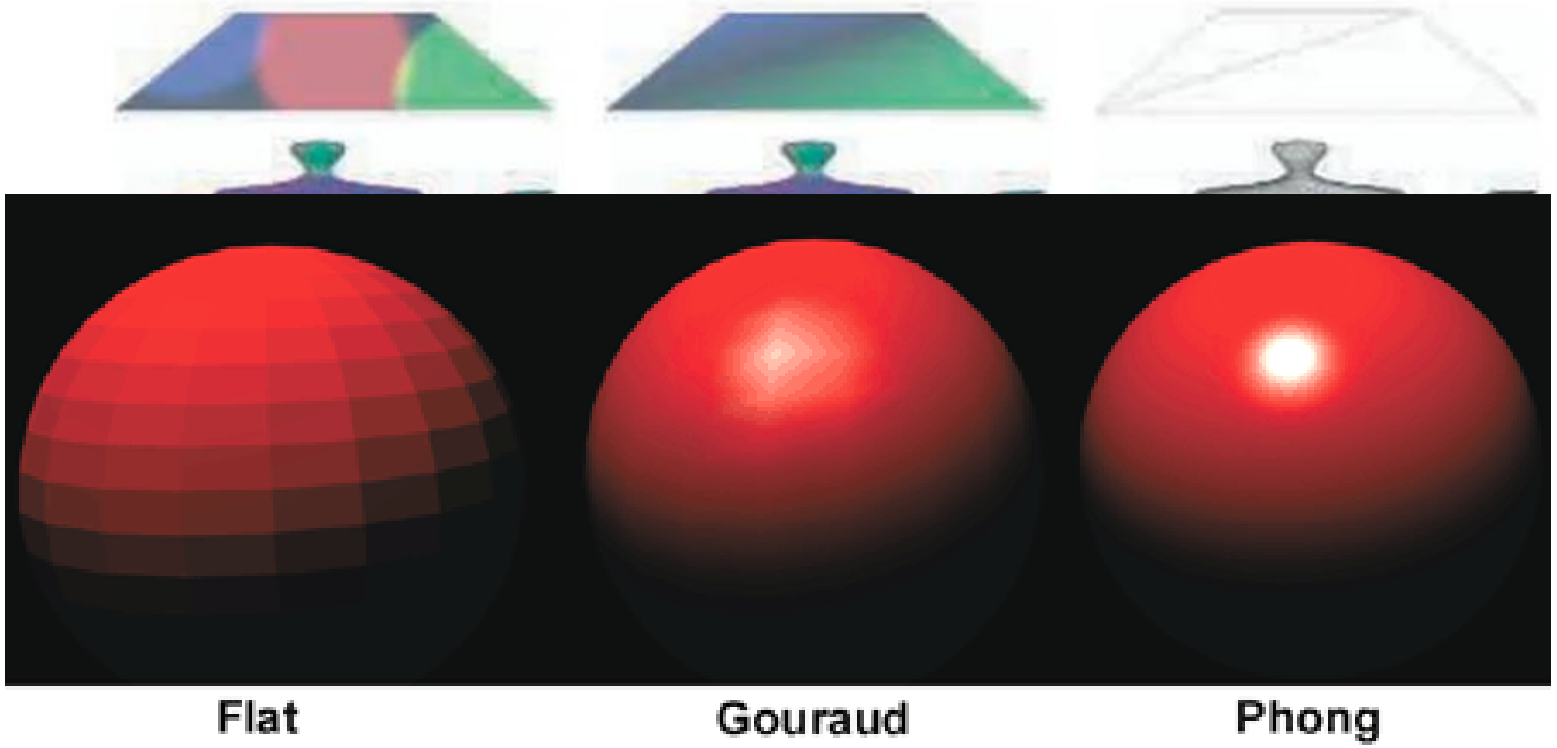
• 셰이딩 계산은 모든 programmable 단계에서 진행
평가 빈도는 모두 다름

- Vertex shader
- Hull shader
- Domain shader
- Geometry shader
- Pixel shader

Frequency of Evaluation

• 셰이딩 계산은 모든 programmable 단계에서 진행
평가 빈도는 모두 다름

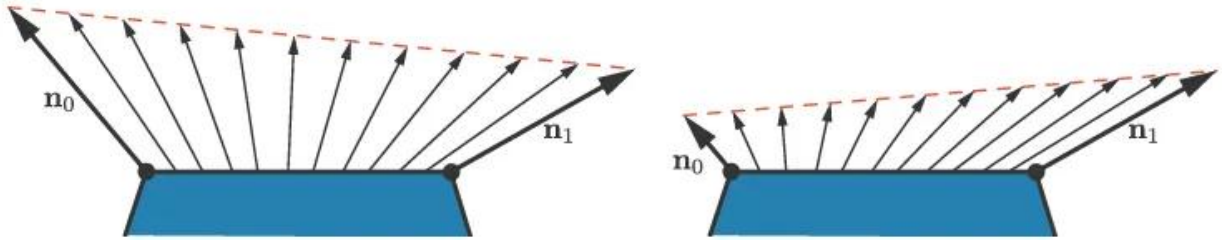
- Vertex shader
- Hull shader
- Domain shader
- Geometry shader
- Pixel shader



Frequency of Evaluation

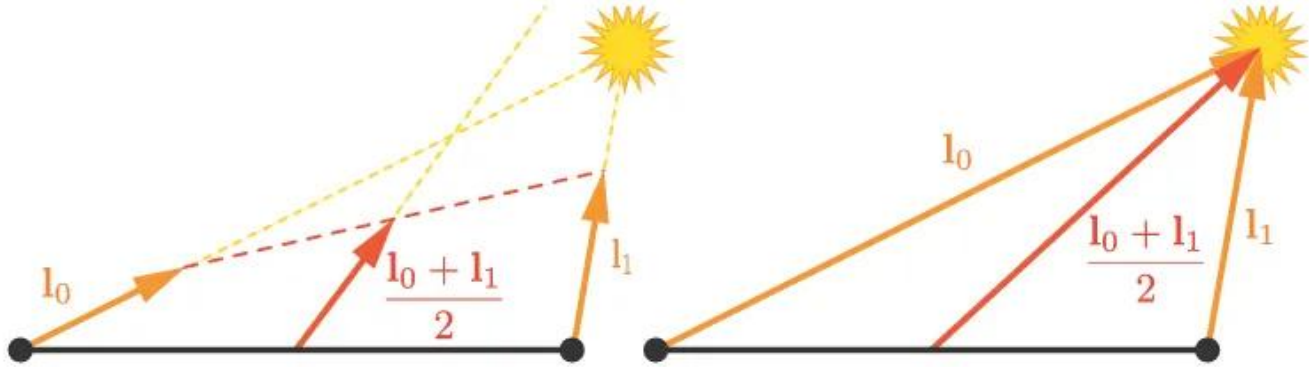
- 법선의 보간 과정

➔ Pixel shader에서 재정규화



- 벡터 보간 과정

➔ 정규화 이전에 보간



Material Systems

- Material : object의 표면 property
  - 표면의 외형을 쉽게 다루기 위한 고수준의 캡슐화
  - 색, 반사도, 거친 정도, 투명도 등
- Material template : 파라미터 집합
- Material instance : template + parameter의 값
- 계층 구조 형성 -> 상속 형식

- 파라미터 변경 시점
  - Shader program / compile time (Boolean switch)

Material Systems

Composition type

- Geometric processing
- Compositing operation (pixel discard, blending..)
- Reuse computing parameter
- Individually selectable material feature / logic
- Light source evaluation

- Shader code의 부분적인 모듈화 -> 소스 코드 수준에서 구현

Material Systems

- Code reuse
  - #include 활용
- Subtractive
  - ubershader / supershader : 주요 shader 1개만 사용

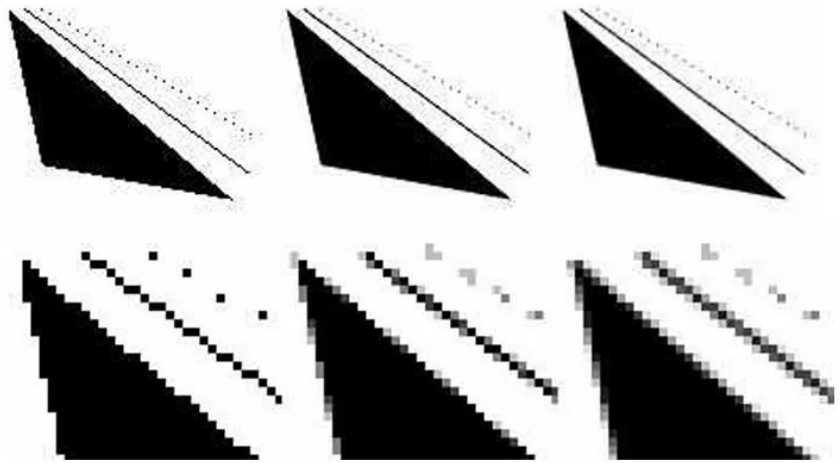
```
#ifdef USE_NORMAL_TEXTURE
Texture2D NormalTex : register(t1);
#endif
```

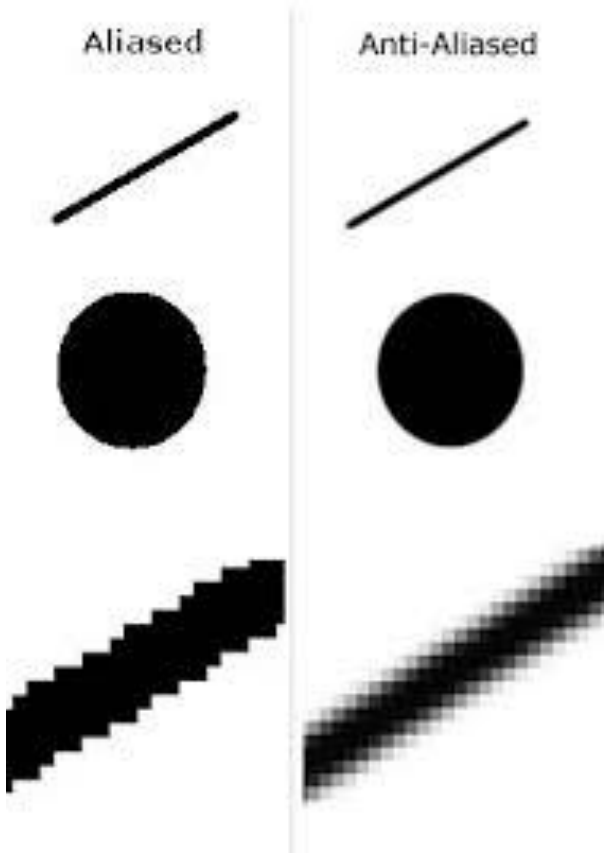
- Additive
  - 노드의 구성 및 추가를 구조화

Aliasing and Antialiasing

- Aliasing : 샘플링, 디지털화 과정에서 발생하는 왜곡
  - Jaggies / crawlies

➔ Antialiasing 기술이 필요
& 샘플링, 디지털화(필터링) 과정에서 중요

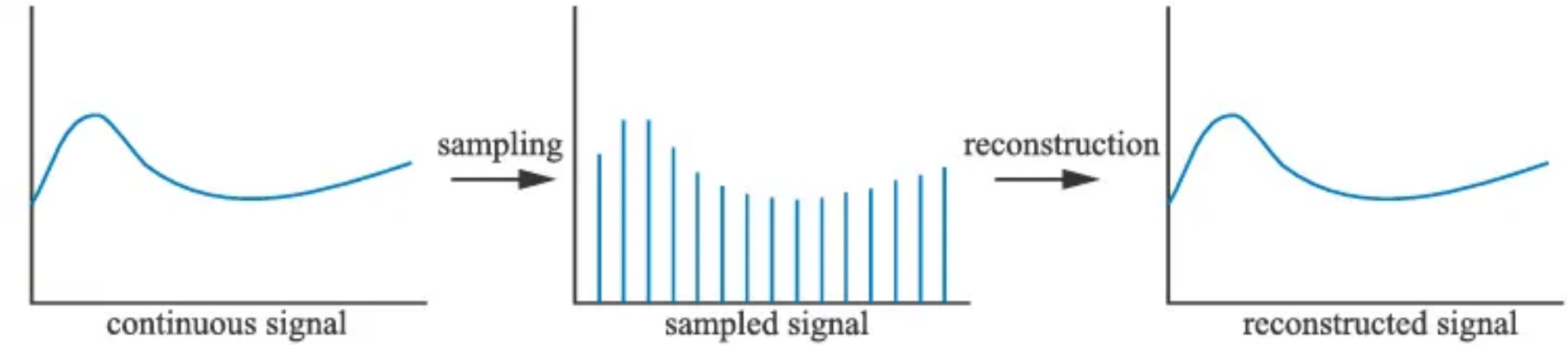




Sampling

- 연속적인 데이터에서 이산적인 데이터를 추출

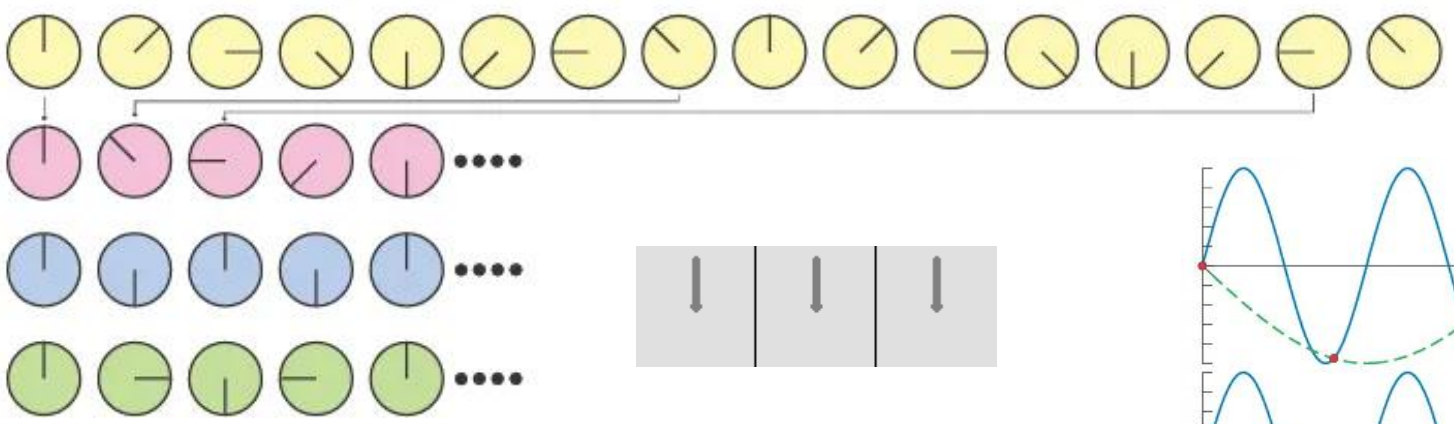
정보의 양 축소 -> 앨리어싱
재구축 -> 안티앨리어싱 필요

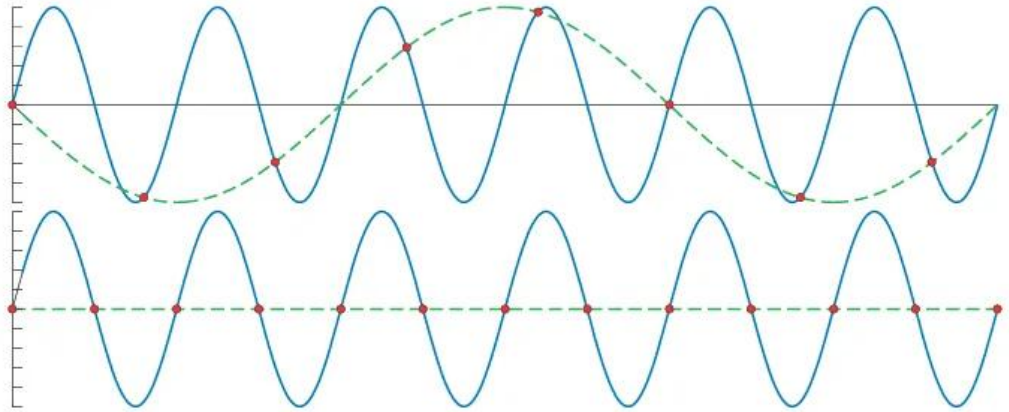


Sampling

- Temporal aliasing
  - 진동수가 감소한 형태로 나타남

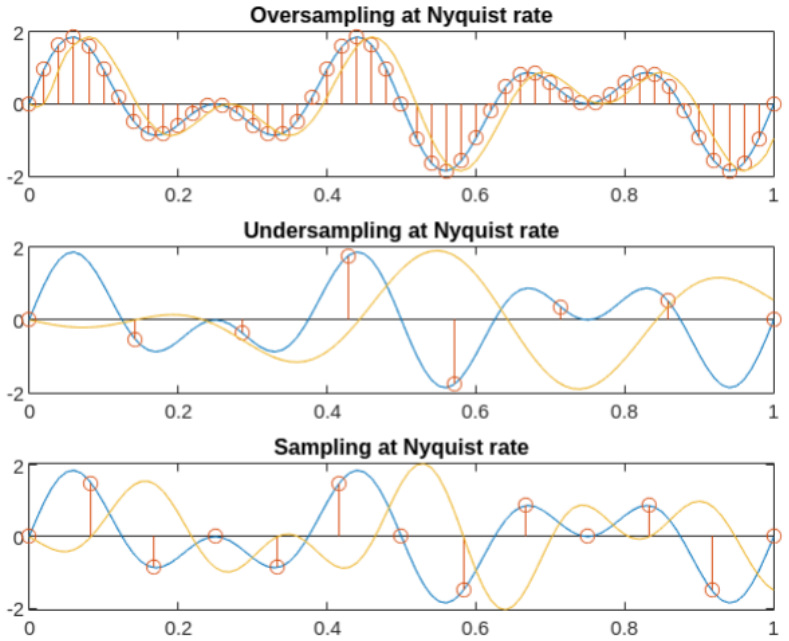
flickering / stroboscopic



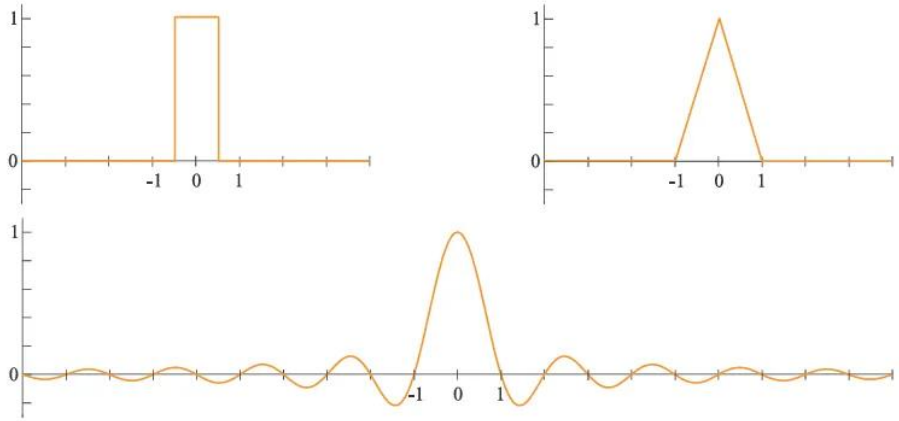


Sampling theorem

- Nyquist rate(limit) $f_s \geq 2f_c$
- 샘플링 frequency는 신호의 최대 freq보다 2배 이상
- Band-limit은 항상 존재하지만, 충분히 부드러워야 함
- 너무 높은 주파수 대역 또한 제한

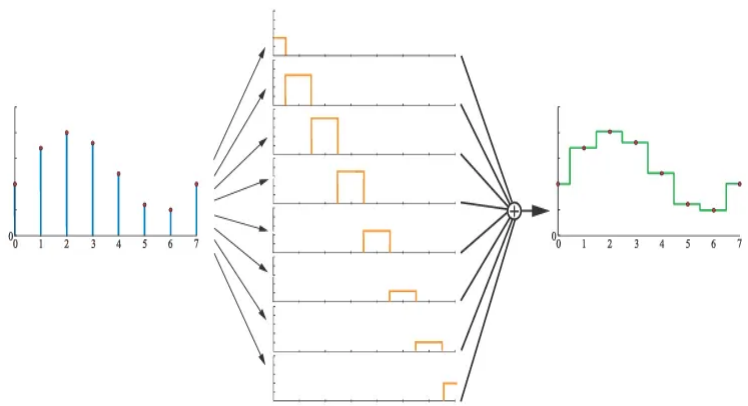


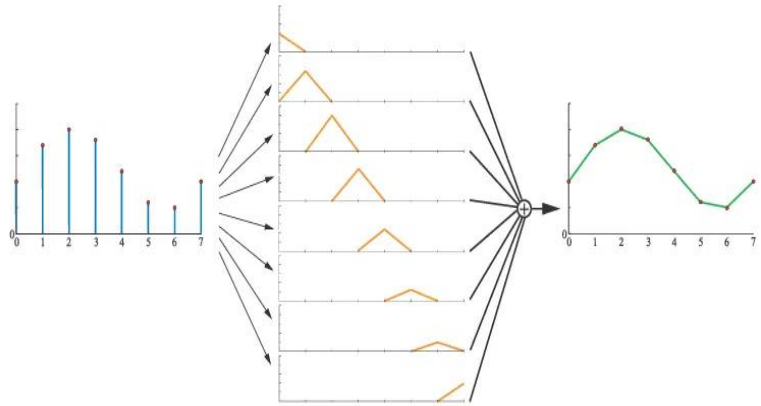
Reconstruction(Filtering)

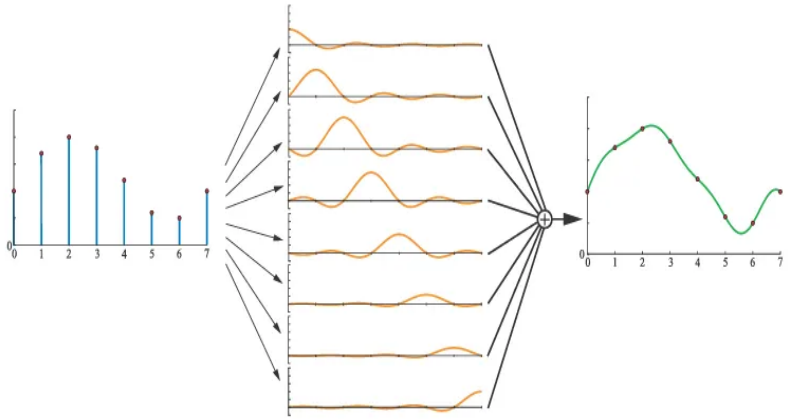


- Box filter
- Tent filter
- Sinc filter
  - 가장 이상적인 low-pass filter

$$\mathrm{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

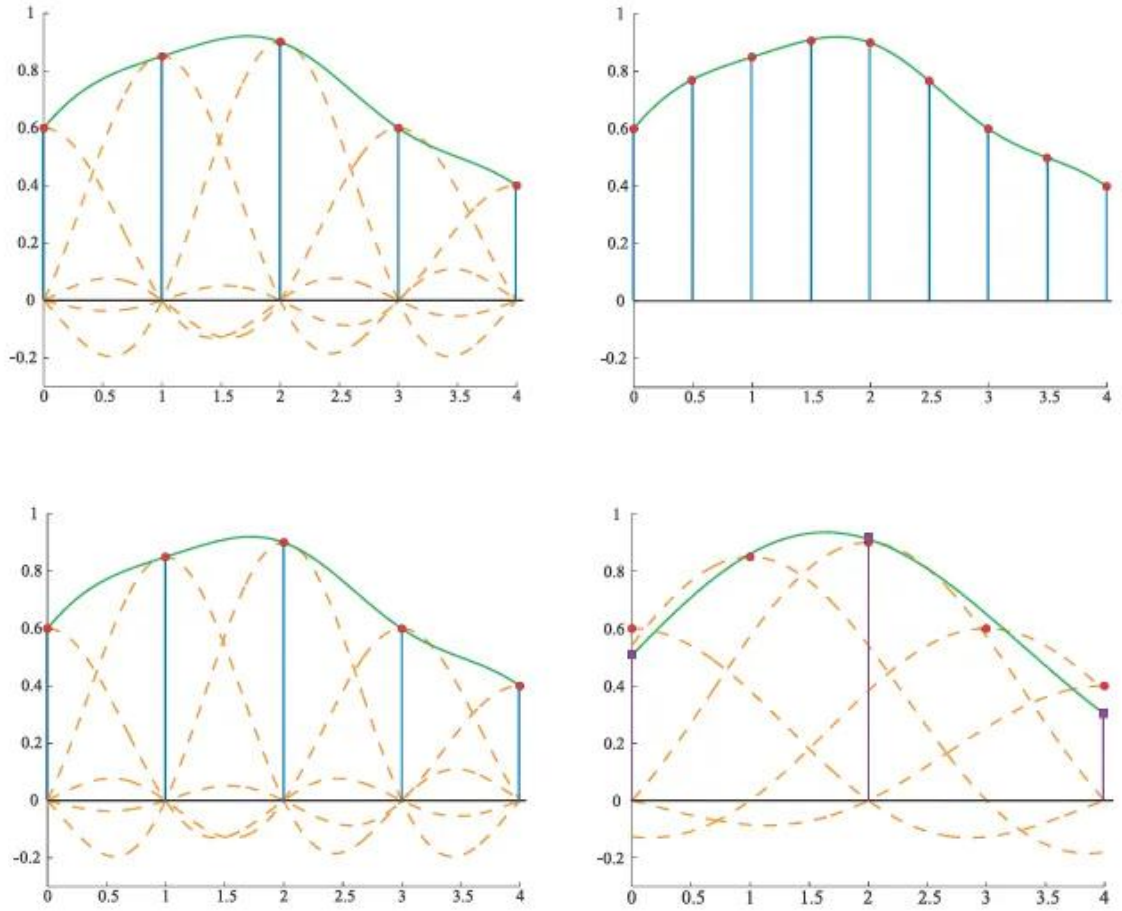




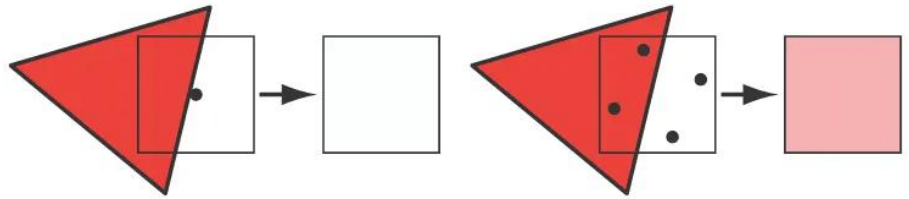


Resampling

- 샘플된 신호를 재조정
- Magnification
  - 원하는 간격으로 자유롭게
- Minification
  - 앨리어싱 발생 가능
  - Sinc(x/a) 사용
    - 저역 필터의 폭 증가
    - 높은 주파수 성분 제거

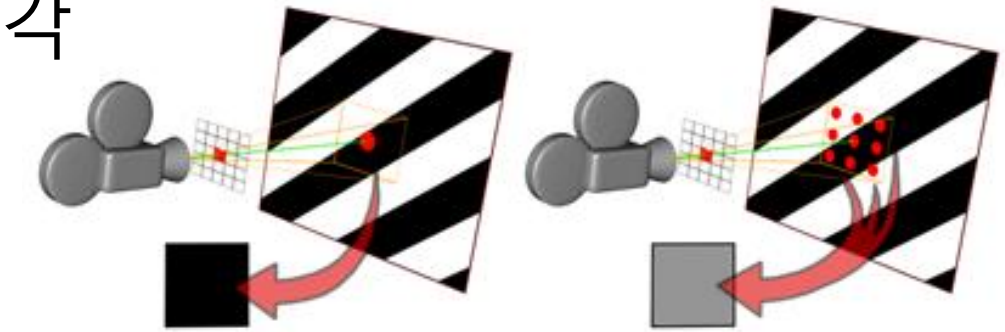


Screen-Based Antialiasing



- 렌더링 품질 향상 목적
- 정답 X -> 다양한 기법이 존재
- 기법마다 가중치, sample 위치는 제각각

$$\mathbf{p}(x,y) = \sum_{i=1}^{n} w_i \mathbf{c}(i,x,y)$$



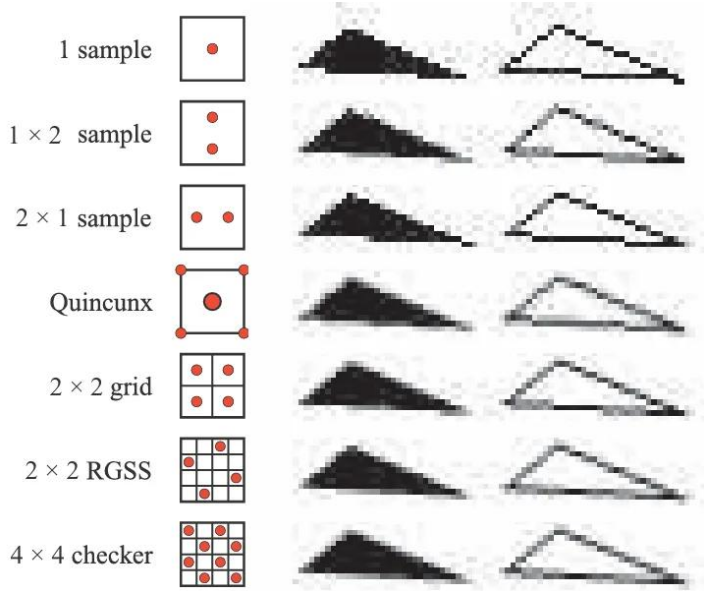
- Super sampling Antialiasing(SSAA) -> 단순, 정확, 비쌈
  - 인접한 픽셀들의 평균 계산 (픽셀 셰이더 여러 번)
  - Accumulation buffer : 동일한 해상도에 더 많은 색상 bit 저장

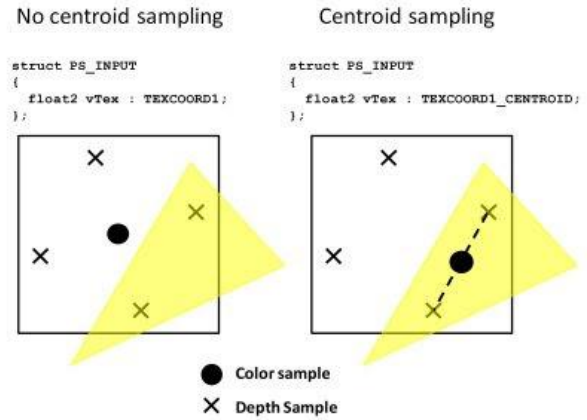
Multisampling AA(MSAA)

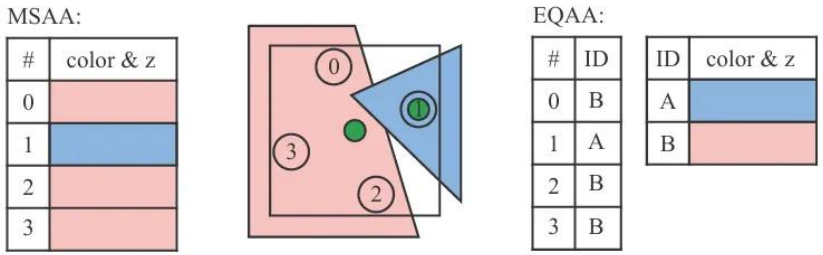
- 픽셀당 여러 위치의 샘플 활용
  - 픽셀 셰이더 한번에 비슷한 효율

- Centroid sampling(interpolation)
  - 셰이딩 샘플의 위치를 조정하는 기술

- EQAA
  - Coverage sample을 추가하여 면 위에 있는지 확인







Multisampling AA(MSAA)

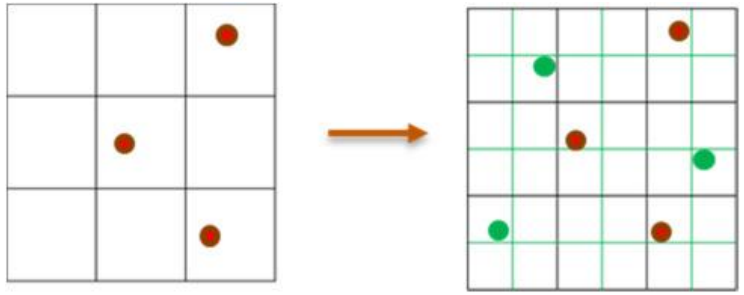
- 모든 도형이 multiple-sample buffer에 렌더링되면
  - resolve op : 샘플 색상을 평균하여 픽셀의 색상 결정

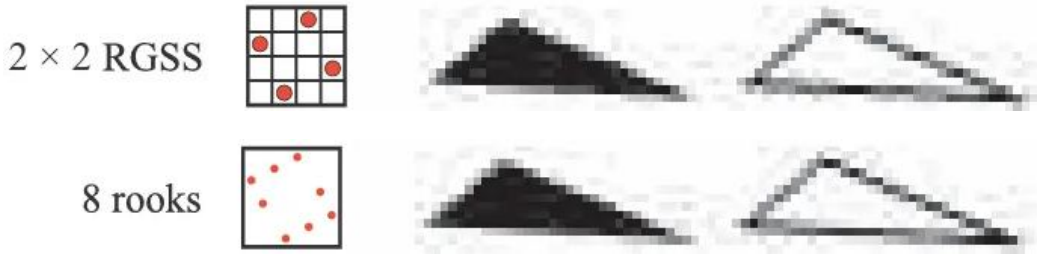
- Dynamic range가 높다면
  - Tone map : high(HDR) -> standard(SDR)

- Sampling pattern을 수동으로 수행한다면
  - Temporal AA algorithm 사용
  - 이전 이미지와 블렌딩 / 샘플로 2~4프레임만 사용

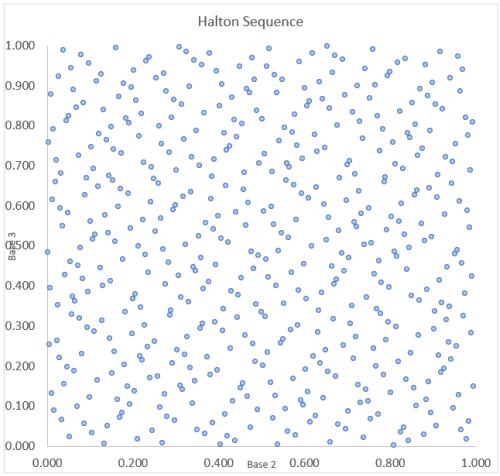
Sampling pattern

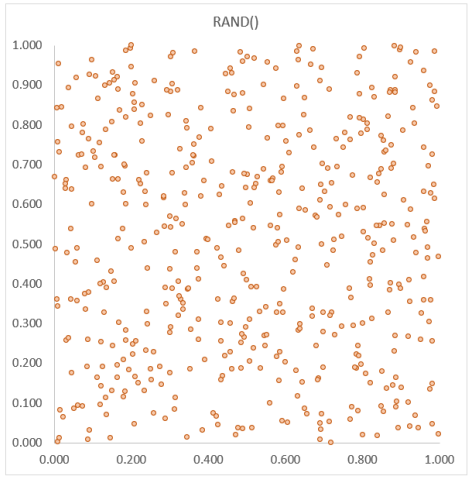
- 효과적인 AA를 위한 패턴 찾기는 중요
- Edge, corner 부근에서 앨리어싱 자주 발생
  - RGSS
  - Latin hypercube (고르게 분포)
  - N-rooks





- Stratified sampling이 추가로 필요
  - Halton sequence





Stochastic sampling

- 반복되는 앨리어싱 효과를 random의 노이즈로 변환

- 픽셀 단위에서는 여전히 문제
  - 픽셀마다 다른 pattern
  - Interleaved sampling
    - 시간에 따라 sampling 위치 변경
    - ATI's smoothvision

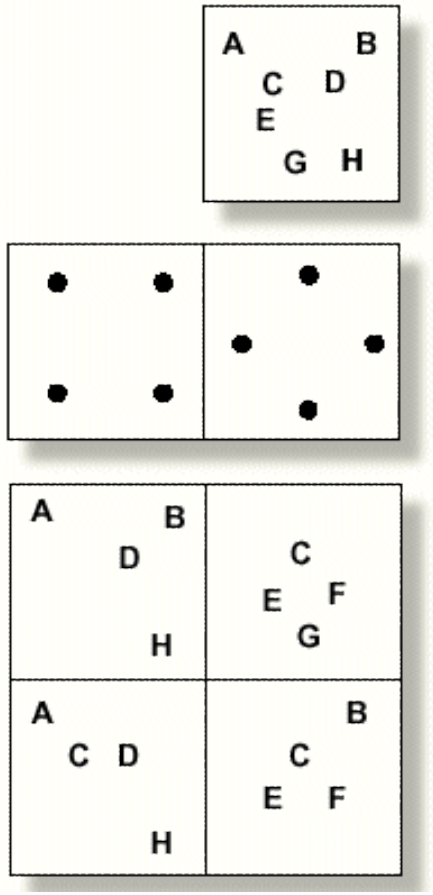


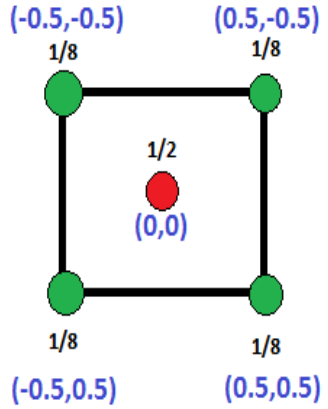
Figure 11: *The sampling template is used to determine where every pixel gets its sample location.*

Figure 12: *Non-programmable sampling patterns provide a redundant anti-aliasing solution, which leads to lower visual quality, and possible image artifacts*

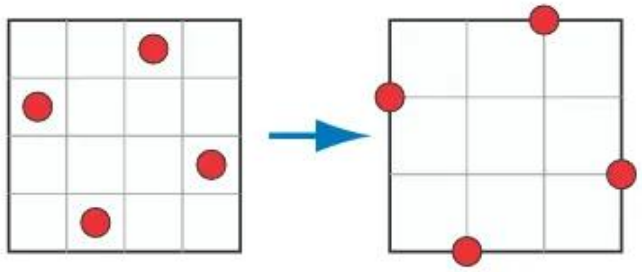
Figure 13: *Four pixels with 4x SMOOTHVISION™ applied: pixel sample locations are programmed on a per pixel basis, allowing for a more intelligent and high quality anti-aliasing solution. Each square represents a single pixel. Each letter represents a single sample location.*

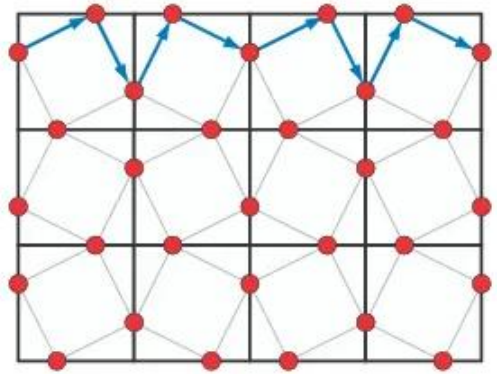
Quincunx & Rotated Grid(RG)

- Quincunx
  - 픽셀당 평균 2개의 샘플
  - Temporal AA로 사용 가능
  - 움직이는 객체는 어려움



- Rotated Grid
  - Edge 부근 변형에 특화
  - flipquad



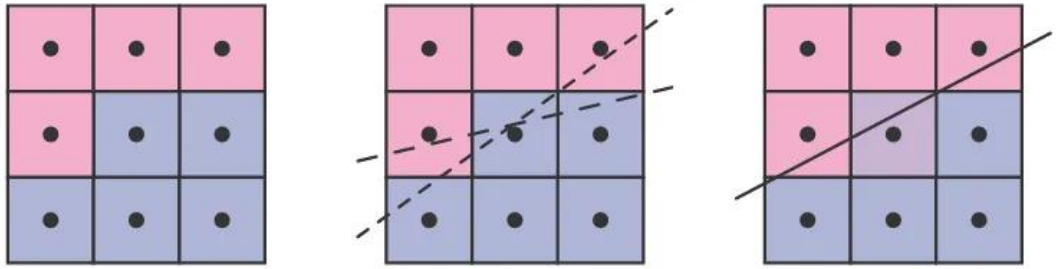


Morphological methods

- Geometry의 edge 에서 앨리어싱이 자주 발생함

➔구조와 관련지어 해결 (morphological AA[MLAA])

- Subpixel reconstruction AA(SRAA)
  - 깊이, 법선 등 사용



- Geometry buffer AA(GBAA) & Distance-to-edge AA(DEAA)
  - Renderer가 삼각형 edge의 위치를 찾을 수 있음

- 후처리(그림자, 하이라이트…)에서도 개선 가능

Morphological methods

- 두 object 간 색상 차이가 낮을 경우
- 3개 이상의 표면이 겹칠 경우
- 표면에 high-contrast, high-frequency가 있을 경우
- Corner가 둥글게 표현될 수 있음
  - MSAA coverage mask로 개선

- 제공된 정보만을 사용 -> 더 많은 샘플 & 한계점 존재

Morphological methods

- Fast Approximate AA(FXAA)
  - 이미지 필터 -> fast & low-quality

- Subpixel Morphological AA(SMAA)
  - 서브픽셀 단위

- Color-only input
- 1~2ms per frame
- Temporal AA

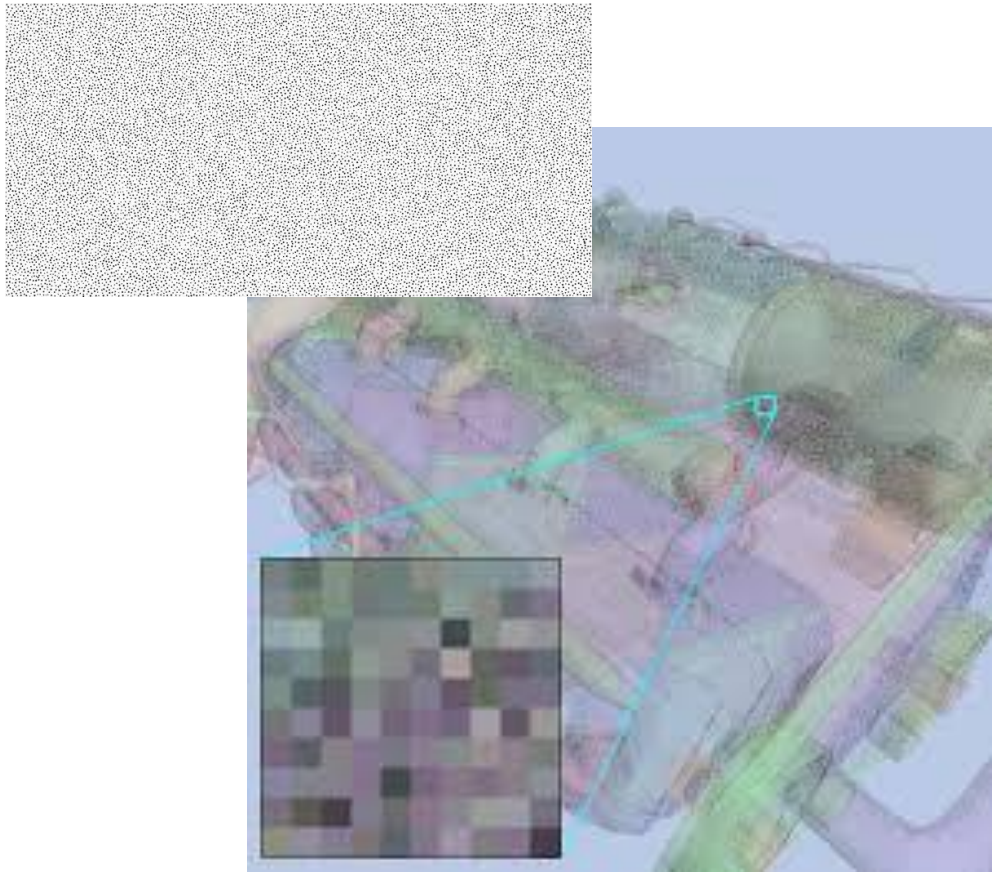
Transparency, Alpha and Compositing

- 렌더링에서 반투명(semitransparent) 물체를 다루는 방식
  - 빛 기반 효과(light-based effect)
  - 시점 기반 효과 (view-based effect)
- 우선은 view-based 투명도에 대하여!

- Screen-door 투명도
  - 체커보드 형태로 렌더링
  - 단순함!
    - 한 번에 한 물체만
    - 2개까지 겹칠 수 있음

Stochastic transparency

- Subpixel screen-door mask + stochastic sampling
- random한 stipple pattern 사용
  - 합리적이나, 노이즈 존재
  - 픽셀당 더 많은 샘플 -> 성능 향상
  - 블렌딩 필요x



Alpha blending

- Alpha : 물체의 덮인 정도를 나타내는 요소
  - Opacity, coverage, or both

- AA(MSAA 등)에서 coverage는 샘플에서 고려

- Ex) 비누방울의 edge가 픽셀 ¾ 을 덮음 & 투명도 : 0.1
  - Alpha = 0.75 * 0.1 = 0.075
  - AA 방식에선, 각 샘플의 alpha = 0.1

Blending Order

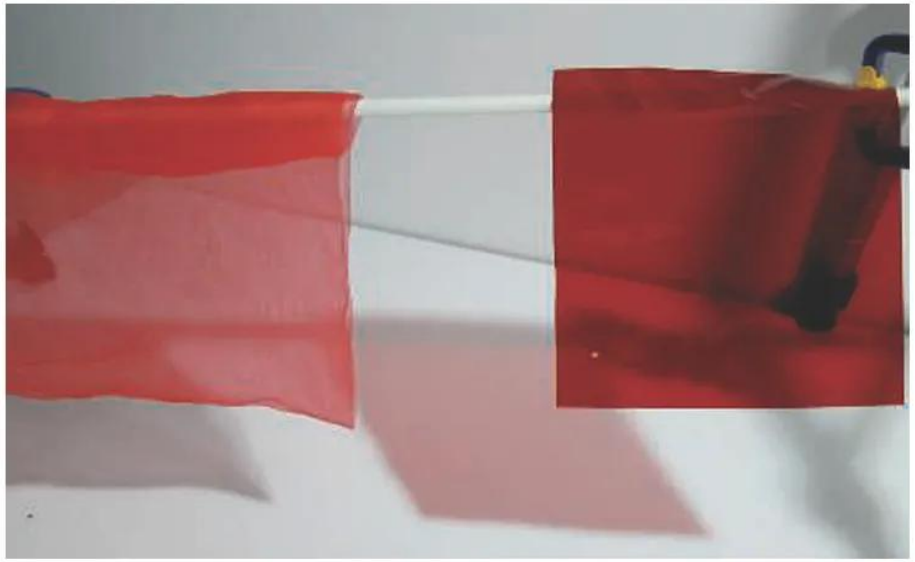
- Object를 덮는 픽셀들은 픽셀 셰이더에서 RGBα(RGBA)를 받음
- 전달받는 값을 원래 색상과 혼합하는 과정이 필요

$$\mathbf{c}_o = \alpha_s \mathbf{c}_s + (1 - \alpha_s)\mathbf{c}_d \quad [\textbf{over operator}]$$

- 렌더링되는 object의 반투명한 모습
- 다양한 투명 효과는 어려움

$$\mathbf{c}_o = \alpha_s \mathbf{c}_s + \mathbf{c}_d$$

- Additive blending
- 발광효과에 최적



Blending Order

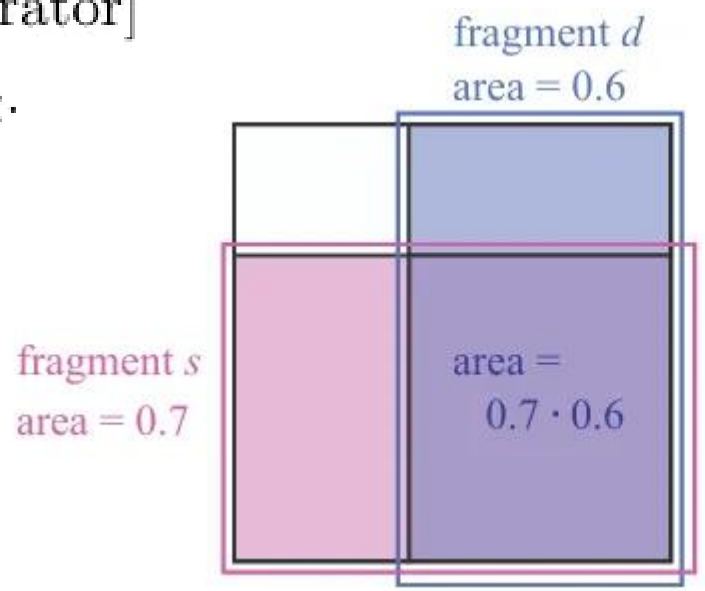
- z-buffer의 한계 : 한 픽셀당 object 한 개만

- Object 겹칠 경우, 저장 후 후처리 X
- 뒤에서 앞 순서로 처리 (over)
  - View 방향에 따라 거리 정렬이 필요
  - 겹쳤으면, per-mesh 기준으로 해결 X

Blending Order

- 앞에서 뒤 순서
- Alpha를 따로 계산함(순서에 무관)

$$\mathbf{c}_o = \alpha_d \mathbf{c}_d + (1 - \alpha_d)\alpha_s \mathbf{c}_s \quad [\textbf{under operator}]$$
$$\mathbf{a}_o = \alpha_s(1 - \alpha_d) + \alpha_d = \alpha_s - \alpha_s\alpha_d + \alpha_d.$$

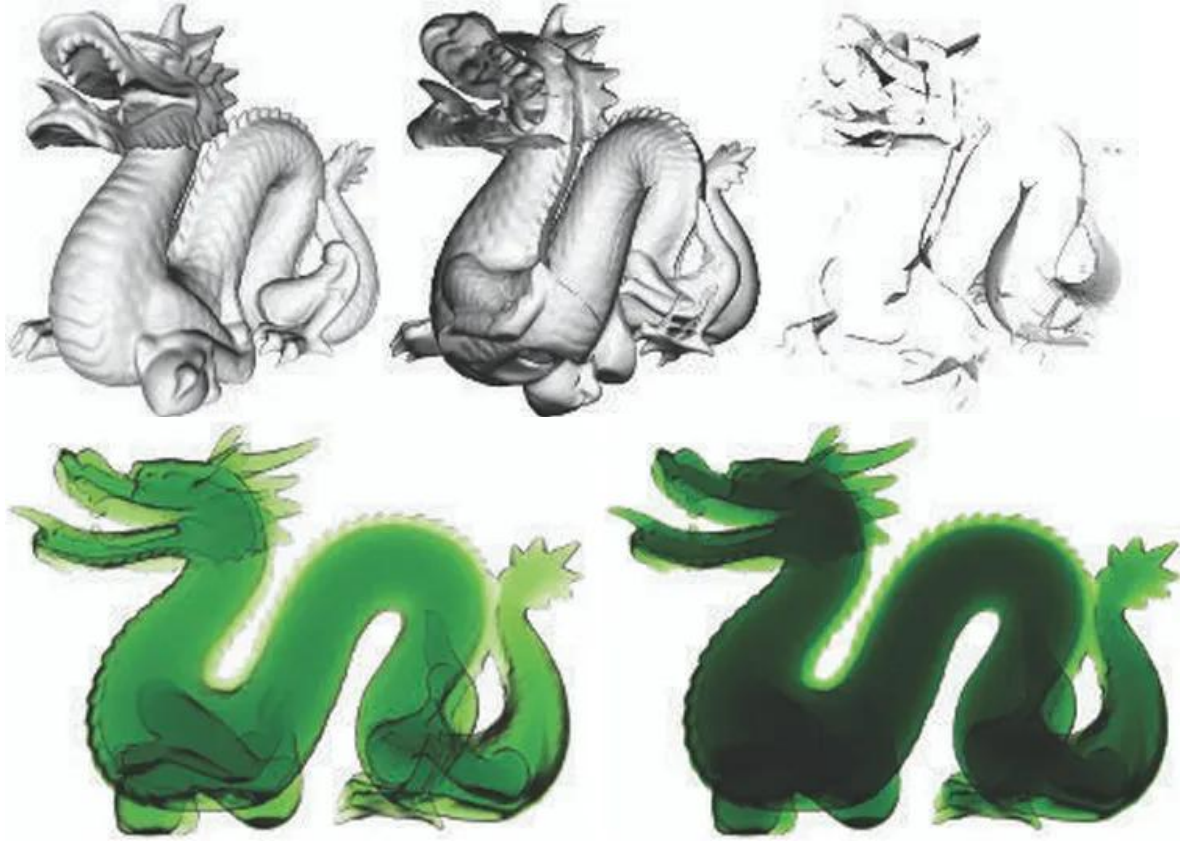


Given two fragments of areas (alphas) 0.7 and 0.6, total area covered = $0.7 - 0.7 \cdot 0.6 + 0.6 = 0.88$

Order-Independent Transparency

- under : 모든 투명 object를 별도의 색상 버퍼로 그림
- over : 색상 버퍼들로 불투명 장면 위에 겹침

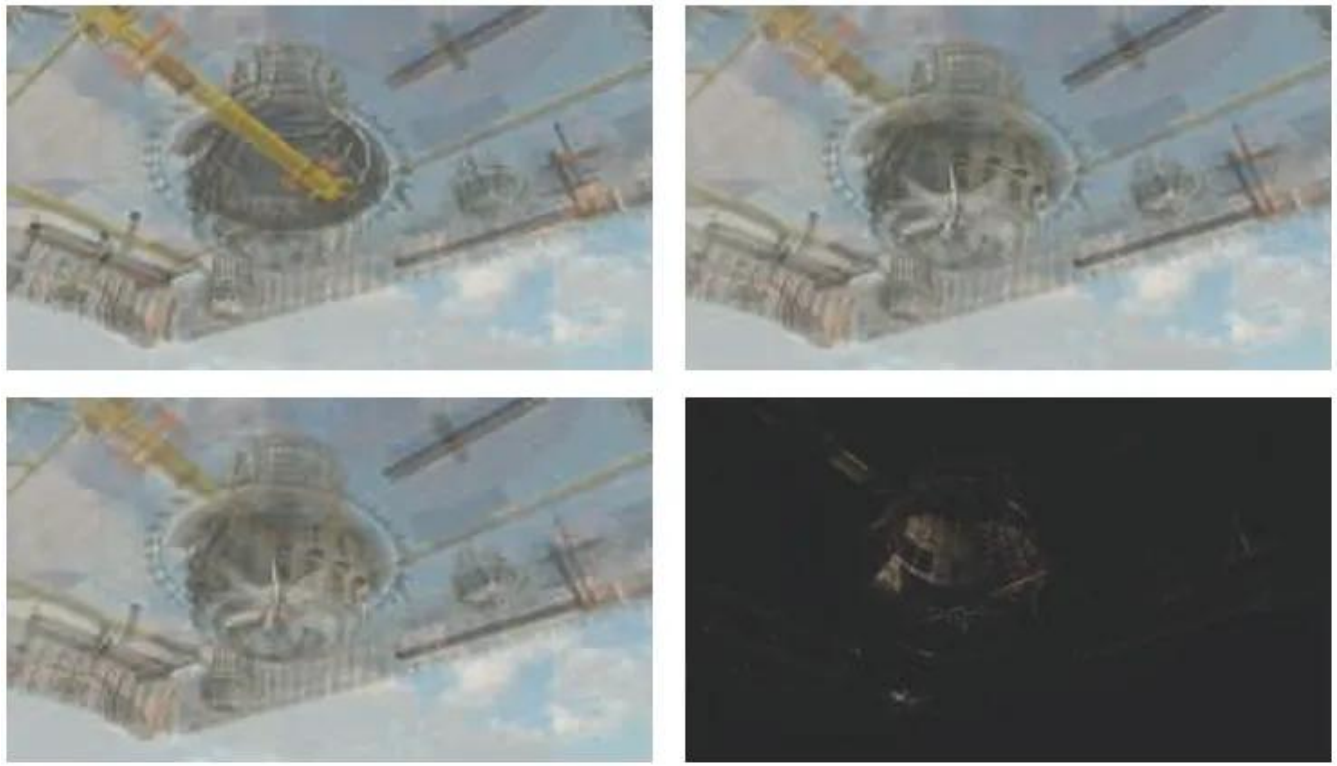
- Depth peeling
  - 모든 표면의 z-depth 저장
  - 모든 투명 object 렌더링
  - z-depth가 작은 object 렌더링
  - 진행한 z-depth보다 작으면 peel
  - z-depth에 맞춰 순차적 렌더링



Order-Independent Transparency

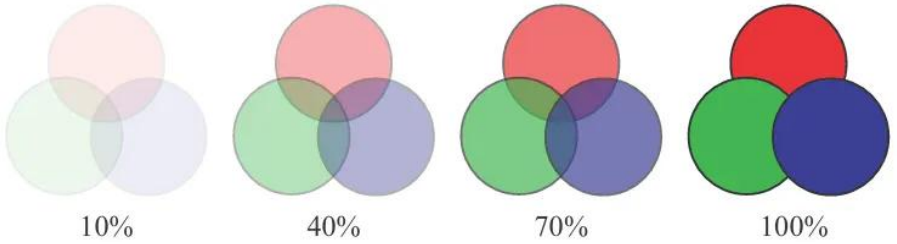
- A-buffer : 렌더링된 삼각형이 coverage mask 생성 후,
  픽셀의 모든 fragment 저장

- Multi-layer alpha blending
  - GPU의 픽셀 동기화를 활용



Order-Independent Transparency

- K-buffer
  - 처음 몇 개의 레이어만 최대한 저장 & 정렬
  - 더 깊은 레이어는 삭제 & 병합
  - weighted average 활용
  - 낮은 불투명도에서 좋음
  - weighted sum



$$\mathbf{c}_o = \sum_{i=1}^{n}(\alpha_i \mathbf{c}_i) + \mathbf{c}_d(1 - \sum_{i=1}^{n}\alpha_i)$$

weighted average

$$\mathbf{c}_{\text{sum}} = \sum_{i=1}^{n}(\alpha_i \mathbf{c}_i), \quad \alpha_{\text{sum}} = \sum_{i=1}^{n}\alpha_i,$$

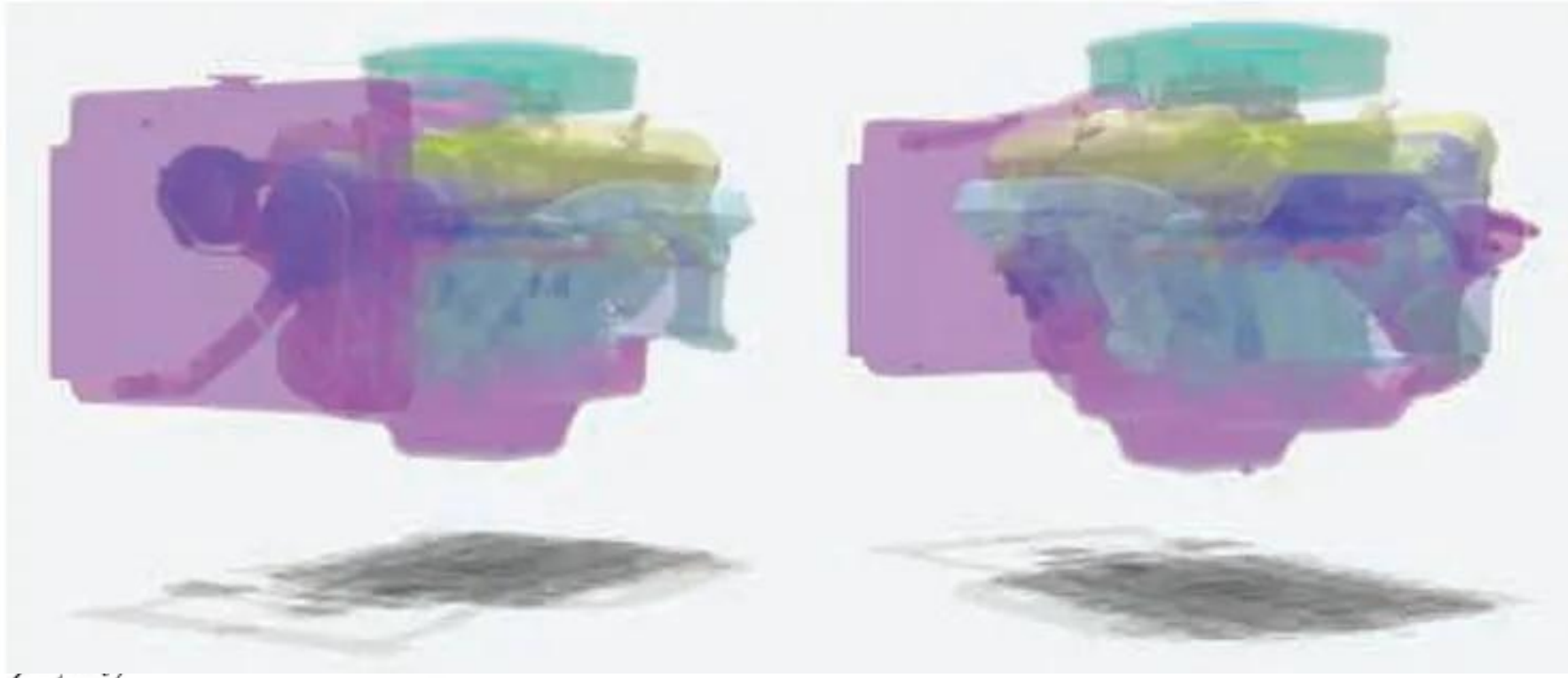
$$\mathbf{c}_{\text{wavg}} = \frac{\mathbf{c}_{\text{sum}}}{\alpha_{\text{sum}}}, \quad \alpha_{\text{avg}} = \frac{\alpha_{\text{sum}}}{n},$$

$$u = (1 - \alpha_{\text{avg}})^n,$$

$$\mathbf{c}_o = (1 - u)\mathbf{c}_{\text{wavg}} + u\mathbf{c}_d.$$

Order-Independent Transparency

- K-buffer
  - 처음 몇 개의 레
  - 더 깊은 레이어
  - weighted avera
  - 낮은 불투명도(
  - weighted sum



$$\mathbf{c}_o = \sum_{i=1}^{n}(\alpha_i \mathbf{c}_i) + \mathbf{c}_d(1 - \sum_{i=1}^{} \quad ,$$

$$\mathbf{c}_{\text{wavg}} = \frac{\mathbf{c}_{\text{sum}}}{\alpha_{\text{sum}}}, \quad \alpha_{\text{avg}} = \frac{\alpha_{\text{sum}}}{n},$$

$$u = (1 - \alpha_{\text{avg}})^n,$$

$$\mathbf{c}_o = (1 - u)\mathbf{c}_{\text{wavg}} + u\mathbf{c}_d.$$

Premultiplied Alphas and Compositing

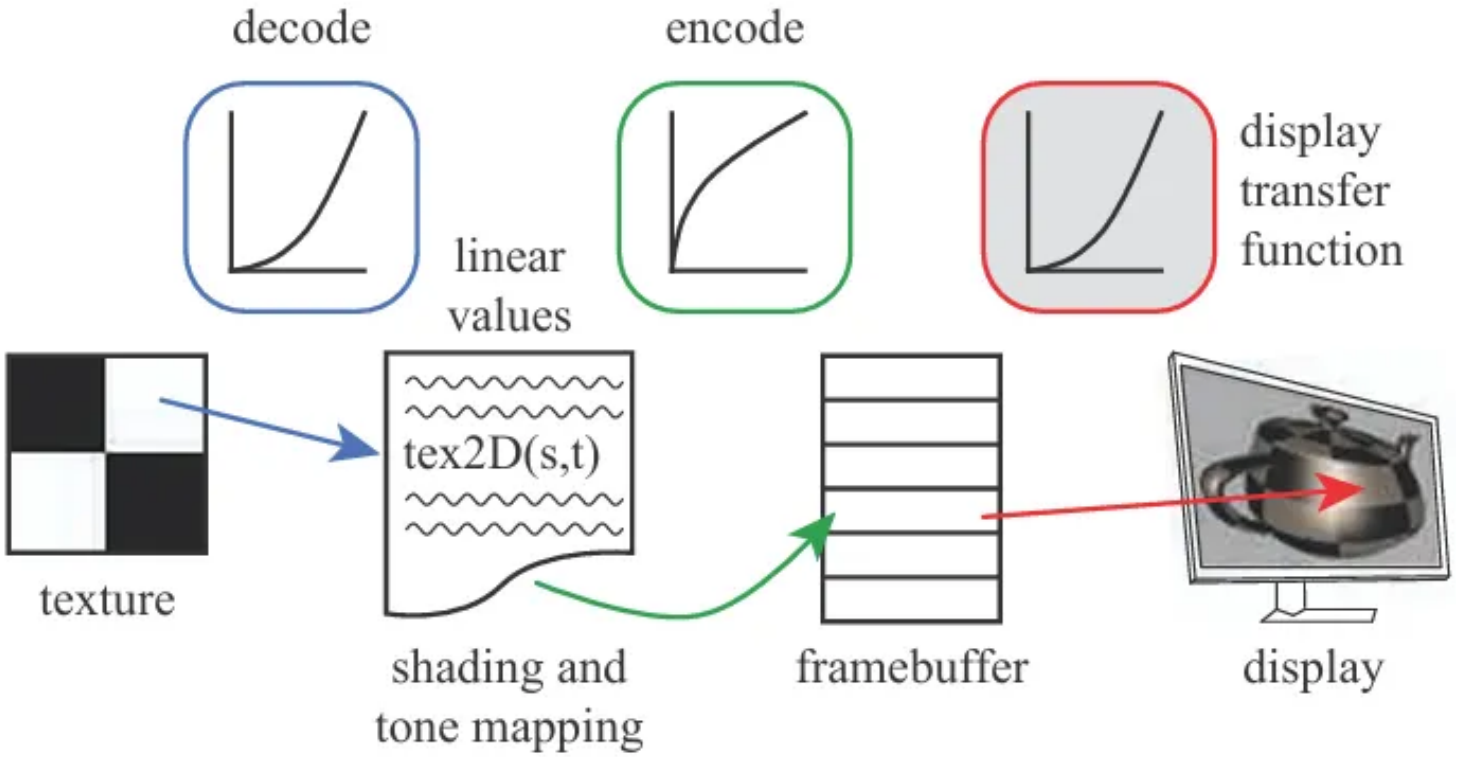
- Compositing : over op.가 여러 object의 결과를 블렌딩
- Matte : 알파 채널에 의해 형성된 이미지 
- 합성된 RGBα를 사용 -> premul alpha를 사용

$$\mathbf{c}_o = \mathbf{c}'_s + (1 - \alpha_s)\mathbf{c}_d$$

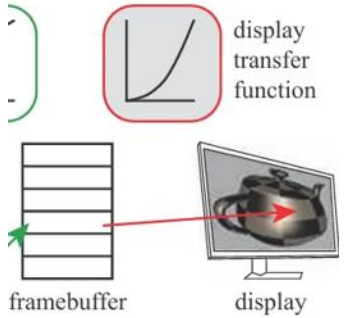
- unmul alpha : 원색 저장
- Chroma-keying : 특정 색을 투명하게

Display Encoding

- 모든 연산 : linear / display : nonlinear?
- Gamma correction : [0,1] 정규화 & power (1/2.2)



Display Encoding



- sRGB : display transfer function에서 정의되는 색 영역

- 디코드 함수

$$y = f_{\text{sRGB}}^{-1}(x) = \begin{cases} 1.055x^{1/2.4} - 0.055, & \text{where } x > 0.0031308 \\ 12.92x, & \text{where } x \leq 0.0031308 \end{cases}$$

$$y = f_{\text{display}}^{-1}(x) = x^{1/\gamma}$$

$$y = f_{\text{simpl}}^{-1}(x) = \sqrt{x},$$

- 인코드 함수

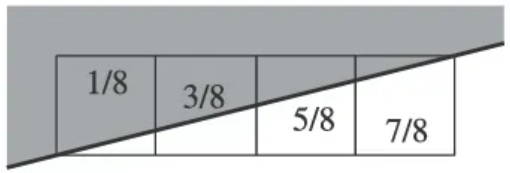
$$x = f_{\text{sRGB}}(y) = \begin{cases} \left(\frac{y + 0.055}{1.055}\right)^{2.4}, & \text{where } y > 0.04045 \\ \frac{y}{12.92}, & \text{where } y \leq 0.04045 \end{cases}$$

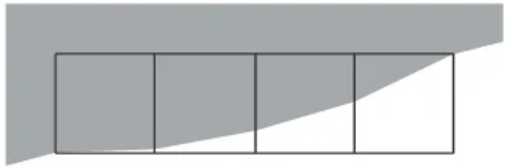
$$x = f_{\text{display}}(y) = y^{\gamma}$$

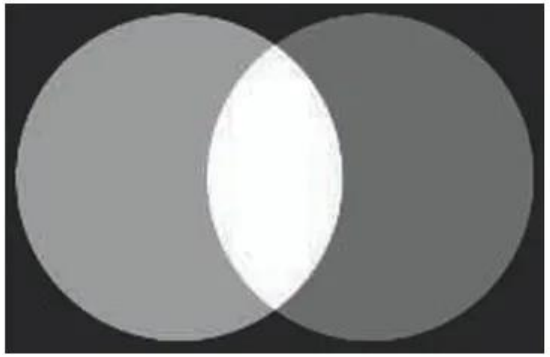
$$x = f_{\text{simpl}}(y) = y^2;$$

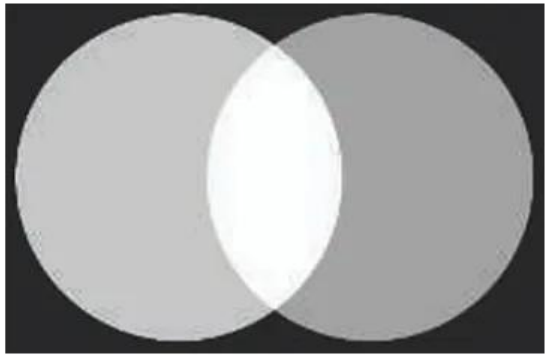
Display Encoding

- 보정을 무시한 경우
  - 낮은 선형 값에서 더 어두움 (0.1 -> 0.351 / 0.5 -> 0.730)
  - 모서리 품질
  - 픽셀의 발광이 어두워짐









ropping