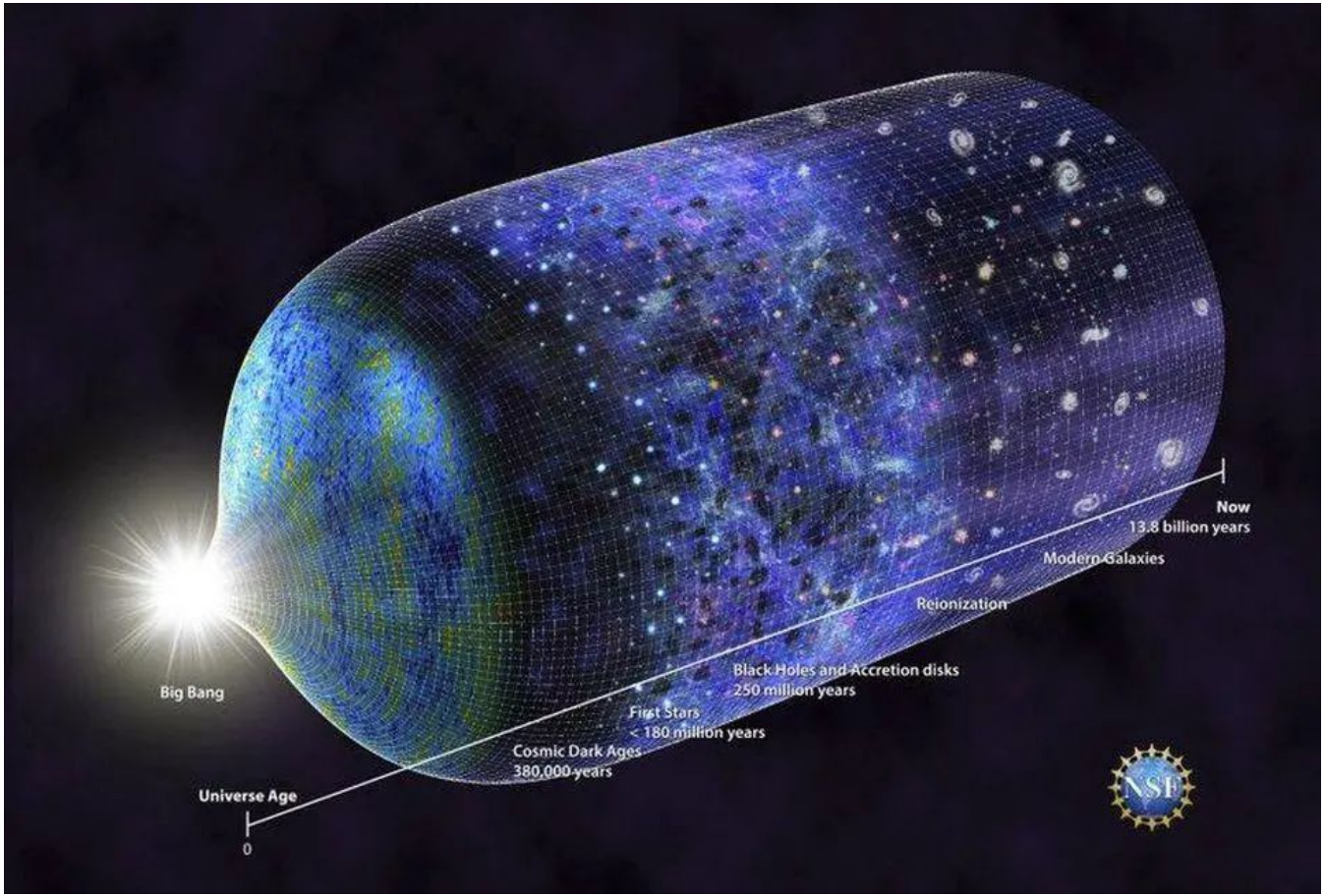
Subsurface scattering

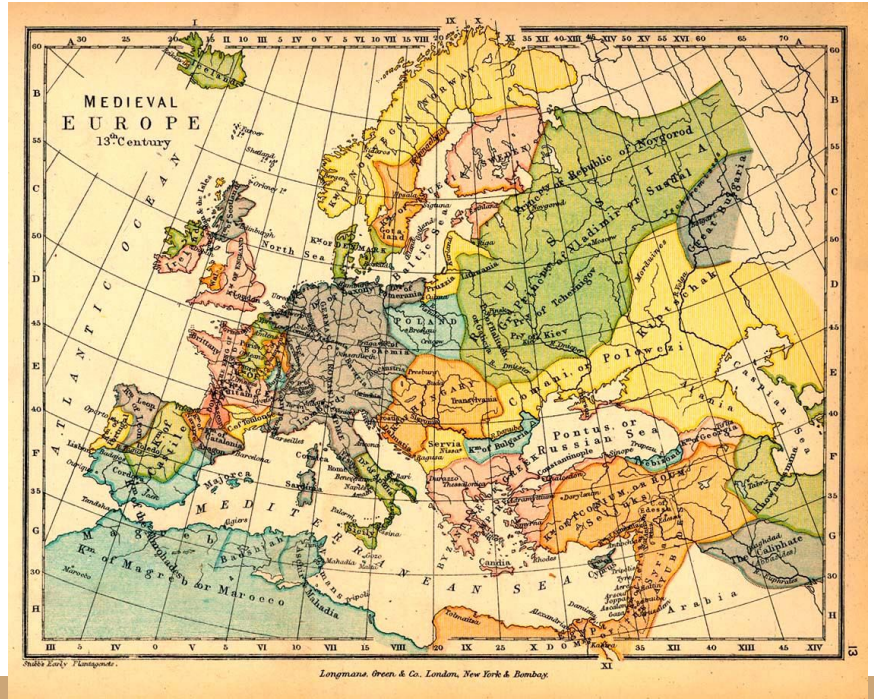
Karl-Hendrik Veidenberg

Let's start at the beginning



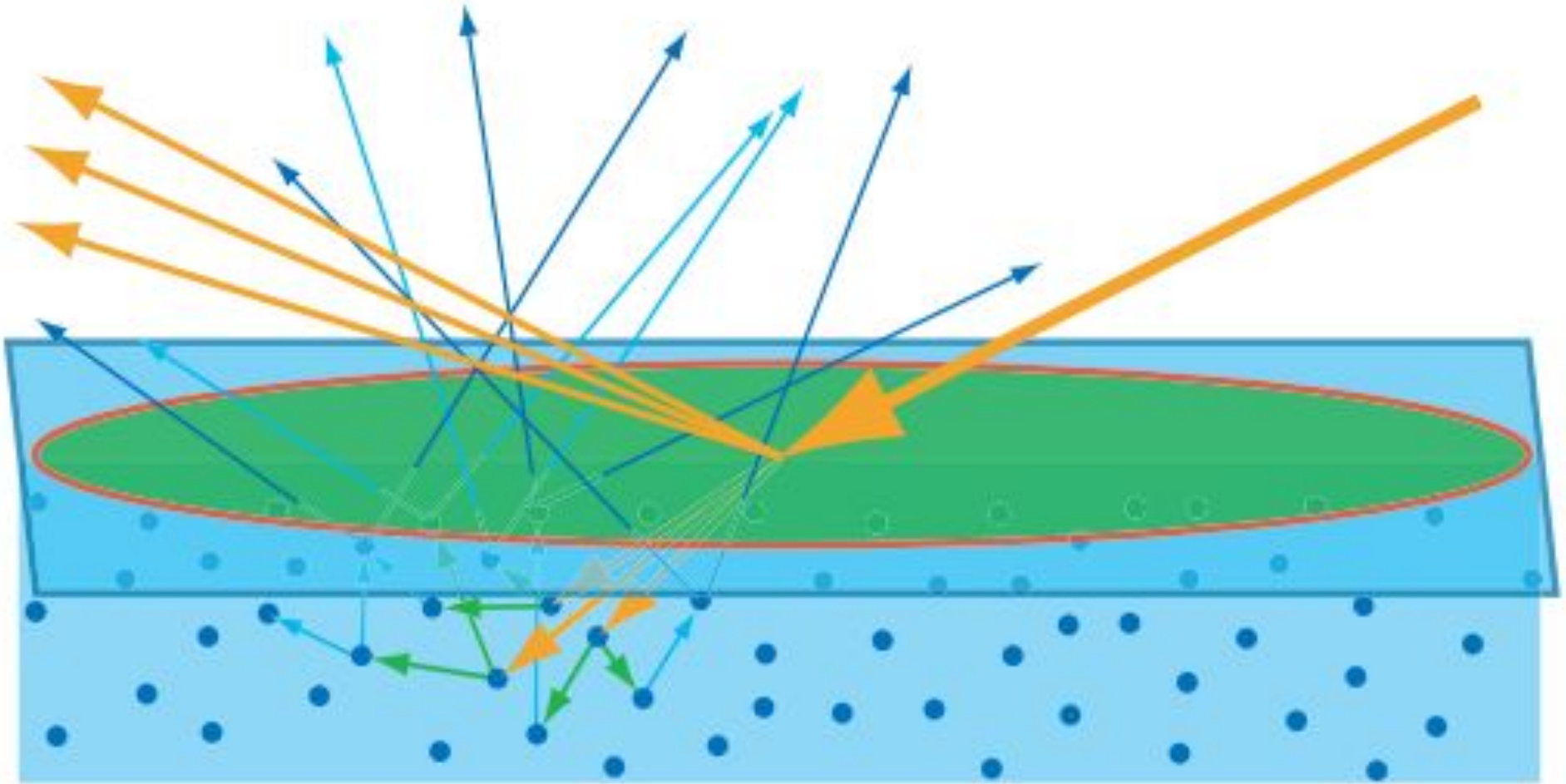
Nevermind, let's skip ahead

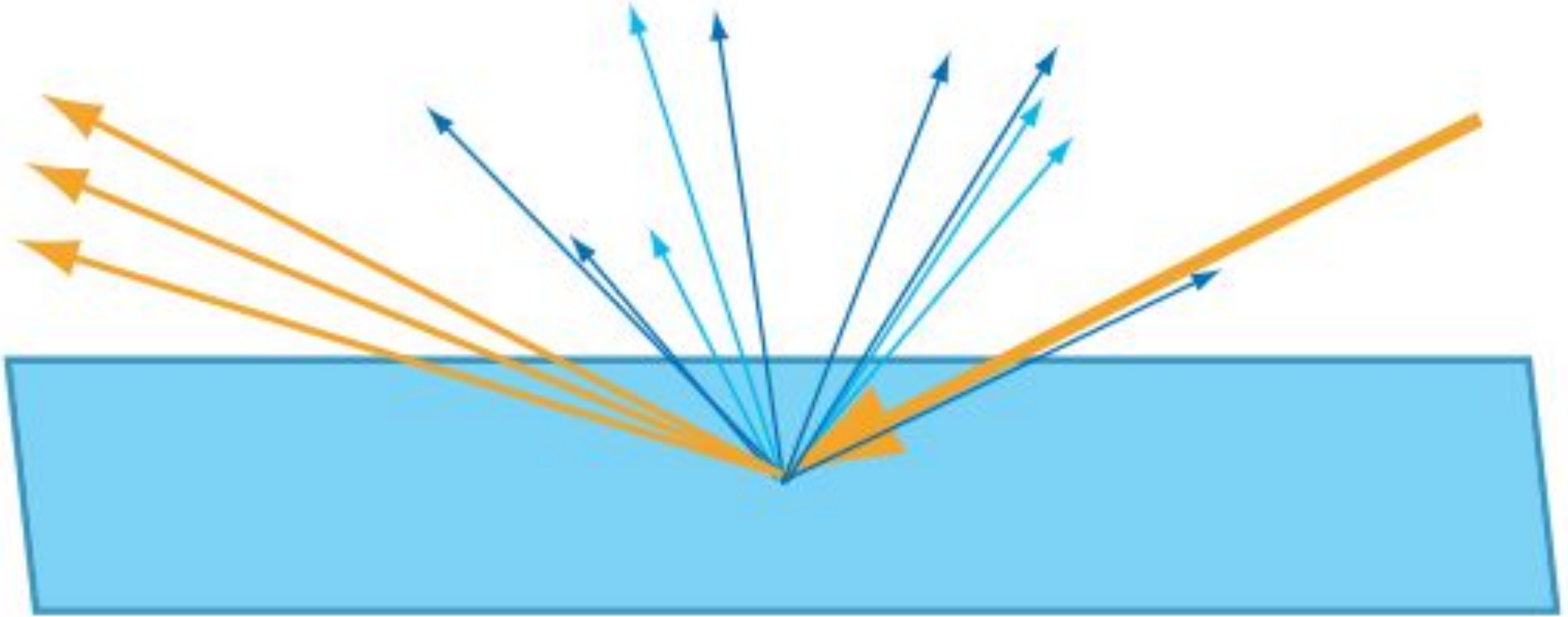
- Sub - under (latin)
- Surface - to come out of hiding (french)
- Scatter (sceaterian) - to go or flee in different directions (old-english)

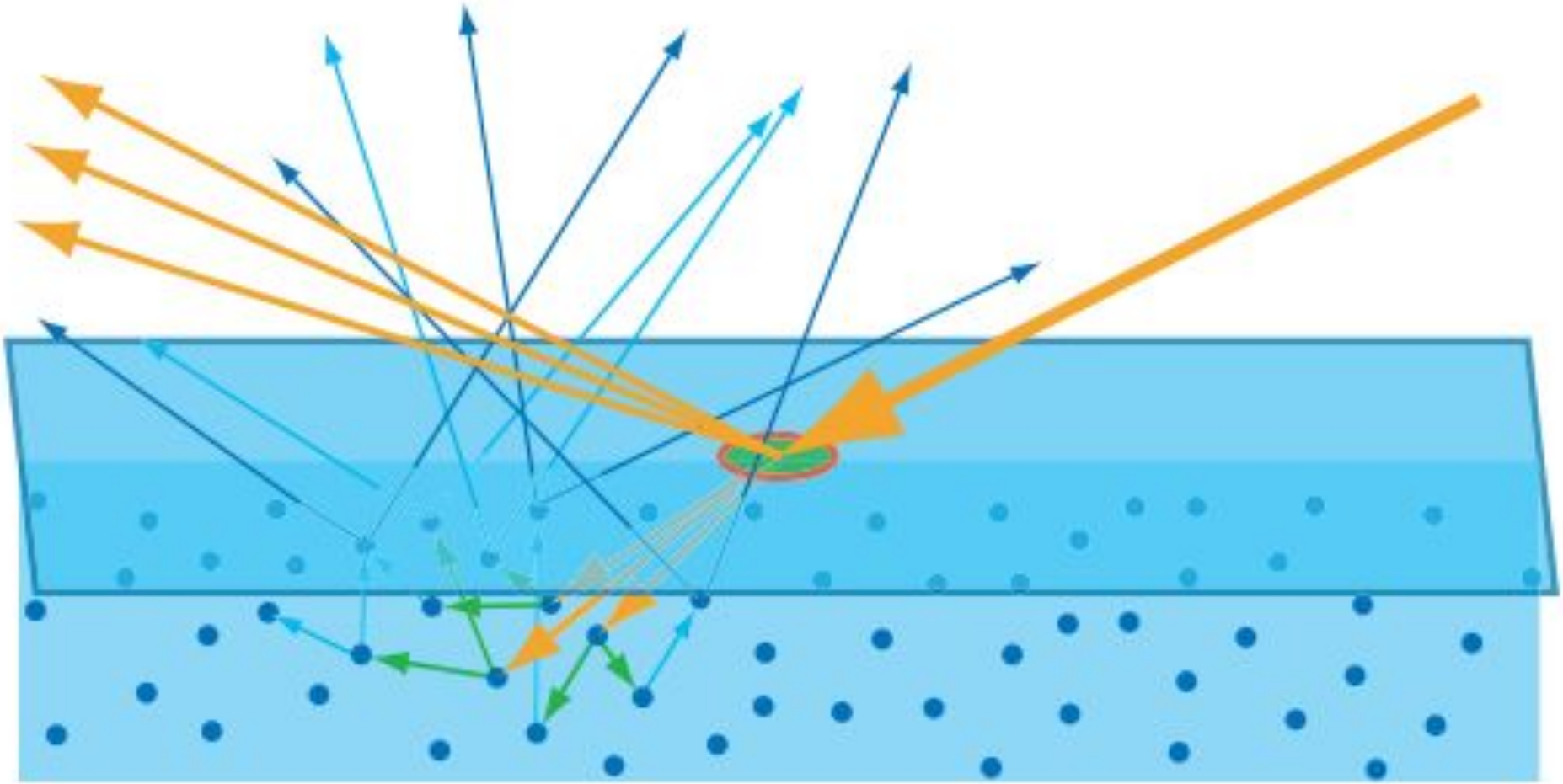


Light

1. Light penetrates a translucent object (skin, wax, milk etc.)
2. Light bounces around in the object
3. Light passes back out of the object at a different angle







BRDF and BSSRDF

- BRDF - Bidirectional Reflectance Distribution Function
  - w_i - incoming light direction (vector)
  - w_r - outgoing light direction (vector)
  - L - radiance
  - E - irradiance (power per unit of surface area)

$$f_r(\omega_i, \omega_r) = \frac{dL_r(\omega_r)}{dE_i(\omega_i)} = \frac{1}{L_i(\omega_i)\cos\theta_i}\frac{dL_r(\omega_r)}{d\omega_i}$$

- BSSRDF - Bidirectional Scattering Surface Reflectance Distribution Function
  - p_o - point where light resurfaces from the material
  - w_o - direction from surface to point p_o
  - p_i - point where light enters the material
  - w_i - direction from surface to point p_i

$$S(p_o, \omega_o, p_i, \omega_i) = \frac{dL_o(p_o, \omega_o)}{d\Phi(p_i, \omega_i)}$$


BRDF
[Jensen et al. 2001]
BSSRDF
[Jensen et al. 2001]

Depth Map based subsurface scattering

1. Render scene from the light source
2. Convert that into a depth map
3. Project that back onto the scene using projective texture mapping
4. Surface distance from light - exit point distance from light = distance light has traveled through the object

NB! Only works with convex objects.

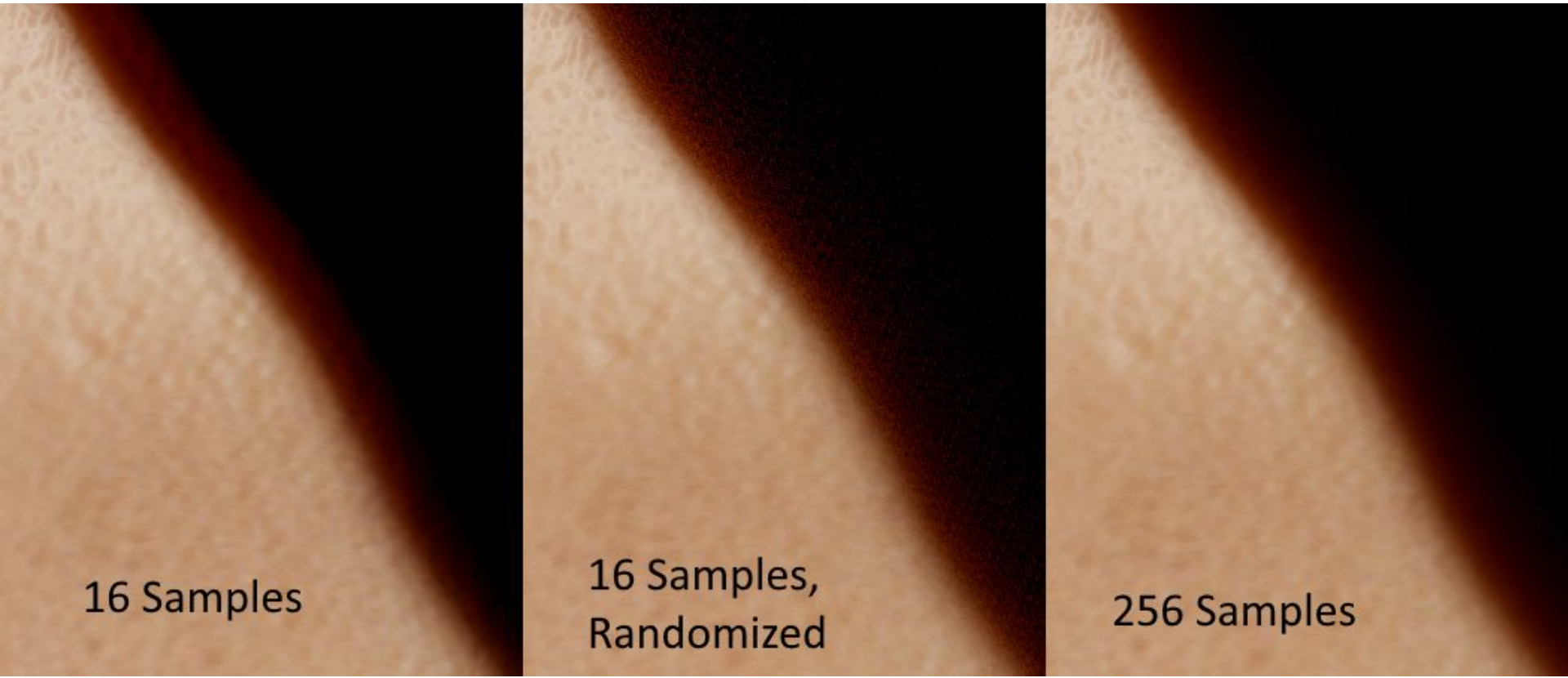
Texture space diffusion

1. Draw the mesh using a special vertex shader and a pixel shader to compute the irradiance map in texture-space
2. Blur the irradiance map with bunch of Gaussian kernels
3. Draw the mesh for real using the camera’s projection matrix
4. In the pixel shader sample the convolved irradiance maps
5. Apply the diffuse albedo map to get the diffuse reflectance with subsurface scattering

Screen space subsurface scattering

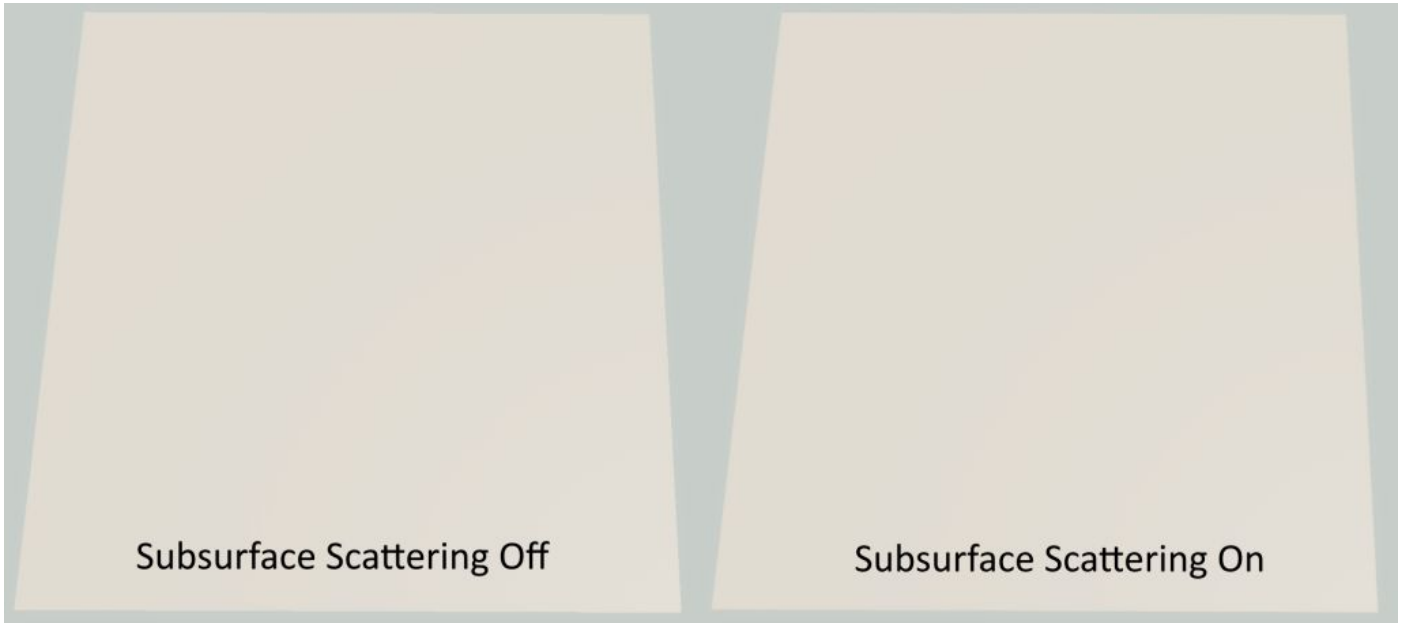
1. Hope that 3D locality of a sample point's neighboring surfaces is preserved when it's rasterized and projected into 2D
2. Do a few texture fetches from neighboring texels in order to convolve the lighting with your diffusion profile

NB! Similar to texture-space diffusion, you can't really account for light that transmits through thin surfaces like the ear.


16 Samples
16 Samples,
Randomized
256 Samples

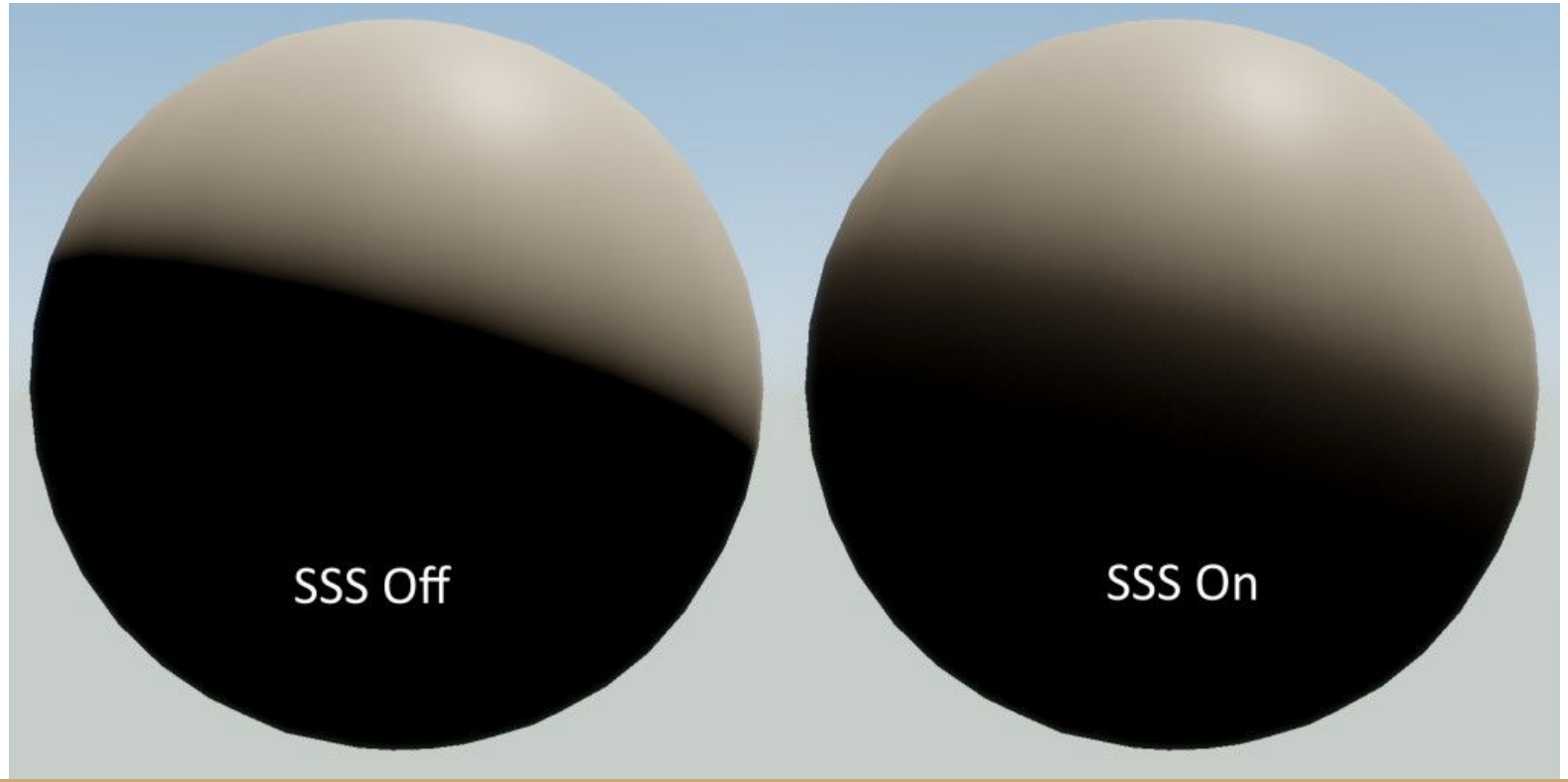
Pre-integrated subsurface scattering

- Pre-compute the scattered results, and use the correct result based on a few properties of the point being shaded
- Sub-surface scattering isn't visible if the surface is flat and the light is uniform



Pre-integrated subsurface scattering

- Therefore, SSS **is** visible when those two things aren't true



Separable subsurface scattering

- The best option for screen space subsurface scattering
- Same result as 12-pass Gaussian, using only a 2-pass method

Used literature

http://graphics.ucsd.edu/~henrik/images/subsurf.html

https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch16.html

https://cs184.eecs.berkeley.edu/sp20/lecture/15-71/advanced-topics-in-material-mode

https://www.pbr-book.org/3ed-2018/Color_and_Radiometry/Surface_Reflection

https://therealmjp.github.io/posts/sss-intro/

http://www.iryoku.com/separable-sss-released

Thank you for listening!