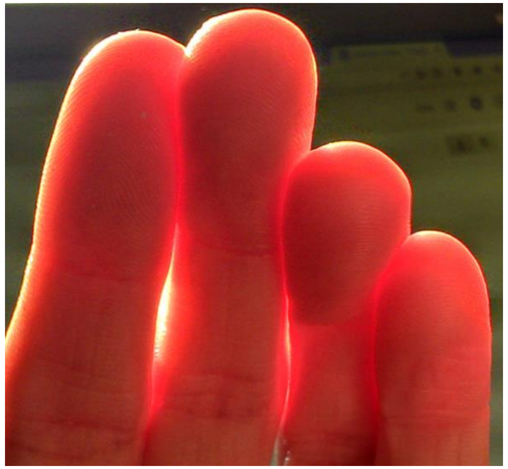
Computer Graphics

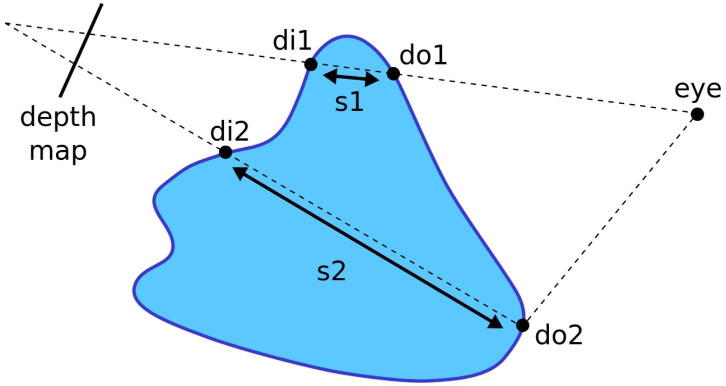
Sub-Surface Scattering

Tobias Isenberg

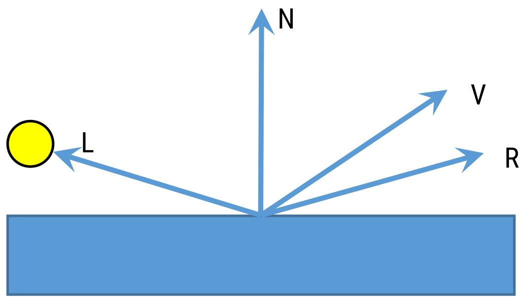
Overview

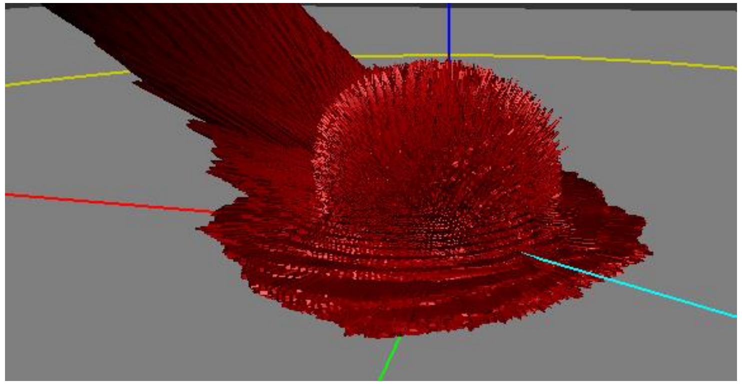
- motivation – 2 approaches - applications

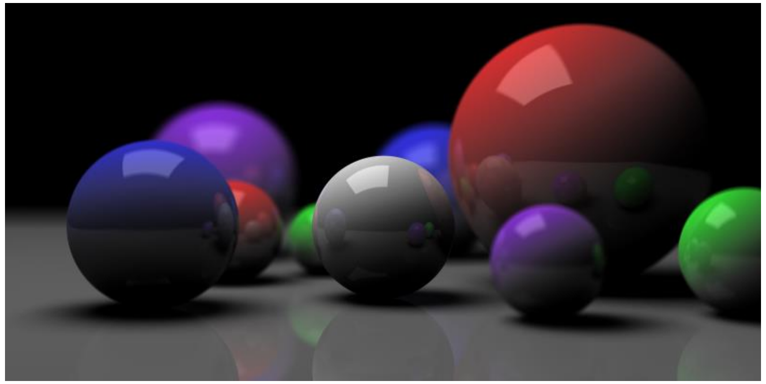




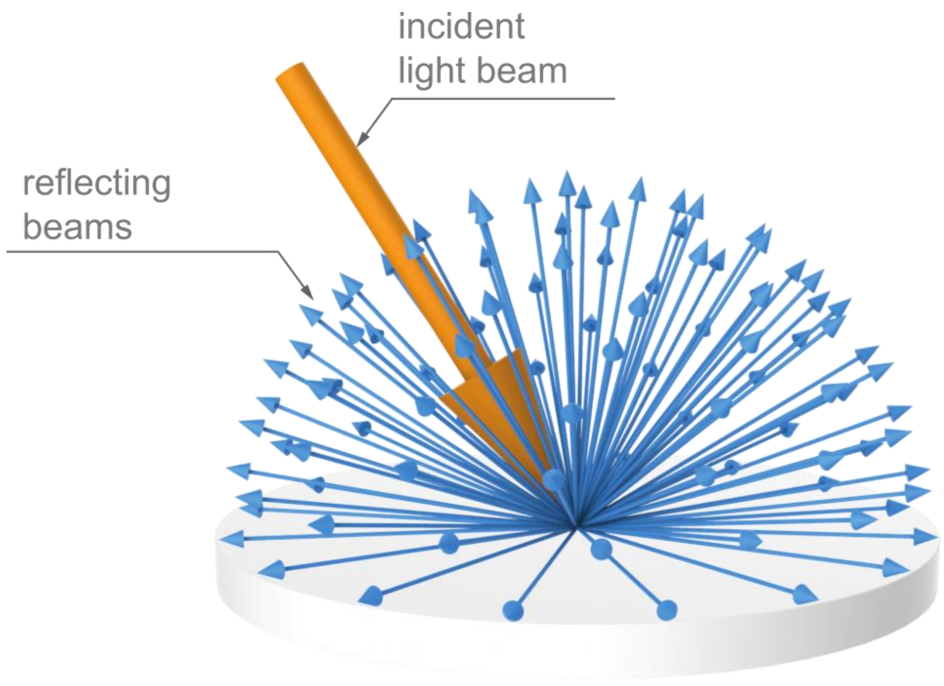
Reflection models so far

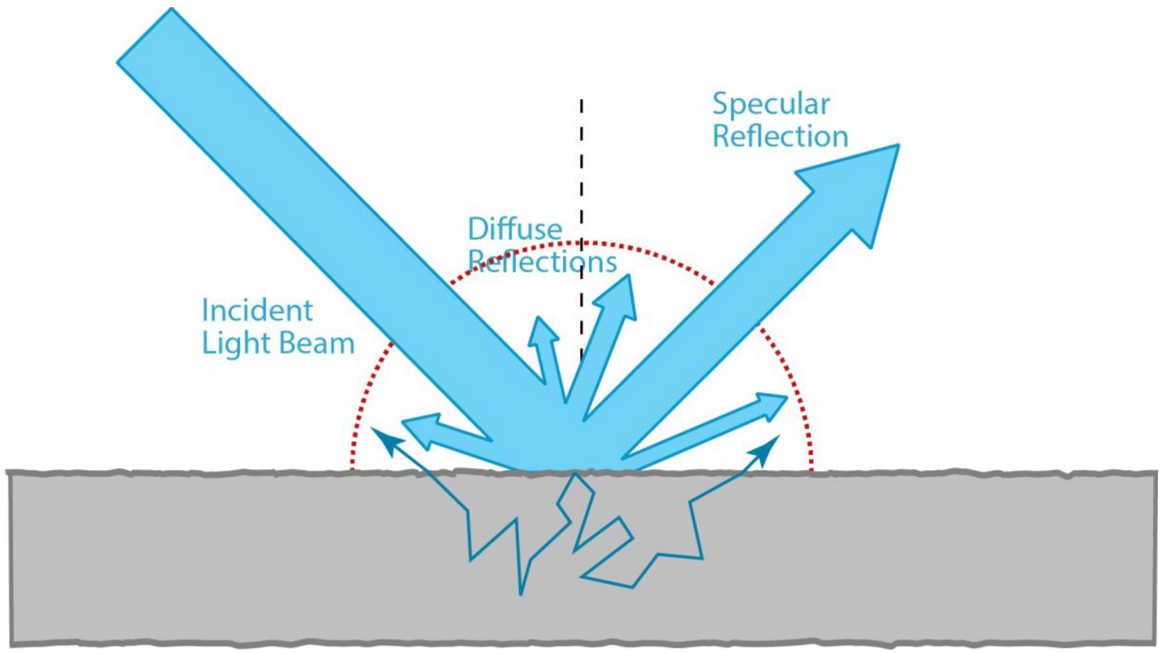




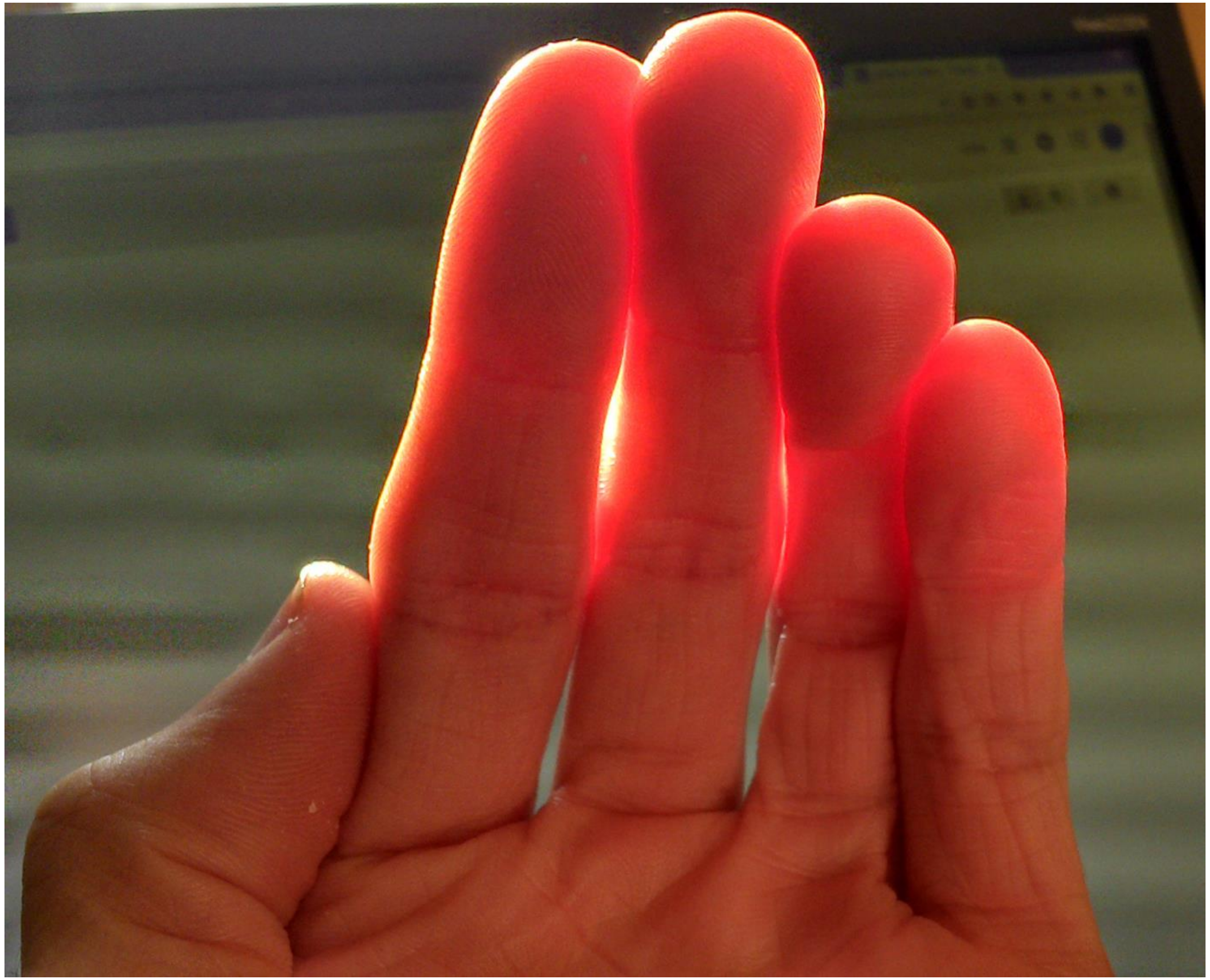


Reflection models so far





Motivation



Motivation

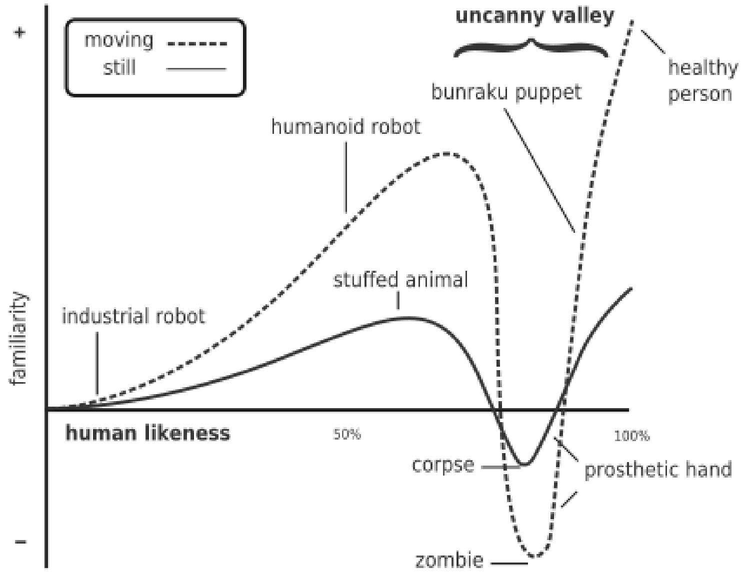
Motivation

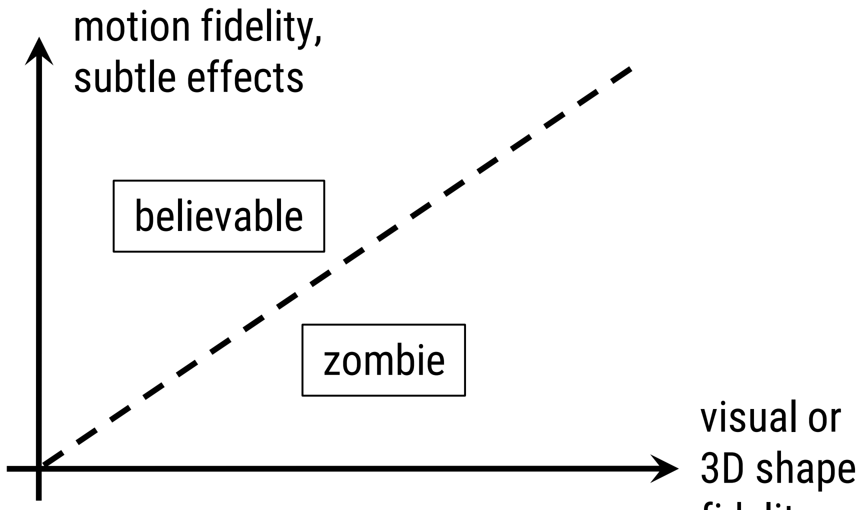
Motivation

Motivation

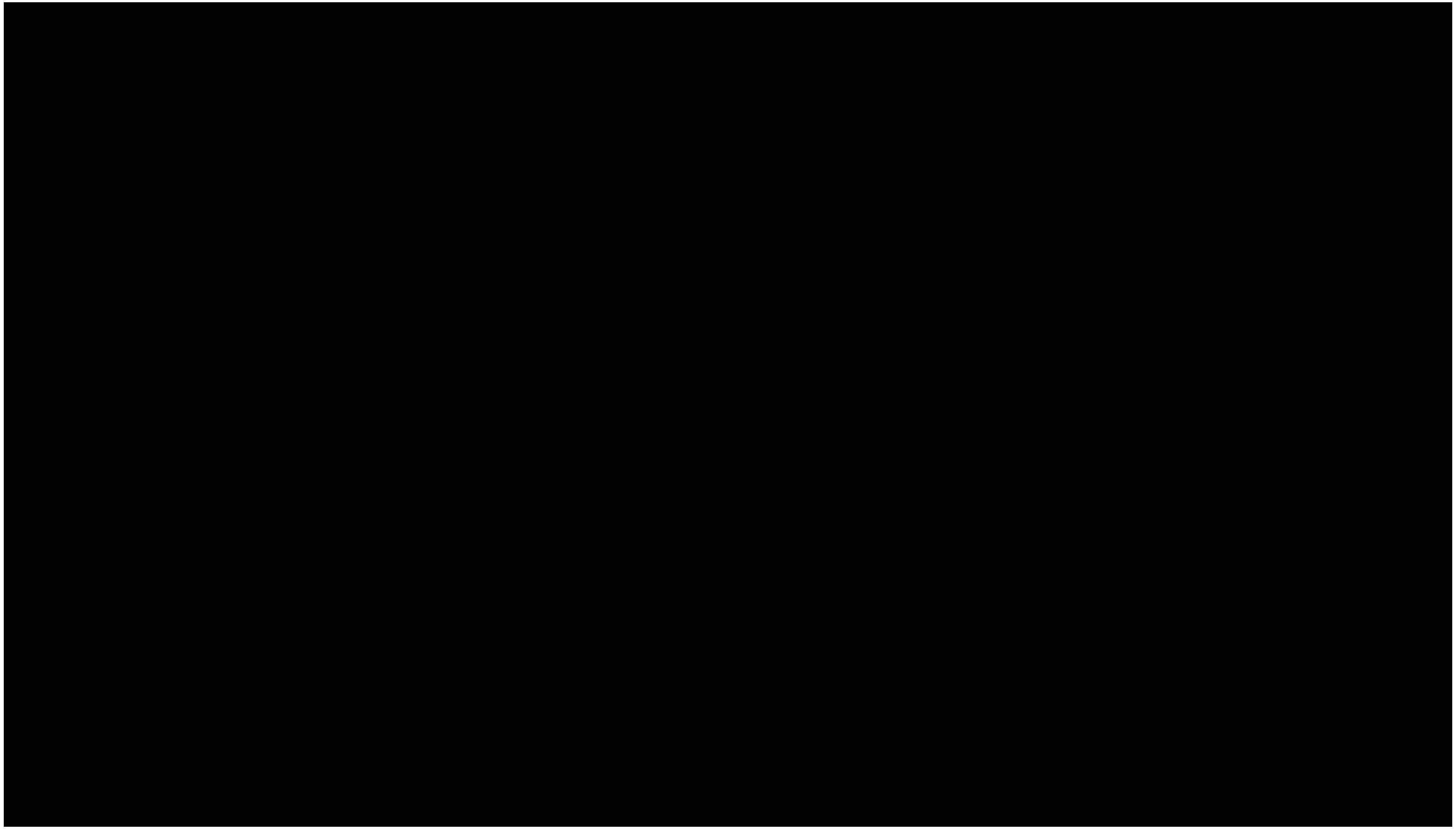
Side note: The Uncanny Valley

- observation: the more “real” humanoid robots or animated characters become, the more they seem to look freaky
- applies to many cases with simulated reality (games, movies, VR, etc.)
- relation of shape, visual, motion, behavioral fidelities: “zombie line”
- **here:** if the appearance of materials does not keep up with the geometry



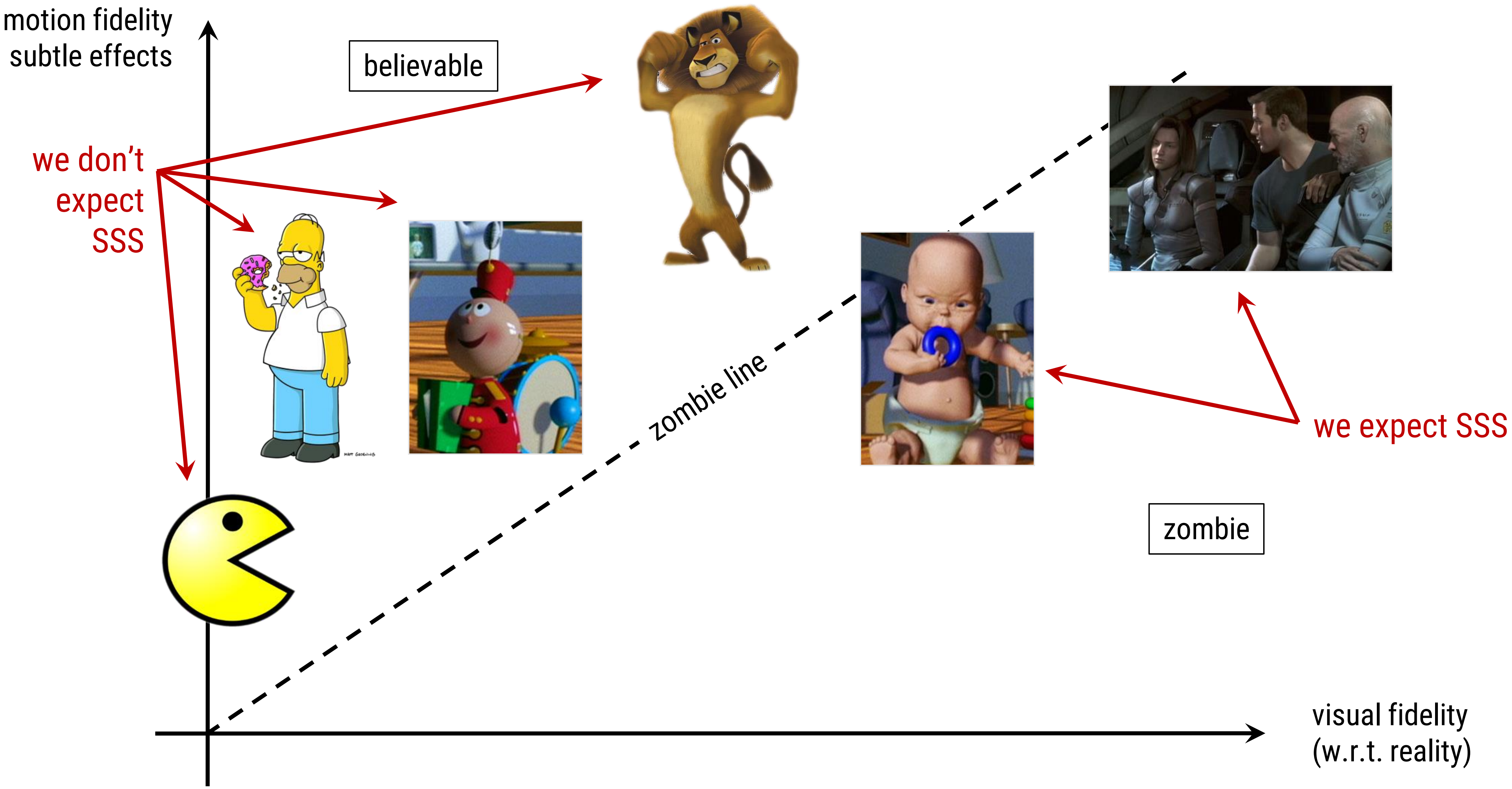


Uncanny Valley Intermission: Tin Toy



PIXAR
ANIMATION STUDIOS

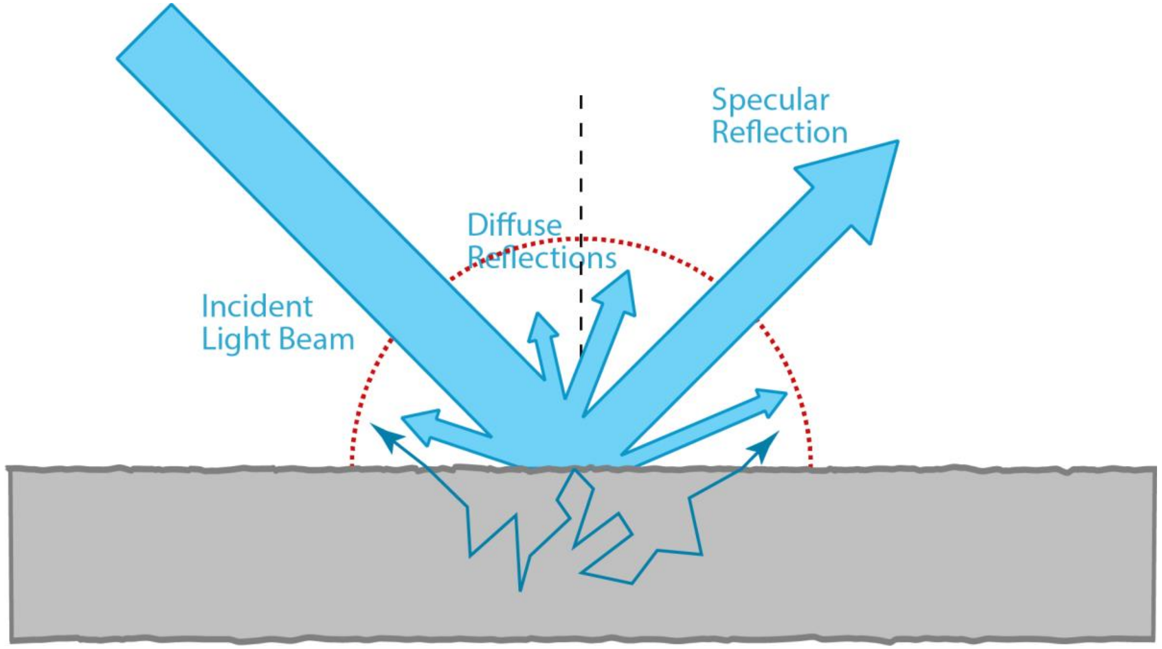
Uncanny Valley/Zombie Line



Sub-surface scattering

- in addition to “normal” reflection
  - light penetrates the surfaces of translucent materials
  - is reflected/refracted multiple times
  - leaves the surface again at a different point

$\rightarrow$ sub-surface scattering



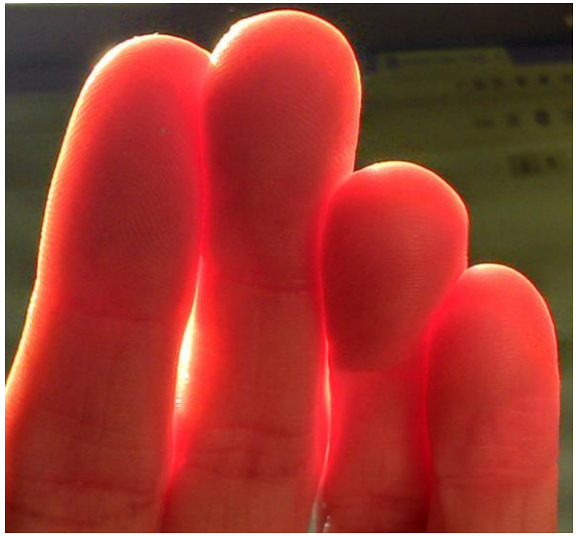
Materials that are affected

- marble
- leaves
- wax
- milk
- fruits
- skin

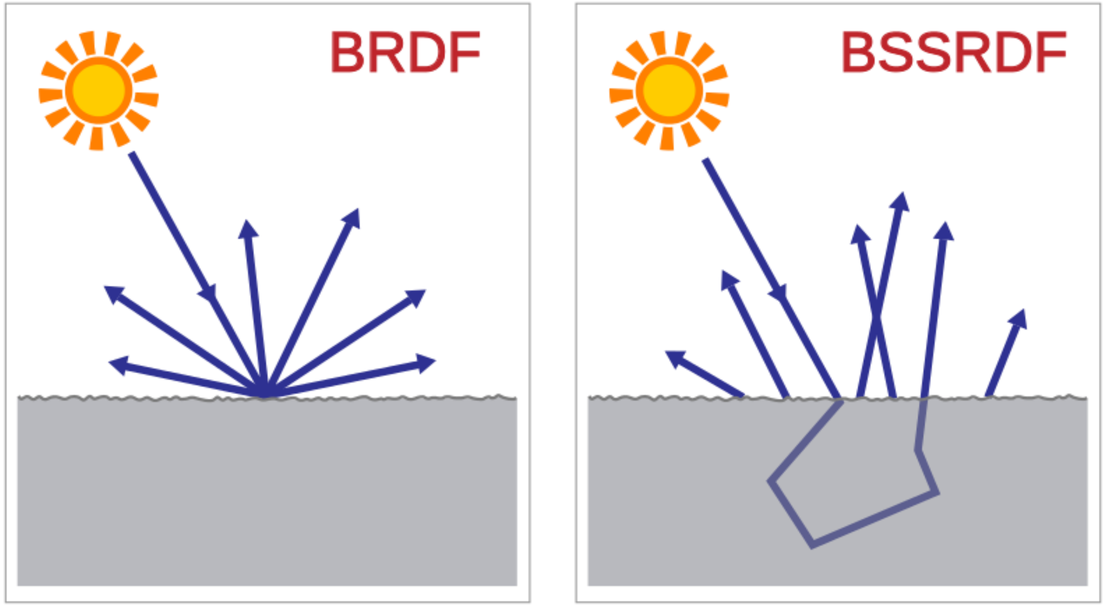
reflectance:
- 6% direct reflection
- 94% sub-surface scattering



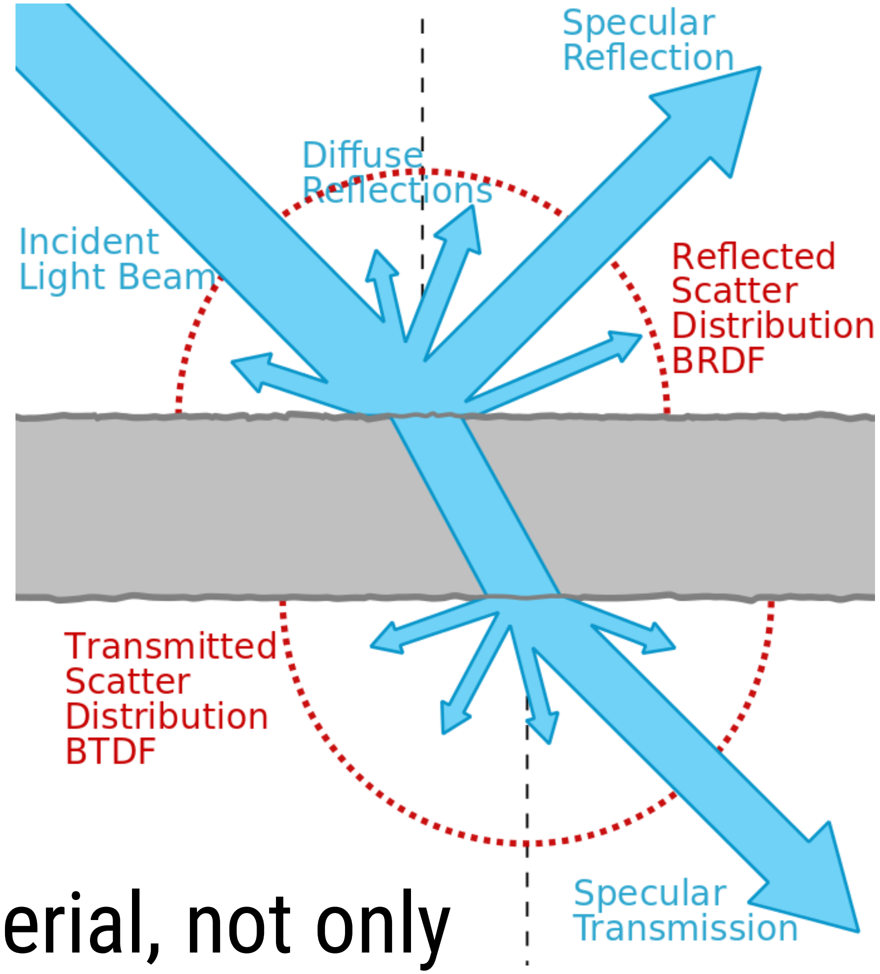





BRDR – BS(S)RDF



Bidirectional Scattering Surface Reflectance Distribution Function



- needs to be measured for each material, not only on reflection point and also for different depths

Results BSSRDF (Raytracer)

skim milk,
whole milk, and
“diffuse” milk

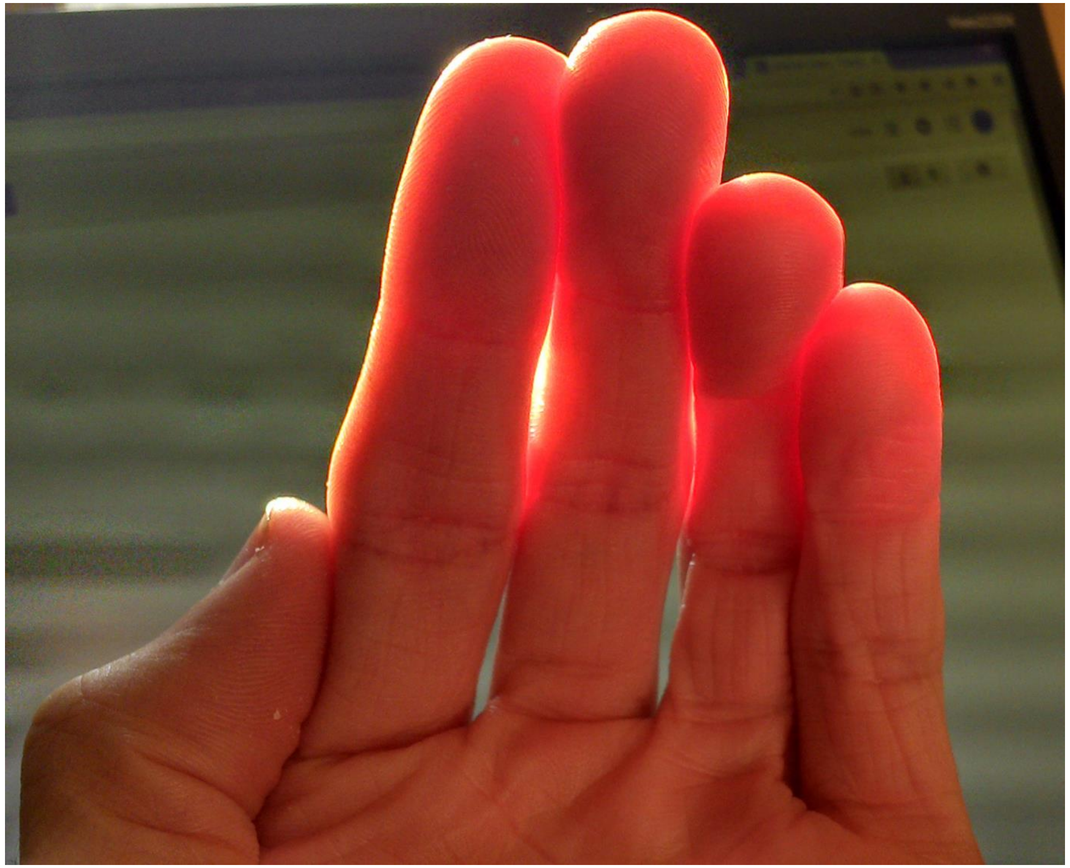
Results BSSRDF (Raytracer)

BRDF

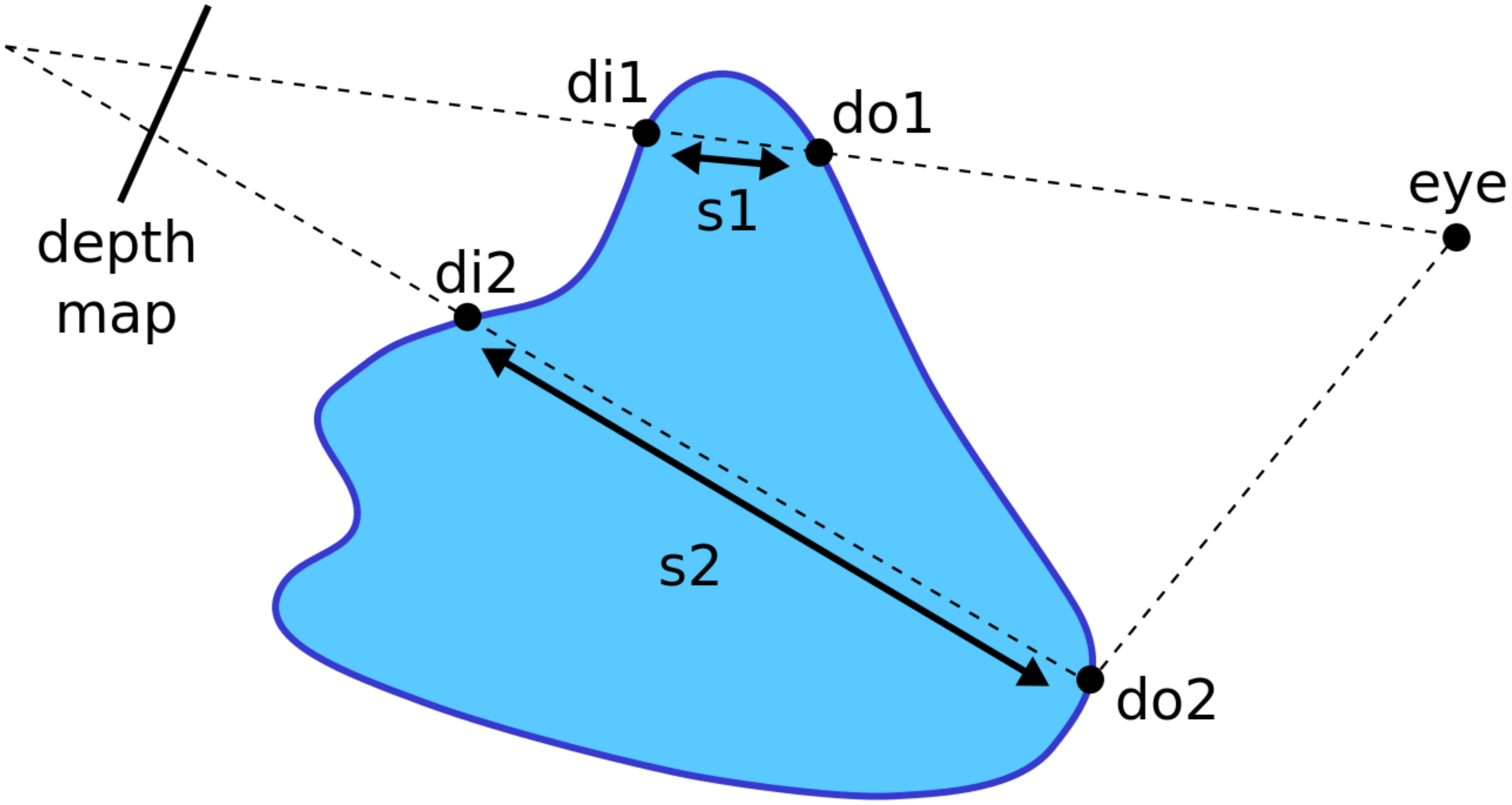
BSSRDF

Observation

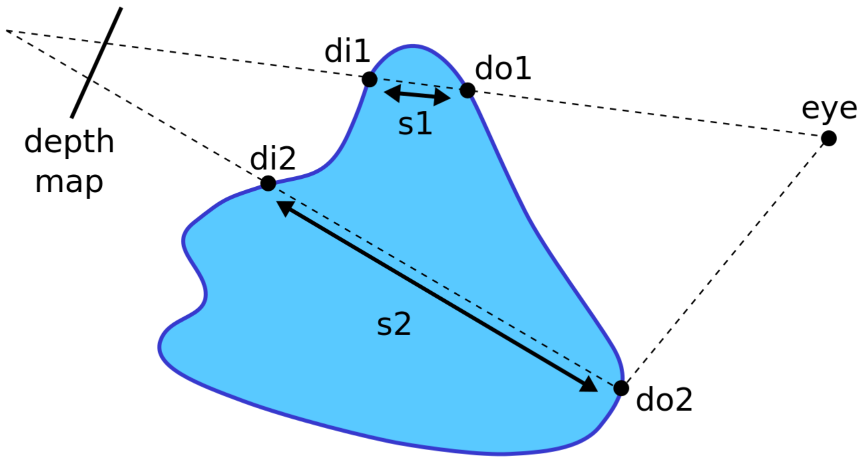
- the thinner an object, the more likely we will see SSS
- idea for SSS:
  - model light absorption based on the thickness of translucent material
  - use z-buffer to record light distances, in a similar way to shadow mapping



z-Buffer-based SSS



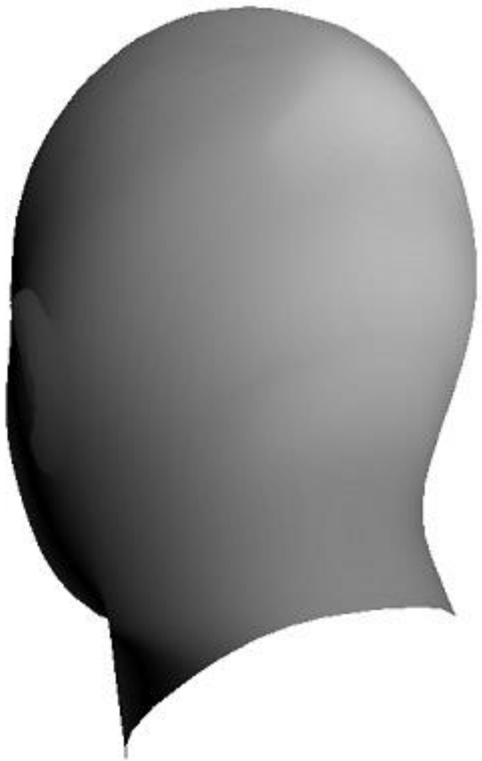
z-Buffer-based SSS



- 1st render pass:
  depth map from the POV of the light
- 2nd render pass:
  find first hit point, unproject, and project with light's matrices, then compute the distance between entry and exit points
- use this computation to modify the local illumination

z-Buffer-based SSS

depth map from light source

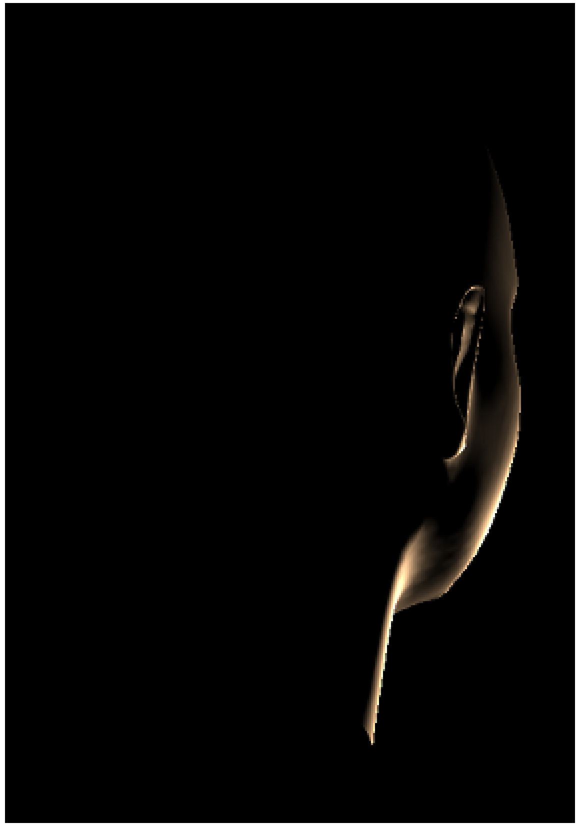


distance traveled by light
from point of entrance in object

resulting SSS,
light behind the object

images: Clément Landrin

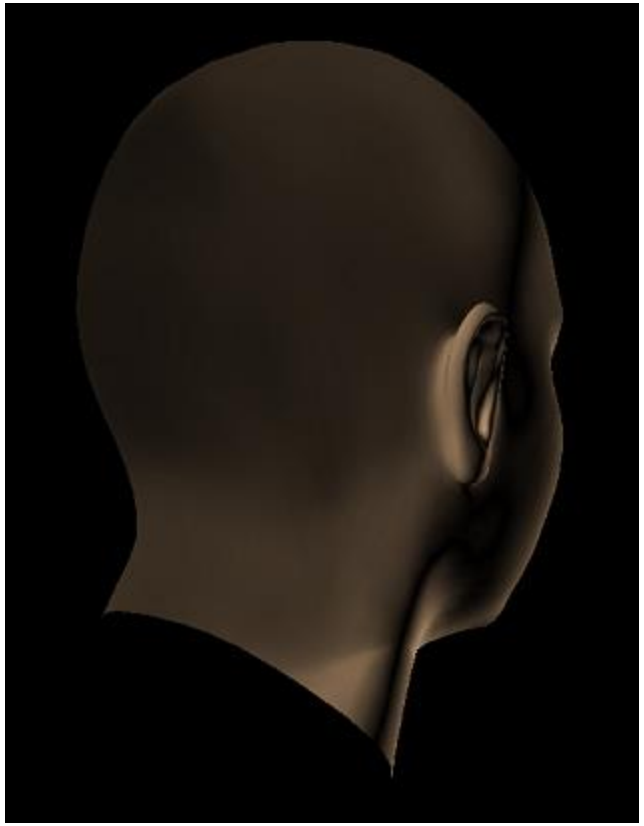
z-Buffer-based SSS



w/o SSS

w/ SSS

only SSS contribution



3× SSS

images: Clément Landrin

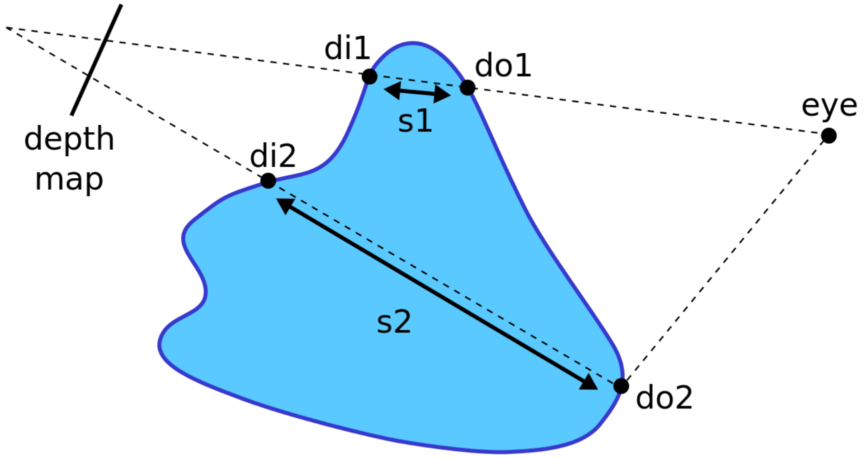
z-Buffer-based SSS

- problem:
  SSS not linear w.r.t. distance

- 1D texture to record fall-off:

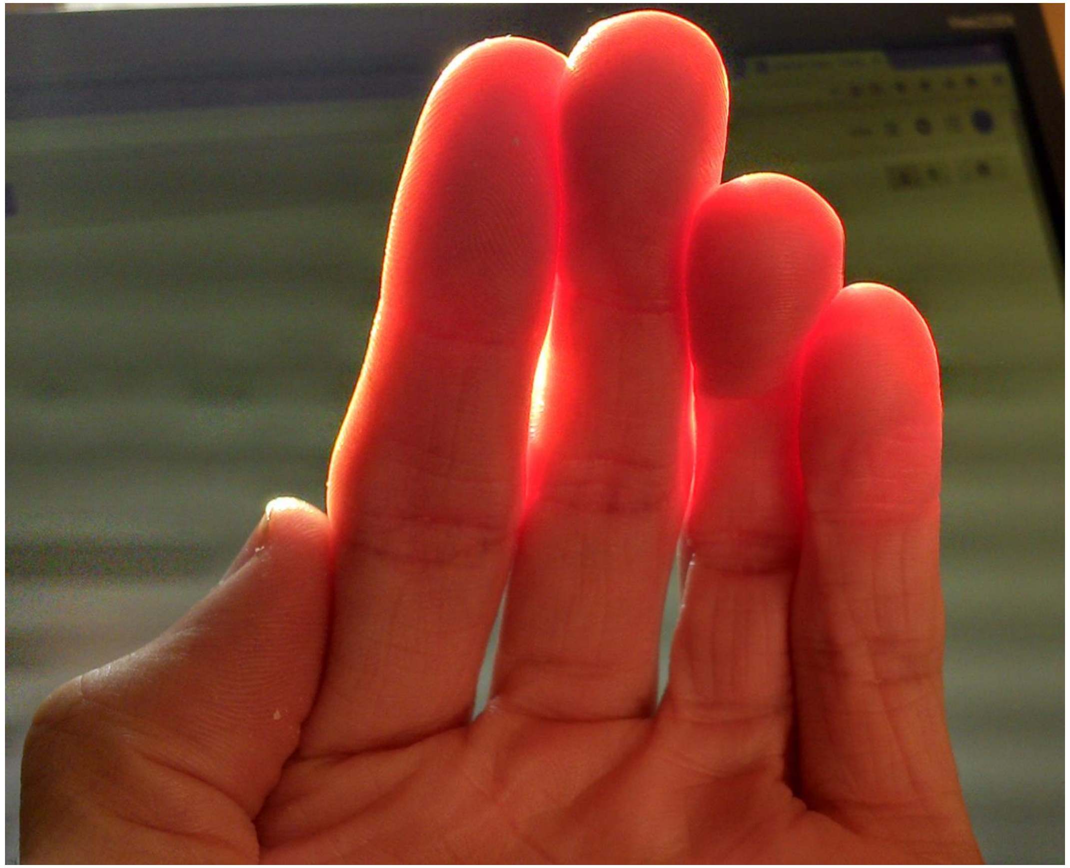
$$e^{(-d\sigma_t)} light\_color$$

  - d: distance through the material
  - σ_t: SSS material constant



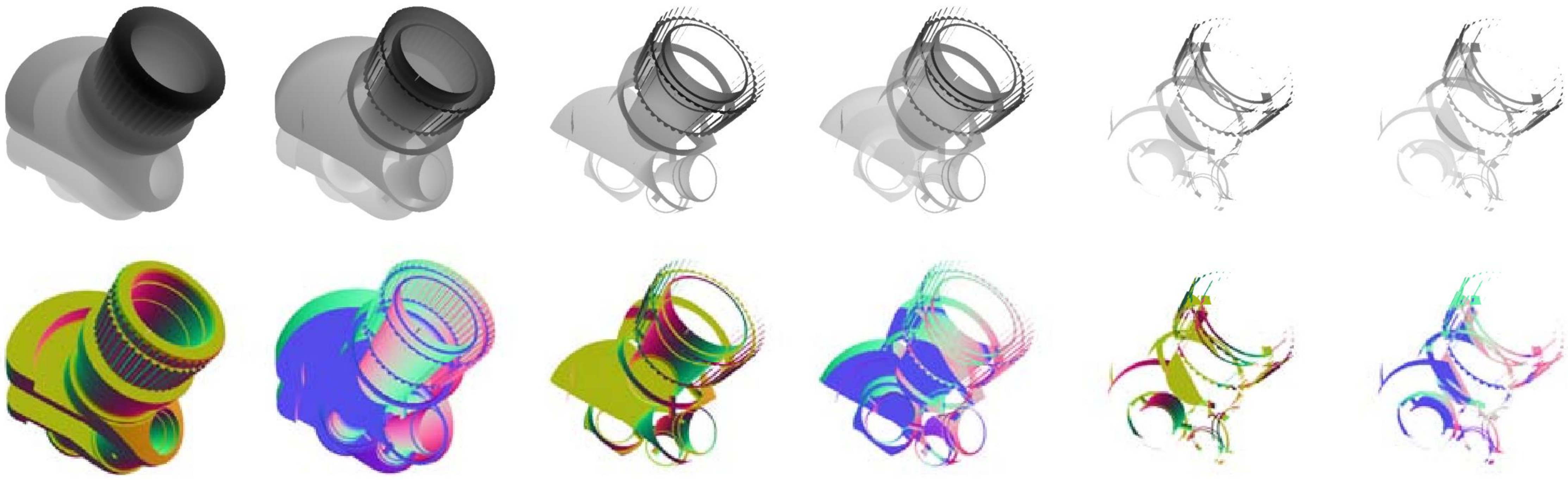
z-Buffer-based SSS: Problems

- 1st assumption: material is homogeneous
- not all materials are:
  - bones – muscles – skin layers
- solution:
  depth peeling



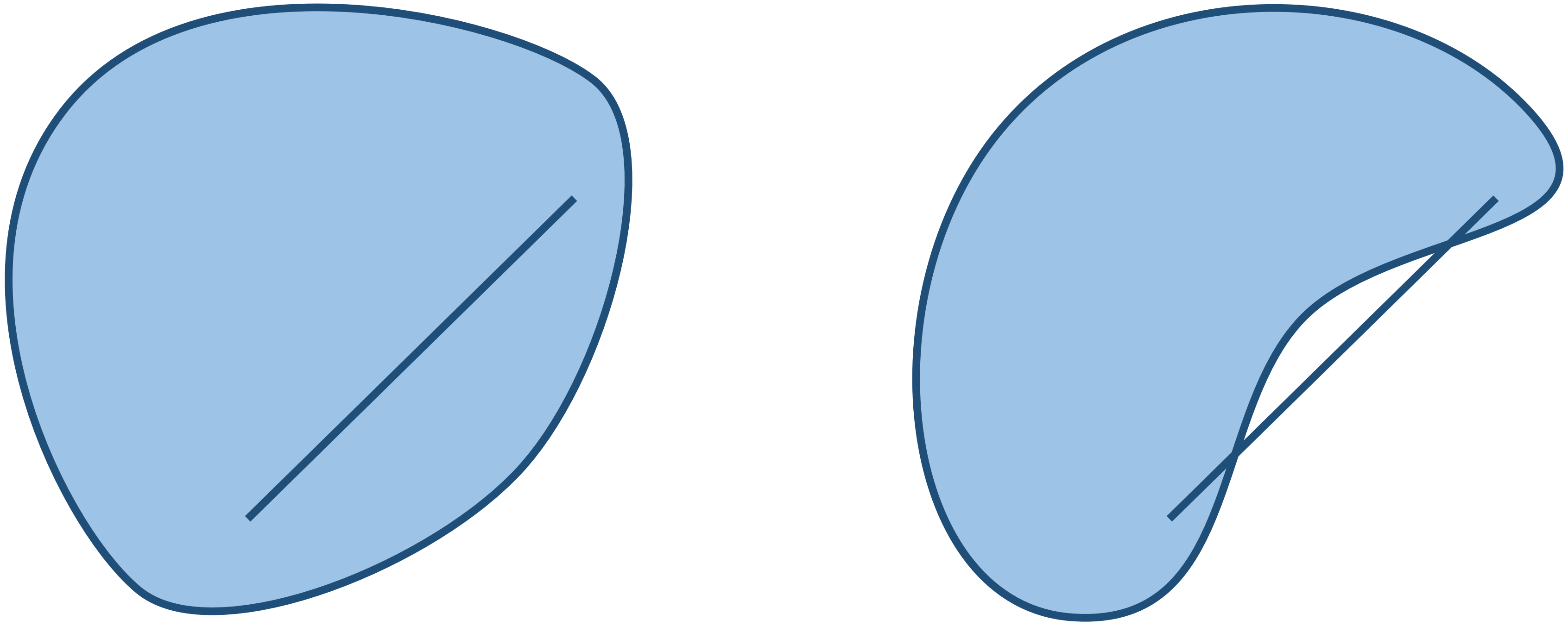
Depth peeling

- technique to remove layer by layer through repeated render passes that compare with previous *z*-buffer



z-Buffer-based SSS: Problems

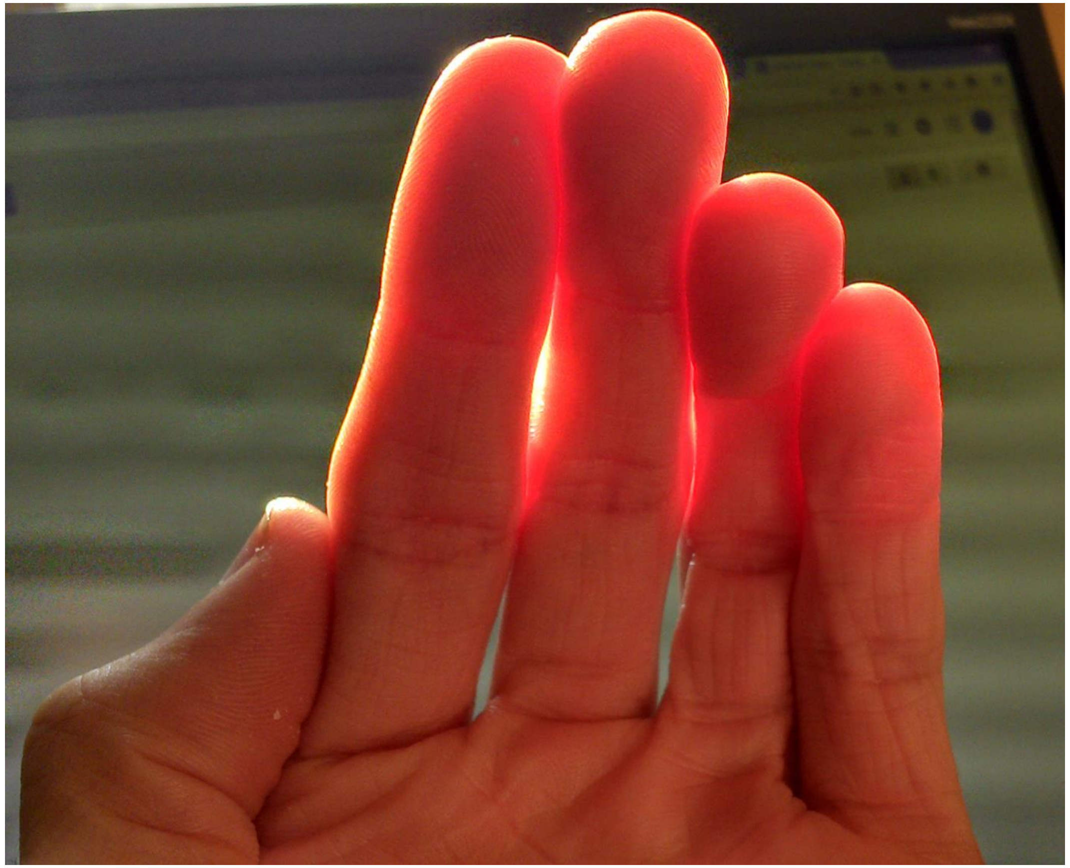
- 2nd assumption: material is convex, but not all shapes are



- solution: depth peeling

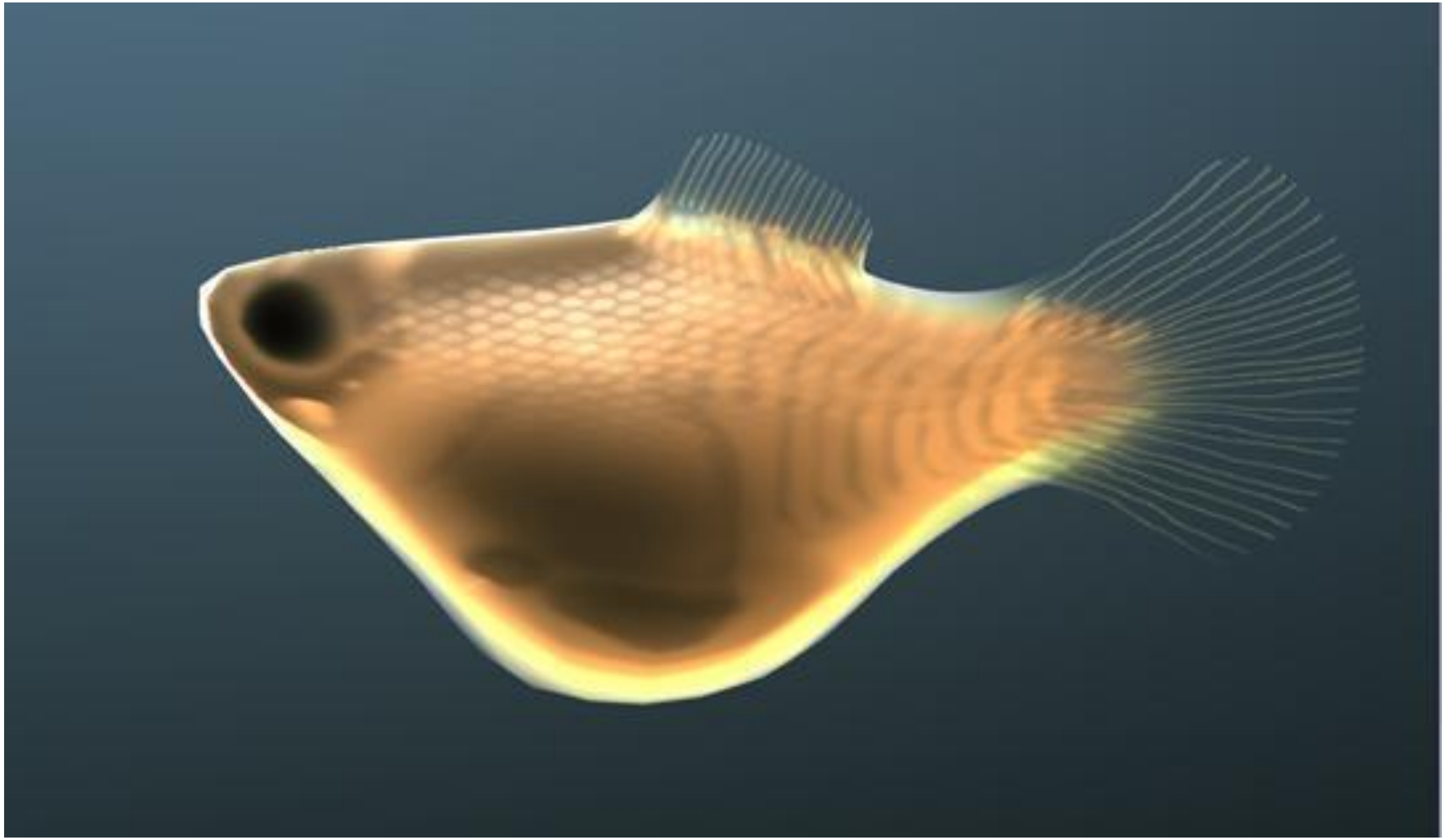
z-Buffer-based SSS: summary

- approach based on observation that thin objects are more likely to exhibit SSS
- similar to shadow mapping
- ideal for shapes illuminated from the back
- assumption that material is homogeneous
- assumption that object is largely convex



Results

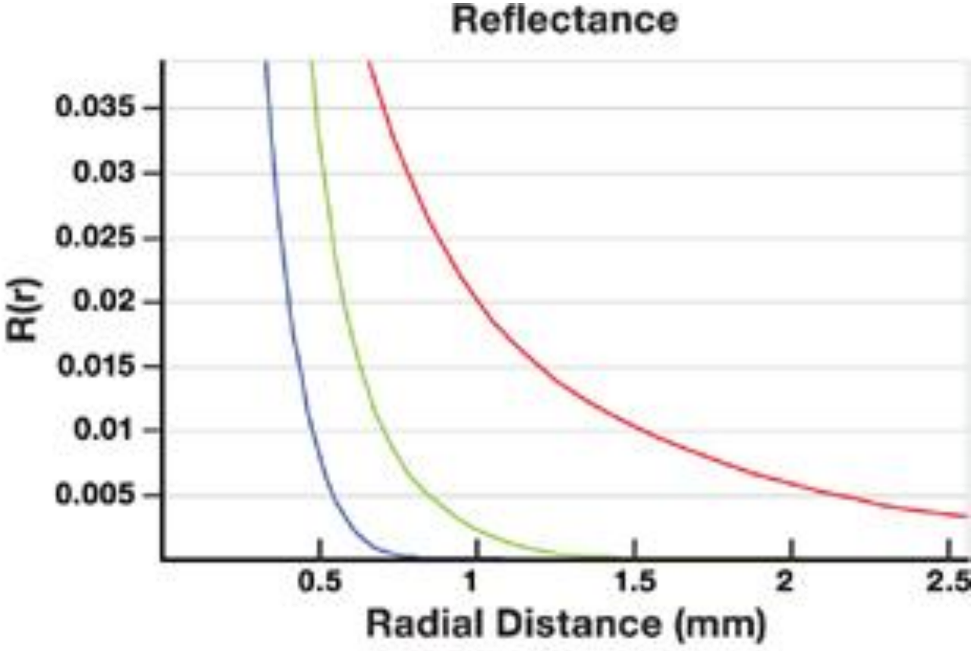
Results



Texture-space diffusion

- better technique for front-illuminated objects?
- observation: SSS generally leads to a blurring of the diffuse reflectance, in particular for skin

(a)



(b)

Texture-space diffusion

- diffusion modeled in texture space
  - vertex shader unwraps the geometry into *u*-*v*-space
  - its diffuse illumination is thus mapped to texture space
  - diffusion in texture space, and storing as light map; used to render

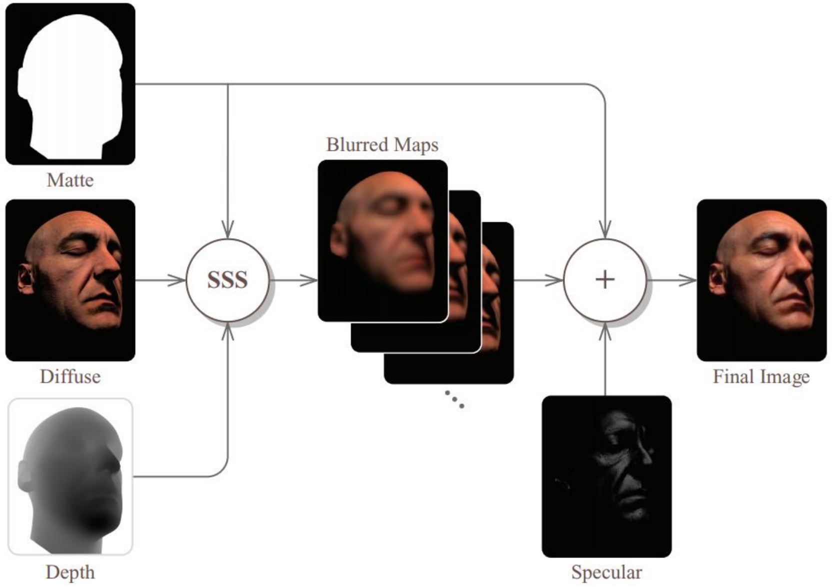
Texture-space diffusion (Matrix Reloaded)

CG

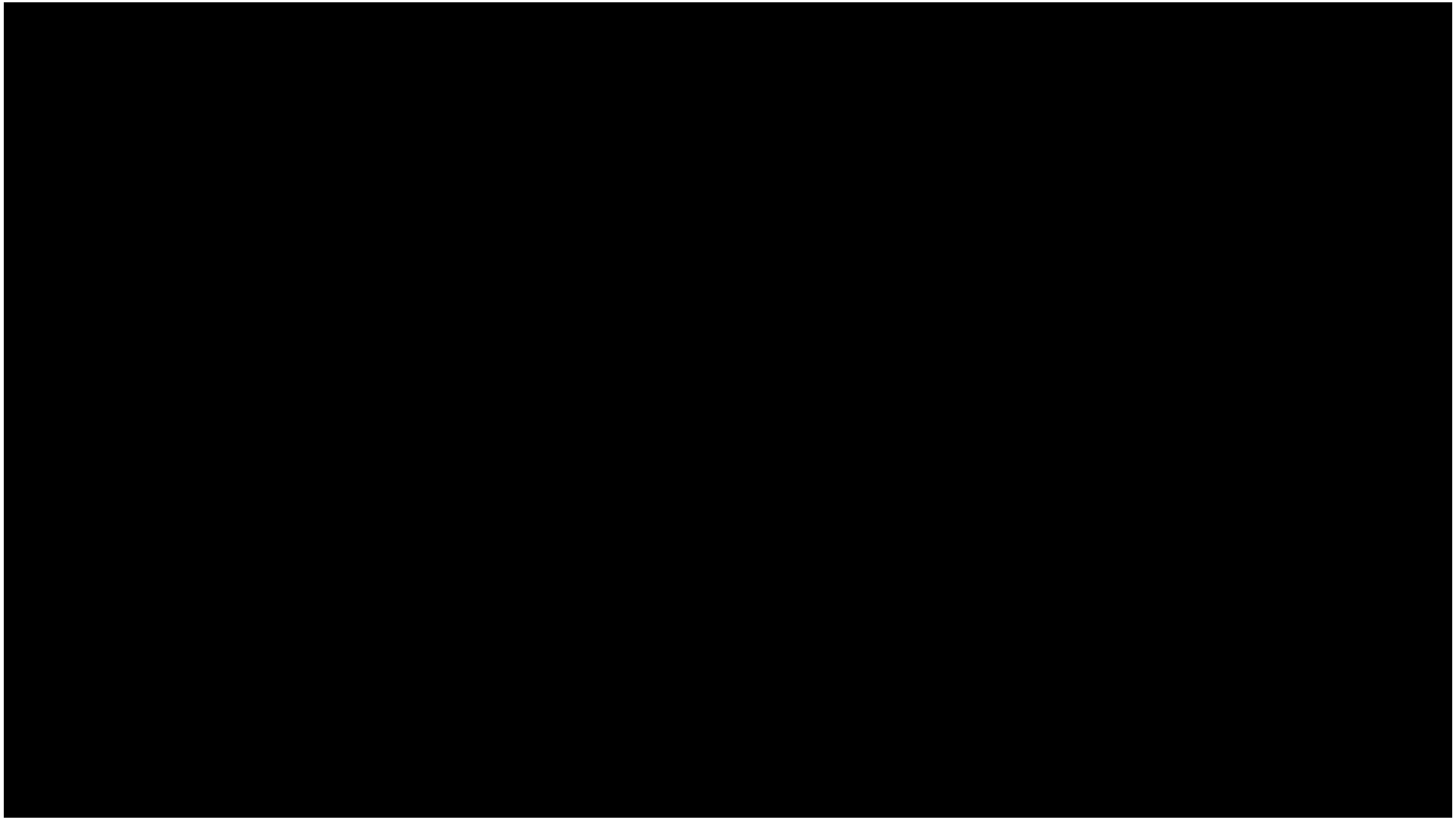
real

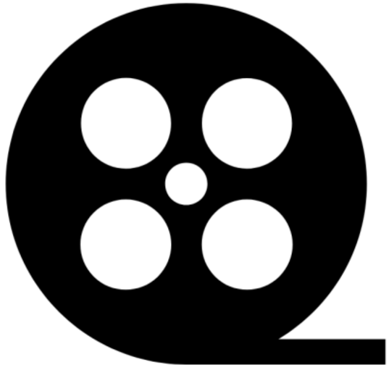
Screen-space subsurface scattering

- problems of texture-space computation:
  - each object needs own texture
  - real-time rendering thus bound by # of objects
- solution: direct computation in screen-space
- computation as a post-process



Screen-space subsurface scattering



Applications of SSS

Application in games: Unreal Engine 3

Application in games: Skyrim

Application in games: Tomb Raider

Application in movies: LOTR & Hobbit

Application in movies: Avatar

Subsurface scattering: Summary

- essential for “photorealistic” rendering
  - many translucent materials, in particular skin
  - considering reflection exclusively on the surface insufficient
  - needed in games and movies
- three approaches
  - BSSRDF
  - based on *z*-buffer
  - based on illumination map diffusion (texture or image space)
- can now be implemented in real-time (pixel shader)
- applications in games and movies