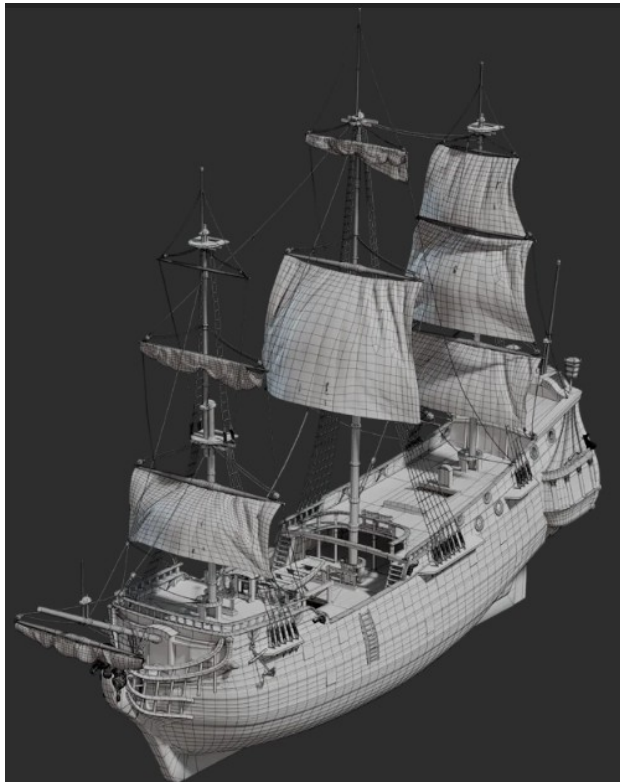
3D Models: Material

Syllabus

- TODO: point where we are

Recap: shape and material

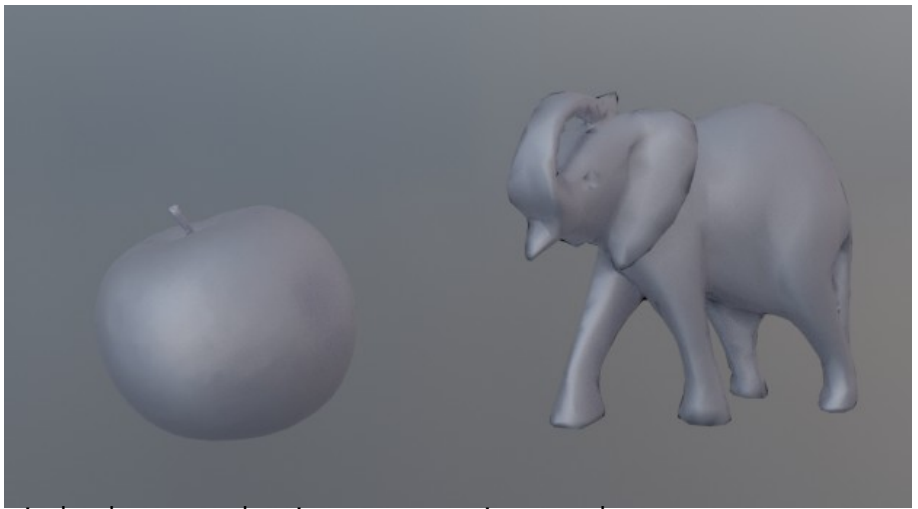
- Rendering and creating objects in 3D scene to represent a real world objects requires:
  - Shape modeling
  - Visual **Appearance** modeling



https://polyhaven.com/a/ship_pinnace

Recap: shape and material

- We separated 3D object in shape and material.
  - Material characteristics of object are independent of shape and position.
  - This enables us to model material separately from shape and its position, since it is enough to model how one tiny bit of material interacts with light, we reuse this model for arbitrary shape.



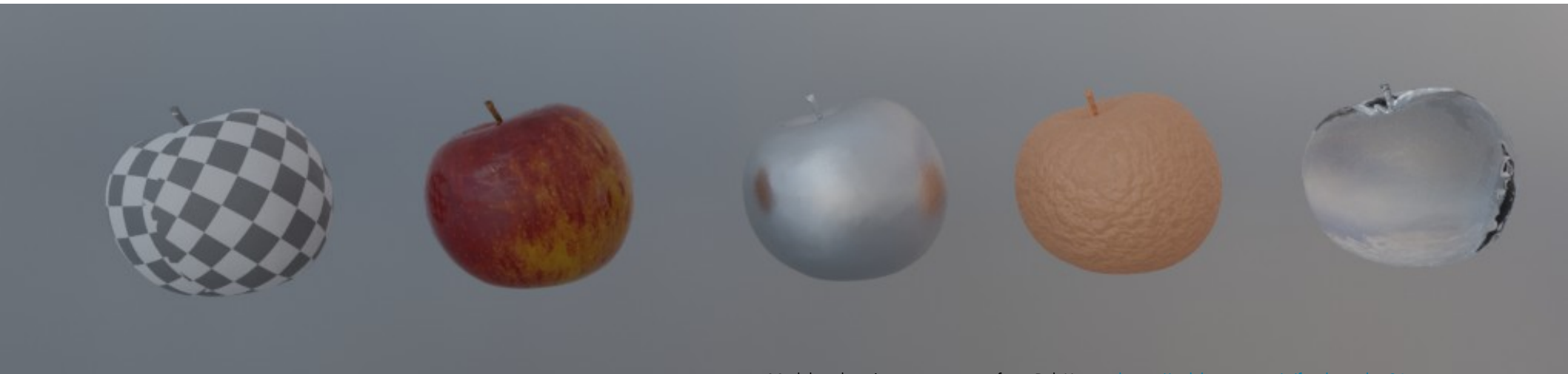
Aluminum apple and aluminum statue – in both cases aluminum properties are the same.
Also, changing position of aluminum apple in space doesn't change aluminum properties.

Appearance of 3D object

Material representation

"Let the form of an object be what it may, - light, shade and perspective will always make it beatuiful" – John Constable

- To takeaway: modeling of shape of an object and material can be separated



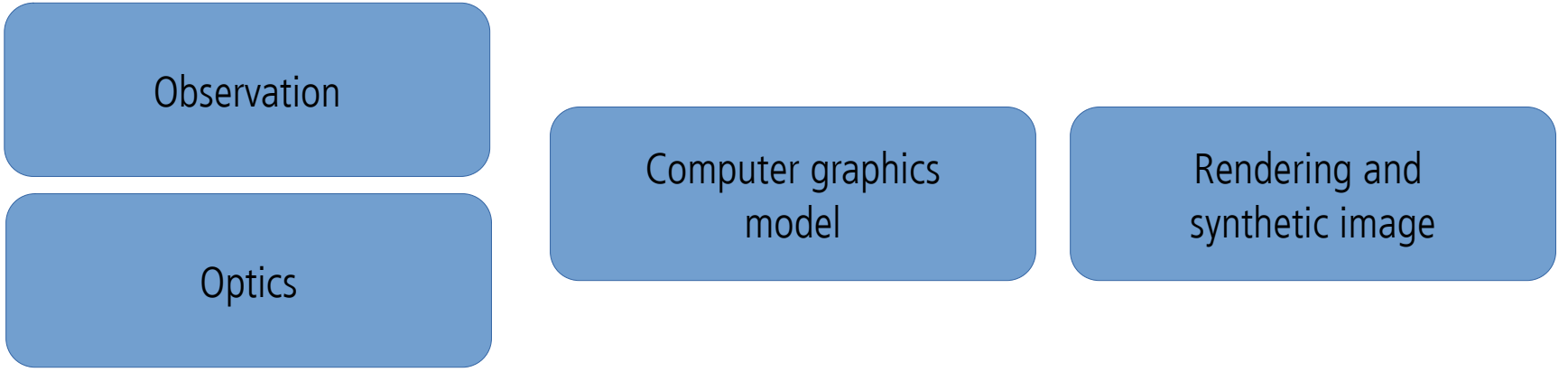
Model and environment texture from PolyHaven: https://polyhaven.com/a/food_apple_01

Real-world phenomena

- Examples of various real-world objects that we would like to represent
  - Metals
  - Plastics
  - Glass
  - Wood
  - Fabric
  - Stone
  - Clouds
  - seas

Prior to appearance modeling

- Two main disciplines influence modeling:
  - Observation
  - Physics - optics



Understanding appearance

Modeling begins with **observation**

Observation

- Understand **what makes each material look different** than other materials
- Observe **characteristics which are responsible for object appearance**:
  - Shape: large scale form or geometry of object. Shape is needed to place object correctly in the scene with respect to other objects, to determine which objects are occluded and areas into which shadow is cast by object
  - Material: for computer graphics modeling purposes this are fine-scale geometrical variations and substance properties
  - Illumination
  - Sensor/Perception
  - **Material**

<EXAMPLES: CHARACTERISTICS RESPONSIBLE FOR APPEARANCE>

Note: real world and models

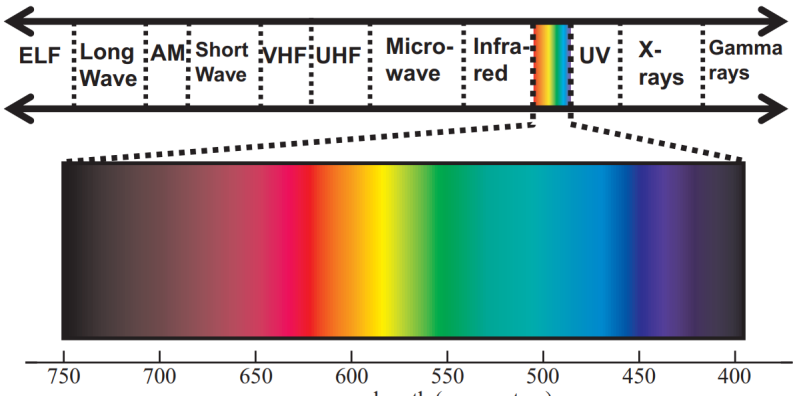
- Real world objects are very complex phenomena which have to be simplified using a **model** for desired application.
- Best description of real world is physics which again describes a world using models – simplifications
  - Geometric optics
- Computer graphics takes step further into simplification for creation and computational purposes
  - Separation of objects into shape and material
- Finally, artists who create based on real world use their perception – subjective model of the world – they draw what they see, not necessary what is true and correct.

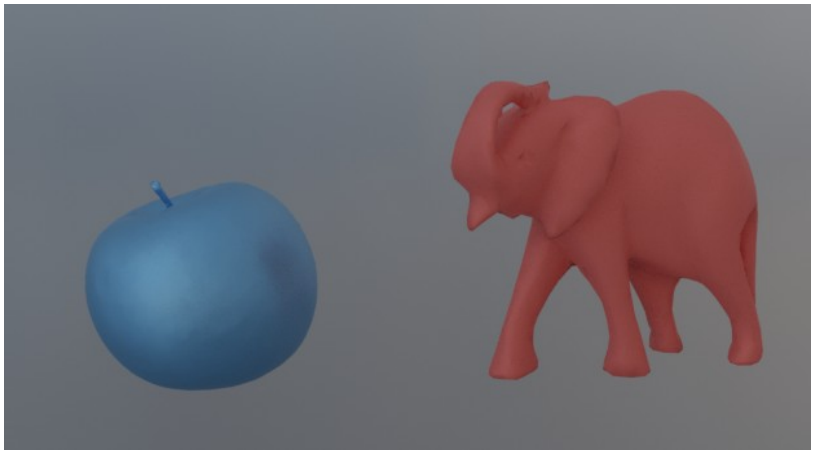
Classifying materials

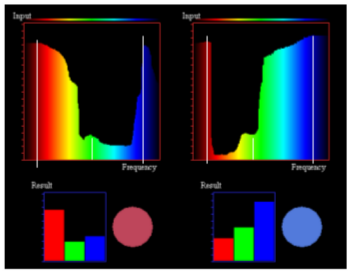
- Classifying materials enables us to **understand which characteristics are needed to be modeled** in order to obtain required appearance. We can classify any material by following variations:
  - Spectral: color
  - Directional
  - Spatial

Material characteristics: color

- Light is a electromagnetic wave, thus characterized by wavelength. Small part of wavelengths are interesting for material modeling
- Light scattering from object surface (its material) is described using color and brightness
  - Applications concerned with color reproduction: photography, print and television
- Color is described with red, green and blue floating points (R, G, B) in [0,1]
- Brightness is floating point value [0, inf]



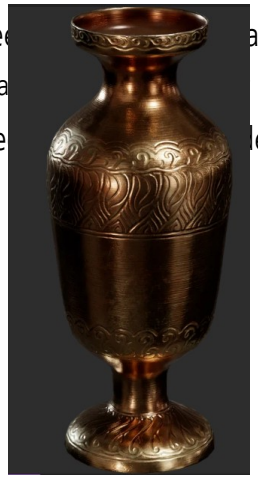




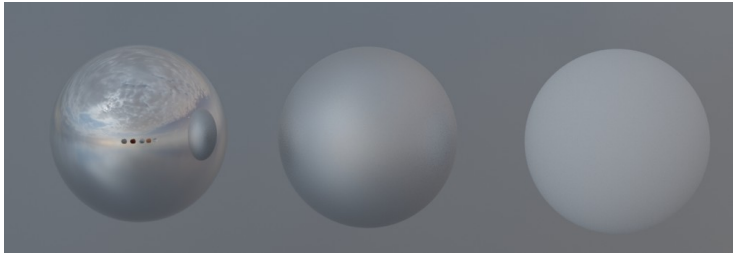
https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf
https://www.researchgate.net/publication/234814482_Digital_modeling_of_the_appearance_of_materials

Material characteristics: directional

- Object appearance that results from **directional of scattering of light** by the object
- **Directional effects:** whether we se as we change view
  - Those effects are due to structures a htan the visible to the eye, can be observes sider flat by looking at it)
- Directional effects that are attache lepen
  - Shiny
  - Hazy
  - Glossy
  - Matte
  - Dull
  - Translucent
  - Transparent



- Directional effects are modeled using **scattering function.** There are many perceptual dimensions of material types. Most important light scattering models are:
  - Specular (mirror direction)
  - Glossy (preferred direction)
  - Diffuse (all directions)

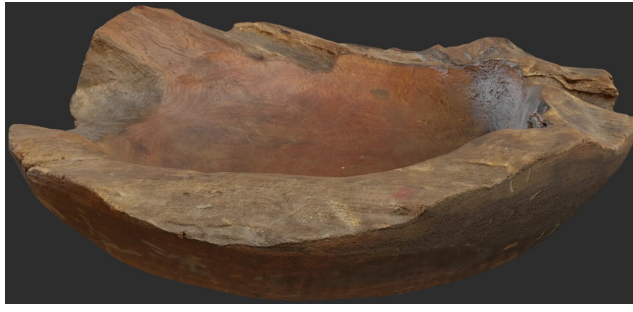


Material characteristics: texture

- Visual variations on the object surface, much smaller scale than size of the object but larger than the wavelength of light - **spatial variations**:
  - Directional or spectral (color)
  - Small scale geometric variations: bumps and pores
- Surfaces with spatial variation are observed as non-uniform: **texture or pattern** can be observed

Color variation

Directional variation



Small scale geometrical variation

Understanding appearance

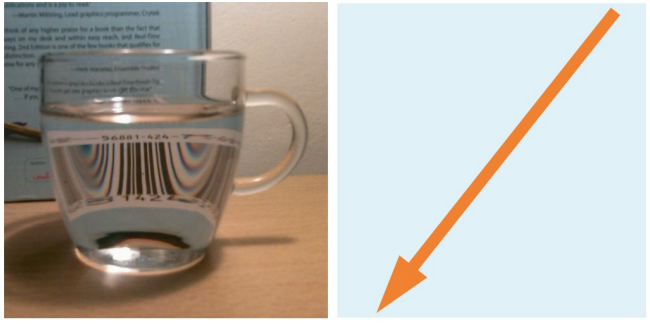
Modeling is founded on **optics**

Light-matter interaction

- Effect that material has on light is defined by property called **index of refraction** (IOR) – complex number
  - Light is electromagnetic wave.
  - We use **geometrical optics** model which approximates light as rays and materials with index of refraction:
    - Real part of IOR determines speed of light
    - Imaginary part if IOR determines absorption of light

Light-matter interaction

- Simplest light-matter interaction is light propagating through **homogeneous medium**.
  - Uniform IOR: light travels on a straight line, it can only be **absorbed**, meaning that direction is same, but intensity might be attenuated
  - Examples: transparent materials; glass or water.



Transparent media: straight line with same intensity (scattering only happens when light crosses air-glass-water homogeneous medias since they have different (but constant) IORs.

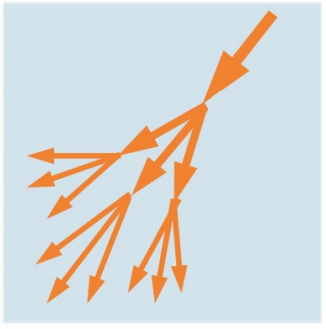
Clear absorbent media: straight line with loss of intensity – selective absorption (color is changed)

Slight absorption becomes significant with distance

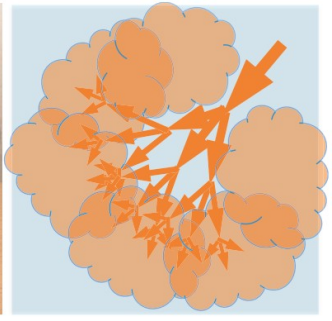
https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf

Light-matter interaction

- In **Heterogeneous medium** IOR varies which cause **light scattering**
  - Direction of light is changed but not the intensity
  - Light can scatter in all directions, mostly non-uniformly: forward or back scattering (in or reverse of incoming direction)



Microscopic particles cause varying of IOR and light to scatter continuously in all possible directions.



Translucent or **opaque materials** → light is scattered so much that we can not see (clearly) through the object

Longer distances cause more scattering (e.g., clean air)

https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf

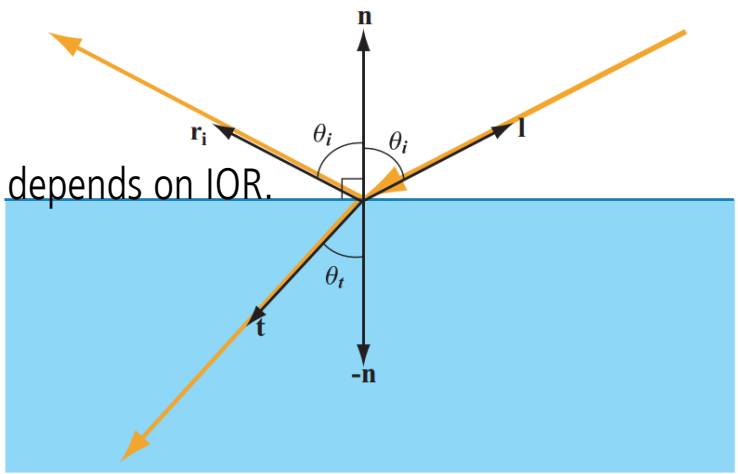
Light-matter interaction: scattering and absorption

- Light traveling through medium, based on **index of refraction** will:
  - **Absorb**
  - **Scatter**
- **Appearance depends on both scattring and absorption**

https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf

Light-matter interaction: between media

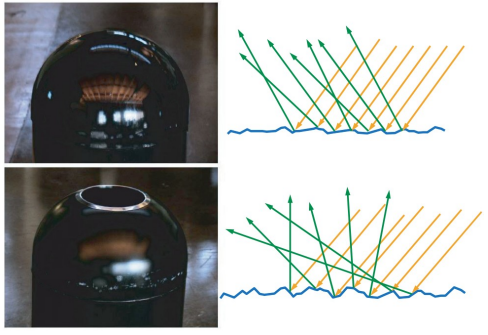
- We have discussed how **light propagates through a medium**.
- Behavior of light when **propagating between media of different index or refraction** is defined with Maxwell's equations
- Those equations are too heavy for computer graphics thus, we:
  - Utilize geometrical optics approximations: rays light representation and IOR
  - Assume that interface between (volumes) media is perfectly (optically) flat, planar boundary* → **object surface**
  - On the one side is IOR of air and on the other IOR of object
- Solution for such surface is called **Fresnel equations**
- For such surfaces, light can **reflect** and **refract** where amount and direction depends on IOR.



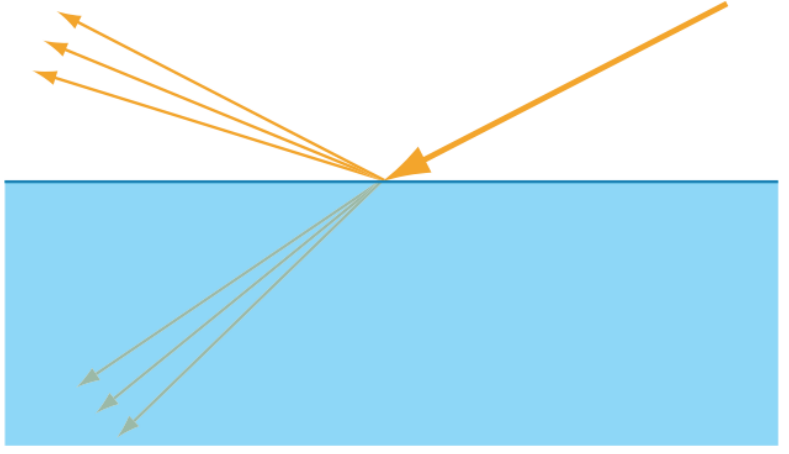
* Surface should be infinetly large, but in comparison with wavelenght of light, surface real objects can be considered as such.

Light-matter interaction: reflection

- Real surfaces are not optically flat.
- Often, irregularities are present – larger than wavelength (causing light to reflect differently) and too small to render since this interaction happens under one pixel.
- In this case, we model such surface as a large collection of tiny optically flat surfaces – facets. Final appearance is aggregate result of relevant facets.
  - Smaller deviation of those facets cause more **mirror-like surface reflection** (small roughness)
  - Larger deviation of those faces causes more **blurred surface reflection** (glossy, high roughness)



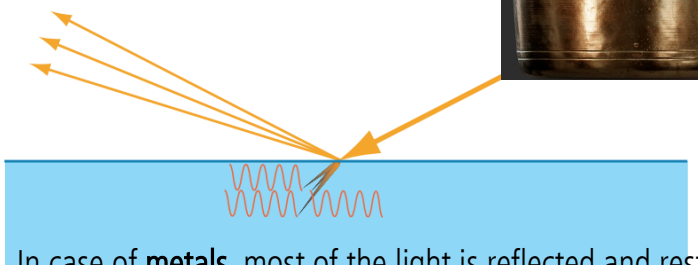
Both surfaces look smooth but surface on bottom is rougher. Roughness difference is on microscopic scale - **microgeometry**.



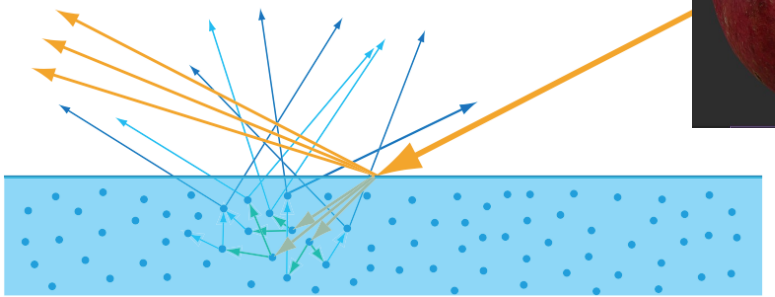
Macroscopically, non-optically flat surface can be treated as reflecting with multiple directions

Light-matter interaction: refraction

- Besides of reflection on surface, light can also **refract**.
- Amount and direction of refracted light depends on material which we can separate in:
  - **Metals**
  - **Dielectrics**



In case of **metals**, most of the light is reflected and rest is immediately absorbed. That is why mirrors are made using metal foundation. Conductors are spectrally selective and thus reflection color may vary

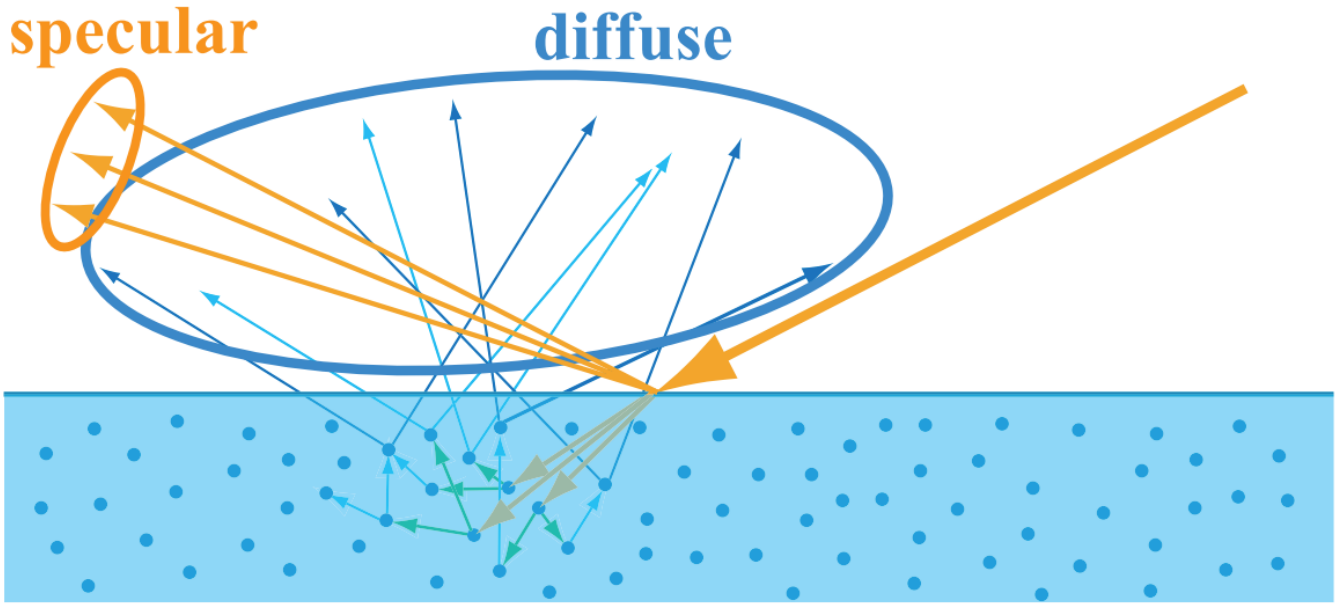


In case of **dielectrics**, light partially reflects and partially refracts. Refracted light is then absorbed and scattered inside surface (**sub-surface scattering (SSS)***). Some of the light can also be re-emitted – causing **diffuse** reflection.

* Note that in this case, light is not exiting from the same point where it has entered, thus we need more than local information for calculating light behavior. Therefore, this is one of more complex effects that require advanced rendering methods or approximation methods.

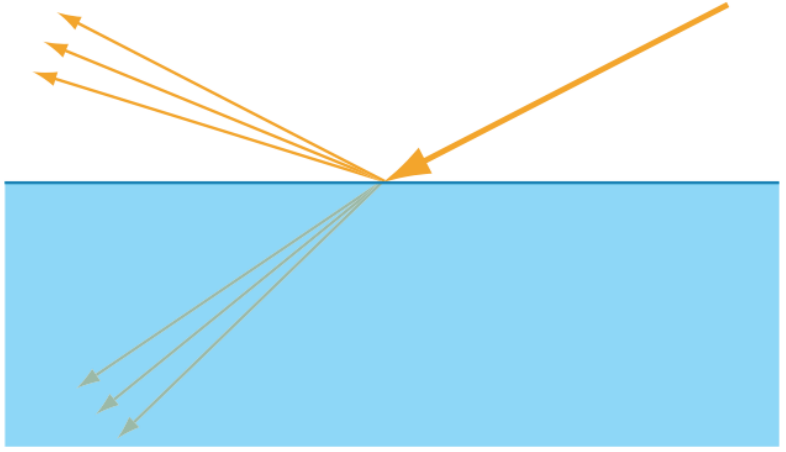
Diffuse and specular reflection

- With these considerations we can understand two foundamental light scattering processes:
  - Diffuse
  - Specular

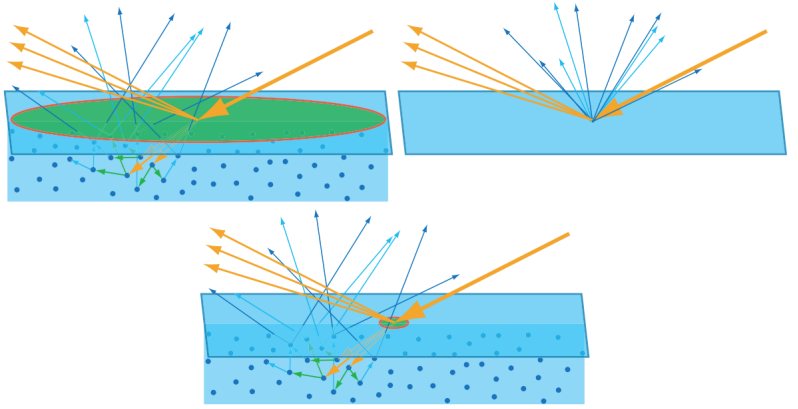


Local models and complexity of SSS

- We will discuss local models: describing what happens when light falls on one point
- SSS is a complex scattering phenomena requiring global model which we will approximate with local model.



We will be focusing on local models which describes what happens at one point/pixel



SSS can be approximated with local model: diffuse scattering.

- With physical considerations we have looked at small scale and gave foundations to:
  - Directional effects and scattering
  - Color
- Larger scale variations visible by eye are simulated by varying scattering model

Modeling appearance

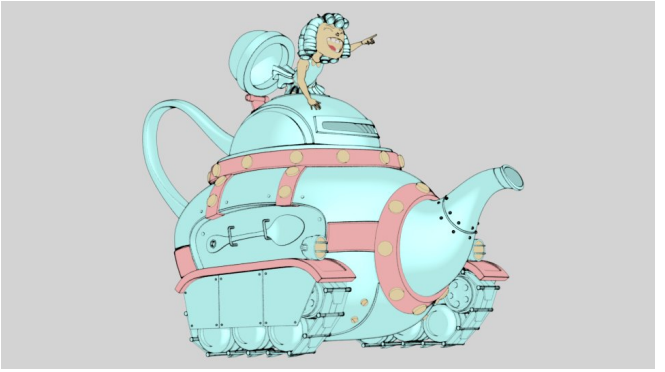
Towards material modeling

- With material in CG we describe light-matter interaction - **shading**
- Material in CG can is decomposed to describe:
  - Color
  - Directional effects
  - Spatial variation:
    - Of color and directional effects
    - **Small-scale geometrical information** that is too small to represent explicitly by geometry/shape.

Diversity of appearance models

- Appearance models, depending on application, can range:
  - Photorealism
  - Stylized
- Note: Stylized is based on photoreal with exaggeration

https://www.artstation.com/artwork/rANRe5



"Rolling Teapot" - Model by Brice Laville, concept by Tom Robinson, render by Esteban Tovagliari - RenderMan 'Rolling Teapot' Art Challenge: https://appleseedhq.net/gallery.html#https%3A%2F%2Fappleseedhq.net%2Fimg%2Frenders%2Frolling-teapot.jpg

Materials and rendering

- Appearance in real world depends on material, shape, illumination and camera position
- Appearance of simulated object also depends on material description
- Shading step in rendering uses material model, alongside with viewing position, object shape and light to calculate the object color
- Final color of surface is calculated by:
  - summing all incoming light that falls onto surface – light transport
  - Calculating color and intensity of light reflected into camera using material description - shading
  - <comparison>

Scattering models: intuition

- Rendering comes down to computing **intensity** and **color** of light – RGB values* - entering the camera along set of **viewing rays.**
- Viewing rays are intersecting objects in the scene. Thus, intensity and color of viewing ray comes from **closest intersected object surface**
  - We will ignore **participating media** for now – medium between objects
- Therefore, our goal is to **compute intensity and color of the surface intersected by viewing ray**
  - <image of big picture: camera object and light>
- For this purposes we use **scattering function.**

* Physically-based color and intensity of light is called radiance but we work with RGB rendering which is simplification of spectral rendering

Material modeling: scattering

- Material modeling describe how light behaves when interacts with objects:
  - Light-surface interaction
  - Light-volume interaction
- Description of light interaction with surface, when considered locally, is called scattering.

Material modeling: texture

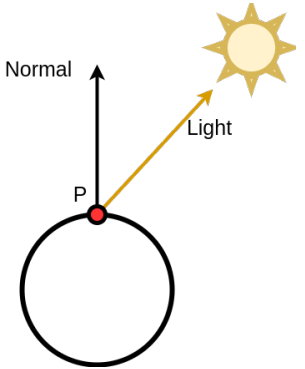
- Same description of material for each point of the 3D object, results in homogeneous material: smooth surface
- Once we have a scattering model, we can parameterize it and vary its properties over the surface.
- Therefore, we create parameterized material models and associate some parameters to each point of the surface. For example, this way, we can model marble which has color variation over surface.

Elements of material model

- Material modeling is separated into:
  - **Scattering** → description of light-matter interaction in a point
  - **Texture** → variation of scattering function properties across 3D object surface

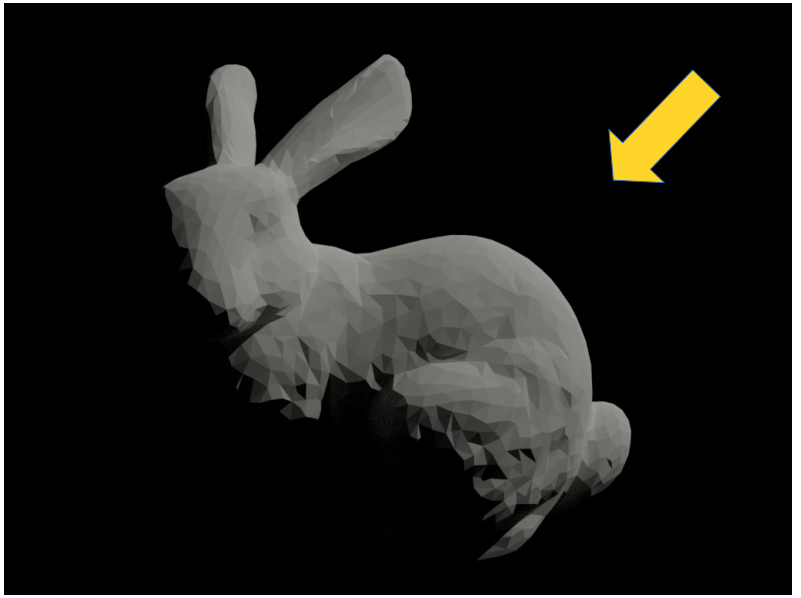
Scattering models

Scattering model



By now we know that amount of light falling on surface depends on shape and light:

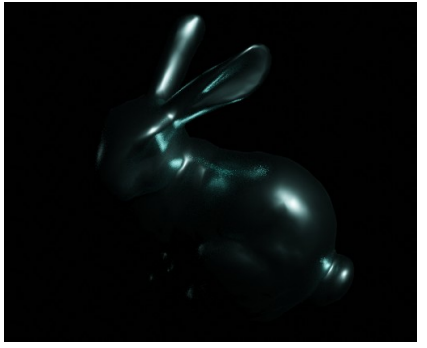
- Surface normal in point is crucial for determining how much the surface is oriented towards light



Scattering model

- Observing objects we can see that they have different appearance, although similar in shape
  - This particular look of objects is defined by how light scatters when it falls on surface point.
  - This behavior is defined by **scattering function** - surface response to light



Scattering function

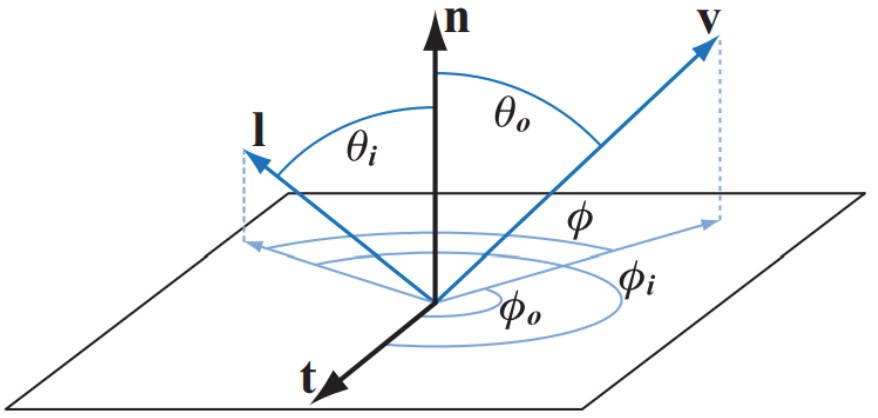
- Scattering function can be separated in reflection and transmission
  - Model describing reflection is called "bidirectional **reflectance** distribution function" – **BRDF**.
  - Model describing transmission is called "bidirectional **transmission** distribution function" – **BTDF**.

- **Reflective** – all light is scattered above surface
- **Transmissive** – all light is scattered below surface
  - Refractive – special case of transmissive

https://www.pbr-book.org/3ed-2018/Reflection_Models/Specular_Reflection_and_Transmission

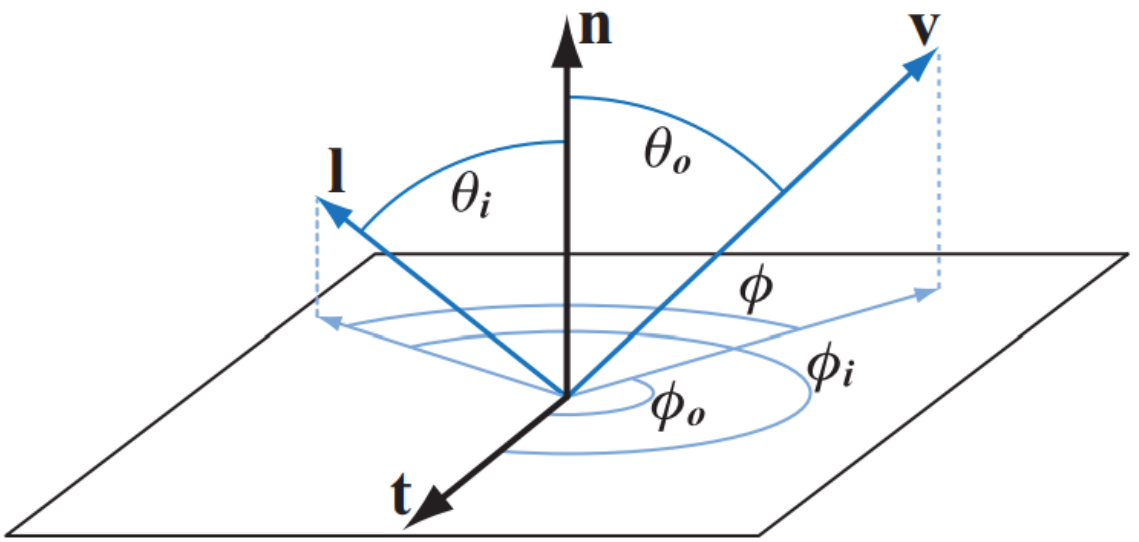
BRDF

- Description of how the light is **reflected** when light falls on **opaque** surface
- Describes **local interactions:** how reflects from a surface point
- BRDF f(v,l) describes surface response which depends on:
  - Incoming – unit length vector - light directior
  - Outgoing – unit length vector - view directio



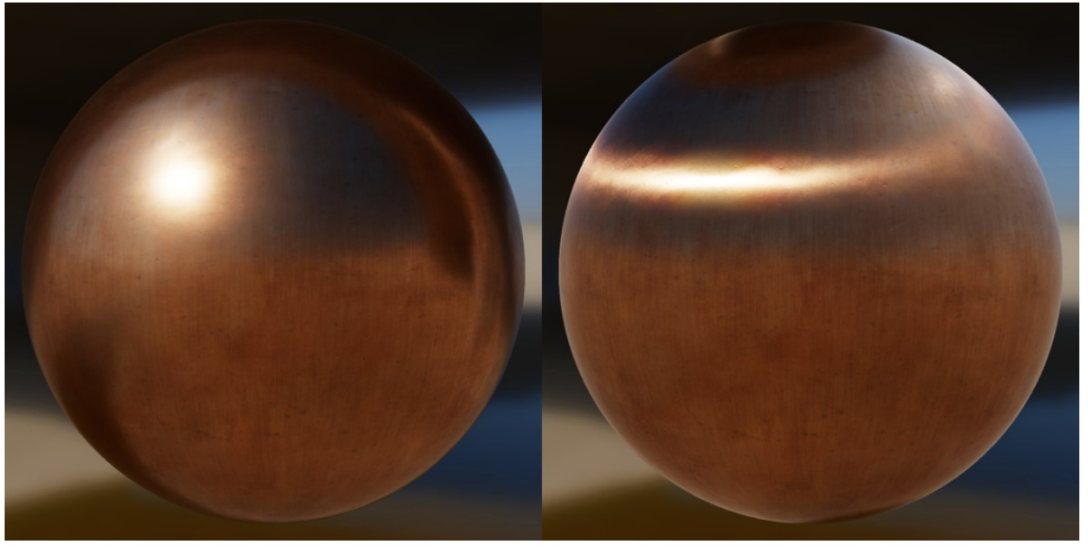
BRDF

- Incoming (l) and outgoing (v) directions have 2 degrees of freedom, two angles relative to surface normal:
  - Elevation (θ)
  - Azimuth (Φ)
  - Dimensionality of



Isotropic and anisotropic BRDF

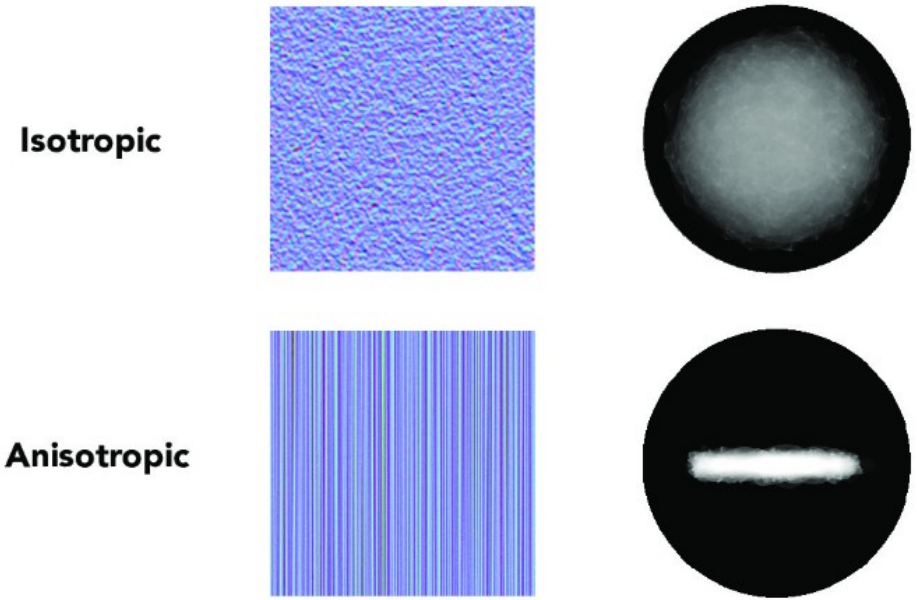
- **Isotropic BRDF**: rotating light and view directions around the surface normal does not affect the BRDF.
  - Incoming and outgoing directions have the same relative angles between them: such BRDF can be parameterized with three angles.
- **Anisoptropic BRDF**: reflection behavior changes when light and view vectors are rotated around normal



https://google.github.io/filament/Filament.html

Isotropic and anisotropic BRDF

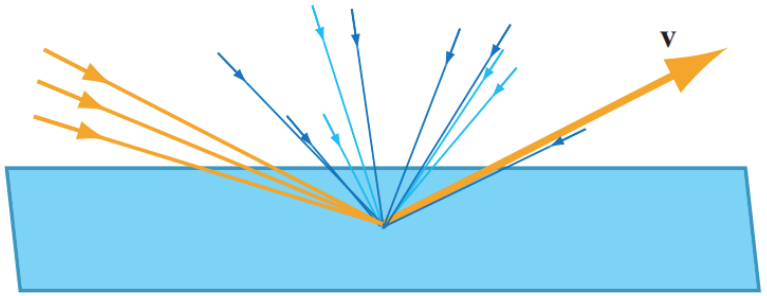
- **Anisoptropic behavior** is present due to underlying surface sturcture which is directional



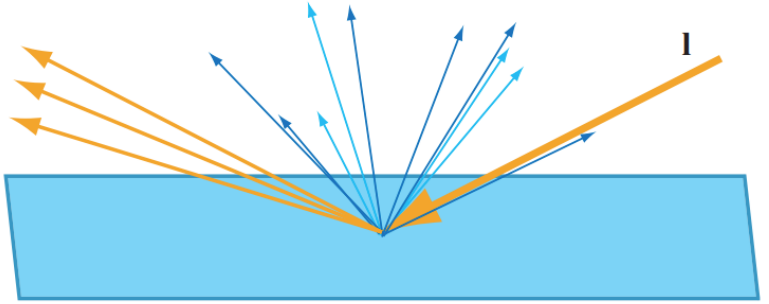
https://cs184.eecs.berkeley.edu/sp22/lecture/14-29/material-modeling

Note: bidirectionality of BRDF

- "**bidirectional**" in BRDF means that given given incoming and outgoing direction, we can compute **amount of reflected light** in outgoing direction.
- Further, bidirectionality can be used for:
  - Given outgoing (view) direction, it specifies the relative contributions of incoming light
  - Given incoming light direction, it specifies distribution of outgoing light



Contributions of incoming light given view direction

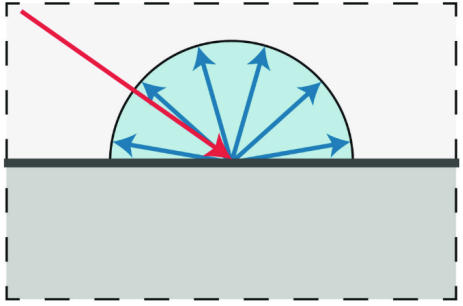


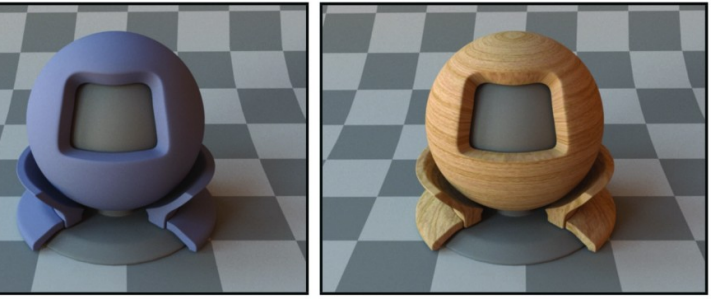
Distribution of outgoing light directions given incoming light direction

https://blog.selfshadow.com/publications/s2013-shading-course/hoffman/s2013_pbs_physics_math_notes.pdf

https://mitsuba2.readthedocs.io/en/latest/generated/plugins.html#bsdfs

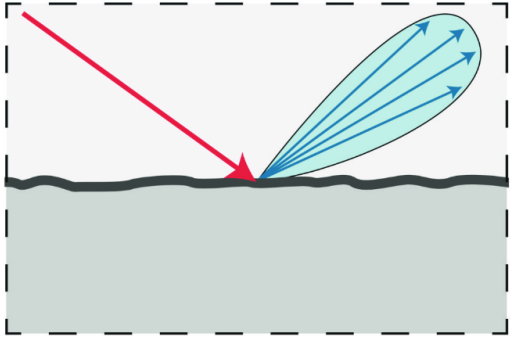
BRDF reflection types

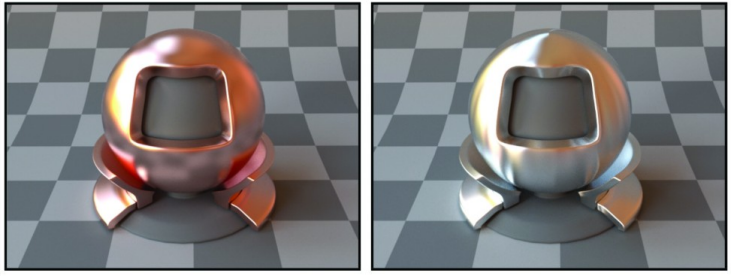




Diffuse, Diffuse textured

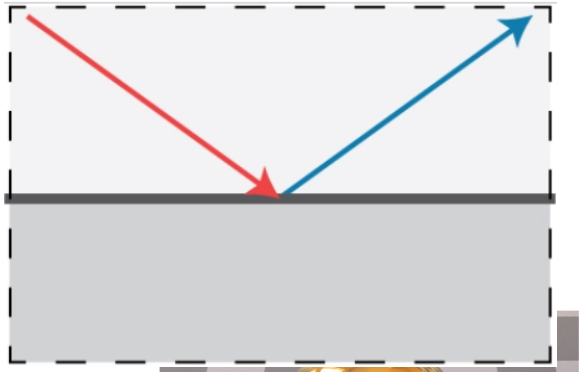
Light is scattered in all possible directions. Independent of viewing direction. Equally bright from all directions.

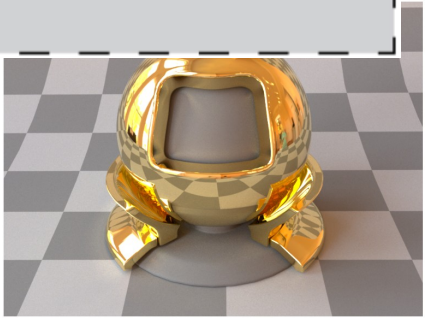




Glossy: Copper, Aluminium

scattered light is concentrated around particular direction (lobe). Appears blurred. Dependent of viewing direction.





Specular (gold)

light is scattered in single direction (mirror-reflection direction). Perfectly sharp. Dependent of viewing direction.

Generally, impulse scattering is term when light is scattered in single direction, but not necessary mirror-reflection direction

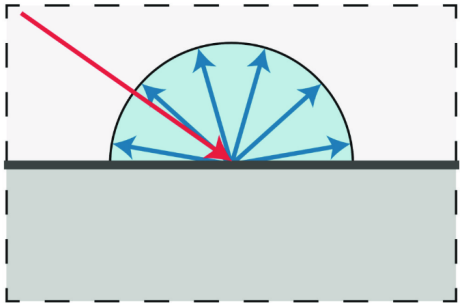
Scattering models

- Main types are:
  - **Empirical**
    - Created to simulate observed scattering phenomena
  - **Data-based**
    - Scattering is measured from real world and stored in tables on which lookup with direction (li, lo) is peformed
  - **Physical-based**
    - Based on physical interaction of light with matter

Empirical models

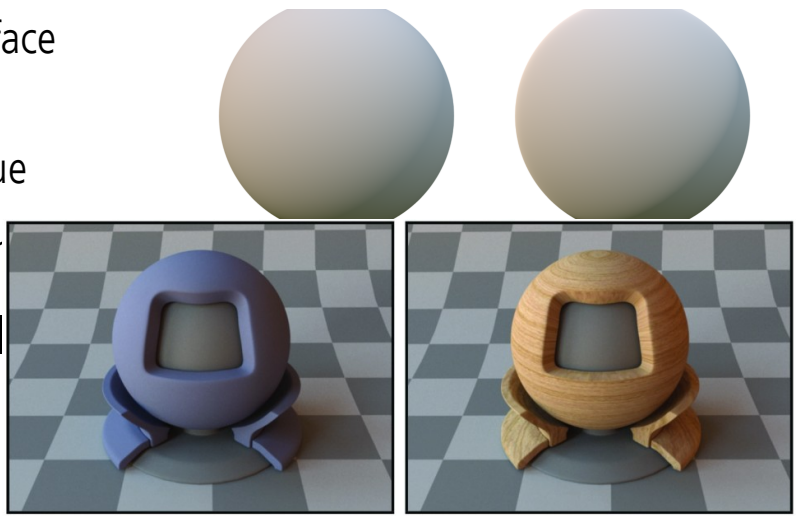
- Models based on observation of scattering phenomena
- Phenomenological models: describe the quantitative properties of real-world surfaces by mimicking them
- Easy to implement and use
- Types:
  - Lambertian
  - Mirror
  - Phong and Blinn-Phong

Lambertian model



- Scattered amount of light in all directions is the same and linearly depend on incoming light
- Simplest BRDF model:
  - Doesn't depend on surface or view direction but note that surface orientation will attenuate or increase incoming light making surface darker or brighter
  - **Constant reflectance value: diffuse color** or **albedo: (**R, G, B) value
- Often used as cheap approximation for sub-surface scatterir
- Although simple, it forms the basis for more complex model

$$f(v,l) = \frac{albedo}{\pi}$$



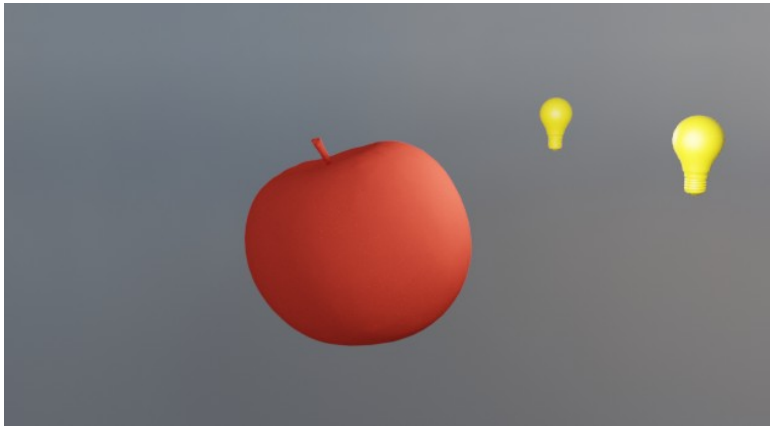
https://google.github.io/filament/Filament.html

Digression: where is light?

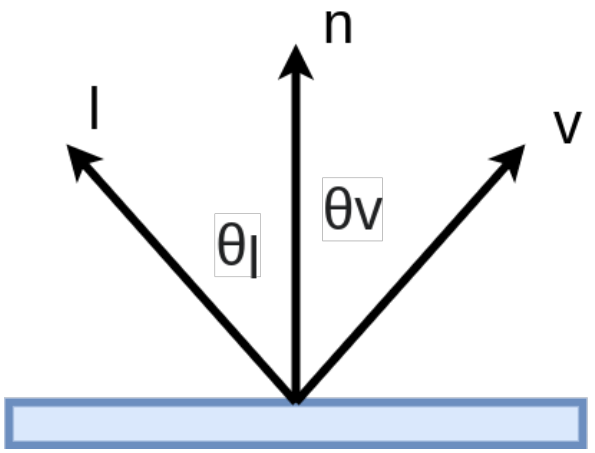
- BRDF is used in shading step of rendering.
- Shading "collects" all incoming light and multiplies it with BRDF – reflectance equation:

$$L_o = \sum_{l}^{n} L(l) \cdot f(v,l) \cdot (n \cdot l)$$

$$f(v,l) = \frac{albedo}{\pi}$$



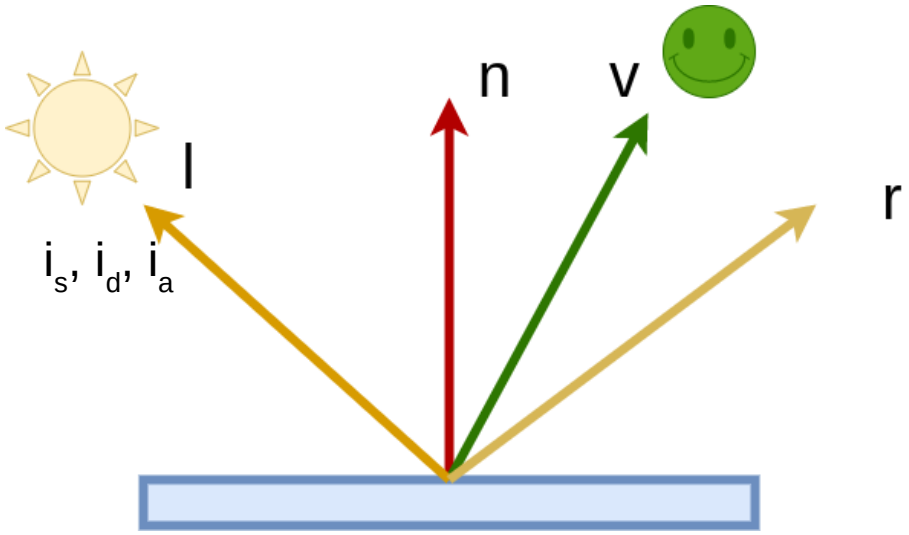
Mirror (specular) model



- Ideal reflection where incoming light is reflected completely in a single outgoing direction
- Reflects only in mirror reflectance direction:
  - Depends on view and light directions
- Single parameter: **reflectivity** – a constant (R,G,B)
  - Conductors have colored reflection due to spectral selectiv

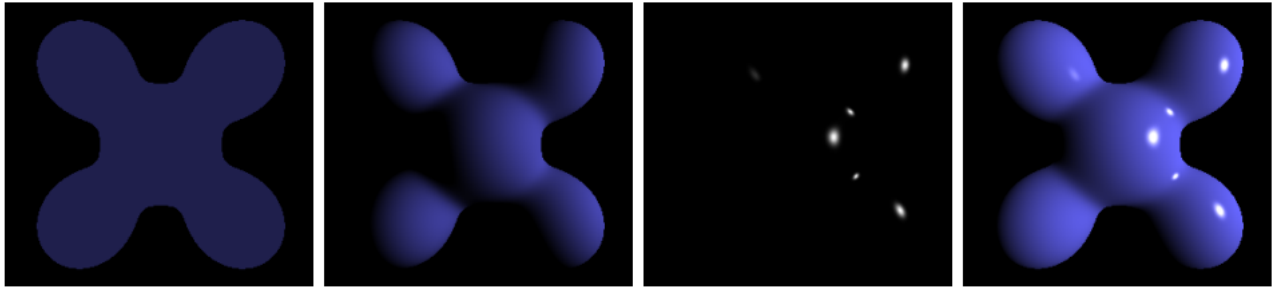
$$f(v,l) = \begin{cases} reflectivity \text{ if} & v = l - 2(l \cdot n)n \\ 0 & otherwise \end{cases}$$

Phong model



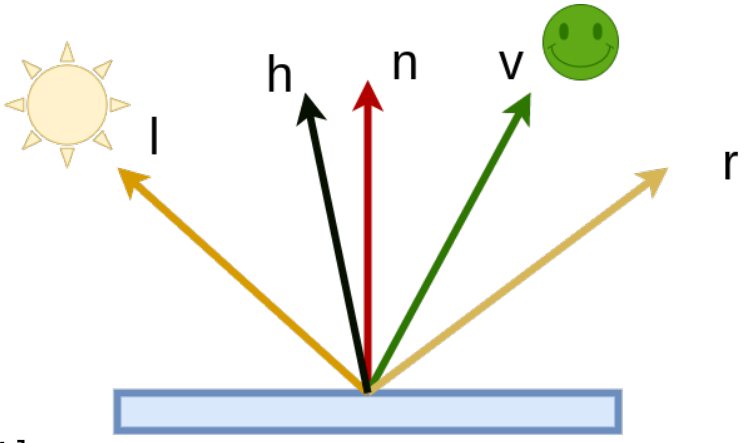
- Rather than BRDF, Phong model is a complete reflection model
- A phenomenological model using:
  - **Diffuse** reflection: rough surfaces; large highlights
    - Parameter: k_d – ratio of reflection of the diffuse term of incoming light
  - **Specular** reflection: shiny surfaces; small highlights
    - Parameter: k_s – ratio of reflection of the specular term of incoming light
    - Additional parameter: alpha – shininess constant: larger for smooth and mirror-like surfaces and small specular highlights
  - **Ambient** term: small amount of light that comes from around the scene
    - Parameter: k_a – ratio of reflection of the ambient term for all points on the surface

$$L_o = k_a i_a + \sum_j^n (k_d(l_j \cdot n) i_{j,d} + k_s(r_j \cdot v)^\alpha i_{j,s})$$



Blinn-Phong

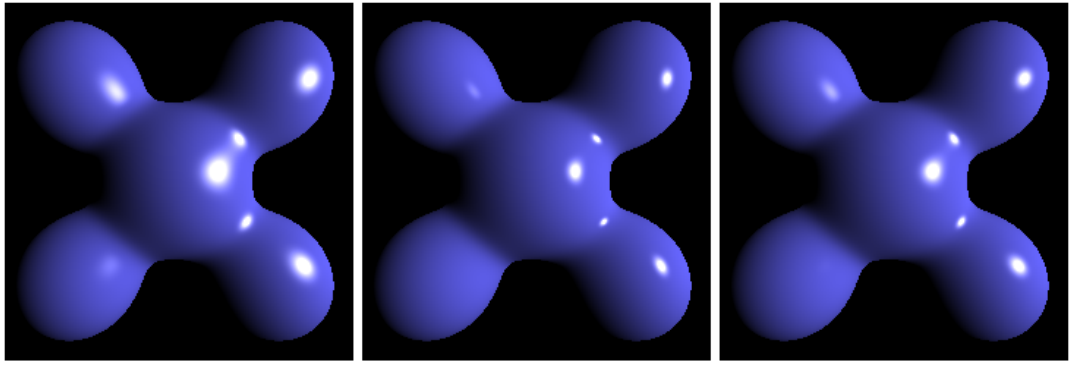
- Modification to Phong model, actual BRDF
- Introducing **half vector** between light and view vectors
- Parameters: Lambertian (k_L) and glossy (k_G), range: [0,1]
- Energy conserving if $k_L + k_G <= 1$



$$f(v,l) = \frac{k_L}{\pi} + k_G \frac{8+s}{8\pi} z^2$$

$$z = max(0, h \cdot n)$$

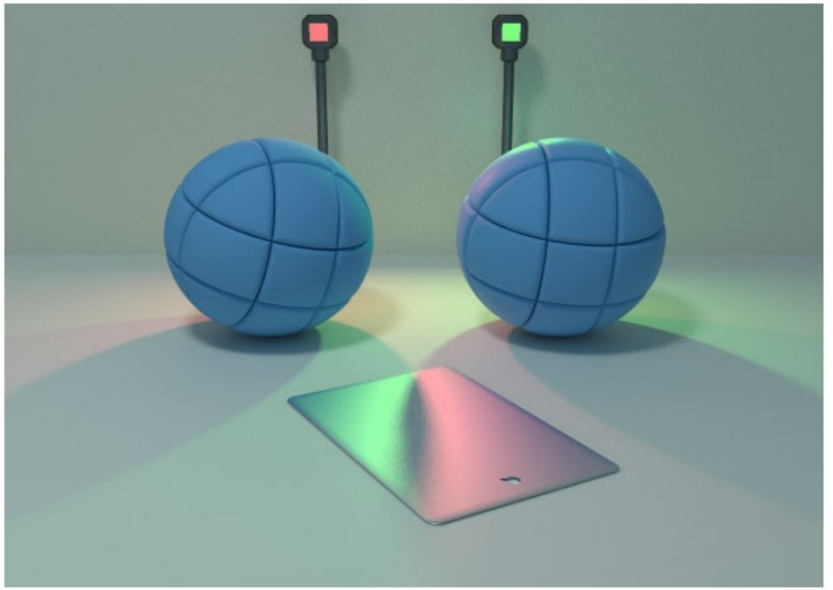
$$h = \frac{v+l}{2}$$



https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_mod

Other empirical models

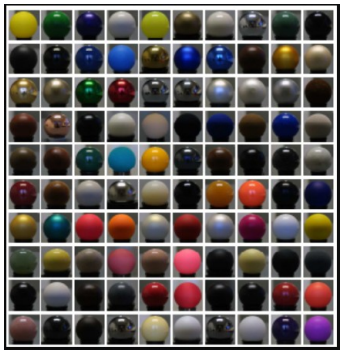
- Lafortune model
  - Generalization of Phong's model
  - Richer appearance with multiple lobes



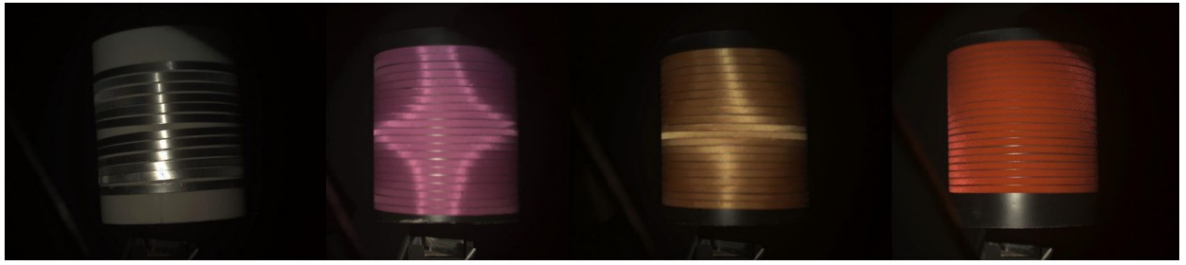
http://people.csail.mit.edu/addy/research/ngan05_brdf_eval_ppt.pdf

Data-based models

- BRDF measurement of real-world can be used for:
  - Evaluation of phenomenological and physically based models
  - Modeling of material
- Different real-world materials have been measured:
  - Isotropic BRDFs
  - Anisotropic BRDFs
  - Texture characteristics
  - Sub-surface scattering
- Problems:
  - Costly for rendering
  - Memory



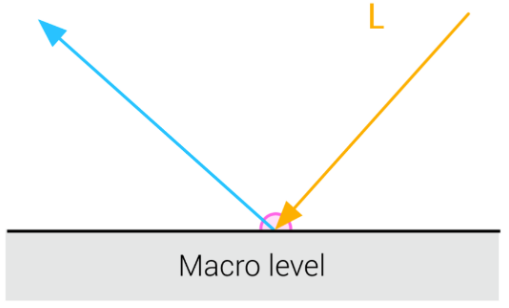
Isotropic materials:
https://www.merl.com/brdf/

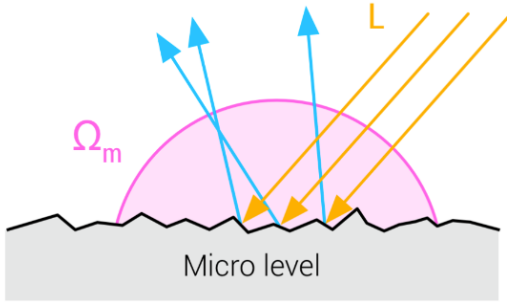


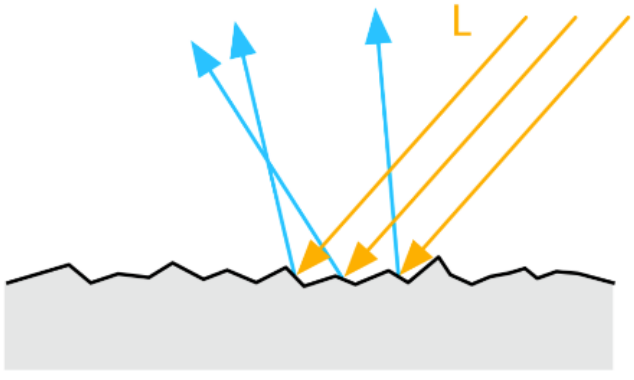
Anisotropic materials: Brushed aluminium, Yellow satin, Purple satin, Red velvet
http://people.csail.mit.edu/addy/research/brdf/

Physically-based models

- Real-world surfaces have geometrical detail on multiple and thus very small scales
  - Although we can not see those small-scale details individually, we can see their aggregate response and they define how surface actually appears
- **Small-scale surface irregularities** (smaller than a pixel) can not be modeled explicitly, therefore this is modeled using BRDF
  - **Geometric optics** assumption: these irregularities are much smaller than light wavelength (they have not effect of appearance) or much larger than light wavelength (they cause light redirection). Wave optics describes phenomena in between.

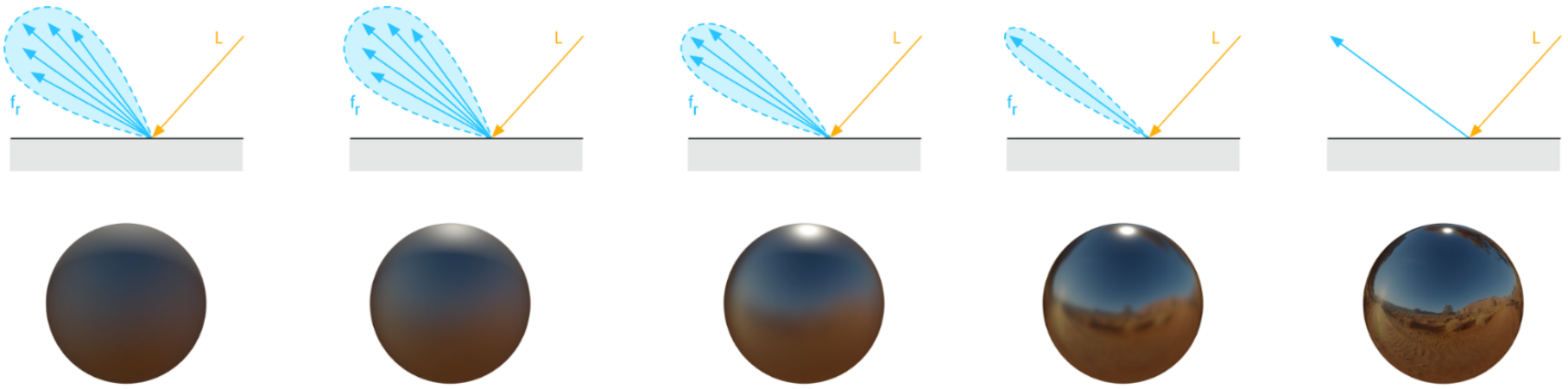






https://google.github.io/filament/Filament.html

Physically-based models

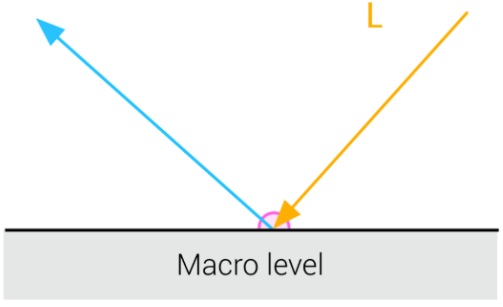


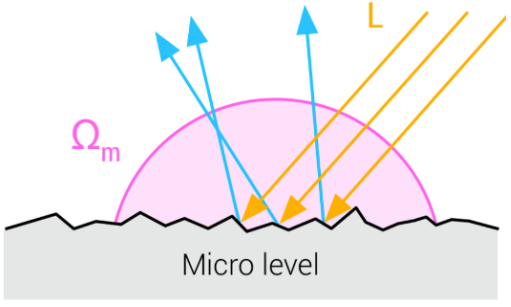
Varying roughness

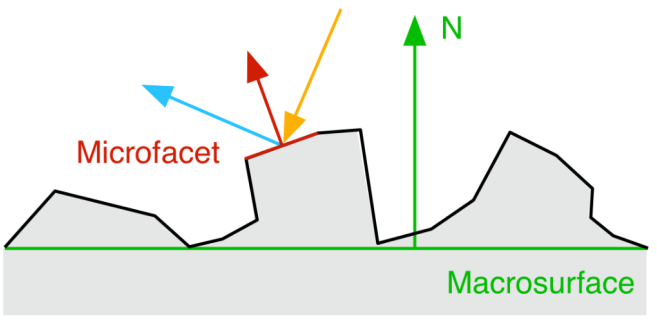
https://google.github.io/filament/Filament.html

Physically-based models

- Physically based models represent small scale irregularities as **facets** with **microsurface normals***.
  - **Micro-facet theory**: statistical distribution of microfacets which have strong peak at macrosurface normal. Spread of this distribution is called **roughness**
  - Higher roughness means more blurred surface since micronormals will be more spread (**glossy-diffuse**). Small roughness gives mirror-like surface (**specular**).
- With assumption of microfacets which are **optically-flat**, geometric optics can approximate what happens with light in interaction with such interface using **IOR, Fresnel equations and Snell's law**







https://google.github.io/filament/Filament.html

* Note how important normal information in computer graphics is. Different scales of geometry are always described with normal information.

Physically-based models

- Physically based rendering imposes two laws:
  - **Helmholz reciprocity:** f(l,v) = f(v,l) – input and output can be switched and the function value will stay the same
  - **Conservation of energy*:** outgoing energy can not be greater than incoming energy. BRDF which significantly violates this property leads to too bright and thus not realistic surfaces.
    - Note that BRD the distribution it describes is highly non-uniform. An example ar

White furnace test: white sphere is illuminated with white light from all directions. If energy conserving is satisfied, the sphere will disappear in white background

* Energy conservation is measured with directional-hemispherical reflectance R(l). It measures amount of light coming from given direction that is reflected into any outgoing direction in the hemisphere around normal – it measures energy loss for a given incoming direction. If BRDF is reciprocal then hemispherical-directional reflectance can be calculated as well giving the same value. Term for both reflectances in Directional albedo. The value must be in [0,1] to satisfy energy conservation: 0 is completely absorbed, 1 is completely reflected. Note that this restriction doesn't apply to BRDF since it can have arbitrarily large values in certain direction (e.g., highlight direction).

Physically-based models

- Microfacet-based BRDFs are state of the art reflectance models using in professional modeling and rendering software.
- Fundamental models are:
  - Torrance–Sparrow microfacet model for glossy surfaces
  - Oren-Nayar model for diffuse surfaces

https://www.pbr-book.org/3ed-2018/Reflection_Models/Microfacet_Models

Scattering models: practical tip

- In graphics, **various scattering models have been developed** (and still are!) to represent surface even more correctly or efficiently. R&D approaches can be classified in: empirical, data-based and physically-based
- Choice of the model depends on application and what is trying to be achieved.
- Practical tips:
  - **When modeling a material in DCC Tool**, you will be often offered with multiple implementations of basic (or more advanced) models that you further combine to achieve desired material description. In this case, it is good that you are familiar with how they work and their parameters because a lot of time is actually spent on "tweaking" parameters to achieve desired appearance. Understanding parameters of scattering models help very much with upcoming topic: texturing. This is huge and important topic.
  - If you are more interested in **developing your own scattering models** to achieve different appearance (not necessary photo-realistic, rather non-photorealistic which will be discussed later) then understanding of existing scattering models is great foundation to build on: you will see that advancements of scattering models just added more complexity to basic ones.

Exploring BSDFs

- Various BSDFs are available in modeling tools for material creation. Core scattering models are deeply integrated into renderer source code and user is provided with an interface to those for combining and creation of complex material.
  - Cycles/EEVEE (Blender): https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/index.html
  - Appleseed: https://appleseed.readthedocs.io/projects/appleseed-maya/en/master/shaders/shaders.html#materials
- Similarly as for object shape (e.g., mesh), materials are meant to be transferable between applications. Note that it is up to applications renderer which scattering models are supported. Therefore, it is often a case that material defined in one modeling tool can not be easily fully and exactly transferred to another application.
  - This requires matching supported scattering functions between applications. Example is Blender to Unity
- In order enable easier communication and transfer, standardized BSDF is created and supported by diferent applications.
  - Principled BSDF: Blender to Godot: https://docs.godotengine.org/en/3.0/tutorials/3d/spatial_material.html
- Tendency is towards integrations of material modeling tool standards into game engines for easier transfer
  - Example: https://substance3d.adobe.com/plugins/substance-in-unreal-engine/

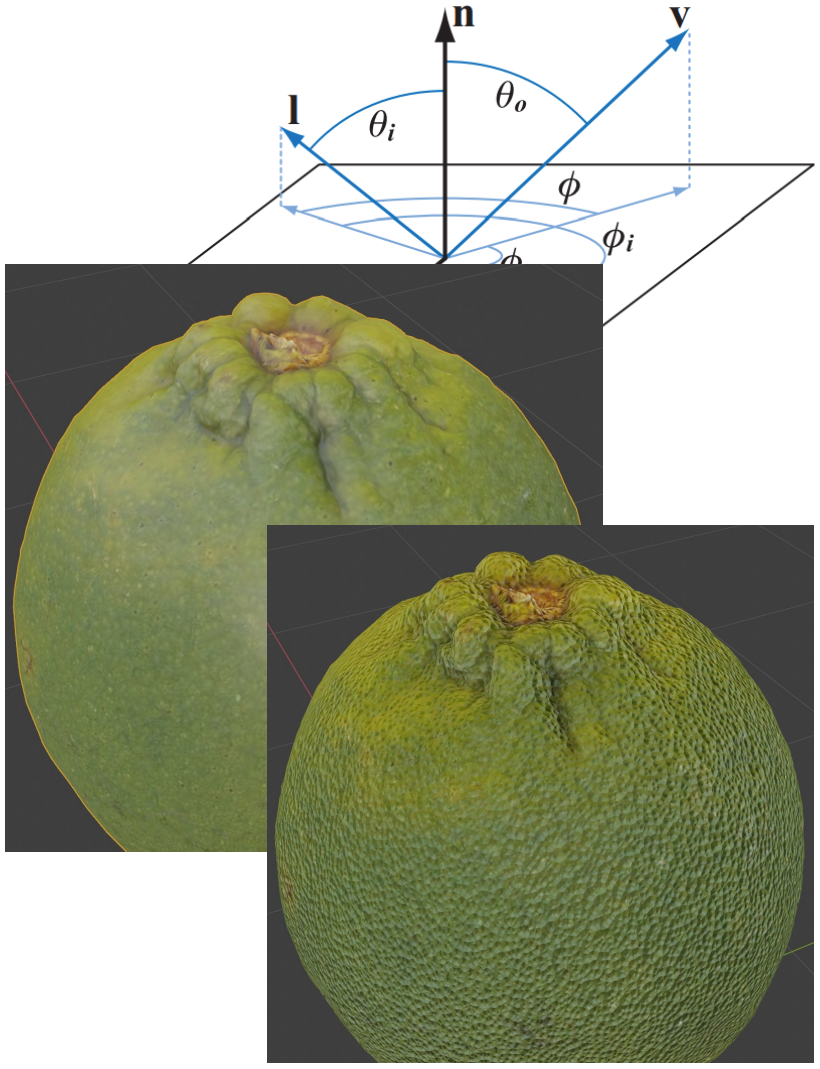
More into topic

- Note that BSDFs describe interaction of light with surface (opaque or transparent)
- Other objects have specific appearance due to sub-surface scattering. Example for such material is wax. Such material is not transparent, but it is important what happens under surface since some light scatters outside and influences appearance – such material is called **translucent** and requires **BSSRDF***.
  - For simpler applications these materials can be approximated with diffuse BRDF (note that diffuse scattering is actually a result of sub-surface scattering and re-emitting as well as surface roughness)
  - Different approaches take phenomenological approach were they model what we observe in reality. They result in realistic appearance but are not physically correct: https://www.ea.com/frostbite/news/approximating-translucency-for-a-fast-cheap-and-convincing-subsurface-scattering-look
  - Finally, phyiscally based approaches model actual scattering of light inside of surface and its absorption, reflection and re-emission.
- Next to translucent surfaces, there are many phenomena for which is important to model light scattering inside a volume. These are called **volumetric rendering** approaches and they rely both on scattering function and light transport.
  - Volumentric rendering is highly researched and developed field: http://advances.realtimerendering.com/s2015/The%20Real-time%20Volumetric%20Cloudscapes%20of%20Horizon%20-%20Zero%20Dawn%20-%20ARTR.pdf
- Often cloth is important material to render. **BRDF for cloth** (e.g., sheen BRDF) are developed and investigated.
- We discussed light scattering on geometric optics. **Wave optics** is active research area.
- https://www.realtimerendering.com/#visapp

* https://www.pbr-book.org/3ed-2018/Volume_Scattering/The_BSSRDF

Normal and BRDF

- BRDF evaluation depends on surface normal
  - Normal defines so called **basis**
- Note that **perturbing the normal,** would tilt the basis and reflected light would be different!
- This enables modeling small **scale geometrical details**
- Variation of surface normal is task for **texturing**.



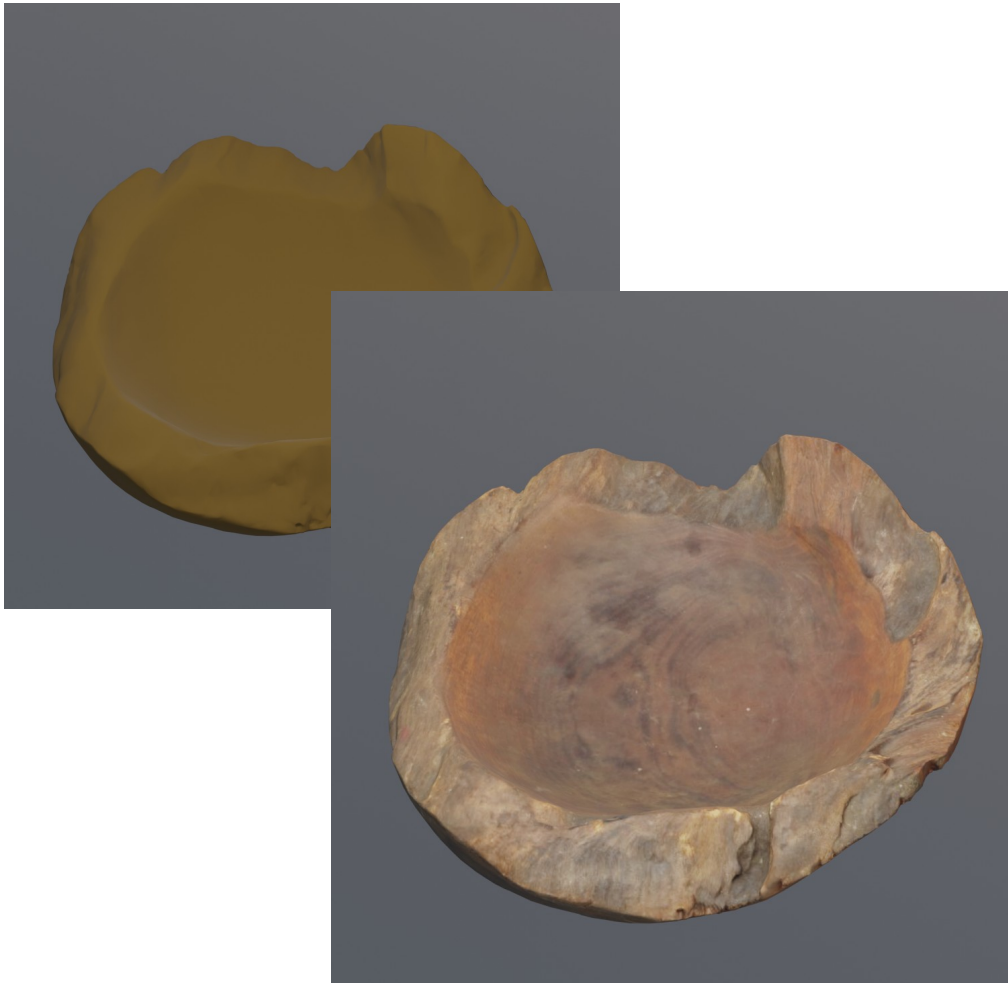
Scattering models parameters

- Scattering models are parameterized: color, roughness, etc.
- **Uniform parameters** result in overly smooth and perfect surface – not realistic.
- If BRDF depends on position on which is evaluated then it is called spatially varying BRDF
  - Variation of parameters over surface is done using **texturing**

Texture

Variation of material over surface

- Real objects rarely have only one material or material with same properties in each point.
  - Let's consider wood. Wood has structural texture (e.g., grain) at a scale of about 1mm. Also it has cellular texture at lower scale. The material of wood fiber is different. Therefore, the rich appearance of wood comes from material variation of fibers. This richness is called texture.
- Homogeneous description of material using uniform scattering functions makes objects look too perfect (too smooth and artificial).
- Realistic surface require variation of scattering functions or its properties over it (e.g., color). One material parameter is color.
- In graphics, we use **texture function** to vary material properties over surface.



Texture and texture function

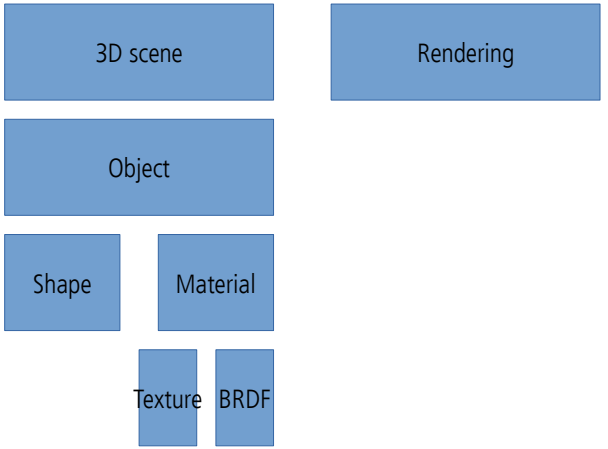
- Term **texture** is used in various disciplines and everyday life: texture of fabric, texture of food, texture in music, texture in image processing, texture in...
- Texture doesn't have well established definition. It means differently in different disciplines. It spans over multiple scales and over space. But, for our purposes of 3D modeling, we can try to define it as **variation of material properties over surface**. Therefore, a texture is something that we can completely define and model.

Texture and shading

- Texture model is used alongside scattering model during shading rendering step to calculate color of surface in each point. Looking in the rendered image, we will perceive texture on the surface.
  - Note that color of texture in most cases (especially in physically based rendering) is not one on one in the rendered image. Texture merely defines properties of scattering function which is used in rendering process.
  - Note the difference in texture function and texture. Texture is generally overloaded term and it is furthermore overloaded in computer graphics as we will see.

Texturing: practical example

- Surface texture is its look and feel
  - Bark of a tree, surface of stones, oil painting, etc.
  - <IMAGE>
- In computer graphics, texturing is process that takes surface* and modifies its appearance at each location using image, function or other data source.
  - <IMAGE FOR INTUITION>
    - Texture is property of object
    - During rendering texture is obtained for each point of object



* Or volume. For example, variation of gas density can be defined using texture.

Texturing: practical example

- Texture is often used to add more details* to the surface.
  - <example>
    - Tree trunk: (1) model shape using cylinder, (2) add color image to add variation (3) add roughness to vary reflection (4) add normal map to break flatness (5) add displacement map to break illusion of normal map

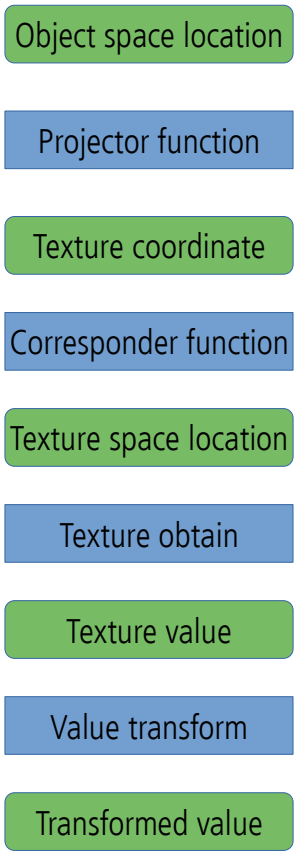
* Remember that computer graphics is all about simulation and approximation in order to achieve required appearance. It is a trade-off between quality and speed. Depending on viewing distance and size of objects in 3D scene, some details can be represented with lower quality. For example, modeling all details on geometrical (shape) level can be very expensive. Thus, texturing come in which represent cheaper way to model details

Texturing: variation

- In previous examples we have seen problems that are solved with textures:
  - Color variation
  - Roughness variation
  - Shading normal variation
  - Geometry variation
- Note that those are exactly parameters of scattering models!
  - Some of those describe variation on small scale: **microscale**
  - Some of those describe variation on bigger scale: **mesoscale**
  - **Macroscale** is given by the object shape.
- Different algorithms solving those problems with varying complexity (and thus quality) exists.

Texturing pipeline*

- Texturing: variation of scattering/shading** parameters over surface – **position dependent**
- **Location in space of object surface/volume**
  - Often, **object-space locations** are used so that is object moves, texture stays the same
- **Projection function.**
  - This function maps location in object space into location in texture space → **texture coordinates**
  - This process is called mapping and thus texture mapping (and sometimes texture image is called texture map which is not strictly correct)
- **Corresponder function**
  - Transform texture coordinates in **texture space location**s
  - For image texture, this might be indices to retrieve pixel values
  - For procedural texture, this might be domain of procedural function
- **Obtaining texture values (sampling)**
  - Texture space location is used for reading image texture or evaluating procedural texture
- **Value transform**
  - If needed additional transformations are done on value that is obtained from texture
- **Usage**
  - Final value is used to modify property of surface at that point: scattering function parameter or normal.



* Note that all steps do not have to be used in different applications!

** Value of texture is used for shading to obtain the final color

Texturing pipeline: example

- Wooden floor
  - During rendering, **object position** (x,y,z) on wooden floor is found
  - **Projector function** finds mapping from (x,y,z) to (u,v). In this case simple projection can be used: u~=x, v~=y so that u,v are in [0,1] – texture coordinates
  - **Corresponder function** uses texture coordinates for finding corresponding pixel in the image of wooden floor. If image has resolution 1024x1024 then pixel positions are integer of 1024*u, 1024*v
  - **Texture reading** is performed using texture space locations
  - Finally, color encoded in texture can not be directly used for rendering, **transformation on values** is done (e.g., sRBG → linear). Resulting values are used.
  - <ILLUSTRATE>

Projector function

- First step in texturing process:
  - Obtaining surface location (given by renderer)
  - Projecting surface location into texture coordinate space (often, but not always, 2D space (u,v)).
- Texture coordinates can be calculated:
  - On the fly during rendering or
  - During object modeling and stored per-vertex – Modeling packages allow artists to edit (u,v) coordinates.
- Many possible ways of obtaining texture coordinates exists:
  - Texture projections
  - Mesh unwrapping

Projector function: mesh unwrapping

- Objects are, often, surfaces in 3D space. Image texture lives in 2D space.
- Mesh unwrapping tries to "unwrap" 3D surface onto 2D space.
  - Analogy: world globe and world atlas
- Different methods exists. Unwrapping belongs to larger field called **mesh parametrization**
  - TODO
- <Example of mesh unwrapping>
- Modeling packages offer different methods for unwrapping and a way to manually enhance the results
  - Goal is that each polygon gets fair share of texture area (e.g., evade stretching) but mantaining connectivity – which determines where separate parts of texture meet and visible seams.
  - **EXAMPLE**

Projector function: texture projection

- The goal is to transform 3D point in space into texture coordinate*
- Commonly used methods in modeling programs are:
  - Spherical
  - Cylindrical
  - Planar
  - <EXAMPLES>
- For highly complex shapes, separating into simpler shapes can be done and then texture projection can be done separately.

<EXAMPLES>

- Spherical projection can be used in environment mapping
- Planar projection uses orthographic projection (usefor for, e.g., decals)

* Note that the goal is really to find texture coordinates and not to find a way of pasting a 2D image on 3D object. This is more general. Once texture coordinates are found, different types of textures can be used: image or procedural, etc.

Projector function: texture projection

- Often **surface normals** are used:
  - This enables **triplanar mapping** – which of six planar projection directions are used for projecting
  - For curved and tilted surfaces efficient **blending** of multiple projection planes must be used.
  - EXAMPLE

Projector function: different approaches

- Encapsulating object in volume of simpler shapes can make finding texture coordinates more tractable
  - Polycube maps
  - TODO
-
  -

Projector function: different approaches

- View direction can be used for defining projection
  - example
- If object is supplied with any kind of additional information (e.g., curvature or temperature) this can be used for texture coordinate generation
  - Main point is that texture coordinates must be generated in this step. Using position and normal for deriving those is only one way of doing it.

Projector function: inherent texture coordinates

- Parametric surfaces have natural set of (u,v) which defines any (x,y,z) point on surface
  - <EXAMPLE>
- In this case, we can say that surface representation already has inherent texture coordinates and projector function is not needed.

Projector function: different texture coordinates

- Texture coordinate space doesn't have to be always 2D
- If texture is used for volumetric objects, then texture coordinates are (u,v,w)
- For animation, texture coordinates can be (u,v,w,t) where t is time and determines change in texture
- Furthermore, procedural texturing can directly rely on object locations – solid texturing.

Projector function: different texture coordinates

- Texture coordinate, next to positional, can also be directional
  - Each point in space is accessed by input direction
- Common type of using directional parametrization is **cube map**
- EXAMPLE

Projector function: multiple texture coordinates

- Multiple textures can be applied on objects and thus multiple sets of texture coordinates can be applied
- Idea is the same: those texture coordinates are interpolated across surface and used for obtaining texture value
- <example of multiple textures>

Corresponder function

- Convert texture coordinates to texture space locations
  - e.g., u,v to pixel index in image
- Corresponder functions:
  - Which part of sub-texture will be used
  - Transformations: rotate, scale, translate, shear or project
  - How image is applied if (u,v) is larger than 1 or smaller than 0: wrap, mirror, clamp, border, repeat, wang tiles, etc.
- TODO

Obtaining texture values

- Corresponder function gave texture space coordinates
- Those coordinates are used for:
  - Obtaining image texture values
  - Evaluating procedural textures values
- TODO

Transforming texture values

- Image textures contain RGB(A) values.
- Many data types can be stored/encoded using those values. Thus, although image textures always contain color values, those values can be interpreted differently then color.
- Therefore, values obtained from texture are optionally transformed to required datatype.
- <example: encoding vector in image texture>

Image textures

- Image texturing process can be seen as "gluing" image on 3D surface
- For particular position on surface we explained how to obtain texture coordinates. These coordinates are (u,v)
- Image textures are usually 2^m x 2^n texels. Thus called power-of-two textures (POT)*
- Texture image can be created:
  - By drawing
  - Taking photographs
  - Measurements
  - Baking

* Older GPUs couldn't support mipmapping for non-POT textures. Modern GPUs handle all textures correctly.

Magnification and minification

- Example: smaller texture image (256x256) is used and projected on square. Number of texels may be different as number of pixels.
- If projected square on screen contains more pixels than original image then it comes to magnification. Texture system must magnify the texture. Magnification types:
  - Nearest neighbour: pixelization
  - Bilinear interpolation: blur
  - Cubic filtering
  - <examples>
- If several texels cover pixel cell, then cumulative effect of texels influencing the pixel must be taken in account.
  - Nearest neighbour: heavy aliasing and temporal aliasing (when camera is moving)
  - Bilinear interpolation: if pixel is covered by more than four texels then aliasing again becomes the problem
  - Problem of aliasing is solved using sampling and filtering techniques
  - <examples>

Anti-Aliasing: sampling and filtering

- Frequency of a texture depends how closely are spaced texels on the screen.
- Nyquist limit – texture frequency must be not larger than half the sample frequency. To properly display a texture on a screen, we need at least one texel per pixel:
  - Pixel sampling frequency must increase
  - Texture frequency must decrease

Anti-aliasing: mipmapping

- Most popular anti-aliasing method
- MIP – multum in parvo (latin) – many things in small place – minimization filter:
  - Process in which original image is filtered down repeatedly into smaller images.
- Before rendering, original image is augmented with sets of smaller versions of this image
  - Image pyramid: <image>
- Mipmap chain:
  - Original texture is downsampled by quarter of original area. New texel is calculated as average of four neighbour texels in original image. Newly create texture is called sub-texture
  - This process is repeated until one or both of image texture dimensions equal to one.
  - <image>
- High quality mip mapping requires:
  - Good filtering: when averaging new texel value from 2x2 of texels it is recommended to use Gaussian, Lanczos or Kaiser filter (Box filter gives bad results)
  - Gamma correction. Images are often stored in sRGB colorspace. This colorspace is useful for displaying images but not computing. Therefore, always before filtering, image must be in linear colorspace.

Anti-aliasing: mipmapping

- Using mipmaps to achieve 1:1 pixel to texel ratio. TODO
- Texture level of detail
  - TODO
- RTR: 6.2.2

Examples of image textures

- Substance:
  https://substance3d.adobe.com/assets/allassets?assetType=substanceMaterial&assetType=substanceAtlas&assetType=substanceDecal
- PolyHaven: https://polyhaven.com/textures

Volume textures

- Extension of 2D image is 3D image accessed with (u,v,w) coordinates
- Example: medical imaging data
  - Three dimensional grid is visualized as slices by moving polygon through it
  - <image>

Volume textures

- Complex problem of finding good 2D parameterize for 3D mesh is not needed since object locations can be used as texture coordinates.
  - example
- Volume texture can represent volumetric structure of material such as wood or marble.
  - Image
- Problem: volumetric image textures are very space demaning

Procedural textures

- Instead of image lookup for texture values, those can be evaluated from a function
- Procedural texturing is much more popular in offline applications while in real-time rendering image textures are far more common*
- Procedural texturing is based on algorithm which defined values for each point of 3D space or object surface.
  - Parametrization of surface is therefore not needed**

<IMAGES: PROCEDURAL TEXTURING>

<IMAGES: NODE SYSTEM FOR PROCEDURAL TEXTURING>

* GPUs are extremly efficient with image textures and antialiasing is easier.

** Procedural texturing of complex shapes on complex surfaces is technically challenging and active area of research. For example, textures can be decomposed in stationary and globally varying. Stationary textures are easly to be procedurally generated on any shape. Globally varying textures have certain elements which have direction and shape, thus procedurally modeling of those on complex surfaces is hard

Procedural textures: volumetric textures

- Storing volume textures into image is very costly
- Therefore, volume textures are attractive application for procedural texturing since they can be synthesized with various methods
- Often, **noise function** is used:
  - Sampled at power-of-two frequencies called **octaves**. Each octave has weight which falls as frequency increases
  - Sum of weighted samples is called **turbulence** function
  - <example>
- Lattice points in 3D array are often precomputed and used to interpolate texture values

Procedural textures: Perlin noise

- Most popular noise function upon which wide range of methods is built
- TODO

Procedural textures: Cellular texture

- Widely used texture for representing Voroni-like textures and many natural textures: cells, flagstones, lizard skin
  - Another fundamental procedural texture upon which diferent methods are built
  - examples
- It is based on measuring distances from each location to a set of feature points
  - Intuition

Procedural texture: physical simulation

- Perlin and Cellular textures are mathematical models and phenomenological models for representing natural phenomena
- Phyiscally based simulation can also be used to create textures
  - Diffusion-rection: TODO
  - Practical: houdini simulation to texture

Procedural textures: Antialiasing

- Procedural textures can be hard to antialias
- Helping factor is that "inside information" about the texture is available:
  - For example, if procedural texture is built on summing noise functions, then frequencies which would cause aliasing can be omitted

Material mapping: variation of parameters

- Until now we have discussed how to obtain texture value given location on object surface.
- Now, we will discuss how texture values are used to modify material across object, that is scattering function and shading parameters over surface
  - Example: 3D scene → rendering → shading: shading reads texture information and uses it for calculating surface color

Usage of texture values

- Modify scattering/shading parameters
  - Surface color using albedo/diffuse texture values (Example of linear parameter*)
  - Roughness values using roughness texture values
- Distributing different materials over surface
  - Texture can be used as a "mask" which determines which scattering function and shading calculations will be performed where: rusty and shiny material example
- Modify scattering basis
  - Bump, normal (Example of non-linear parameter*)
  - Parallax
  - Displacement

* Antialiasing must be also performed on shading results since pixel may cover large patch of the scene containing different materials. Linear parameters are easier to antialias then non-linear ones.

Alpha texture values

- Alpha values are used for blending or alpha testing for transparency
- This enables **decals/cutouts**: rendering only parts of the image on object surface.
  - Example
- **Bilbording** method, which is used for standalone flat polygons with image texture, is also using alpha texture values
  - Example: large forest - https://80.lv/articles/how-to-fake-a-large-scale-forest-in-blender/
- Alpha value is very much used when adding VFX to rendered or real images - **compositing**.
  - Example

Bump mapping

- Large family of small-scale detail representations
- Typically, texture value is processed in shading step during rendering to achieve more 3D appearance of surface details than compared to color texture
  - No additional geometry is generated thus this is often used approximation to achieve required surface details
  - <examples>

Bump mapping: scale of modeling

- Details on object can be classified:
  - **Macro-features**: cover many pixels. Represented by model shape (e.g., polygonal mesh). For character modeling, limbs are modeled at macroscale
    - example
  - **Meso-features**: cover few pixels across. Everything between two scales. It represents detail that is too complex to render using shape representation (e.g., polygonal mesh) but large enough to be observable. For character modeling: skin or cloth wrinkles are modeled on mesoscale. For this scale family of **bump mapping methods** is used.
    - example
  - **Micro-features.** Described with scattering function in shading step. It uses texture values for computing the color. It simulates the microscopic response of surface geometry and substance. Shiny and diffuse objects are modeled on micro-scale
    - example

Bump mapping: idea

- Difference between methods is how they represent details
  - Tradeoff: level of realism and complexity
- Main idea (Blinn):
  - Surface will appear to have small scale details is during shading process, surface normal is slightly perturbed. <example: 2d sketch and rendered images>
- Information for perturbing surface normal can be encoded in texture just like color.
- Problem:
  - Actual geometry is not changed.

Bump mapping: frame of reference

- Perturbing surface normal must be done with respect to some frame of reference
- To do so, **tangent frame** is used – tangent frame **basis**
  - TBN: TODO
- On this basis, scattering model is evaluated
  - Perturbing the basis by perturbing the normal we achieve different incoming light and viewing directions – just like the surface contain small bumps and dents.
  - Also, directions of light and viewing are often transformed in this space – so they are in the same space as surface normal which is needed for correct for evaluation.
  - Tangent frame is used to determine orientation of material on the surface which is important for **anisotropic** surfaces

Bump mapping: Blinn method

- Original bump mapping – **offset vector bump map** - method required two signed values to be stored per texel in texture
  - Those correspond to amount of normal to be varied in u an v image axis – which way surface faces at the point.
  - Those values are used to scale vectors perpendicular to normal (tangent and bitangent)
  - Resulting vectors are added to normal to change direction
  - <example>
- Another approach is to use **heightfield** to modify surface normal direction
  - Image texture contains one floating point number per texel representing height
  - Those values are used to derive u and v signed values similar to those in original method – this is done by taking differences between neighbouring columns are rows to obtain u and v (Sobel filter an be used)

Normal mapping

- Instead of calculating perturbed normal using height values, normal vectors can be directly stored into image texture or evaluated from procedural texture.
- Normal map image texture encodes (x,y,z) vector with values in [-1,1] to rgb image texture
  - For PNG images, 8bit unsigned int is used. Therefore 0 represents -1 and 255 represents 1
- Normal map are often defined in **tangent space***
  - Normal map can be then maximally reused for different objects, transformations and deformations.
  - Color of normal map is determined with x,y and z directions of encoded vectors <example>
- <examples>

* Originally, they were defined in world or object space but this was limited to specific geometry for particular

Normal mapping: practical notes

- Relationship between normal and shaded color is not linear*. Therefore, filtering of normal maps for **anti-aliasing** requires additional work.
- Normal map can be easly **derived from height map**
  - Example
- Problem with normal mapping, as with bump mapping, is that additional geometry is not created. Thus effects of light being shadowed** by geometry and casting shadow on its own surface is not possible.
  - Solution is **horizon mapping** method which enables normals having bumps which cast shadow on their surfaces.

* Normal map applied to specular surface results in non-linear relationship between shaded color and normal map. Lambertian (diffuse) surface has almost linear relationship between shaded color and normal map since base of lamberitan shading is based on dot product with normal which is linear operation.
** Generally, when light falls on surface with bumps, multiple scattering, masking and shadowing effects happen which are not possible with normal mapping.

Creating bump/normal mapping

- Often for purposes in real-time graphics, high quality mesh is created. This mesh is then simplified and normal map is **baked** from high quality mesh.
  - Low quality mesh and baked normal are then used for real-time applications
  - Example
- Another possibility is to create bump map by scanning heights from real world
  - Example

Texture modeling and texture application

- When we talk about textures in computer graphics, we can separate the work in creating/modeling a texture and applying it. Often, line between these two is blurred.
- Reminder: we said that material modeling (which includes texture) can be performed separately of modeling 3D shape.
  - Creation of texture is often separated of modeling 3D shape and it is done in so called “texture space”.
  - Texture application is process where we “apply” texture on a 3D shape. That is, map it from texture to object space.
- Texture modeling can be described as process of developing a function which maps some property to each point of the surface. Texture modeling can be separated into:
  - Creating image textures
  - Creating procedural textures
- Texture application can be described as a process of adding actual texture on the object.
  - Often, term texture mapping is used. This term is due to historical reasons where details were painted on 3D model and those details were stored in array of images called textures. The process of corresponding each vertex of model to a location in image was called mapping.
  - Main task in texture application is to find mapping between texture and object space.
- Texture, simply speaking, contains information about surface details. It can be color (albedo), normals, roughness, etc.

Storing and transferring materials

- Similarly to mesh information, standards for material storage and transfer are defined
- Material standards:
  - MaterialX
- Certain formats that we mentioned for mesh storage are used for storing the whole scene including the material:
  - GLTF, USD

Practical note: textures

- Generally, outside of computer graphics field, term texture is overloaded
- In computer graphics this term is also overloaded
- On modeling level of abstraction, texture is a way to modify scattering function/shading parameters over surface.
- On development level of abstraction, especially while programming in graphics API, texture is nothing more than a memory buffer that is sent from CPU to GPU.
  - RGBA channels can encode any information. Examples: octree, mesh, etc.

Exploring textures

- Industry standard texturing tool: https://www.adobe.com/products/substance3d-painter.html
- Textures are often used for layering materials (RTR 9.12)

More into topic

- Procedural texturing
- Texture mapping
- https://www.realtimerendering.com/#texture

Practical insights

- Show how is material generated in practice using different scattering functions and textures

Further into topic

- TODO

Literature

- https://github.com/lorentzo/IntroductionToComputerGraphics/wiki/Foundations-of-3D-scene-modeling