Ray Tracing in Entertainment Industry

Tanaboon Tongbuasirilai
Dept. Computer Science
Kasetsart University

Week 7

Texture and reflection models

Texture and reflection models

Why texture ?

- In most cases, we need a man-made appearance requiring artistic skills for representing a material.
- It can add natural and/or unnatural effects to the materials.
- It’s easier to imitate material appearance with artistic skills.

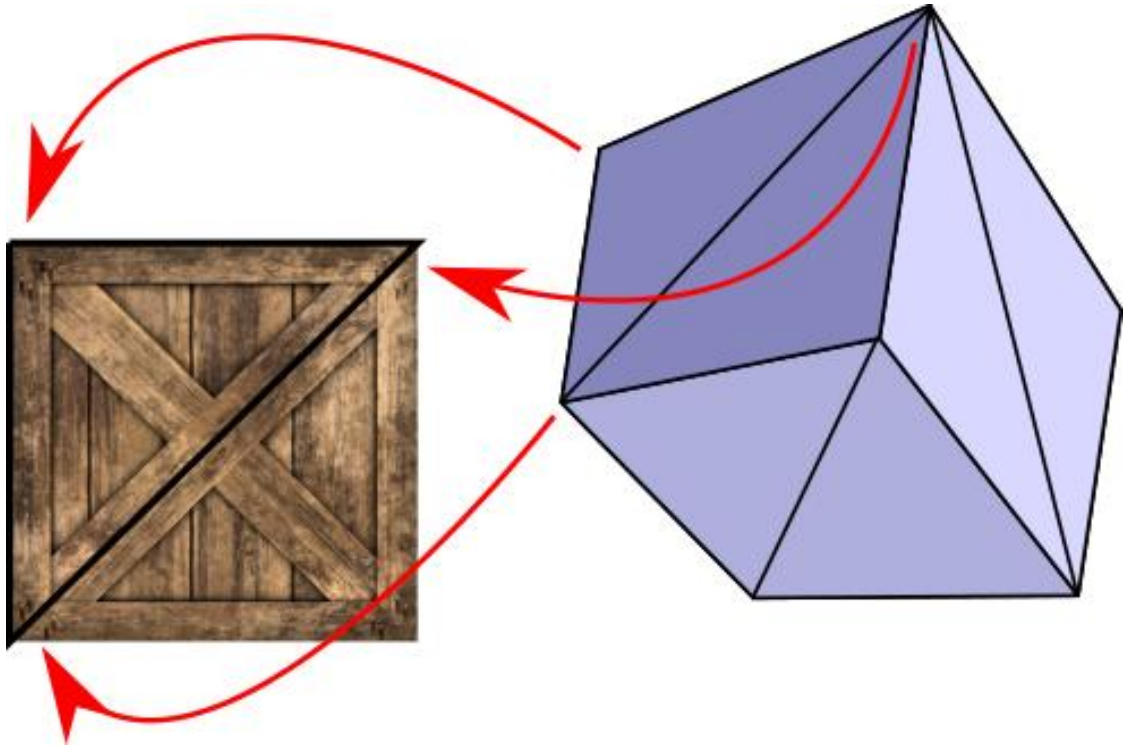
Why reflection models ?

- Geometric optics and image texture do not account for how much light is scattered on a surface.
- We would like to model light quantity related to the material scattering properties. [accuracy, efficiency]

mc.dean.lsa.umich.edu/threejs/manual/en/textures.html

Texture

- A texture, in general, is some sort of variation from pixel to pixel within a single primitive.
- We will consider only one kind of texture: image textures.
- An image texture can be applied to a surface to make the color of the surface vary from point to point, something like painting a copy of the image onto the surface



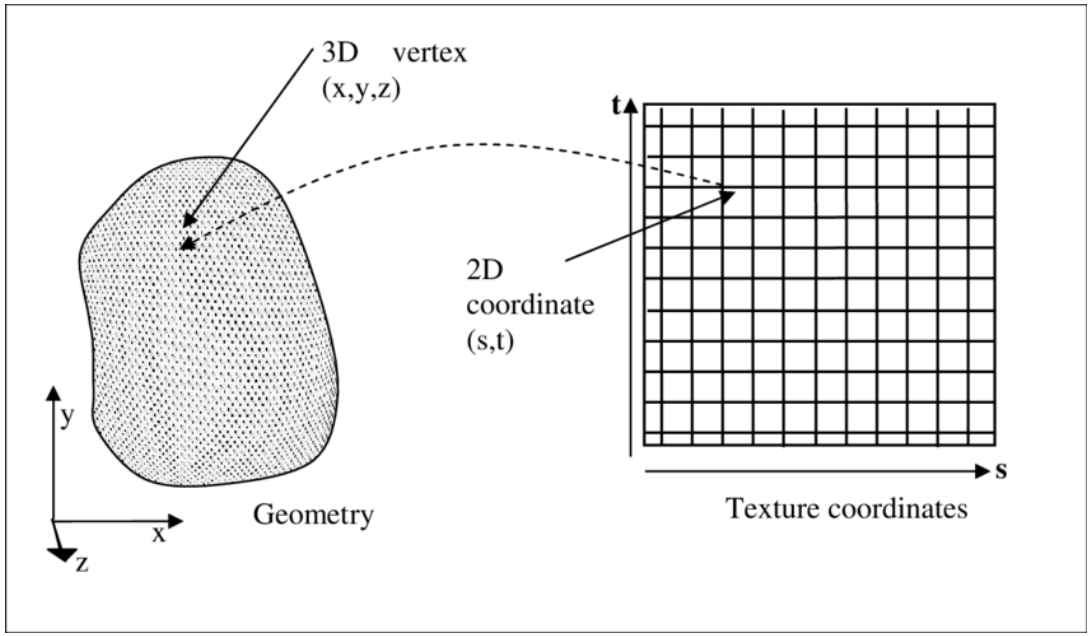
math.hws.edu/graphicsbook/c4/s3.html

gabrielgambetta.com/computer-graphics-from-scratch/14-textures.html

Texture mapping

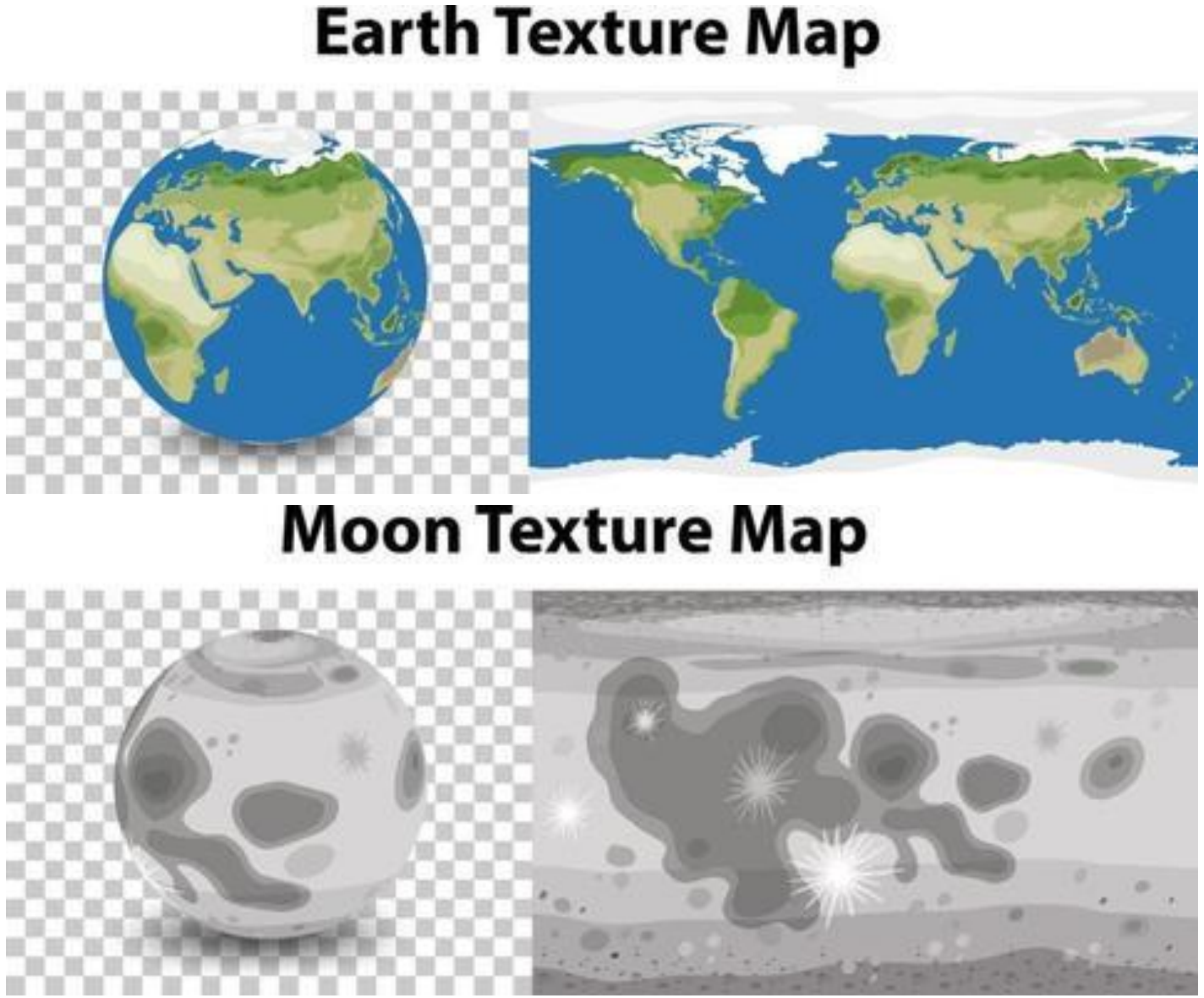
- Texture coordinates
  - Defined by 2 parameters : (u,v)
- Object coordinates
  - Defined by a 3D point : (x,y,z)

- Texture mapping
  - A function to map from a 3d point to a uv point.
  - T : (x,y,z) → (u,v)



www.researchgate.net/publication/228394843_Using_Haptics_in_a_Networked_Immersive_3D_Environment

Spherical mapping



www.vecteezy.com/vector-art/2785881-earth-planet-with-earth-texture-map

Texture mapping in ray tracing

1

Update Hitinfo class

- Declare : u and v parameters
- Methods to set and get uv parameters.

2

Update all Object classes

- Only update ‘intersection()’ method of each object class.
- Once a ray hits an object, we compute the uv coordinates to texture.

3

Implement Texture classes

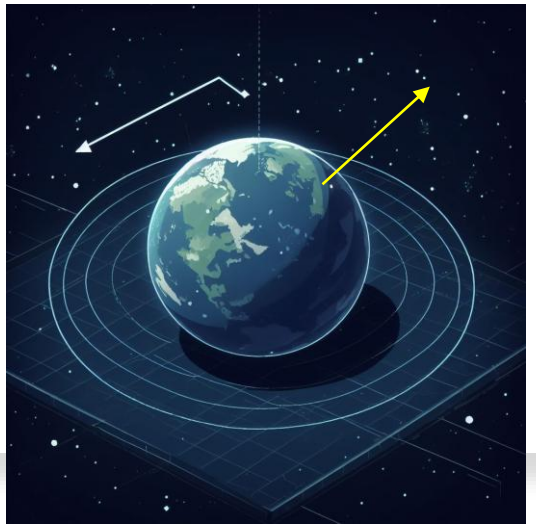
- Parent class : Texture
  - Method – tex_value() : return texture value at a specified (u,v) coordinate.
- Child classes
  - Solid color – return only its color
  - Procedural texture – return procedural texture
  - Image texture – load up a rectangle image and use it as a texture

```python
# set uv coordinates of the hit point.
def set_uv(self, fu, fv):
    self.u = fu
    self.v = fv
```

```python
class Hitinfo:
    def __init__(self, vP, vNormal, fT, mMat=None) -> None:
        self.point = vP
        self.normal = vNormal
        self.t = fT
        self.front_face = True
        self.mat = mMat
        # handling texture
        self.u = 0.0
        self.v = 0.0
        pass
```

Update Hitinfo

Update intersect()



Sphere class

```python
# return uv coordinates of the sphere at the hit point.
def get_uv(self, vNormal):
    theta = math.acos(-vNormal.y())
    phi = math.atan2(-vNormal.z(), vNormal.x()) + math.pi

    u = phi / (2*math.pi)
    v = theta / math.pi
    return (u,v)
```

Compute theta and phi of the intersection point.

Why normal is used to compute (theta,phi) angles ?

normalize

Don’t forget to add (u,v) coordinates.

```python
# set uv coordinates for texture mapping
fuv = self.get_uv(hit_normal)
hinfo.set_uv(fuv[0], fuv[1])
```

Texture class

```python
# solid color as a texture
class SolidColor(Texture):
    def __init__(self, cColor) -> None:
        super().__init__()
        self.solid_color = cColor

    def tex_value(self, fu, fv, vPoint):
        return self.solid_color
```

Solid color

```python
def tex_value(self, fu, fv, vPoint):

    if self.invalid:
        return rtu.Color(0, 1, 1)

    # clamping values to (0,1)
    u = rtu.Interval(0, 1).clamp(fu)
    v = rtu.Interval(0, 1).clamp(fv)

    # scale to pixel location and get the pixel value
    i = int(u*self.img.width)
    j = int(v*self.img.height)
    pixel = self.img.getpixel((i,j))

    # return the pixel color of the texture
    # remember to scale the value to (0,1)
    scale = 1.0/255
    return rtu.Color(pixel[0]*scale, pixel[1]*scale, pixel[2]*scale)
```

Image as a texture

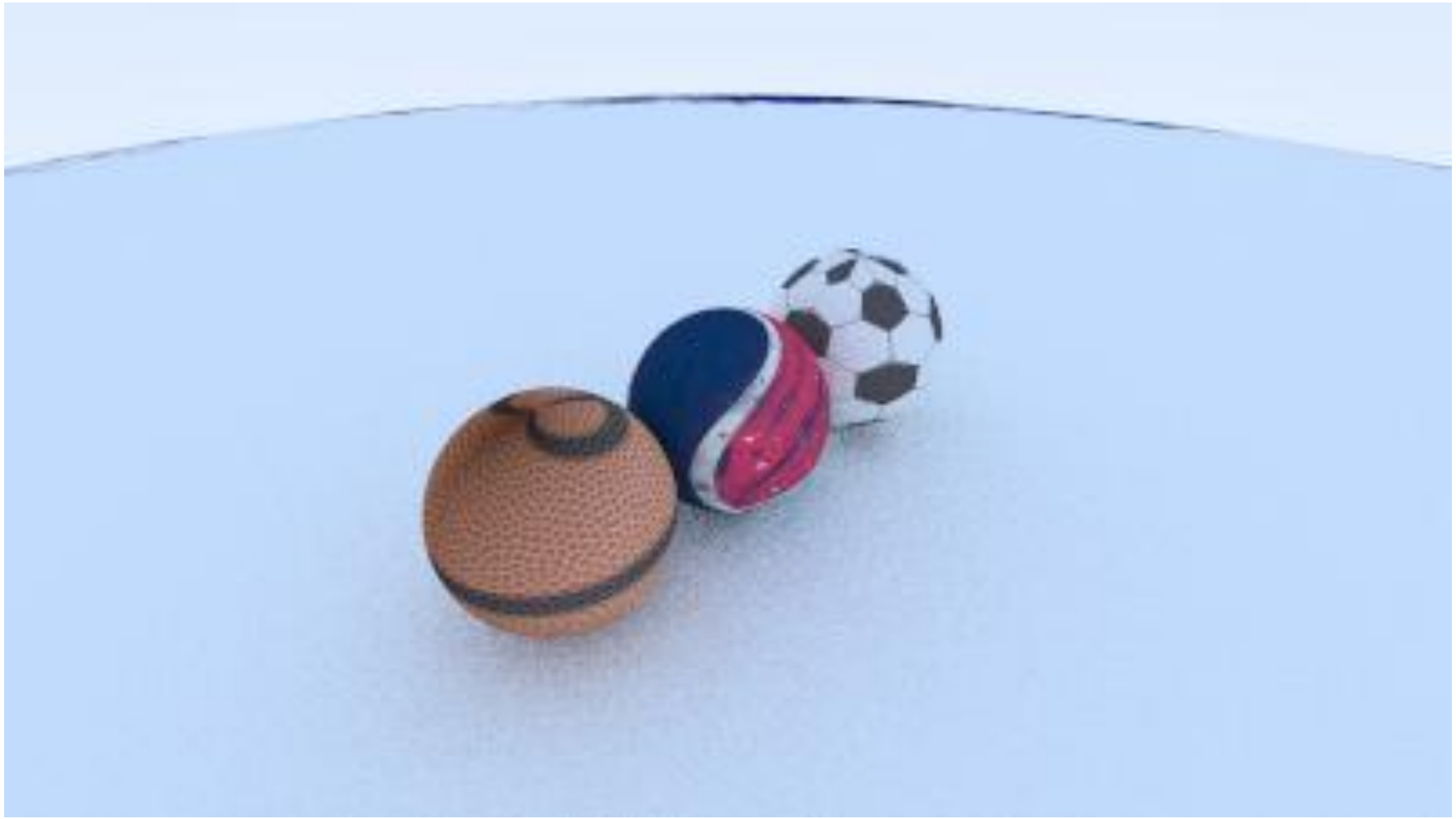
```python
# checker board as a texture
class CheckerTexture(Texture):
    def __init__(self, fScale, cColor1, cColor2) -> None:
        super().__init__()
        self.inv_scale = 1.0/fScale
        self.even_texture = SolidColor(cColor1)
        self.odd_texture = SolidColor(cColor2)

    def tex_value(self, fu, fv, vPoint):

        xInteger = int(math.floor(vPoint.x()*self.inv_scale))
        yInteger = int(math.floor(vPoint.y()*self.inv_scale))
        zInteger = int(math.floor(vPoint.z()*self.inv_scale))

        isEven = (xInteger + yInteger + zInteger) % 2 == 0

        if isEven:
            return self.even_texture.tex_value(fu, fv, vPoint)

        return self.odd_texture.tex_value(fu, fv, vPoint)
```

Checkboard

A little more work to finalize texture mapping

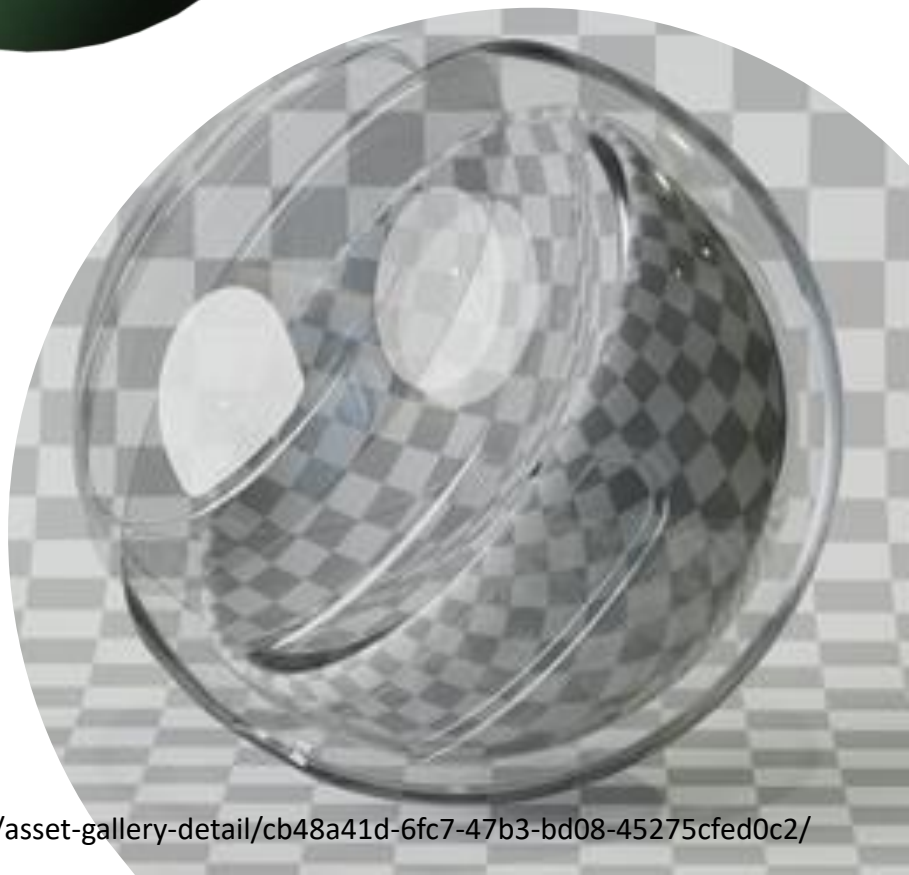
```
# a texture
class TextureColor(Material):
    def __init__(self, color_or_texture) -> None:
        super().__init__()
        if isinstance(color_or_texture, rtu.Color):
            self.albedo = rtt.SolidColor(color_or_texture)
        else:
            self.albedo = color_or_texture

    def scattering(self, rRayIn, hHinfo):
        scattered_direction = hHinfo.getNormal() + rtu.Vec3.random_vec3_unit()
        if scattered_direction.near_zero():
            scattered_direction = hHinfo.getNormal()

        scattered_ray = rtr.Ray(hHinfo.getP(), scattered_direction)
        attenuation_color = self.albedo.tex_value(hHinfo.u, hHinfo.v, hHinfo.point)

        return rtu.Scatterinfo(scattered_ray, attenuation_color)
```

Example



Reflection models

- Previously, we studied how light changes its direction on a variety of materials.
  - Matte plastic (dull-looking materials)
  - Mirror
  - Transparent medium (dielectric materials)

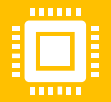


www.blenderkit.com/asset-gallery-detail/cb48a41d-6fc7-47b3-bd08-45275cfed0c2/

Surface reflection models come from a number of sources

Measured data: Reflection distribution properties of many real-world surfaces have been measured in laboratories. Such data may be used directly in tabular form or to compute coefficients for a set of basis functions.

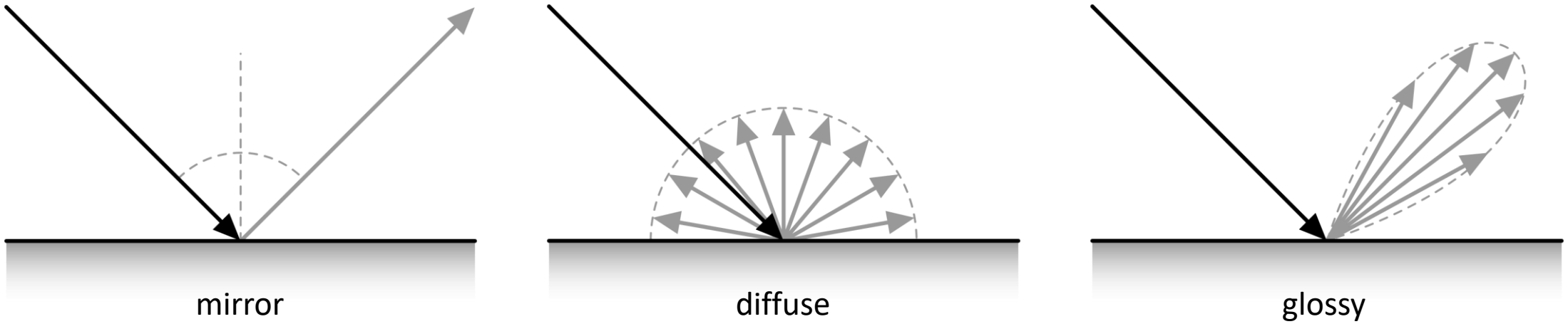
Phenomenological models: Equations that attempt to describe the qualitative properties of real-world surfaces can be remarkably effective at mimicking them.



Simulation: Sometimes, low-level information is known about the composition of a surface. For example, we might know that a paint is comprised of colored particles of some average size suspended in a medium or that a particular fabric is comprised of two types of threads with known reflectance properties.

Physical (wave) optics: Some reflection models have been derived using a detailed model of light, treating it as a wave and computing the solution to Maxwell's equations to find how it scatters from a surface with known properties.

Geometric optics: As with simulation approaches, if the surface's low-level scattering and geometric properties are known, then closed-form reflection models can sometimes be derived directly from these descriptions.



Types of reflection (Recap)

- Diffuse reflection
  - Equally scattered ray in all directions.
- Specular reflection
  - Weighted scattered ray based on the surface parameters.

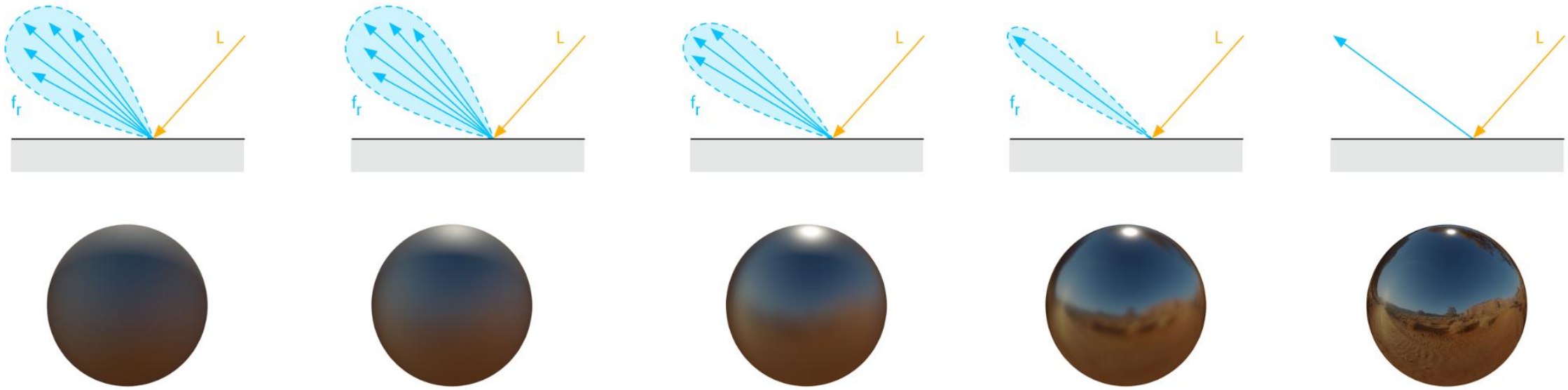
en.wikipedia.org/wiki/Bidirectional_reflectance_distribution_function

How much light is scattered ?

- Integration over all possible scattered rays.
- We implemented this part as scattering properties of materials.

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_\Omega \underbrace{\rho(p,\omega_o,\omega_i)}_{reflectance} \underbrace{L_i(p,\omega_i) \mid cos(\theta_i) \mid d\omega_i}_{irradiance}$$

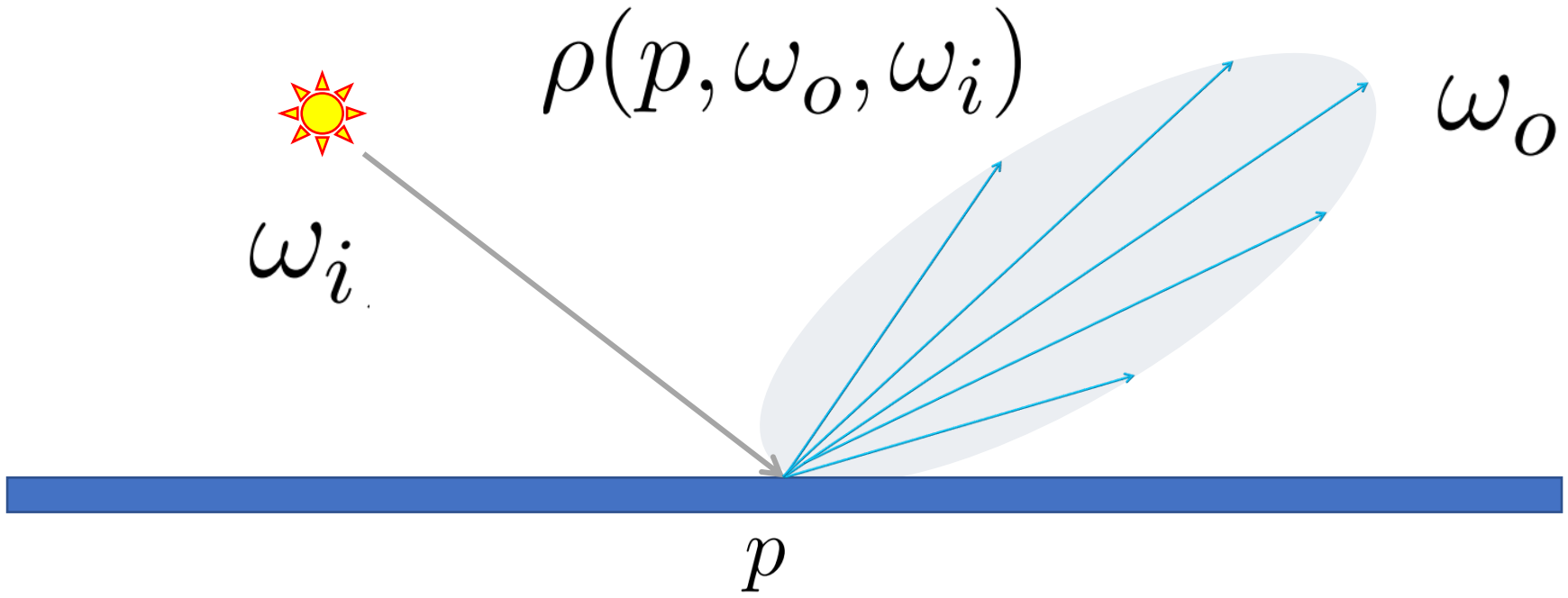
- The quantity that light scatters in the direction.
- This parameter can be represented by reflection models.



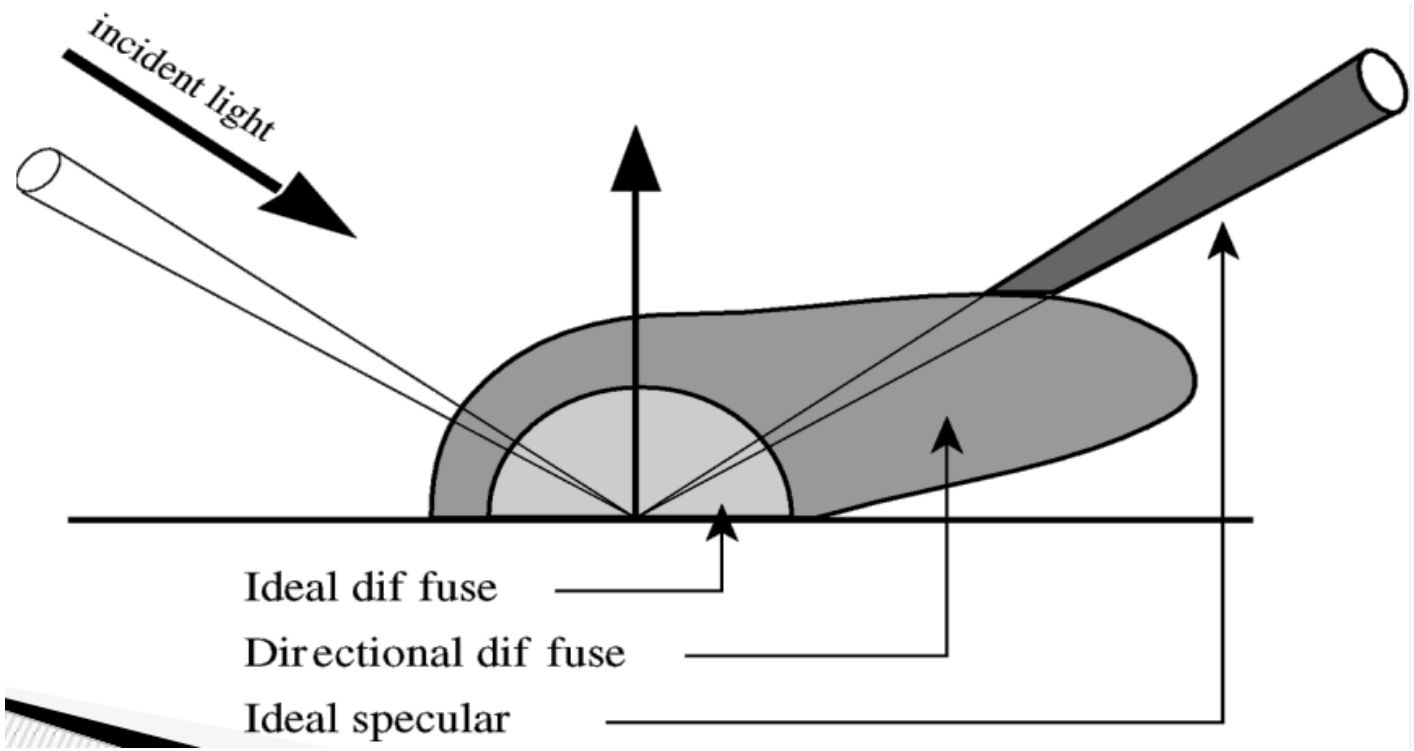
Bidirectional Reflectance Distribution Function

- BRDF is a function describing how much light scatters for a given pair of directions.
- A pair of directions is defined by incoming light direction and outgoing (exitance) viewing direction.

Illustration - BRDF



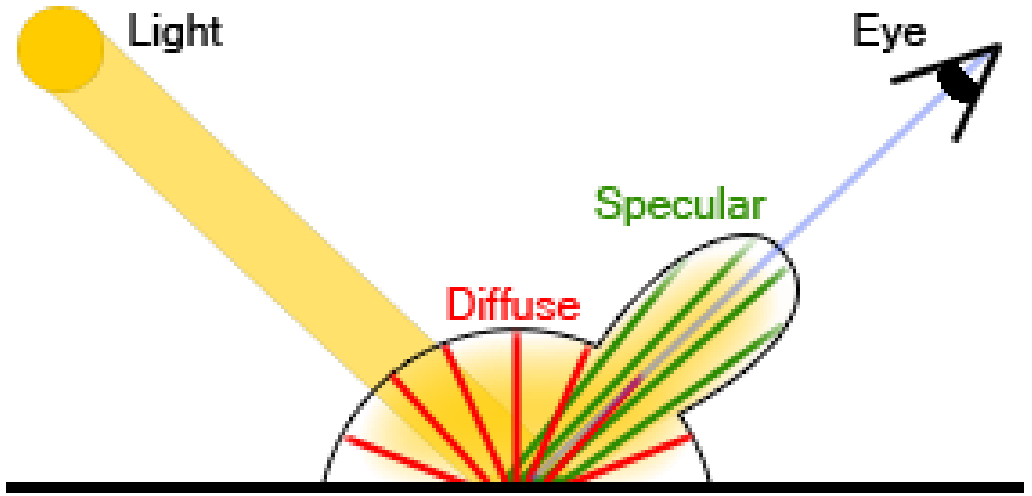
BRDF lobes



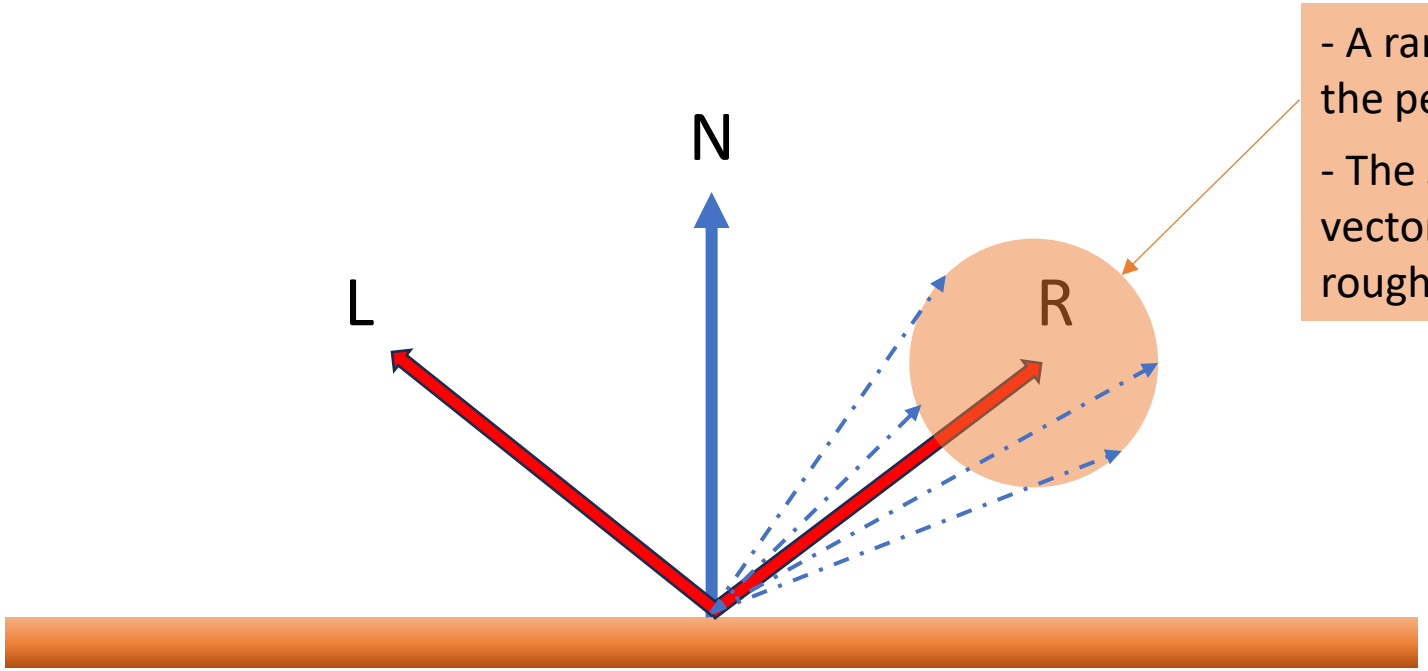
maverick.inria.fr/Members/Nicolas.Holzschuch/cours/Slides/1b_Materiaux.pdf

A metal class

- This material exhibits some specular reflection depending on a roughness parameter.
- Attributes
  - Color
  - Roughness
- Methods
  - scattering()



Specular reflection



- A random unit vector around the perfect reflection vector.
- The size of the random vector depends on the roughness parameter.

Codes and class assignment !

- Github : RT-python-week07
  - https://github.com/KUGA-01418283-Raytracing/RT-python-week07