Foundations of 3D Scene Modeling

Recap

Cornerstones of image generation:

- **3D scene**
- Rendering algorithm
- Raster image

<IMAGE: high-level overview of three main components>

Where we are today

<IMAGE: tree-like structured knowledge of the course>

Introduction

Elements of any 3D scene:

- 3D model(s)
- Light source(s)
- Camera(s)

<IMAGE: high-level overview of three main components>

3D scene

- 3D scene modeling goes hand in hand with object oriented design.
- 3D scene representation has inherent tree-like structure thus often represented with so called **scene-graph**
  - Book: Foundations of Game Engine Development: Rendering (E. Lengyel)

<IMAGE: COMPONENTS OF 3D SCENE AND SCENE GRAPH>

Complex scene

<IMAGE: An motivation image that we will understand by the end of the lecture.>

3D Models

Foundations of 3D models

To create objects in a 3D scene, we need a way of representing:

- Shape
- Material

3D models creation is tightly connected to rendering:

- From the user point of side: we need a representation of shape and material which is intuitive to handle (author)
- From the rendering point of side: we need to make sure that shape and material representation can be used for rendering process
  - Shape representation will be used for determining visibility problem*: what objects we see from certain point of view
  - Material representation will be used for determining the shading problem**: how the objects that we see appear

Representing 3D models requires:

- Representation of **shape** - surface geometry
- Representation of **material** – surface appearance

* As we will get to know more complex material representations we will see that this is not completely true. Visibility of objects will also depend on material. Imagine glass cup on the top of a book – book will be visible since glass is transparent.

** Shape and material information are both important for calculating how objects appear. We separate them to show that you can first purely focus on modeling a 3D shape and then model material afterwords.

Shapes

- Description of 3D objects in the scene can be done using:
  - Surface representation
  - Volume representation
- Based on phenomena that we are modeling, we can choose different representations
- Simple shapes can be described mathematically (parametrically), e..g., spheres, disks, planes
- Complex shapes are described using different representations: polygons (mesh), subdivision surfaces, curved surfaces, voxels, SDFs, etc.

<IMAGE: DIFFERENT OBJECTS, DIFFERENT REPRESENTATIONS>

- For now, we will focus on fundamental representations for surface: meshes and curves

Foundations of 3D surface representation

Foundational shape representations in geometrical modeling are*:

- Meshes
- Curved (subdivision) surfaces

*Note that these representations are used to describe surface of the shape (a manifold – 2D surface in 3D world). Later, we will discuss how to describe interior of object (its volume). Interior of object can be described purely with spatially varying material enclosed in described surface. Also, advanced shape representations (e.g., voxels) can be used to efficiently describe the mesh. Since the topic of volumetric representation requires more knowledge about material and/or advanced shape representations, it will be covered later.

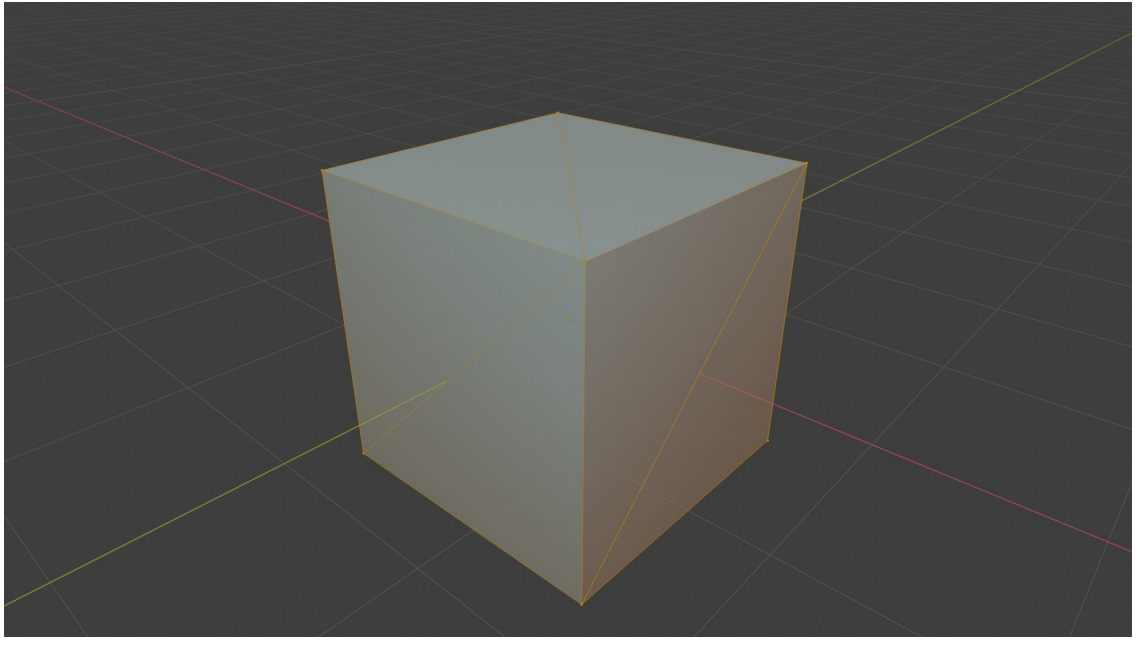
Foundations of Meshes

- Pros:
  - Simple, a lot of effort has been made to represent various shapes with meshes
- Cons:
  - Not every object is well suited to mesh representation:
    - Shapes that have geometrical detail at every level (e.g., fractured marble)
    - Some objects have structure which is unsuitable for mesh representation, e.g., hair which has more compact representations
- Common types:
  - Triangle mesh
  - Quad mesh

Triangle mesh introduction

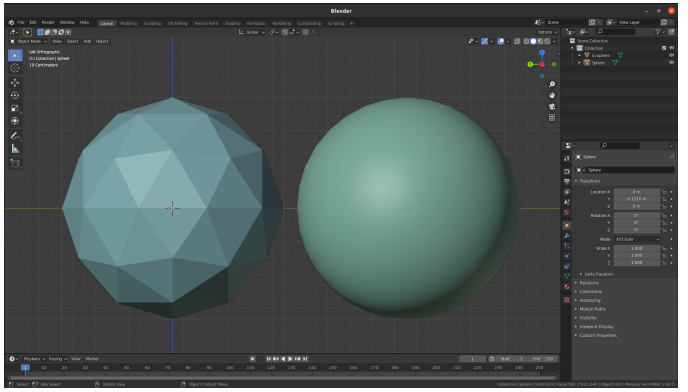
- Triangle mesh is foundational and most widely used data-structure for representation of a shape in graphics
- Triangle mesh consists of many triangles joined along their edges to form a surface
- Triangle is fundamental and simple primitive:
  - All vertices lie in the same plane
  - GPU graphics rendering pipeline is optimized for working with triangles
- Triangle mesh has nice properties:
  - Uniformity: simple operations
    - Subdivision: single triangle is replaced with several smaller triangles. Used for smoothing
    - Simplification: replacing the mesh with the simpler one which has the similar shape (topological or geometrical). Used for level of detail

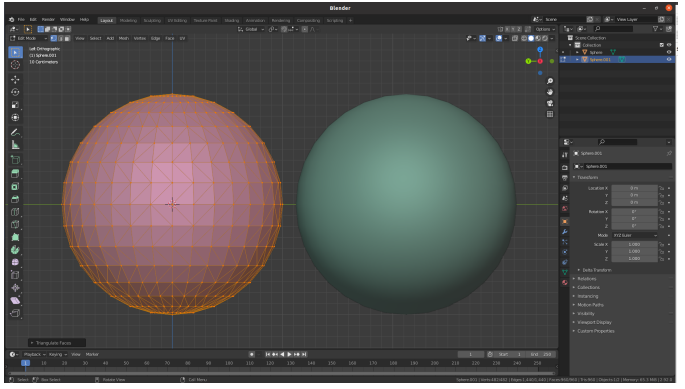
Example how certain flat shapes are created with triangles



Example how certain curved shapes are approximated with triangles

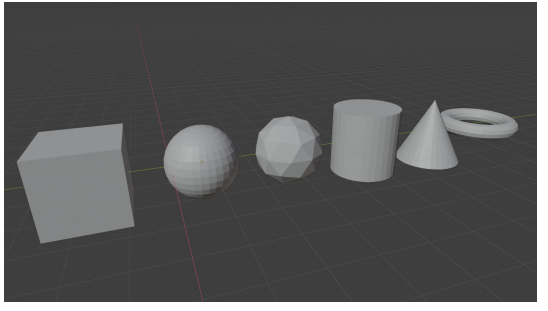
- Conceptual approximation: find points on complex shape and connect adjacent points with a mesh structure
  - For example: scanning and reconstruction
- Example: sphere vs icosahedron
  - Each point on icosahedron is close to point of sphere
  - Each normal vector of icosahedron is close to vector normal of the sphere in the same point. But, function that assigns normals to the sphere is continuous while for icosahedron is piecewise constant → this influences reflection of light!





Common basic shapes

- Now we can understand how to represent basic shapes using triangle meshes
- Every DCC Tool provides basic shapes:
  - Blender[1], Maya[2], 3DSMax[3], Houdini[4], etc.



1. https://docs.blender.org/manual/en/latest/modeling/meshes/primitives.html
2. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2022/ENU/Maya-Basics/files/GUID-45D2EAD4-5BCF-42DA-A1AB-EC6EE09FE705-htm.html
3. https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2021/ENU/3DSMax-Modeling/files/GUID-66152BDE-BA64-423F-8472-C1F0EB409E16-htm.html
4. https://www.sidefx.com/docs/houdini/model/create.html

Complex shapes?

- How to represent complex shapes with triangle meshes?
- Digression: drawing complex form (3D shape)
  - Anything can be decomposed in simple forms[1,2]: box, sphere, cylinder, torus, cones, etc.

1. http://www.thedrawingwebsite.com/2015/02/18/practicing-your-draw-fu-forms-forms-are-like-sentences/

2. https://www.youtube.com/watch?v=6T_-DiAzYBc&list=RDCMUClM2LuQ1q5WEc23462tQzBg&start_radio=1&rv=6T_-DiAzYBc&t=1343&ab_channel=Proko

Complex shapes

- https://www.youtube.com/watch?v=Q0qKO2JYR3Y&ab_channel=BlenderSecrets

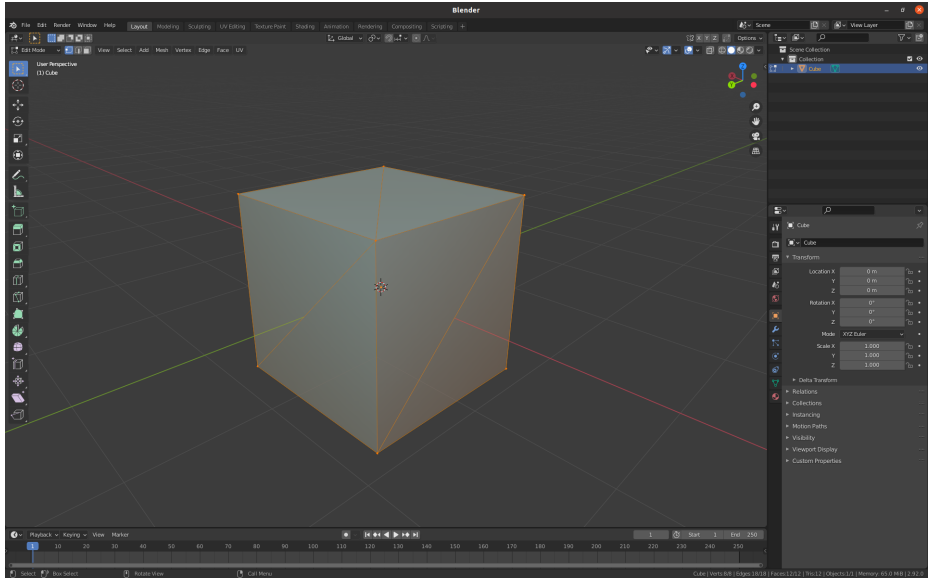
Modeling of complex shapes in DCC Tools

- Choose right basic shape: https://www.youtube.com/watch?v=DcyY4RAHcA4&ab_channel=CGCookie
- Modeling with basic shapes (Blender):
  - https://www.youtube.com/watch?v=AD3gn2AyzgA&ab_channel=LeeDanielsART
  - Procedural (and funny one): https://www.youtube.com/watch?v=Hf8s1Ckycdo&ab_channel=CGMatter
- Modeling with basic shapes (Maya)
  - https://www.youtube.com/watch?v=j3jwVfN8EcU&ab_channel=AnetaV
- Procedural shapes (Houdini):
  - https://www.youtube.com/watch?v=afHVjiNeH7A&ab_channel=AdrienLambert
  - https://www.youtube.com/watch?v=fxOxygaEOFk&ab_channel=SimonHoudini
  - Note that other geometry representation are used as well

Basic description of meshes

Description of mesh requires:

- List of vertices and triangles (edges are inferred from triangles)
  - Vertex table – geometry
  - Triangle (faces) table - topology




```
# Blender v2.92.0 OBJ File: ''
# www.blender.org
o Cube_Cube.002
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v -1.000000 1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v 1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
vt 0.625000 0.000000
vt 0.375000 0.250000
vt 0.375000 0.000000
vt 0.625000 0.250000
vt 0.375000 0.500000
vt 0.625000 0.500000
vt 0.375000 0.750000
vt 0.625000 0.750000
vt 0.375000 1.000000
vt 0.125000 0.750000
vt 0.125000 0.500000
vt 0.875000 0.500000
vt 0.625000 1.000000
vt 0.875000 0.750000
vn -1.0000 0.0000 0.0000
vn 0.0000 0.0000 -1.0000
vn 1.0000 0.0000 0.0000
vn 0.0000 0.0000 1.0000
vn 0.0000 -1.0000 0.0000
vn 0.0000 1.0000 0.0000
s off
f 2/1/1 3/2/1 1/3/1
f 4/4/2 7/5/2 3/2/2
f 8/6/3 5/7/3 7/5/3
f 6/8/4 1/9/4 5/7/4
f 7/5/5 1/10/5 3/11/5
f 4/12/6 6/8/6 8/6/6
f 2/1/1 4/4/1 3/2/1
f 4/4/2 8/6/2 7/5/2
f 8/6/3 6/8/3 5/7/3
f 6/8/4 2/13/4 1/9/4
f 7/5/5 5/7/5 1/10/5
f 4/12/6 2/14/6 6/8/6
```

Normals and appearance

- We mentioned that both shape and material are used for calculating appearance (shading step in rendering).
- Orientation of surface towards light determines how bright it is (again, appearance).
- Orientation of surface in each point is determined by normal vector.
- Normal vector is core information that we can obtain from shape representation
- <IMAGE OF SURFACE NORMALS>
- Normal in surface point is vector perpendicular to tangent in that surface point
- Light direction and normal direction determine amount of brightness – facing ratio → this is core step in shading process as we will discuss later, now we focus on this geometrical nature of normal.
- TODO: how we calculate normal (sphere intuition)
- TODO: how we calculate normal for triangulated meshes
  - TODO: face normals vs vertex normals

Storing and transferring mesh objects

- Mesh datastructure can be stored in various formats which:
  - Are more or less compact
  - Are more or less human-readable
  - Can contain additional object data which is described with the mesh (textures, materials, etc.)
  - Can contain various metadata (e.g., physical behavior of object described with mesh)
  - Store only mesh information
  - Store whole scene and mesh is only one of elements
- Popular formats:
  - https://all3dp.com/2/most-common-3d-file-formats-model/
  - https://www.sidefx.com/docs/houdini/io/formats/geometry_formats.html
- 3D scene is not necessary created, rendered and used in same software. Usually, whole pipeline of software is used, at least:
  - DCC → game engines
- Formats: OBJ, GLTF, USD

Important properties of meshes

- Goal: not to go deep into definitions but rather to verify properties using simpler methods
- **Mesh boundary**: formal sum of vertices
- **Closed mesh**: mesh boundary is zero. Required for defining what is "inside" and "outside" by winding number rule
- **Manifold mesh**: each vertex has arriving and leaving edge
  - Manifolds are desired since it is easy to work with them (both manually and algorithmically)
  - Smooth vs not smooth manifolds (e.g., cube)
  - Self-intersecting meshes are not manifolds
  - In graphics we generally use polyhedral manifolds
- **Oriented vs unoriented meshes**
  - We use oriented meshes so that boundary can be defined

Quad mesh

- Often used as a modeling primitive
- Complexity:
  - Easy to create a quad where not all vertices lie on a plane
- In graphics pipeline it is always transformed to triangle.
- In raytracing rendering plane-ray intersection must be defined

Foundations of Curved surfaces

- Representing surface using mesh is the most used and widespread option for both authoring and transfer.
- Triangle mesh and thus triangle is basic atomic rendering primitive for GPU graphics pipelines and most raytracers.
- However, objects made in modeling systems can have many underlying geometric descriptions.
  - Different geometric descriptions enable easier and efficient modeling and representation of shapes on the user side.
  - On the rendering side, all higher-level geometrical descriptions are evaluated as set of triangles and then used.

<IMAGE: INTRODUCE THE CONCEPT OF USER AND RENDERING SIDE AND HOW OBJECTS CAN BE REPRESENTED>

- Curves and curved (subdivision) surfaces are one of alternative geometric representations that have certain advantages over meshes in certain scenarios.
- Some advantages of curves and curved (subdivision) surfaces are:
  - They are represented by equations and thus have more compact representation than meshes (less memory for storing and transfer) and less transformation operations are needed (
  - Since they are represented by equations they provide salable geometric primitives – geometry can be generated on the fly by evaluating the equations (Analogy: vector and raster images)
  - They can represent smoother and more continuous primitives than lines and triangles, thus more convenient for representing object like hair, organic and curved objects
  - Other scene modeling tasks can be performed more simpler and faster, e.g., animation and collision

<IMAGES: APPLICATION OF CURVES AND CURVED SURFACES>

- To understand curved surfaces, we will start with curves

Fundamental types of Curves

- Bezier
- Hermite
- Catmull-Rom spline
- B-Splines

Foundations of subdivision surfaces

- Generalizing spline curves and subdivision curves leads to spline and subdivision surfaces
- Bezier patches
- Catmull-Clark subdivision surfaces

Foundations of materials: observation of world around us

- Modeling materials begins with observation to understand what makes each material look different than other materials
- We don’t want to observe just appearance (like artists do). We want to observe characteristics which are responsible for object appearance:
  - Shape
  - Illumination
  - Sensor/Perception
  - **Material**
- **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
  - Directional
  - Spatial

<BOOK: DIGITAL MODELING OF MATERIAL APPERANCE (J. DORSEY)>

Foundations of materials: physics

- Material defines **interaction of light with objects**
  - Light is electromagnetic wave (visible part of spectrum). In computer graphics we work with **geometrical optics** which is approximation and light is represented using **rays**.
- Light falling on objects, based on **index of refraction** will*:
  - **Absorb**
  - **Scatter**
- **Homogeneous material:** light travels on a straight line, it can only be absorbed, meaning that direction is same, but intensity might be lower.
  - In **transparent material** (e.g. glass or water) light propagates in straight line – no scattering (scattering happens on interface – boundary between materials). Also, low absorption of light.
  - Homogeneous material with higher absorption will attenuate light traveling through it or completely absorb it if distance is large – but the direction will not change! An example is water over larger distances.
  - Note that also light can be selectively absorbed, meaning that it changes color (alongside intensity).
- **Heterogeneous material**: light is scattered → direction of light is changed but not necessary the intensity
  - Light can scatter in all directions, mostly non-uniformly: forward or back scattering
  - **Translucent** or **opaque materials** are examples of heterogeneous materials → light is scattered so much that we can not see (clearly) through the object
  - Longer distances cause more scattering, but not necessary more absorption (e.g., clean air)

<IMAGES OF REAL WORLD OBJECT SHOWING THOSE CHARACTERISTICS>

* Light interaction with matter (on geometrical optics) is determined by index of refraction – complex number. Real part determines speed of light, imaginary part determines absorption.Constant index of refraction is present in homogeneous materials where index of refraction is constant. For such materials, only absorption happens. Varying index of refraction is present in heterogeneous materials causing scattering (next to absorption).

Foundations of materials: reflection on optically flat surfaces

- Light interaction with matter can be described with Maxwell's equations. Those are too heavy for computer graphics thus we work with (1) geometrical optics approximations and (2) special case which is can be simplified and used in graphics for material modeling.
- Perfectly planar (optically flat, perfectly smooth) surface* between two homogeneous materials with different index of refraction is starting point for describing the behavior of light interaction with surface. This special case is described with Fresnel's equations:
  - Light falling on such surface can: reflect and refract
  - Amount and directions of light depend on index of refraction

<IMAGE: REFLECTION AND REFRACTION>

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

Foundations of materials: reflection general surfaces

- 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)

<IMAGE: variation of facets and roughness>

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

<IMAGES: REFLECTION ON METAL AND DIELECTRIC SURFACES>

<IMAGES: real world objects with such properties>

<IMAGE: reflection from metal and dielectric surface: not the color of reflection!>s

* 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.

Surface material representations in 3D scene

- To simplify and understand modeling of 3D object, we separated 3D object in shape and material.
  - By material we mean characteristics of object independent of shape and position. For example, aluminum sphere and aluminum statue – in both cases aluminum properties are the same. Also, changing position of aluminum sphere in space doesn't change aluminum properties.
  - 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 and reuse this knowledge at other points.
  - Having same description of material for each point of the 3D object, we would end up with homogeneous material. 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.
- By Material modeling we describe how light should behave when it interacts with objects. For now we will limit our discussion to light interaction with surfaces, and leave light interaction with volume for later.
- Description of light interaction with surface, when considered locally, is called scattering. Once we have a scattering model, we can parameterize it and vary its properties over the surface.
- Shading step in rendering takes material information (alongside with viewing position, object shape and light information on the object) to calculate the object appearance → object color
- Based on discussion above, we can separate material in:
  - **Scattering** → description of light-matter interaction in a point
  - **Texture** → variation of scattering function properties across 3D object

Material: scattering function

- We already know that surface orientation (normal) is important shading information since it gives us fundamental information of object appearance – how much is lit.
- How light scatters when it falls on surface point is defined by **scattering function** which takes in account the normal in this point.
- Scattering function can be separated in reflection and refraction. For now we will focus on reflection. Such scattering function is generally called "bidirectional reflectance distribution function" – **BRDF**. Defining this function is fundamental and core part of shading process in rendering.

<IMAGE OF BRDF AND ITS ELEMENTS>

Types of scattering

- **Reflective** – all light is scattered above surface
- **Transmissive** – all light is scattered below surface
  - Refractive – special case of transmissive
- **Mirror-like** (specular) – 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
- **Glossy** – scattered light is concentrated around particular direction (lobe). Appears blurred. Dependent of viewing direction.
- **Diffuse** – light is scattered in all possible directions. Independent of viewing direction. Equally bright from all directions.
- **Retro-reflective** – scattered light is particularly large when viewing and light directions are close

TODO: IMAGES OF NORMAL AND LIGHT REFLECTION TYPES WITH REAL WORLD IMAGES

TODO: BLENDER EXAMPLE OF BASIC SCATTERING FUNCTIONS

Scattering models

- Empirical
  - Mirror
  - Lambertian
  - Phong and Blinn-Phong
- Data based
  - Measurements
- Physically based
  - Microfacets*

* just announce, details in “More on 3D scene modeling”

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.
- By now, we have seen homogeneous description of material using scattering functions. And objects looks too perfect (too smooth and artificial).
- In order to obtain variation of material across surface (spatial variation) we can vary scattering functions or its properties over it. One material parameter is color. In graphics, we use **texture function** to vary material properties over surface.

<IMAGES OF HOMOGENEOUS MATERIAL AND MATERIAL WITH VARIATION>

<EXAMPLE IN BLENDER WITH AND WITHOUT TEXTURES>

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 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.

<IMAGE: IMAGE TEXTURE AND RENDERED OBJECT WITH TEXTURE>

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.

Image textures

- Texture information can be created in form of images:
  - Drawing images
  - Taking photographs
- Once image textures are created, we need to map them on the 3D object:
  - UV parametrization of 3D object
    - Assign texture coordinates to vertices of 3D object
    - Unfold/unwrap surface onto plane
    - Separate object into patches on which texture is applied
    - Paint texture directly on texture
  - Planar mapping
  - Cylindrical mapping
  - Spherical mapping

Procedural textures

- Procedural texturing is based on algorithm which defined values for each point of 3D space or object surface. Examples:
  - Fourier-like synthesis
  - Perlin noise
  - Reaction-diffusion

<IMAGES: PROCEDURAL TEXTURING>

<IMAGES: NODE SYSTEM FOR PROCEDURAL TEXTURING>

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

Light

Foundations of light

- Reminder: when we discussed the material of 3D models, we mentioned shading process which uses material information and light information to calculate appearance of the object surface.
- Light is emitted from a light source, bounces around the scene, interacting with object and some (very small!) portion enters eye or camera enabling us to see objects in the scene
  - Without light in a 3D scene (or real world) we wouldn't see anything. Resulting image would be completely black.
- Light sources define position and distribution of light

Sources of light

- In a real world, every light source has a physical shape and size, e.g., **Sun** or very hot objects.
- Motion of atomic particles that hold electrical charge causes objects to emit electromagnetic radiation over a range of wavelengths (Maxwell's equations)
- Different way how energy is converted into electromagnetic radiation:
  - **Incandescent (tungsten) lamps**: flow of electricity through tungsten filament heats is up and causes to emit electromagnetic radiation with distribution of wavelengths depending on filament temperature. A frosted glass enclosure is often present to absorb some of the wavelengths.
  - **Halogen lamps**: tungsten filament is enclosed in halogen gas. Part of the filament in an incandescent light evaporates when it's heated. Halogen causes evaporated tungsten to return to filament
  - **Gas-discharge lamps**: passing electrical current through hydrogen, neon, argon, or vaporized metal gas, causes light to be emitted at specific wavelengths that depend on the particular atom in the gas. Fluorescent coating on the bulb's interior is often used to transform the emitted frequencies to a wider range.
  - **LED lights** are based on electroluminescence: they use materials that emit photons due to electrical current passing through them.

Light models

- Discussion before showed that lights have shape, size and certain light distribution.
- Representation and modeling of light physically: with shape and size is very computationally expensive. Such lights are called **geometric area light or physical lights***
  - Interaction of physical lights with the scene can not be computed in closed form. Advanced, approximation-based methods such as Monte Carlo integration must be used
- Simplification are lights with no physical size, thus called **delta or non-physical lights**
  - enable us to control light fall-off with distance, which objects are illuminated, which objects cast shadows and so on.

* When we discussed surface material we mentioned scattering and absorption. Also, materials are able of emission. Therefore, physical lights have 3D shape and material which is emissive. Emissive material is modeled as black body meaning that is absorbs all light falling on it.

Non-Physical lights

- Directional (distant) lights
- Point (spherical, omnidirectional) lights
- For physically-based rendering, non-physical light should be avoided
  - Example: size of reflection of object by glossy or mirror-like surface depends on its size and distance to reflective surface. If light source has no shape nor size how reflection should look like? Hack: size parameter which is only used during shading.

Directional lights

- Directional lights are also called distance because they are considered so far from the objects in a 3D scene that they can be represented by parallel rays. As such, we only care of the direction of those parallel rays.
  - Best example of directional light is Sun light on Earth. Sun has spherical shape, but it is so far and Earth is so small compared to it that light rays reaching Earth can be considered parallel (small cone of directions – solid angle).
  - For scenes that cover small area of Earth's surface, Sun light can be assumed parallel.
  - General note: this is another example showing that it is always important to assess the application of Computer graphics and see what is really important to model.

<IMAGE: EXAMPLE OF DISTANT LIGHT>

<IMAGE: EXAMPLE OF DIRECTIONAL LIGHT VECTORS IN THE SCENE>

Point lights

- Most common light sources in nature and around us are spherical (e.g., light bulbs or candle flame) or can be approximated as a collection of spherical light sources.
- As opposed to directional lights:
  - Point light position is important parameter, it is considered infinitesimally small in size
  - Light is emitted in all directions (omnidirectional, isotropic) therefore, no "direction" parameter
- Point light position determines:
  - Direction of incoming light ray for each surface position for each object in the scene
  - Distance to the each surface position for each object in the scene – light falloff

Light intensity and color

- For each light source (physical or non-physical) next to shape, size, position and direction we need to define:
  - Intensity – float value in [0, inf]
  - Color – RGB value in [0,1]

Point lights: light direction and falloff

- Point lights emit radially
- Direction and distance of point light to point on the surface is obtained by tracing ray between these two points.
- Distance of point light to each point of surface of each object in the scene defined how much is that point illuminated
- Energy emitted from point light is distributed across the space as sphere. As the sphere keeps expanding in the space, energy becomes spread across much larger area → more distant objects in the scene receive less light – light falloff. <INVERSE SQUARE LAW FORMULA>

<IMAGE OF POINT LIGHT, ILLUMINATED OBJECTS, SOLID ANGLE>

Point lights: other derivations

- Interesting light sources can be derived by angularly varying distribution of outgoing light:
  - **Spotlights** - emit light in a cone of directions from their position
  - **Textured projections**
  - **Goniophotometric diagram** describes the angular distribution of luminance from a point light source; it is widely used in illumination engineering to characterize lights

Light and shadow

- Light falling on object causes object to cast shadow – blocking light to fall on another surface

<IMAGE WITH AND WITHOUT SHADOW>

- Shadows are important for:
  - Understanding relation of objects to one another
  - Realistic image synthesis
- Simulating shadow depends on rendering algorithm (more specifically, algorithm for solving the visibility problem)
  - As we will see, in rasterization-based rendering shadows will not be inherently present. Advanced, multi-pass rendering is required to obtain visibility of objects looking from the position of light. This way, shadow map is generated and used for rendering.
  - In ray-tracing based rendering, shadows are inherently present due to way incoming light on surface is calculated

Intuition: calculating light and shadow

- How to know if point of surface will be under shadow or not?
- Generate ray from that point in light direction and if no intersections are found, shadow won't be present.
  - For point lights, we need to take in the account their maximum influence!
  - Shadow-acne problem → Later!

Light and Shadow: Tips and Tricks

- Non-physical light sources as well as shadows they cause can be simulated efficiently
- For real-time rendering, full physically based simulation of light transport might not be feasible
- When creating 3D scene, (environment) artists understand the how the scene should be illuminated given some main sources of light. Using this knowledge they place non-physical light to fake actual light flow through the scene
- https://80.lv/articles/dishonored-interiors-lighting-props/

Camera

Foundations of camera

How camera works

Pinhole camera model and parameters

Cameras introduction

- In rendering camera model is required to, at least, define a portion of visible scene and perspective/orthographic projection.
- Now, we will understand how cameras work and how to simulate a real world camera.
  - Such camera model is similar to ones used in production software (e.g., Blender, Maya, 3DSMax, Houdini)
- Simulating real world camera is important for photo-realistic rendering which may be combined with live action footage. Also, camera effects enable certain expressive and artistic possibilities.

Camera model

- Image generation with real world camera is governed by optical laws → very costly to simulate
- Thus, we start with simplest camera model: **pinhole camera**
  - Much easier way to reproduce images, therefore used in most 3D applications and games
- Pinhole camera can be realized in a real world: small box with hole on one side and photographic film on opposite side.

<IMAGE OF PINHOLE CAMERA AND ITS RESULT>

- To simulate creation of image in a camera depends on:
  - Light traveling in space and its interaction with objects (matter) → determined by law of optics
  - Light which is entering the camera
- Once light enters camera, two main processes are important:
  - How image is stored on film → can be simulated (e.g., https://maxwellrender.com/) but out of scope
  - How image is formed in the camera

Image forming basics

<IMAGE OF BASIC PRINCIPLE OF IMAGE CREATION>

- Light rays from the world pass through the small hole and form an (inverted) image on plane (film) opposite to the hole.
  - Camera obscura is real-world realization of the described process without film

Pinhole Camera

- Real world realization: lightproof box with very small hole – aperture, and light-sensitive film on the opposite side.
  - To take an image, open aperture to expose the film to light
  - Aperture is small so that only one ray reflected from the world in point P enters the camera and intersects film in one point – each point in the visible portion of the scene corresponds to a single point on the film (note that in real world such hole must be very small)
  - Geometrically, pinhole is called center of projection – all rays entering the camera converging at this point and diverging on the other size

<IMAGE OF PINHOLE CAMERA and its result>

Pinhole camera: aperture size

- Formed image is sharp if each point of the object maps to the one point of the film
- Ideal pinhole: aperture is so small that only one ray passes through it
  - Not possible in a real world because of diffraction
- It is never a single ray that passes the aperture – the cone of rays (its angle) is determined by a size of aperture. Smaller cone → sharper image

<IMAGES COMPARING LARGE AND SMALL APERTURE SIZE (blur, circle of confusion)>

Pinhole camera: exposure

- If aperture is very small, longer time is needed for image to form on a film.
- Time of which the aperture is open is called exposure time (or just exposure)
  - In real cameras, longer exposure can produce blurred image if camera or objects in the scene are moving
  - Simulated camera do not have problem with this since simulated light transport is considered instant, therefore, simulation of motion blur requires additional simulation
- Therefore, generally, shorter exposure time is better

<IMAGE OF MOTION BLUR>

Pinhole camera vs lens camera

- Since very small aperture requires long exposure times to form an image, it is not possible to obtain sharp images easily (e.g., if camera or objects are not perfectly still)
- Large aperture is again not a solution since blurred images will always be formed
- Solution is to use lens in front of aperture so that rays entering camera are gathered (converged) and focused them to one point on a film plane
- With lenses, aperture can be larger enabling smaller exposure time with sharp images as result

<IMAGE OF LENS CAMERA>

- Introduction of lenses also introduces the depth of field – distance between nearest and the farthest object from the scene that appears sharp in the formed image

<DEPTH OF FIELD IMAGE>

- Pinhole cameras have infinite depth of field
  - Therefore, computer generated images will be sharp. Additional simulation is needed to produce depth of field.

Pinhole camera parameters

- Now when we understand the elements of pinhole camera, we will discuss parameters controlling those elements

Pinhole camera parameters: Focal length

- Moving image plane (film plane) closer to aperture effectively performs zoom out
- Moving film plane away from aperture effectively performs zooming in
- Therefore, distance of film plane from aperture defines amount of scene that we see
- This parameter is called focal length or focal distance

<IMAGE OF PINHOLE CAMERA WITH VARYING FOCAL DISTANCE AND RESULT>

Pinhole camera parameters: angle of view

- Zooming in and zooming out described by Focal length (focal distance) can be also described by angle of the apex of triangle defined with aperture and film edges
- This angle is called angle of view or field of view
- In 3D, this triangle is actually a pyramid and we distinguish horizontal and vertical FOV

<IMAGE OF PINHOLE CAMERA WITH VARYING FOV AND RESULTING IMAGES>

Observation - Pinhole camera: alternative representation

- Triangle introduced in FOV defines how much of the scene is visible
- This triangle can also be viewed as continuation of lines from film edges to aperture and to a scene
- This representation of pinhole camera model is used for simulation

<IMAGE COMPARING TWO REPRESENTATIONS OF PINHOLE CAMERA>

Pinhole camera parameters: film size

- Amount of scene that is captured also depends on film size (image sensor)
  - Film parameters are horizontal and vertical direction
- Smaller surface of film size implies smaller angle of view
- Larger film formats were developed for more details and better image quality
- Capturing the same extent of the scene with larger film requires adjusting focal length

<IMAGE SHOWING DIFFERENT FILM SIZES>

Pinhole camera parameters: quick recap

- Focal length (focal distance)
- Angle of view (field of view)
- Film size
- All three parameters are interconnected, knowing two we can infer the third.
- Angle of view is parameter we usually need for rendering and expose focal length and film size to the user

Pinhole camera parameters: image resolution and aspect ratio

- As discussed, size of film (image sensor) has an effect on angle of view
- Number of pixels (resolution of image) placed on image sensor doesn't have influence on angle of view
- Image quality depends both on image sensor size and number of pixels on it (resolution)
  - Higher resolution images will have more details
- Resolution is determined by width and height which defines number of pixels
- Image aspect ratio can be computed using width and height of resolution, e.g., 4:3, 5:3, 16:9

Transformations

Transformations of 3D scene elements

Translation

Rotation

Scale

Coordinate systems

Look-at notation

Coming together

A 3D scene completed

Back to the complex example

- What have we learned

Literature

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