3D Models

3D models: intuition

Depending on application, we would like our 3D scene to contain different elements:

<IMAGE: different objects and phenomena from a real world>

- Architecture (buildings, cities, interior, etc.)

- Nature (mountains, seas, rivers, clouds, etc.)

- Engineering (car parts, airplanes, trains, etc.)

- Medicine (living beings)

How do we even start with this?

3D models: shape and material

Looking at real world phenomena, we can say that in order to create objects in a 3D scene, we need a way of representing:

- Shape, size, relative position in the space and to other shapes

- Appearance

<IMAGE: shape vs material intuition>

Therefore, we can conclude that representing 3D models requires:

- Representation of **shape** - geometry

- Representation of **material** – appearance

3D models: creation and rendering – a note

Similar as in, e.g., drawing where drawing tools and shapes needed to be chosen, 3D models creation is based on set of representations and operations from which we must choose. Those representations are on the one hand flexible enough to be used for any shape and understandable to a computer, that is rendering program:

- From the user-creation point of side: we need a representation of shape and material which is intuitive to handle (to 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

* 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. Also, separating objects in shape and material is desired for rendering architectures so that are decoupled.

Shape representation: shape and volume

- Description of 3D objects in the scene can be done using:
  - Surface representation
  - Volume representation
- Decision on shape representations depends on phenomena that we are modeling
- 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.
- Complex, natural phenomena such as clouds, mountains, trees, vegetation, galaxies can be represented using fractals

<IMAGE: DIFFERENT OBJECTS, DIFFERENT REPRESENTATIONS>

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

Shape representation: modeling

Modeling, that is, manipulation and creation of object shape representations can be done using – elaborate and extend further:

- CSG
- Regular and free form deformations

Foundations of 3D surface representation

Foundational shape representations in geometrical modeling are*:

- Polygon meshes

- Curved (subdivision) surfaces

<IMAGES: mesh vs curve>

*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

- Polygon mesh (shortly mesh) representation is one of the most oldest, popular and widespread geometry representation used in computer graphics
  - Very often, in professional DCC tools or game engines* we can find mesh representation that is used either for modeling or for rendering

<IMAGE OF MESH USAGE IN DCC AND GAME ENGINES>

- Blender: https://docs.blender.org/manual/en/latest/modeling/meshes/index.html

- Maya: https://help.autodesk.com/view/MAYAUL/2023/ENU/?guid=GUID-7941F97A-36E8-47FE-95D1-71412A3B3017

- Houdini: https://www.sidefx.com/docs/houdini/nodes/lop/mesh.html

- Unity: https://docs.unity3d.com/Manual/class-Mesh.html

- Unreal: https://docs.unrealengine.com/4.26/en-US/WorkingWithContent/Types/StaticMeshes/

* Very often, mesh is commonly used for transporting models and scenes from DCC tools to game engines. DCC tools enable modeling using different shape representations, but in a lot of cases, all shapes are transformed to mesh representation and exported to other programs.

Mesh building block: polygon

- Polygon is planar shape which is defined by connecting array of points.

<IMAGE: single polygon>

- Individual points are called **vertices** (vertex, singular)
  - In 2D they are defined using two coordinates,e.g., (x,y)
  - In 3D they are defined using three coordinates, e.g., (x,y,z)
- Lines connecting two vertices are called **edges**.
- Once edges are presented and connect vertices we can define a **face**
  - Order of connecting vertices matter and it can be clockwise or counterclockwise – **winding direction**
  - Face orientation is defined by **normal** and normal depends on winding direction

Types of polygons

- Face can have minimum three vertices

- In case of three vertices the polygon is called **triangle polygon** (shortly triangle).
  - This is very interesting type of polygon in computer graphics and we will return to this one a lot!

- In case of four vertices, the polygon is called **quad polygon** (shortly quad).
  - This is another interesting polygon that we will also shortly cover.

- Polygon with more than four vertices is called **general polygon**.
  - To make calculations easier, we desire vertices making a plane to be in the same plane. This holds true for triangle, but not necessary any other kind of polygon.
  - Polygons can be convex or concave, and more complex, they may also have holes*. We will not focus on those for now.

* Note that we are currently discussing atomic element of a polygonal mesh. It is a good practice to keep atomic elements as simple as possible (so that computation is easier) and combine those atomic elements into more complex shapes, e.g., convex or concave meshes or meshes with holes. Of course, it is generally hard to say what is better – all depend on application and requirements of application. If for some case complex polygons are needed, then using those is also approved. But in this course, we will keep to the simple building blocks and hint more complex element for you to investigate further if needed.

More complex 3D shapes using polygons

- To create more complex 3D shapes we connect faces to each other.

- Simplest example is cube: 6 faces

  - To describe a cube we need: (1) to define 8 vertices and (2) define how are those vertices connected to form faces

<IMAGE: VERTICES, CONNECTIVITY, FACES>

- Description of how vertices are defined is called **connectivity**

- Vertex and connectivity information is basic information needed for describing 3D shapes.

Representing polygonal shapes in a computer: an intuition

- As discussed, we need at least **vertices** (3D coordinate points) and **connectivity** information to represent a polygonal shape.

- As even single polygon mesh in a 3D scene can can be quite large (10^5-10^6 vertices is not unusual), storing vertices and connectivity information must be performed efficiently.

    - All vertices must be stored

    - Therefore, different techniques exist which try to minimize the amount of data needed for representing connectivity → yielding different standards, formats and API specifications for storing and transferring mesh data*.

* OBJ and FBX are popular and widely used standards. We will discuss them more when we will be talking about triangle meshes. RenderMan, on the other hand, defines API specification for representing mesh data.

Intuition: representing a cube in a computer

- Faces

- Vertices

- TODO: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-polygon-mesh

Common types of mesh representations

- Atomic element – face - of mesh can be any polygon. Common types are:
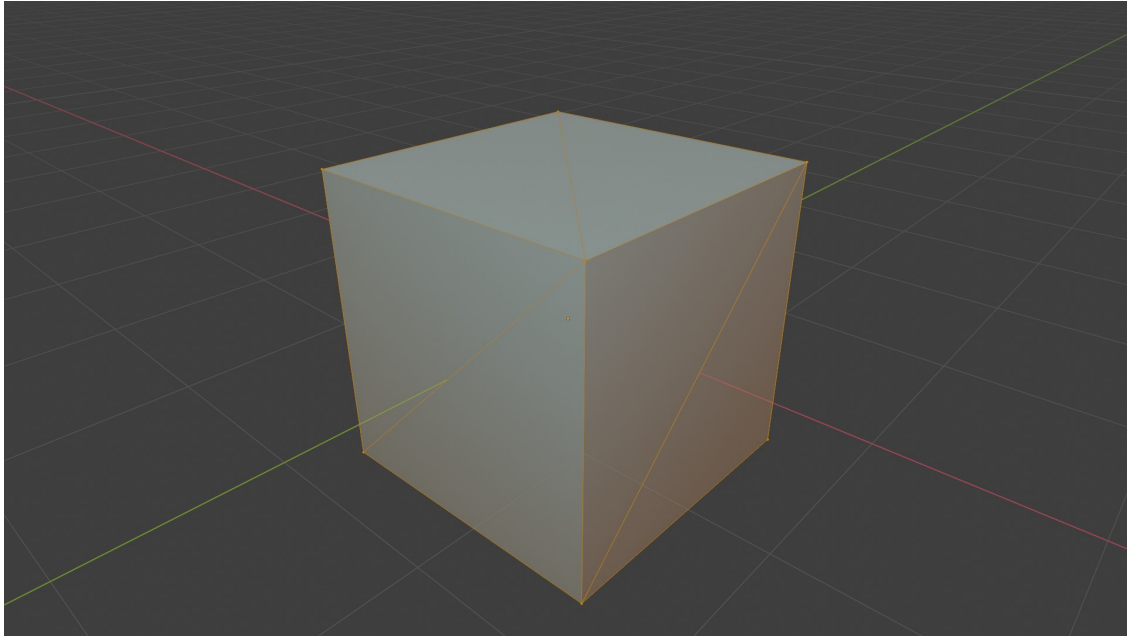 - Triangle mesh
  - Quad mesh

<IMAGE: triangle vs 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

Practical tip: how certain flat shapes are created with triangles

Practical tip: 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!
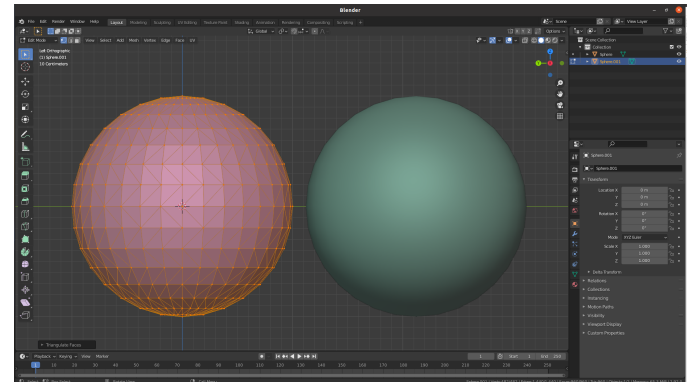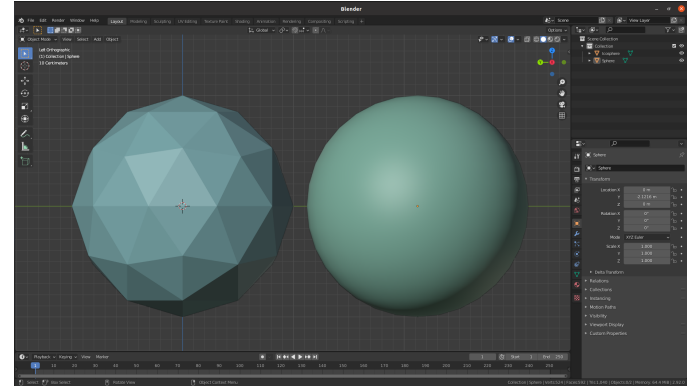
Common basic shapes

- 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.

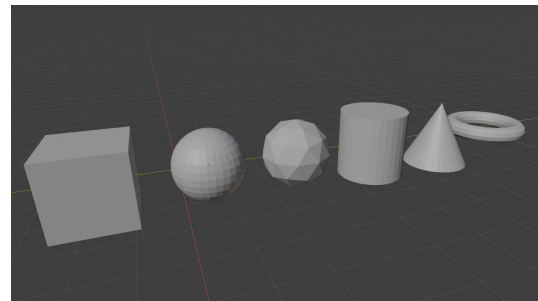1. https://docs.blender.org/manual/en/latest/modeling/meshes/primitives.html
2. https://knowledge.autodesk.com/support/maya/learn-explore/caas/CloudHelp/cloudhelp/2022/ENU/Maya-Basics/files/GUID-45D2EAD4-5BCF-42DA-A1AB-EC6EE09FE705-htm.html
3. https://knowledge.autodesk.com/support/3ds-max/learn-explore/caas/CloudHelp/cloudhelp/2021/ENU/3DSMax-Modeling/files/GUID-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

Practical tip: how complex shapes are made using base shapes

- TODO

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

Tip: more on 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
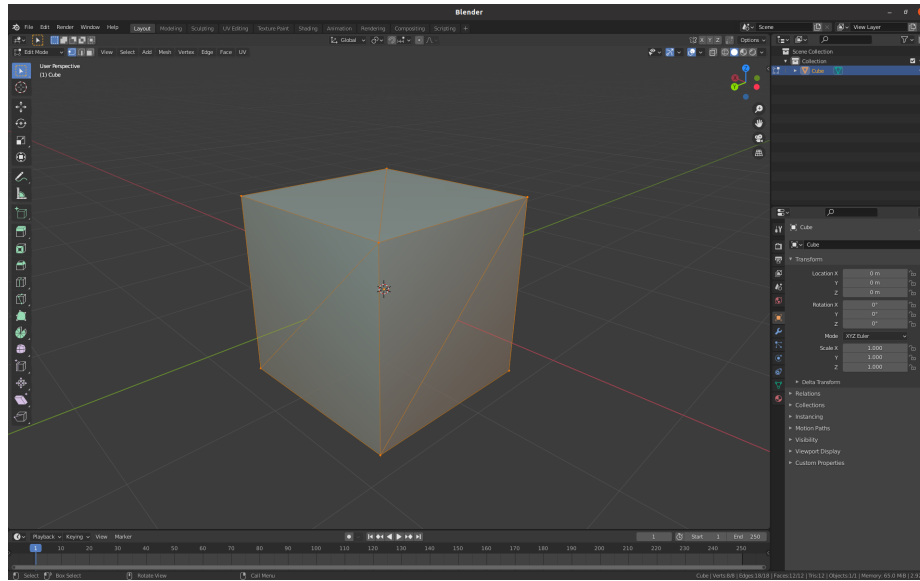

```
1   # Blender v2.92.0 OBJ File: ''
2   # www.blender.org
3   o Cube_Cube.002
4   v -1.000000 -1.000000 1.000000
5   v -1.000000 1.000000 1.000000
6   v -1.000000 -1.000000 -1.000000
7   v -1.000000 1.000000 -1.000000
8   v 1.000000 -1.000000 1.000000
9   v 1.000000 1.000000 1.000000
10  v 1.000000 -1.000000 -1.000000
11  v 1.000000 1.000000 -1.000000
12  vt 0.625000 0.000000
13  vt 0.375000 0.250000
14  vt 0.375000 0.000000
15  vt 0.625000 0.250000
16  vt 0.375000 0.500000
17  vt 0.625000 0.500000
18  vt 0.375000 0.750000
19  vt 0.625000 0.750000
20  vt 0.375000 1.000000
21  vt 0.125000 0.750000
22  vt 0.125000 0.500000
23  vt 0.875000 0.500000
24  vt 0.625000 1.000000
25  vt 0.875000 0.750000
26  vn -1.0000 0.0000 0.0000
27  vn 0.0000 0.0000 -1.0000
28  vn 1.0000 0.0000 0.0000
29  vn 0.0000 0.0000 1.0000
30  vn 0.0000 -1.0000 0.0000
31  vn 0.0000 1.0000 0.0000
32  s off
33  f 2/1/1 3/2/1 1/3/1
34  f 4/4/2 7/5/2 3/2/2
35  f 8/6/3 5/7/3 7/5/3
36  f 6/8/4 1/9/4 5/7/4
37  f 7/5/5 1/10/5 3/11/5
38  f 4/12/6 6/8/6 8/6/6
39  f 2/1/1 4/4/1 3/2/1
40  f 4/4/2 8/6/2 7/5/2
41  f 8/6/3 6/8/3 5/7/3
42  f 6/8/4 2/13/4 1/9/4
43  f 7/5/5 5/7/5 1/10/5
44  f 4/12/6 2/14/6 6/8/6
```

Additional mesh information: Normals

- We mentioned that both shape and material are used for calculating appearance (shading step in rendering).

- **Orientation of the 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

Additional mesh information: texture coordinates

- Later, when we will discuss material part of a 3D object, we will see that, e.g., images or procedural patterns are often used to add more details on object.

- Simplest analogy is hanging posters on the wall.

- The problem is that 3D object are often not simple as flat surface.
  - We need a way for mapping a 2D image/pattern to 3D shape.

- A solution to this problem are texture coordinates.

- TODO: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-polygon-mesh

Storing and transferring mesh objects

- Mesh data-structure 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
- Interesting: https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-polygon-mesh/polygon-mesh-file-formats

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

<IMAGES: DEPICT IMPORTANT PROPERTIES!>

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.
  - Optionally, In ray-tracing-based rendering plane-ray intersection may be defined and then triangle representation is not needed*.

* As we will see, there is always a trade-off between which representation is good for modeling and which representation is good for rendering. Ray-tracing-based rendering can get very flexible with rendering wide representations of shapes but then there is a question if this is feasible to implement and maintain. Often, mapping between different shape representation is researched and used so that on higher level users are provided with intuitive authoring tools and on low level, rendering engine is given efficient representation for rendering process.

Mesh representation: verdict

- Pros:
  - Simple for representing and intuitive for modeling
  - A lot of effort has been made to represent various shapes with meshes
  - Lot of research has been done to convert other shape representations to mesh representation
  - Graphics hardware is adapted and optimized to work with (triangle) 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

Curves, curved surfaces and subdivision surfaces

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

Tessellation of curved surfaces

- For rendering purposes (both rasterization- and raytracing-based) it is beneficial to transform curved surface into triangulated mesh

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.

Notes on material

- In graphics material is used to describe light-matter interaction. It is a model of a real-world. Note that "material" in graphics encapsulates bit more then just chemical property (which might be the common conception we have about term material) – in graphics it also defines small scale geometrical information that is simply too expensive to represent using actual geometry and shape.

- Based on previous discussion we can see that material can be decomposed and describes both:
  - Scale of geometrical information that is too small to represent explicitly by geometry/shape. Note that most important geometrical information in graphics is normal. For smaller scales, we will describe smaller surfaces thus smaller normals.
  - Substance information encapsulating information on absorption that is color and intensity which will be reflected

<IMAGE: SUBSTANCE OF GLASS AND SMALL SCALE GEOMETRICAL INFORMATION OF GLASS>

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>

Note: two faces of BRDF

- BRDF has the name "bidirectional". This means that given given incoming and outgoing direction, we can compute **amount of reflected light** in outgoing direction.

- Incoming and outgoing directions can be viewed in two ways:
  - If light is falling on the surface, what is the distribution of **reflected light directions** (lobe of directions) – this information is particularly useful for ray-tracing (light transport element) as we will see later.
  - If we are looking at the surface from particular direction, how much light will be reflected in this direciton.

<IMAGES: 2 WAYS HOW BRDF CAN BE VIEWED>

<FIND THE BOOK DISCUSSING THIS!>

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

- 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.

Basis and scattering function

- On the different scattering types we have seen that they always require normal around which scattering is computed

- Normal defines so called **basis**

- Note that perturbing the bases can be constructed using geometrical (object shape) normal. So what would happen if we would somehow vary this normal across smooth shape which has constant geometrical normal? More on this will come when we introduce textures!

Empirical models

- Discuss the way of modeling: by observation, phenomenological approach

- Discuss importance of normal, light position and viewing position.

- Mirror: TODO

- Lambertian: TODO

- Phong and Blinn-Phong: TODO

- Discuss parameters

Data-based models

- Discuss measurements need for those models

Physically-based models

- Highlight importance of normal on different scales

- Reflectance model is half substance information half meso and micro geometry information!

- Microfacets*

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

Scattering model parameters

- Discussed models are parameterized

- Parameters can be mainly decomposed in geometrical and substance

- Geometrical: roughness, basis normal

- Substance: color

<IMAGE: variation of scattering parameters>

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>

Values varied by texture

- Texture varies parameters of scattering models

- It is also important to note that texture is used to vary meso-scale geometrical information which is used for evaluating scattering models.

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

Literature

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