Computer image generation overview

Goal of Computer graphics

- Any interactive virtual environment, animation, application, development and research in computer graphics can be boiled down to: **synthesizing images**
- <examples: different applications, actually image is goal>

Example of computer generated image

<IMAGE: synthetic image which motivates this lecture. We will break image into parts which make pillars of computer image generation>

Image synthesis: intuition

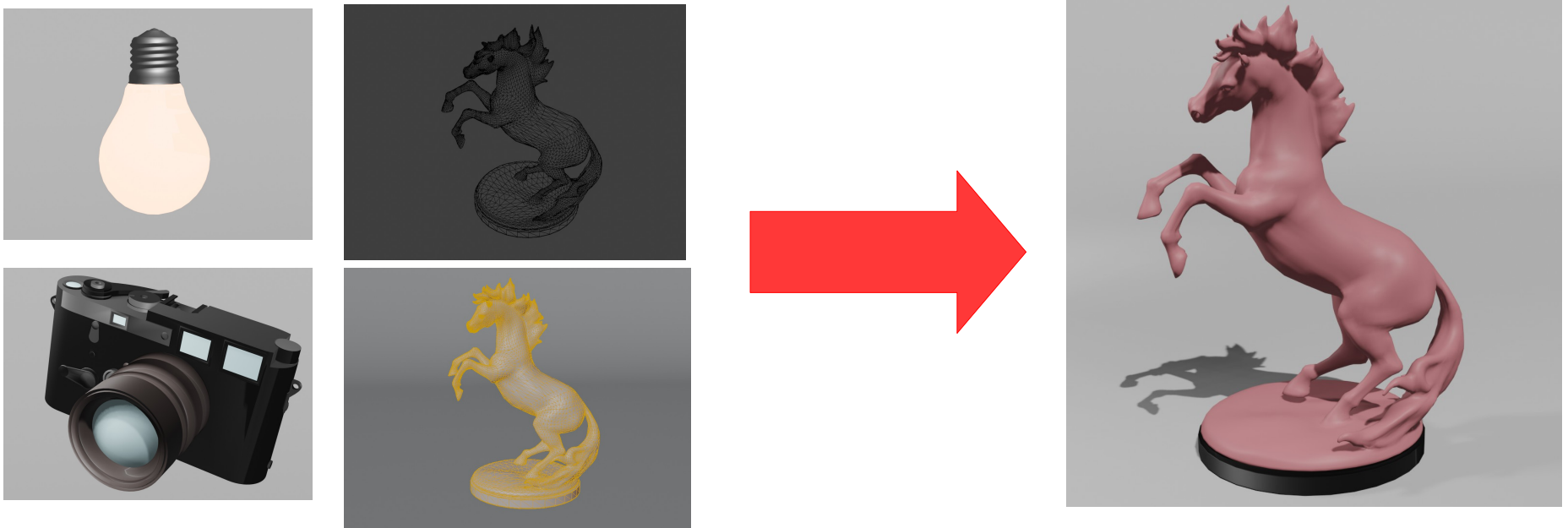
<todo: light rays>

- Real world observation/photography:
  - **Source of light** illuminates **objects** which have different **shapes** and **material**.
  - **Light travels through the space, interacts with objects** and some of it reflects into our eyes or **camera** giving us image of the world around us.

https://polyhaven.com/models

Fundamental image synthesis questions

- Computer graphics gives answer to:
  - How to represent real world objects, lights and cameras in a computer?
  - How to generate image based on computer representation of those elements?



Enter computer graphics

- Computer graphics is solving a problem of:
  - representing real-world in a computer using models
  - generating (synthesizing) images based on those models

Note: computer graphics

- Note that **art of computer graphics is to decompose real world objects and phenomena into separate models and processes** which can be modeled **and then again combined into virtual world.**
  - Understanding this gives a way to understand that all of those parts are deeply intertwined and that one doesn't makes sense without another.
  - Example is light and objects we see. Without light we can't see objects and without objects we can't see light. Nevertheless, we must decompose this phenomena into two models and study them separately
- Therefore, glimpse into bigger picture is important to understand how is everything connected and then studying it separately makes sense.

Simplification of a real world

- As in computer science, computer graphics problems are solved using **simulation and approximation.**
  - The complexity of real world objects and phenomena is simulated and approximated to achieve desired outcomes for specific application.
  - Computer graphics simulations and its results (generated image) are **discrete***.

* For example Computer generated image and display device are 2D array of pixels – raster graphics and display (grid of x and y coordinates on a display device).

Pillars of computer graphics

- To synthesize an image:
  - We need a description of a **3D scene** which contains objects and phenomena we want to visualize
  - A way to transform a 3D scene into 2D array of pixels – **rendering**
  - A way of storing 2D array of pixels for raster display – **image**

Modeling and rendering

- Computer graphics R&D is concerned with*:
  - Techniques and methods for modeling of 3D scene
    - Interaction with 3D scene elements is researched in human-computer interaction (HCI) field, closely related to computer graphics
  - Development of rendering algorithms for creating image out of a 3D scene
  - Storing and displaying image.

* Computer graphics is closely related to other fields such as computer vision and image processing.

- Image generation software can be thus split into:
  - **Modeling** – process in which 3D scene is defined: what is rendered
  - **Rendering** – process in which image of 3D scene is generated

Modeling and rendering are intertwined

- Rendering process must “understand” how 3D scene is represented
- Modeling of 3D scene relies on representations which are understood by rendering algorithm

3D Scene Modeling

A start of image generation journey: answering what we render*

* As discussed, but to highlight again, 3D scene modeling and rendering in order to obtain an image is highly intertwined process. For learning purposes in production and as well as in research in development theses processes are separated. Depending on application and teaching the description of whole process can start from rendering as well.

3D scene modeling

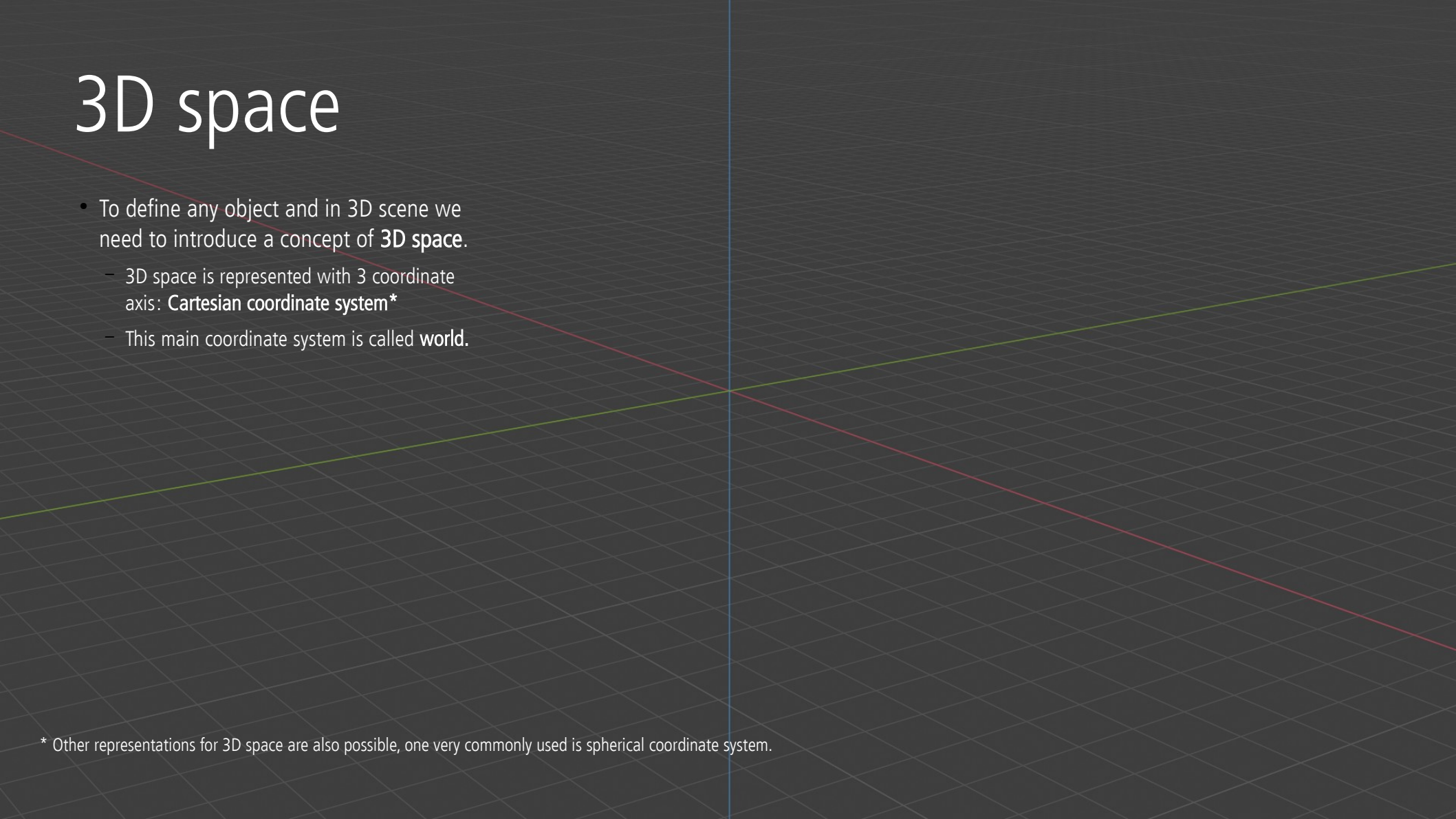
- In order to generate an image (to render it) we need something to render → a 3D scene.
- The real world is made of wide range of phenomena and objects. For modeling of a 3D scene we simplify those. Therefore we describe **objects** by **shape and appearance**.
- The reason why we see objects around us is due to light reflecting in our eyes. Therefore, for purposes of simulation, we can separate objects in those which generate light - **light sources** and those which only reflect light.
- 3D scene is simulated 3D world from which images can be taken - similarly as photographing a real world. Thus, we also need to simulate a device which is needed for taking images – a **camera** – which also defines from where image is taken.

Pillars of 3D scene

- 3D objects
- Lights
- Cameras

3D space

- To define any object and in 3D scene we need to introduce a concept of **3D space**.
  - 3D space is represented with 3 coordinate axis: **Cartesian coordinate system***
  - This main coordinate system is called **world.**



* Other representations for 3D space are also possible, one very commonly used is spherical coordinate system.

World coordinate system

- All objects have unique **transform** in this world coordinate system:
  - Position
  - Rotation
  - Scale

Objects in 3D scene

- Real world objects exhibits wide range of **shapes** and **appearances**
- <IMAGES>

* Representing objects is not only important for visualizing purposes, it is also important for modeling them. Therefore, even some objects for which we only care about the surface appearance (e.g., car rim) we would like to represent them as solids in certain modeling tools which would enable simulation of cutting and modling the surface). For example: https://www.autodesk.com/products/fusion-360/features#3d-modeling or https://pixologic.com/.

Object shapes

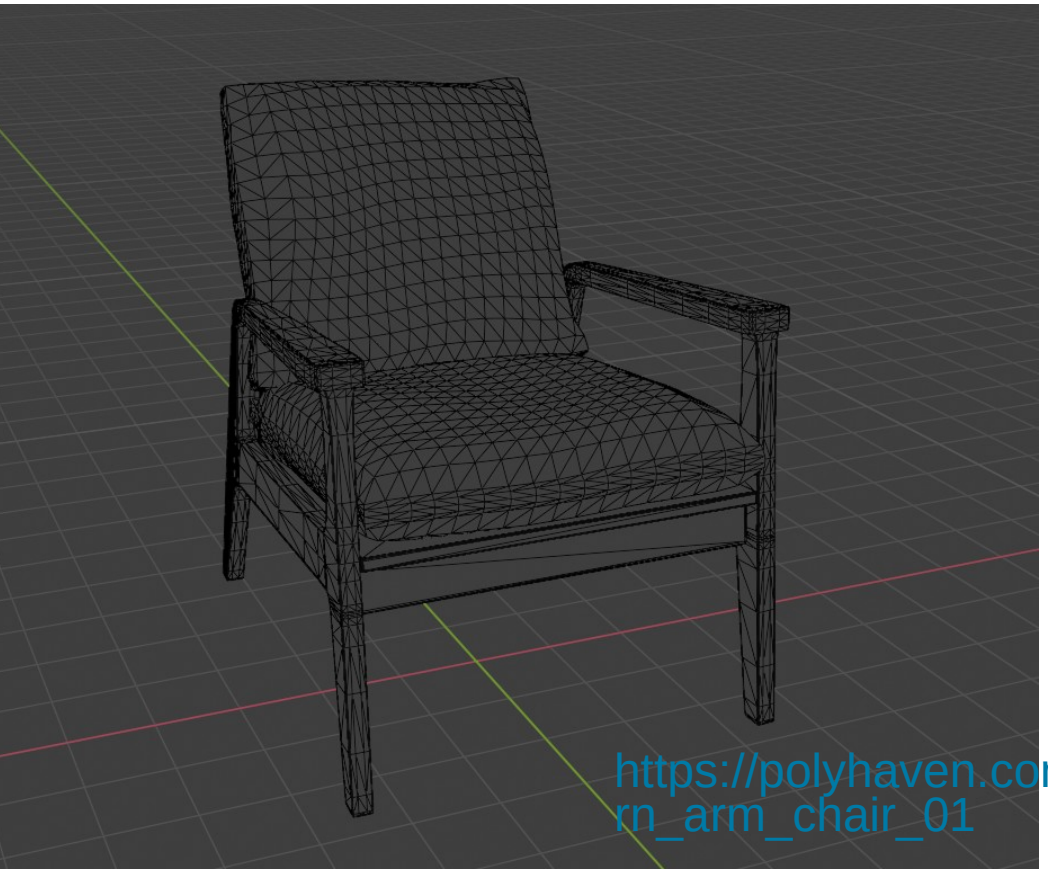
- In graphics, we simplify and categorize real-world objects
- For certain objects, only **surface shape** description is enough
  - Opaque or transparent objects example
- In some cases, **volume interior description** is needed since we can see in or through them (e.g., water).
  - Volume objects examples

Example of Shape representation

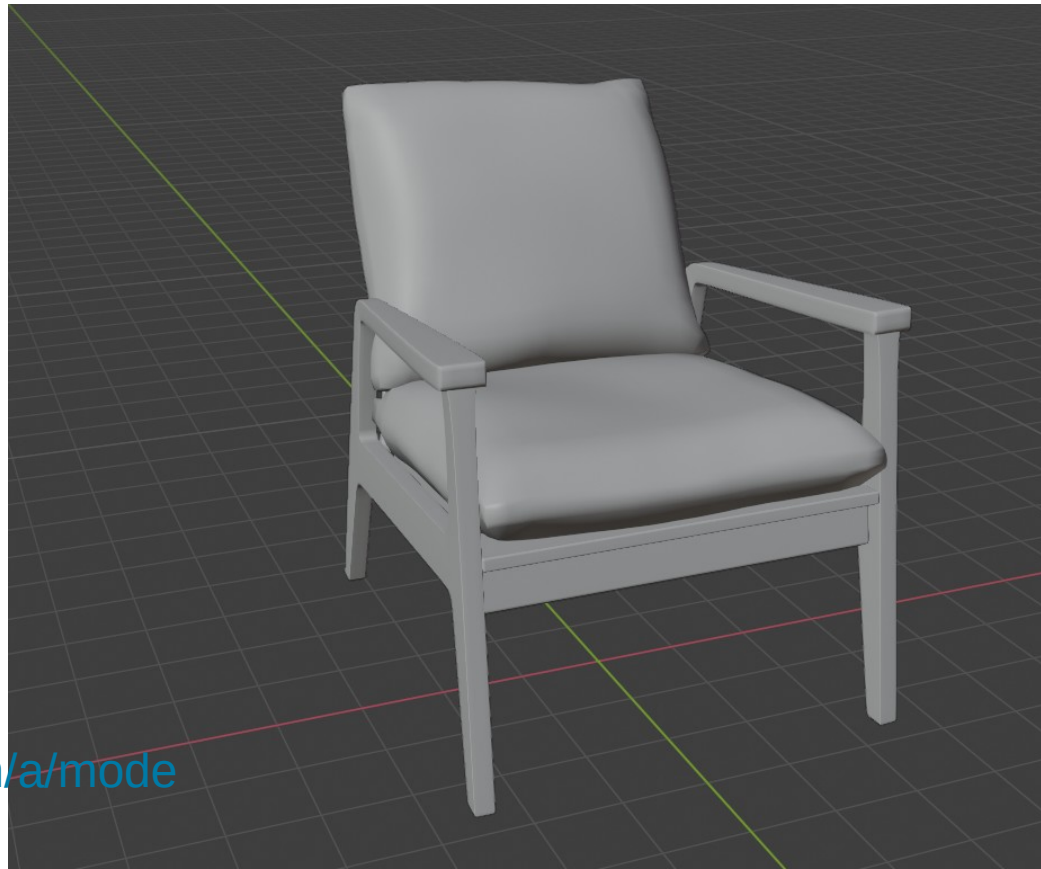
- With the concept of 3D space, we can define object's shape in a 3D scene.
- We start by defining **points** (x,y,z) in 3D space.
- To define a **surface**, simplest way is to connect points to form a **polygon**.
  - In computer graphics, for computation tractability, we often use co-planar polygons - where all points are lying on the same plane
  - Most commonly used polygons are **triangles.**
- Triangle is basic building block for creating more complex shapes. And modeling is all about creating complex shapes using basic building blocks.

<IMAGE: POINTS, TRIANGLES, COMPLEX SHAPES>

Triangle shape representation



https://polyhaven.com/a/modern_arm_chair_01



Reading: about triangles

- Is object that we have seen built using triangles?
  - NO!
- Triangle is most widely used **primitive** for surface shape representation**.** This is because it is very simple and holds great properties for easy calculation. Thus it is highly researched and used for efficient rendering purposes.
- But for easier modeling different shape representations exists.
- It is very often that all representations are turned to triangles before/during rendering stage - using very elaborated method called **triangulation**.

Other shape representations

- Various representations for 3D models exists.
  - Other are better for modeling (e.g., implicit surfaces).
  - Some are are better for rendering (triangulated mesh).

Other shape representations: modeling

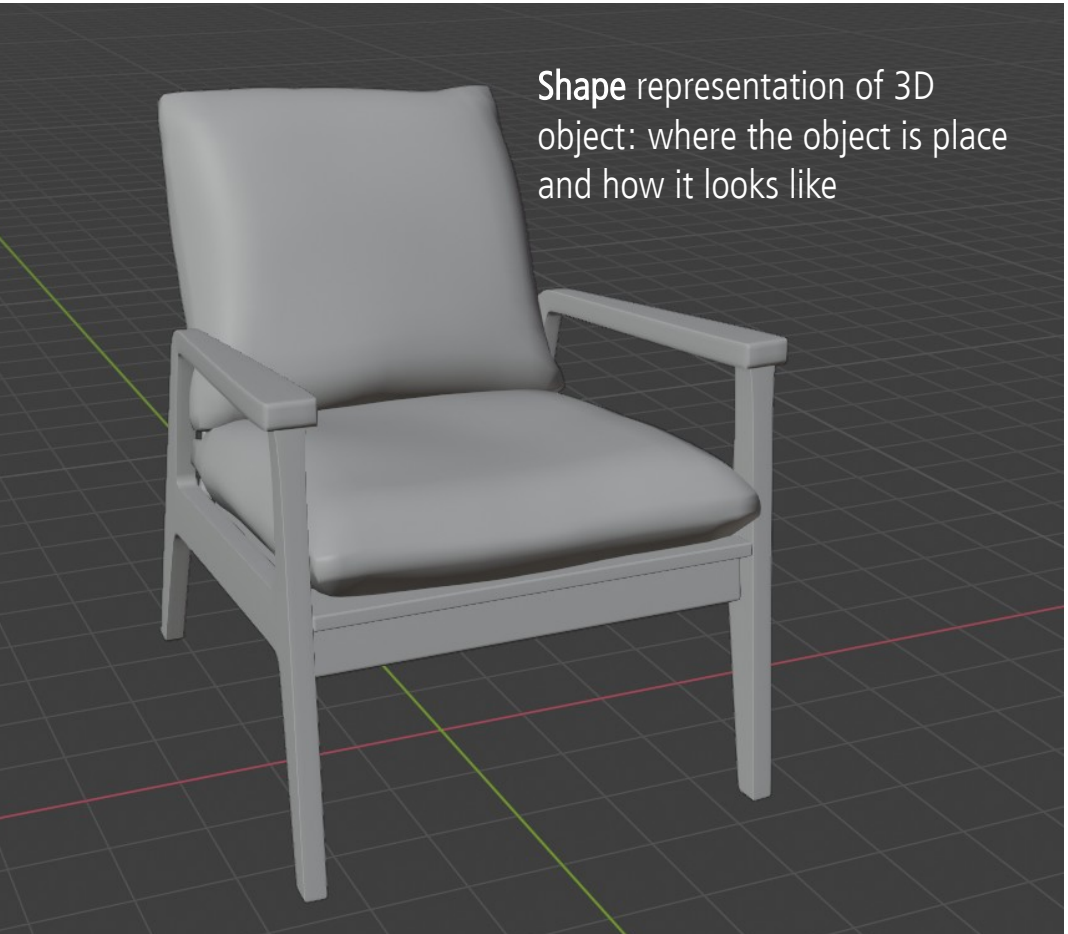
- For easier representation and modeling of different phenomena, various shape representations exist and will be introduced.
- TODO: images: short glimpse into different representation for phenomena

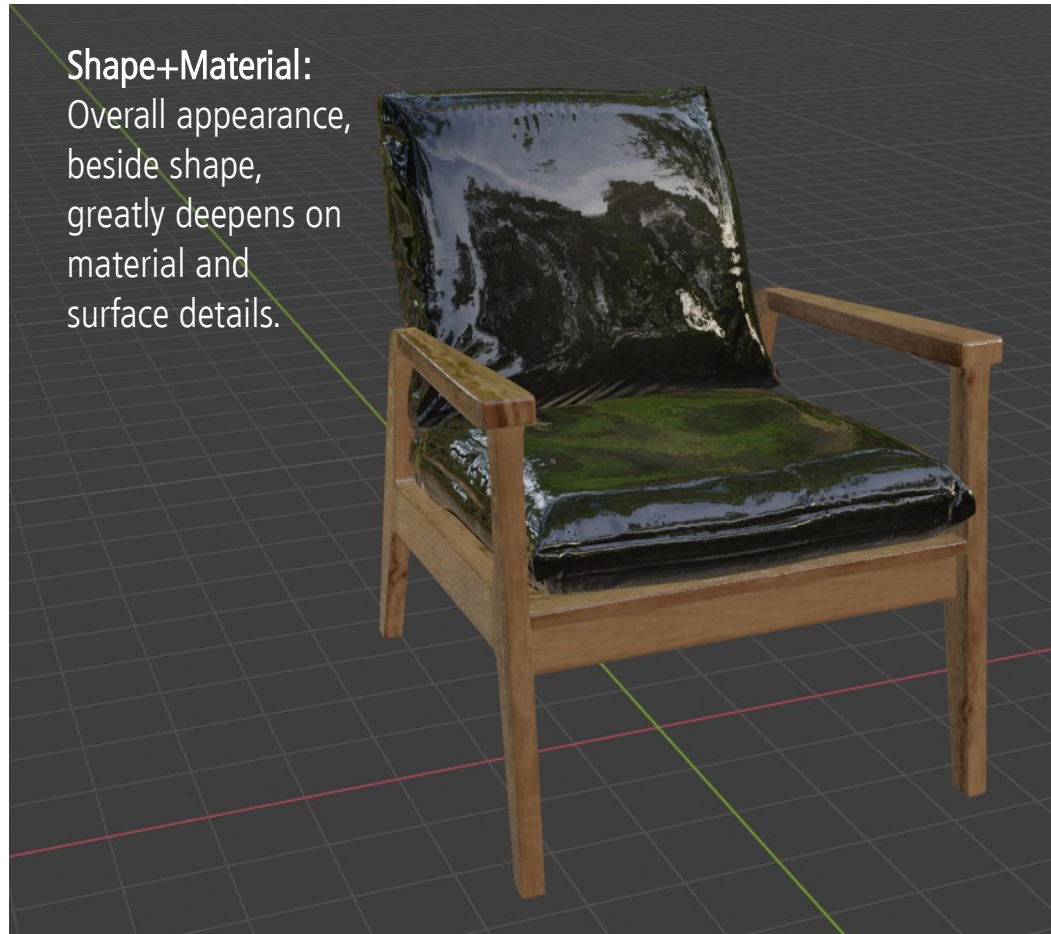
Other shape representations: rendering

- In professional software, various shape representations for efficient modeling exist.
- Prior rendering they are converted to triangulated mesh for efficient rendering.
  - Having one representation for rendering makes rendering process highly efficient since all effort is put into efficient methods for rendering one representation.
  - Converting all objects to same representation (triangulated mesh) is also convenient for applying various rendering effects. For example, techniques for adding more details during rendering time can be only developed to work with triangulated mesh.
  - Lot of research and hardware development was focused onto efficient rendering of triangles as we will see*.

* It is easy to imagine that different rendering primitive is used. But due to historical events this turned out to be a triangle. What is important to note is that mapping of various representations to triangle mesh is possible and thus it is not important which is the rendering primitive as long as it supports various representations.

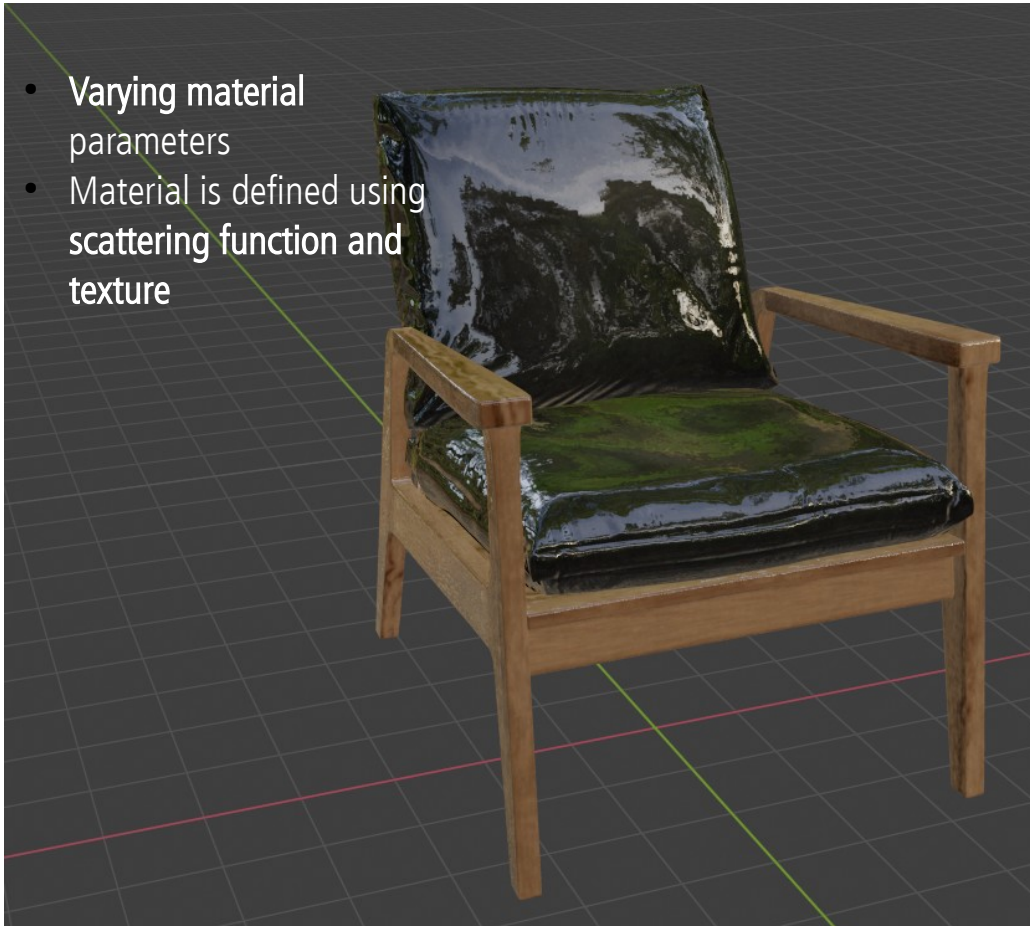
3D objects: appearance





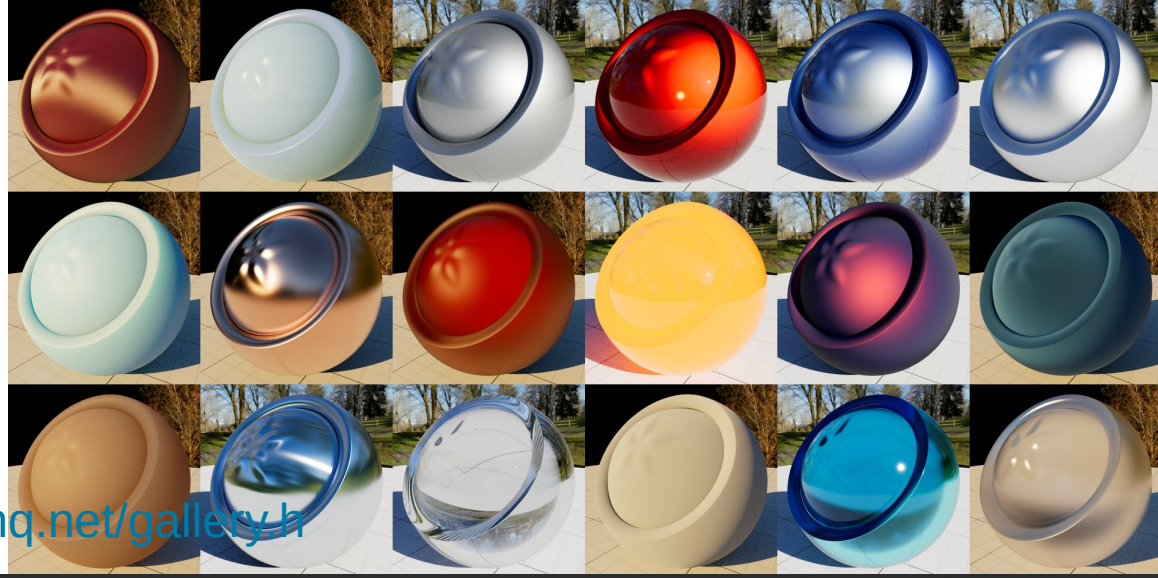
Material


• **Uniform material** parameters
• Material is defined using only **scattering model***

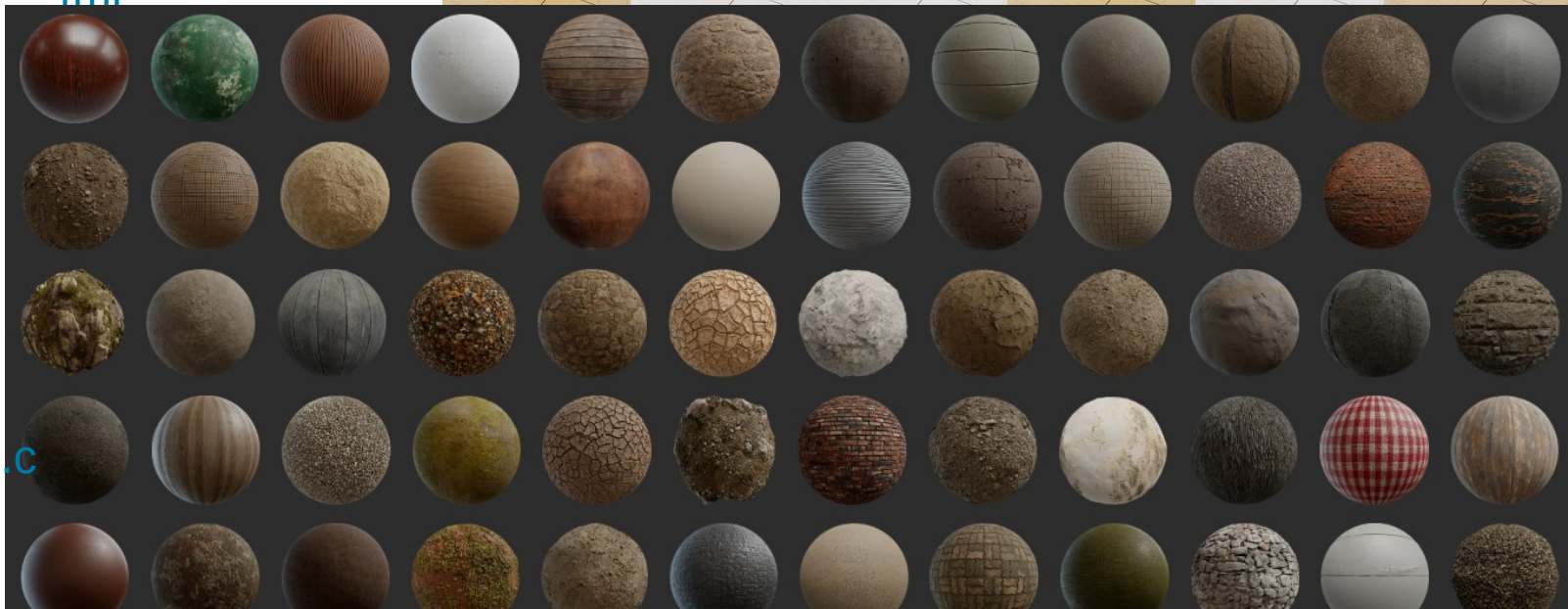

• **Varying material** parameters
• Material is defined using **scattering function and texture**

Material

- In computer graphics, material modeling is combination of:
  - Scattering model
  - textures

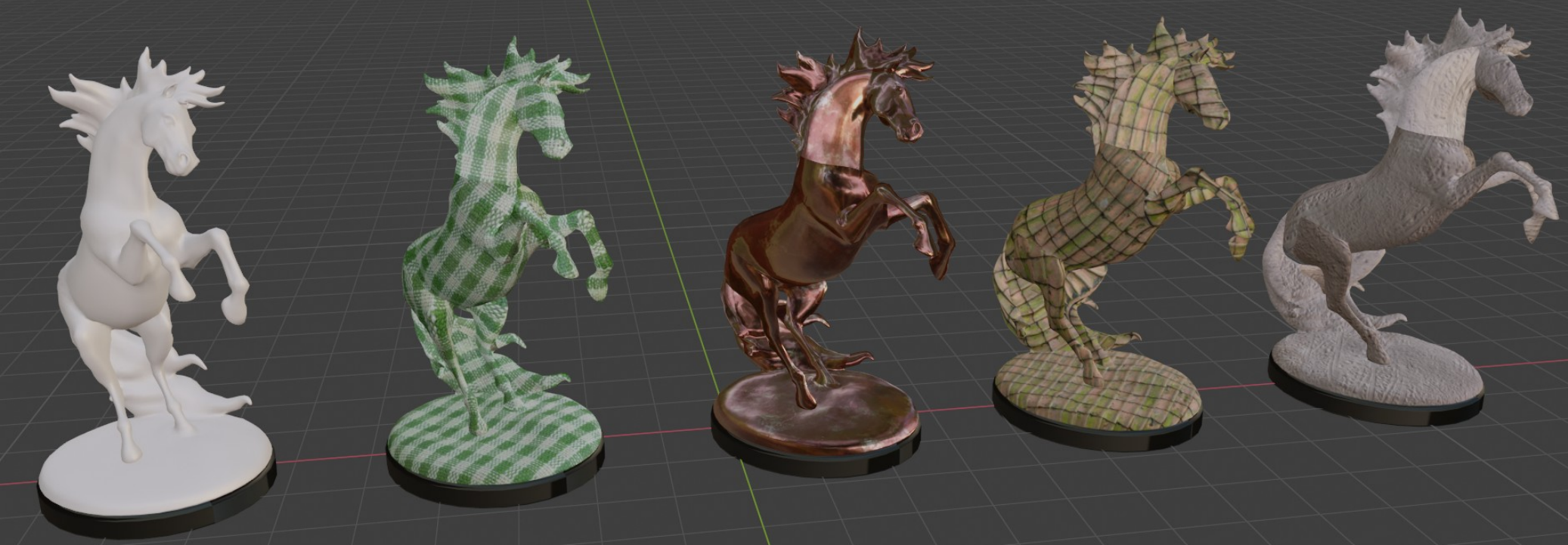


https://appleseedhq.net/gallery.html



https://polyhaven.com/textures

Practical tip: material and shape



Material and shape can be modeled separately.

Creating 3D scene objects

- Artist using DCC tools
  - Shape modeling is done with various representations, methods and techniques
  - Material modeling is done by combining:
    - Scattering functions
    - textures
  - TODO: example

Creating 3D scene objects

- From real world
  - 3D scanning can be used to “import” real object into digital world.
  - CT TODO
- Simulated
  - Some objects are very hard to model by hand and capture from a real world. For example water stream. For these purposes physical simulations are also employed to generate shape. TODO

Lights

- Light enables us to see objects:
  - Some objects emit light – light sources.
  - Some objects reflect/refract light.
- If light is reflect in eyes/camera, we can see that object
- Light significantly determines visibility and appearance.
- TODO

Light in 3D scene

- Example in DCC

Camera

- Camera defines point of view and it is used to simulate effects of real-world cameras.
- Camera is a window to a 3D scene.

Animated objects

- Introducing a time component into 3D scene and moving/rotating objects and their parts as time goes gives foundations for animation.
- Animation can be predefined, generated at render-time or influenced by interaction
- TODO

Animated objects

- Blender
- Maya
- Houdini

Interaction with 3D scene

- Godot, Unity, Unreal

3D scene: recap

- Once **3D scene**, is defined:
  - 3D objects
  - Lights
  - Cameras
- Rendering can take place to generated image.
  - Note again that 3D scene definition must be done with elements understandable to renderer: shapes, material, lights, cameras, etc.

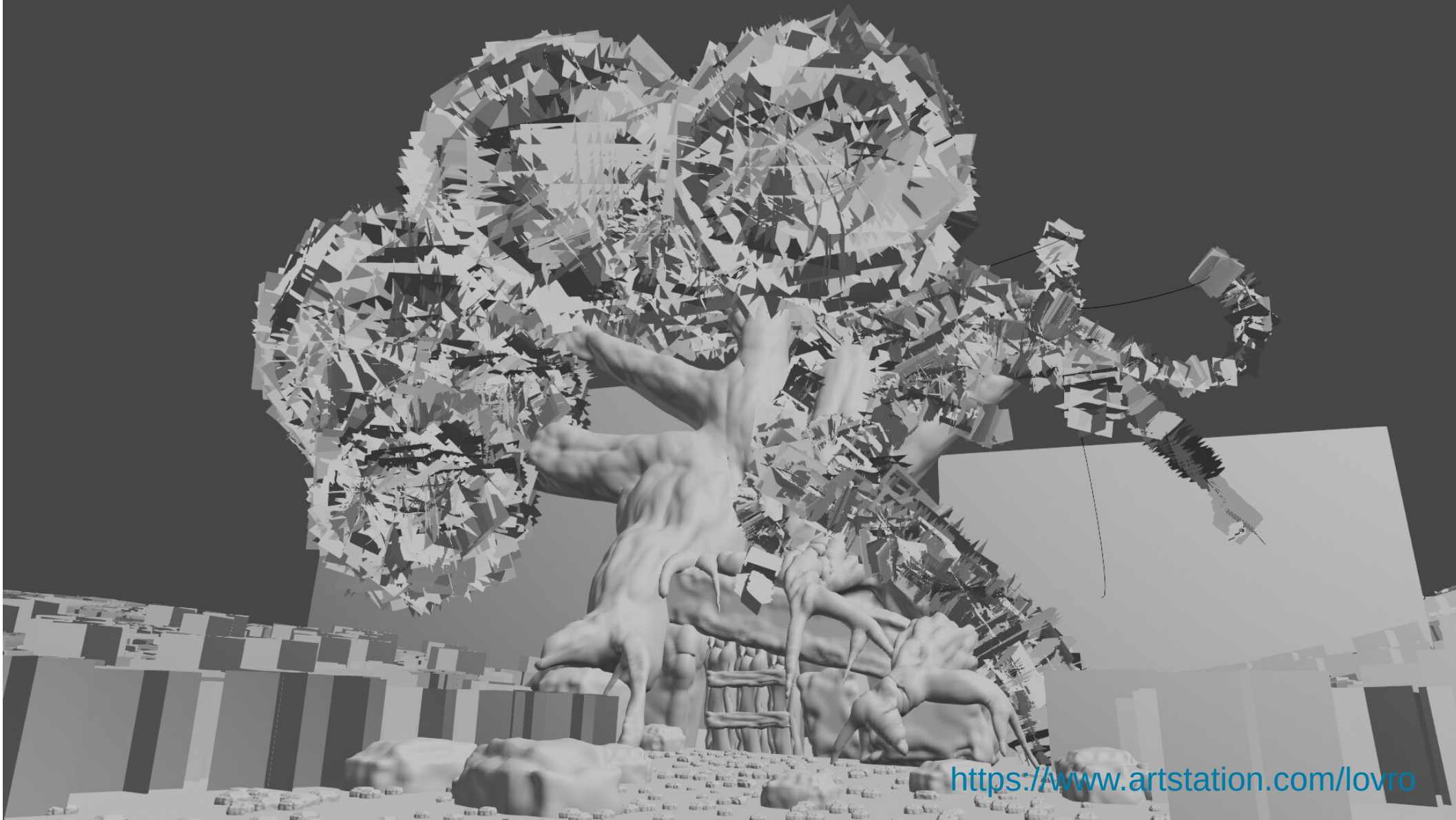
Rendering

A process of generating image from a 3D scene

Rendering

- Rendering is general term for generating images from a 3D scene.

Scene from: https://casual-effects.com/data/index.html (Sponza Palace in Dubrovnik). Rendered in Blender, EEVEE.


https://www.artstation.com/lovro


https://www.artstation.com/lovro


https://www.artstation.com/lovro


https://www.artstation.com/lovro

Rendering: photo-realistic images

- Often, goal is to create **photo-realistic** images – images that look like a photograph* of a real world. This is one spectrum

* Note that there is also distinction between images that look like a real photo and that are "correct" like a real photo. In graphics, almost always it enough to create images that look like a real photo but the way they are produced doesn't necessary be correct in terms of how real world works. Again, this is trade-off between quality and speed and decision depends on application.

Rendering: non-photorealistic images

- On the other hand, creating **non-photo-realistic** (NPR) images* – images that contain wide range of expressive styles, is also often desired.

Enthymeme: https://appleseedhq.net/gallery.html

https://appleseedhq.net/gallery.html

* Specific look of NPR images often comes from exaggerating some characteristics of 3D scene or rendering algorithm. Therefore, it is better to start with photo-realistic rendering and then altering this method to obtain required style in NPR.

- In both cases, image is generated using certain **rendering** method - a simulation of appearance of objects*.
- For photo-realistic images**, to make objects appear as they do in real world, we need to simulate laws of physics that are related to appearance – optics, that is, simulating light transport and interaction with objects in 3D scene and camera.

* Note that the realism/style of image, next to rendering, also depends on the way how 3D scene is simulated as well as its complexity. This is another example showing how rendering and modeling of 3D scene are intertwined processes.

** We will mostly focus on methods needed for photo-realistic image synthesis since different phenomena are modeled for these techniques. Also, understanding photo-realistic rendering makes non-photo realistic rendering easier to grasp.

Rendering: simulating visual system

- As we are simulating images that resemble ones that we take with a real camera or see with our eyes – we need to take in account the **foreshortening effect**.
- This effect is important for photo-realistic rendering since it simulates how our visual system works – it defines shape and size with respect to the distance to the eye.
  - Objects that are further appear smaller than those which are closer
- Method for achieving foreshortening effect is to trace rays from corners of objects to eye and intersecting them with imaginary canvas that lies in between <IMAGE>. https://en.wikipedia.org/wiki/Perspective_machine
  - Direct result of such method is used to simplify the 3D scene while modeling and it is called **wireframe** rendering.
- This example brings the key idea used in every rendering algorithm – **perspective projection**.

Rendering: a visibility problem

- Looking with a camera from particular point of view, we can only see portion of the scene and only some objects.
- Determining which objects and which of their parts are visible is called **visibility problem***
- Visibility problem is used for both determining what is visible from camera and which surfaces are visible to each other in 3D scene (this will be clear later).
- Solution to visibility problem in computer graphics can be solved using two main methods:
  - **Rasterization***
  - **Ray-tracing**

* Also known as: hidden surface elimination, hidden surface determination, hidden surface removal, occlusion culling and visible surface determination
* This is umbrella term and some popular methods are painter's algorithm and z-buffer. Almost all GPUs use an algorithm from this category.

Rendering: appearance calculation

- Once visibility problem for camera view is solved (which inherently takes in account perspective projection giving us information of size of the objects) – once we determine which shapes we see and how large, we would like to calculate their appearance.
  - Note that for solving the visibility we need shape information of the objects.
- Appearance determines the look of objects in terms of color, texture and brightness.

  <EXAMPLE: light traveling from light source, falling on surface, interacting with material, reflecting into eye>
- Appearance is determined with light falling on the object surface and interacting with surface material. What we see is the result of this interaction. Therefore, we can divide appearance computation in:
  - Light interaction with surface - **shading**
  - Gathering light onto surface – **light transport**

Rendering: shading

- When light comes in contact with an object, simply speaking, two things can happen:
  - Light is absorbed
  - Light is reflected into scene
- Absorption gives objects their unique color. If white light (containing all colors) falls onto object which absorbs all colors except red – which is reflected, we perceive this object as red.
  - Color of object is one foundational material information of object as we will see later. It can be defined directly or indirectly using physically-based models which calculate color.
- Reflection. Light which is not absorbed is reflected. We know amount and color of reflection using information on absorption. Direction of reflection depends on:
  - Surface orientation on which light falls – this is described with geometrical property known as **normal** vector.
  - **Scattering** – a model which describes what happens with light as small scale. This model can have uniform parameters or varying parameters – giving textured look.
- Implementation of mathematical model which simulates light interaction with surface is called a **shader**.
- NOTE: Real-world light-matter interaction is complex topic. It has been researched on different scales. We mostly work with geometrical optics and further simplify those methods to generated desired tradeoff between quality of image and rendering speed.

Rendering: light transport

- Crucial information for surface appearance calculation (shading) is information about light falling on it.
- Light is emitted from light sources, reflects from objects and eventually might fall into camera.
  - In rendering, it is extremely expensive to calculate complete light behavior in whole scene. Thus, we are only interested in light which is falling on objects visible from current viewpoint.
- Object surface which is visible from camera might receive light from any light source or surface which is facing.
  - Calculating this amount of light is called **light transport**.
- Note that amount of light which actually falls into camera is extremely small. Thus rendering employs light transport only for visible surfaces starting from the camera and only for relevant light paths.

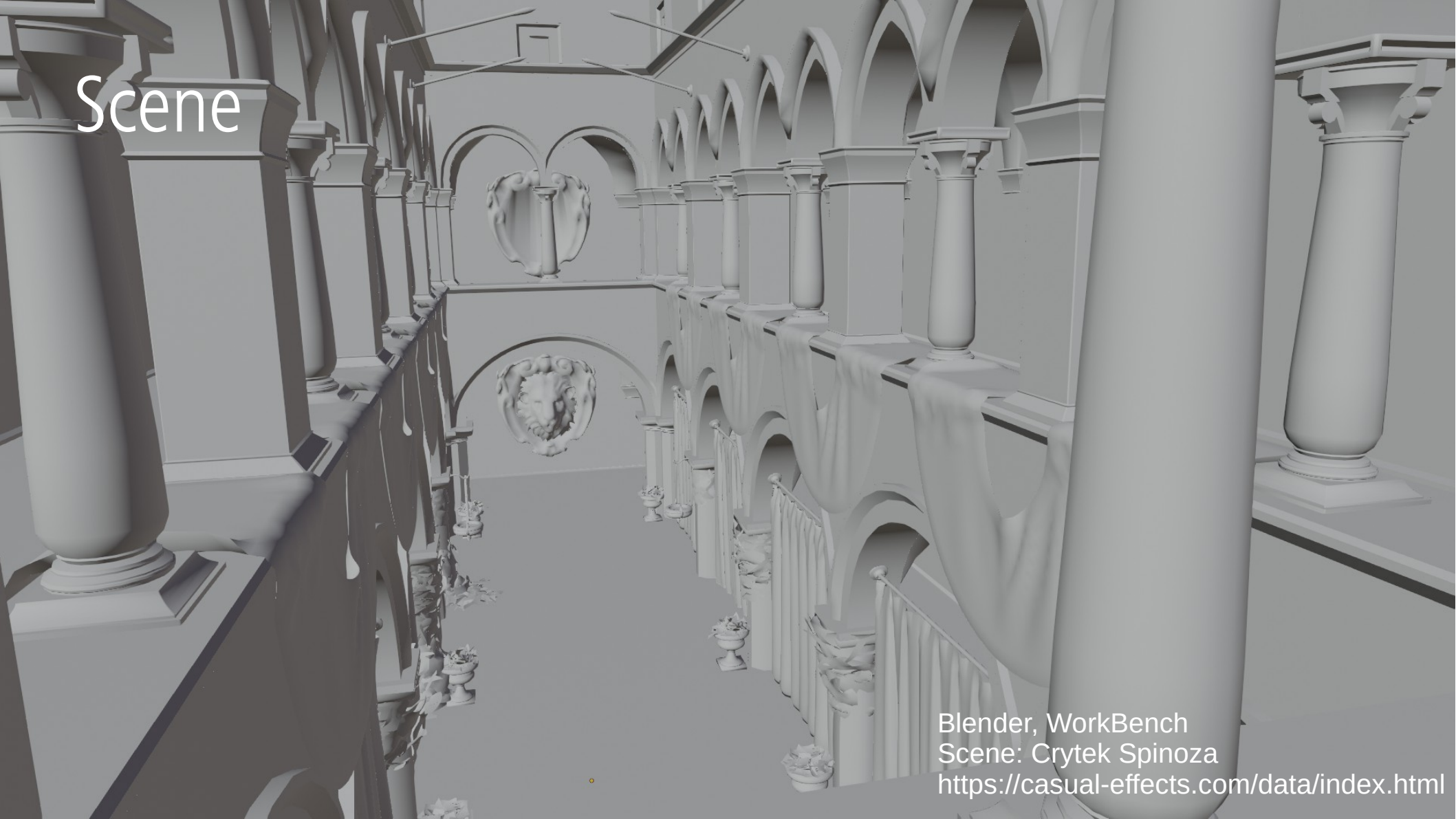
Rendering: recap

- Rendering can be separated in two main steps:
  - Solving **visibility problem**: what is visible from camera point of view with inherent **perspective projection**
  - **Shading**: calculating appearance of visible objects using light-matter interaction information and light falling on the object calculated using **light-transport**.
    - Note again that visibility is general term and it is also used during light transport computation.

Practical rendering

- Rasterization-based rendering
- Ray-tracing based rendering

Scene



Blender, WorkBench
Scene: Crytek Spinoza
https://casual-effects.com/data/index.html

Rasterizer-based render

Ray-tracing-based render

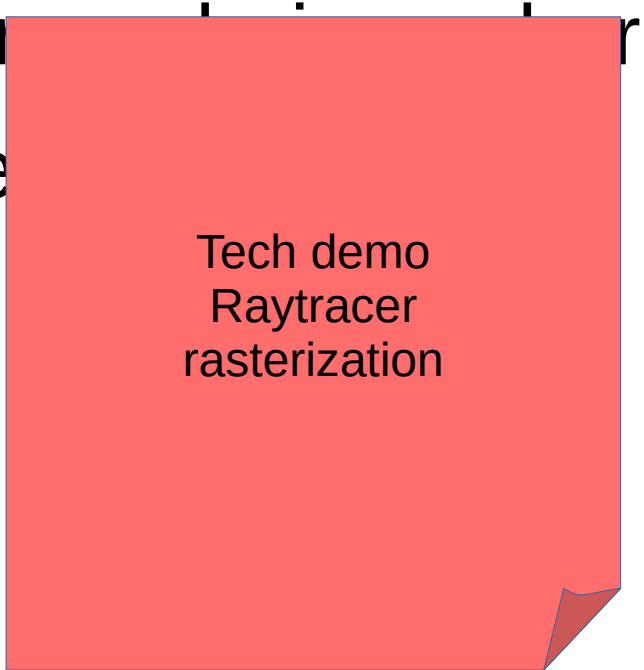
Blender, Cycles
Scene: Crytek Spinoza
https://casual-effects.com/data/index.html

Rasterizator vs raytracer

- Rasterizator can quickly determine visibility but has limited shading capabilities when it comes to light transport
- Raytracer has advanced shading possibilities when it comes to light-transport but it is slow
  - Shadows and soft shadows
  - Reflections

- Give short practical intro in rasterization and raytracing.
- Raytracing: simple renderer
- Rasterization: opengl rende

res



Image

A result of 3D scene rendering

Image

- Array of pixels
- Display device

Merging all together

3D scene, rendering and Image

- 3D scene, rendering and resulting image are highly intertwined elements of image synthesis
- Now that big picture has been discussed, following lectures will dive deeper into each of these topics.

What have we learned?

- Map of computer graphics:
  - 3D scene
  - Rendering
  - Image