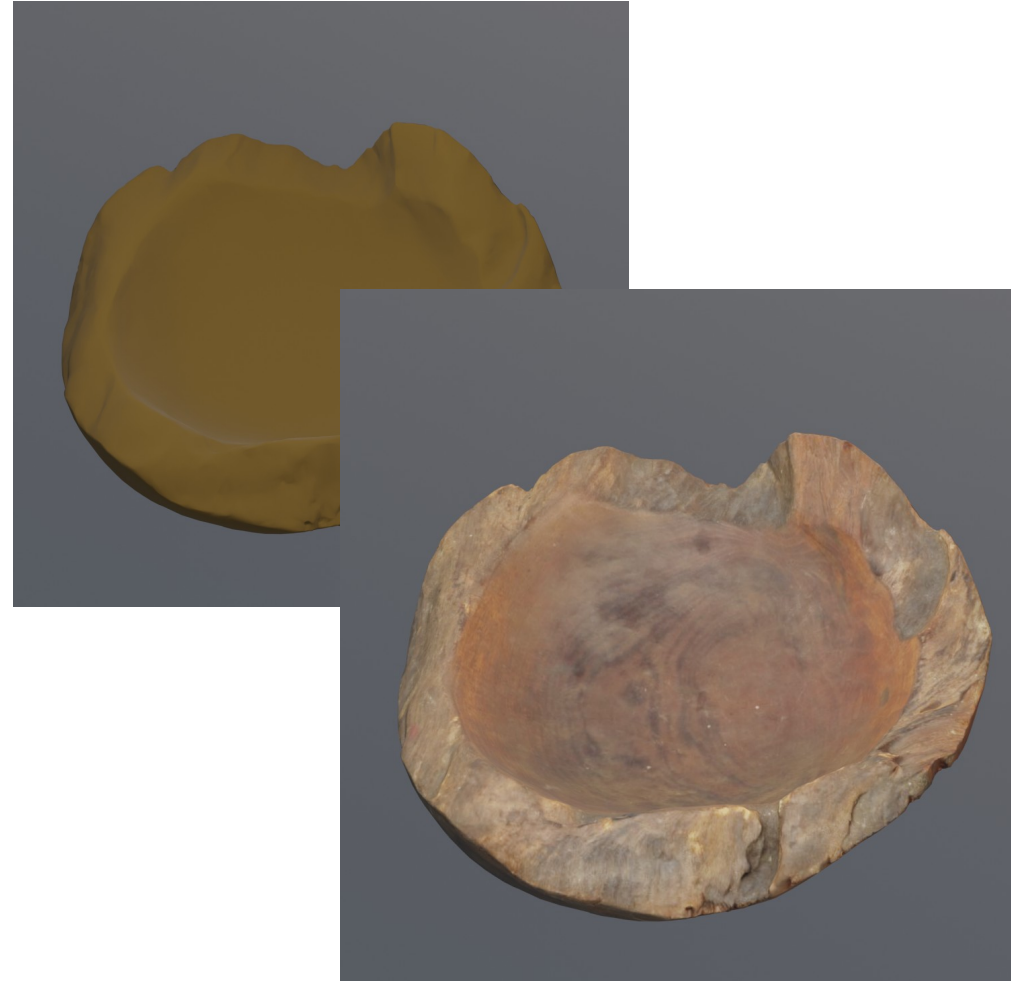


Texture

Variation of material over surface

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



Texture and texture function

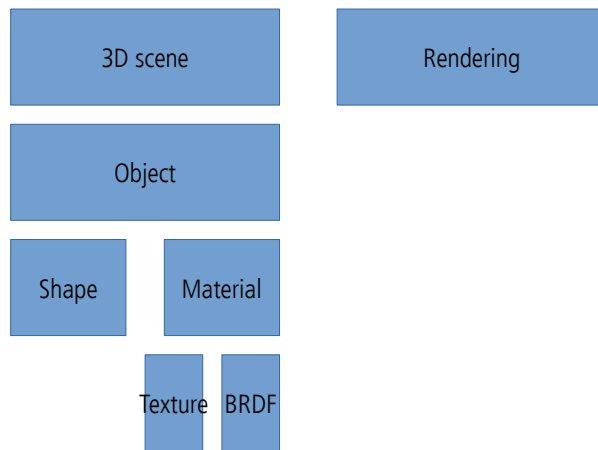
- Term **texture** is used in various disciplines and everyday life: texture of fabric, texture of food, texture in music, texture in image processing, texture in...
- Texture doesn't have well established definition. It means differently in different disciplines. It spans over multiple scales and over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

Texture and shading

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

Texturing: practical example

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



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

Texturing: practical example

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

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

Texturing: variation

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

Texturing pipeline*

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

Object space location

Projector function

Texture coordinate

Corresponder function

Texture space location

Texture obtain

Texture value

Value transform

Transformed value

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

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

Texturing pipeline: example

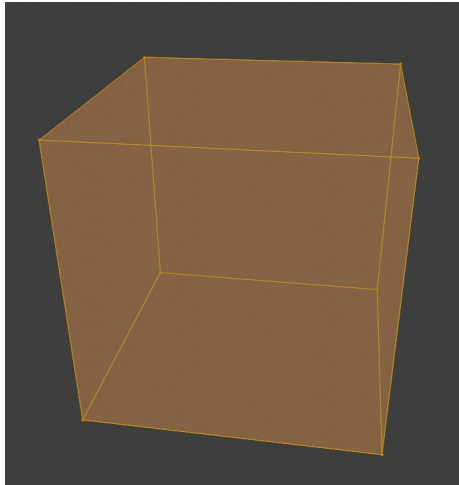
- Wooden floor
 - During rendering, **object position** (x,y,z) on wooden floor is found
 - **Projector function** finds mapping from (x,y,z) to (u,v) . In this case simple projection can be used: $u \sim x$, $v \sim y$ so that u,v are in $[0,1]$ – texture coordinates
 - **Corresponder function** uses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

Projector function

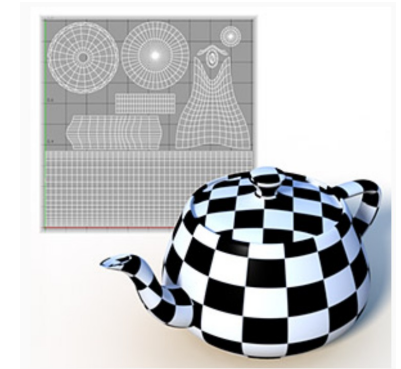
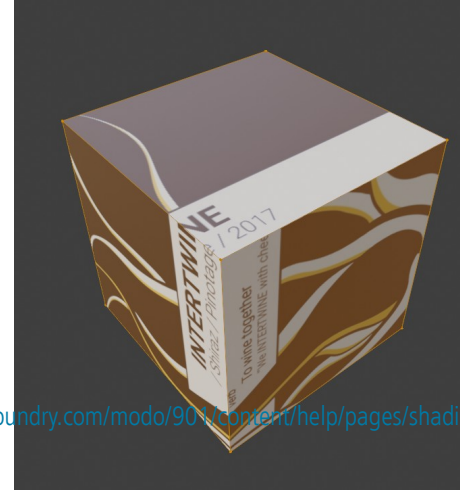
- First step in texturing process:
 - Obtaining surface location (given by renderer)
 - Projecting surface location into texture coordinate space (often, but not always, 2D space (u,v)).
- Texture coordinates can be calculated:
 - On the fly during rendering or
 - During object modeling and stored per-vertex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

Projector function: mesh unwrapping

- Mesh unwrapping tries to “unwrap” 3D surface onto 2D space so that each vertex is assigned with 2D texture coordinate (u,v)
- Unwrapping methods belongs to larger field called **mesh parametrization**
- Modeling packages offer different methods for unwrapping and a way to manually enhance the results
 - Goal is that each polygon gets fair share of texture area (e.g., evade stretching) but maintaining connectivity – which determines where separate parts of texture meet and visible seams.



https://learn.foundry.com/modo/904/content/help/pages/shading_lighting/shader_items/projection_type_samples.html



Projector function: texture projection

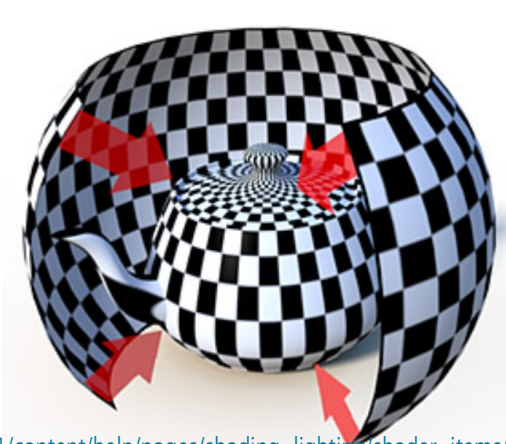
- The goal is to transform 3D point in space into texture coordinate* using projections:
 - Spherical
 - Cylindrical
 - Planar
 - Cubic
 - Box

https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

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

Projector function: spherical texture projection

- Spherical: texture is wrapped into a spherical shape and projected onto a shape. Good for texturing round objects. Surfaces perpendicular to projection cause stretching



Projector function: cylindrical texture projection

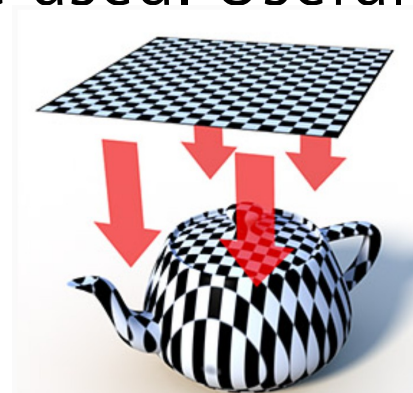
- Cylindrical: texture is wrapped in cylindrical shape and projected onto surface. Good for cylindrical surfaces (cans, bottles). Surfaces perpendicular to projection might cause stretching.



https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

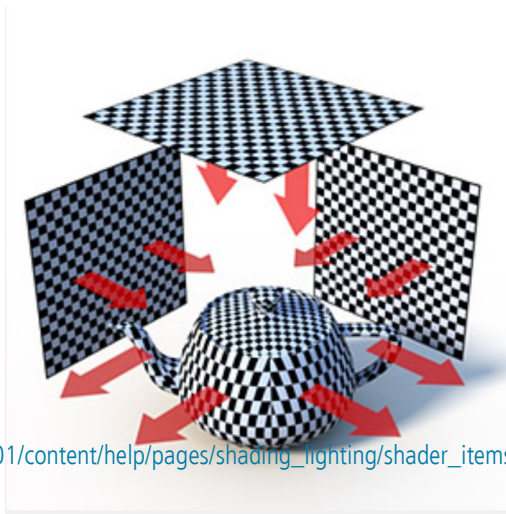
Projector function: planar texture projection

- Planar: similar to concept of movie projector but image is projected orthographically. Great for flat or nearly flat surfaces. Other surfaces might cause stretching of texture. Any vector defining projection plane can be used. Useful for decals.



Projector function: cubic texture projection

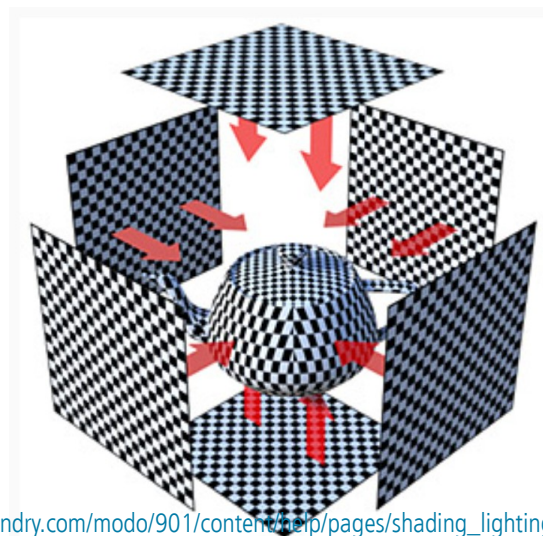
- Cubic: texture is planar-projected on a surface from all three axis directions, X, Y, and Z
- Polygon receives a certain projection, based on its normal direction.
- Best for cube-shaped objects, and occasionally on detailed surfaces where texture seams are not of great concern



https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

Projector function: box texture projection

- Projects a texture in a planar fashion from all six directions, eliminating reverse projections on rear facing polygons inherent to the Cubic method
- Best on cube-shaped objects, and occasionally on detailed surfaces where texture seams are not of great concern



https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

Projector function: front texture projection

- Texture is projected out from a camera or light's position



Projector function: complex shapes

- For highly complex shapes, separating into simpler shapes can be done and then texture projection can be done separately.
- [983] Pagán, Tito, "Efficient UV Mapping of Complex Models," Game Developer, vol. 8, no. 8, pp. 28-34, August 2001

Projector function: texture projection

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

Projector function: different approaches

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

Projector function: different approaches

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

Projector function: inherent texture coordinates

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

Projector function: different texture coordinates

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

Projector function: different texture coordinates

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

Projector function: multiple texture coordinates

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

Corresponder function

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

Obtaining texture values

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

Transforming texture values

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

Image textures

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

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



Magnification and minification

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

Anti-Aliasing: sampling and filtering

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

Anti-aliasing: mipmapping

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

Anti-aliasing: mipmapping

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


Examples of image textures

- Substance:

<https://substance3d.adobe.com/assets/allassets?assetType=substanceMaterial&assetType=substanceAtlas&assetType=substanceDecal>

- PolyHaven: <https://polyhaven.com/textures>

Volume textures

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

Volume textures

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

Procedural textures

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

<IMAGES: PROCEDURAL TEXTURING>

<IMAGES: NODE SYSTEM FOR PROCEDURAL TEXTURING>

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

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

Procedural textures: volumetric textures

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

Procedural textures: Perlin noise

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

Procedural textures: Cellular texture

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

Procedural texture: physical simulation

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

Procedural textures: Antialiasing

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

Material mapping: variation of parameters

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

Usage of texture values

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

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

Alpha texture values

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

Bump mapping

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

Bump mapping: scale of modeling

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

Bump mapping: idea

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

Bump mapping: frame of reference

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

Bump mapping: Blinn method

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

Normal mapping

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

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

Normal mapping: practical notes

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

* Normal map applied to specular surface results in non-linear relationship between shaded color and normal map. Lambertian (diffuse) surface has almost linear relationship between shaded color and normal map since base of lambertian shading is based on dot product with normal which is linear operation.

** Generally, when light falls on surface with bumps, multiple scattering, masking and shadowing effects happen which are not possible with normal mapping.

Creating bump/normal mapping

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

Texture modeling and texture application

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

Storing and transferring materials

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

Practical note: textures

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

Exploring textures

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

More into topic

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

Practical insights

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

Further into topic

- TODO

Literature

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