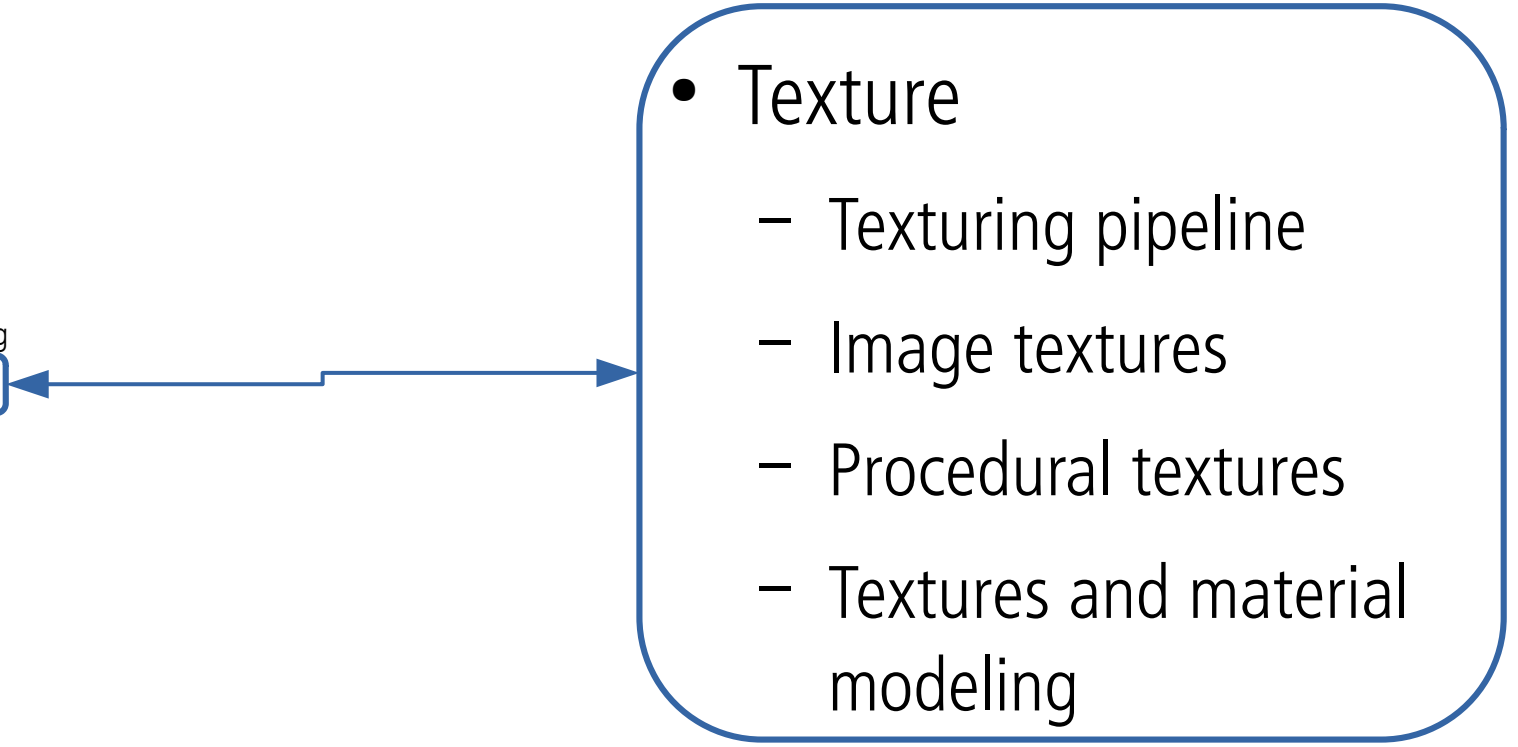

Texture
SUBSTANCE
DESIGNER

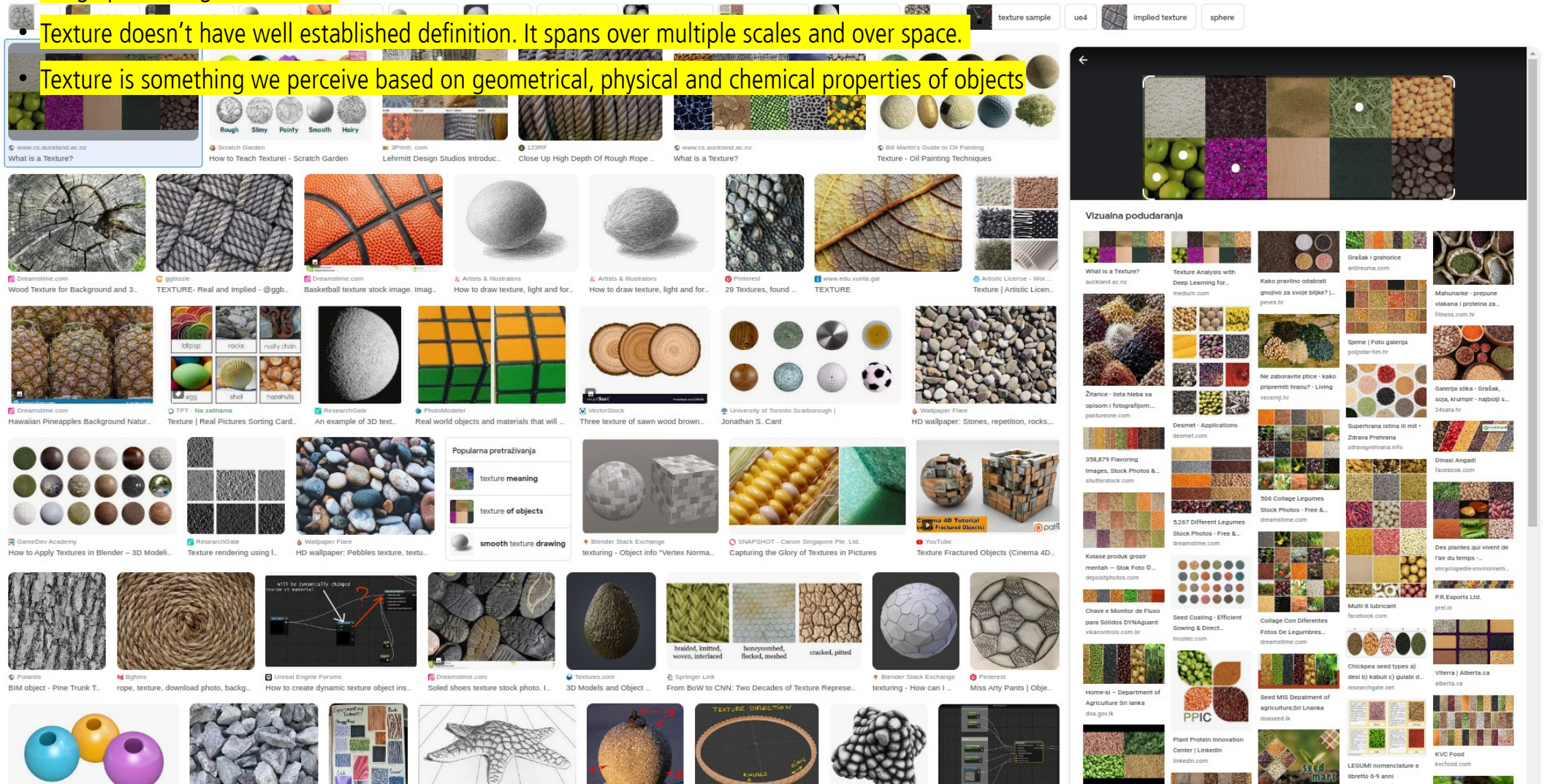
Syllabus

- 3D scene
  - Object
    - Shape
    - Material
      - Scattering
      - **Texture**
  - Camera
  - Light
- Rendering
- Image



Introduction

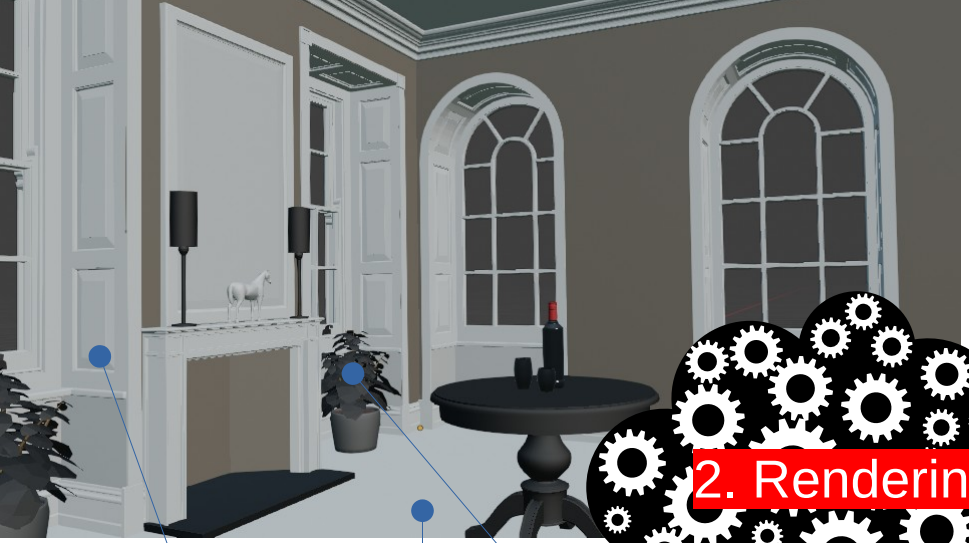
- Term **texture** is used differently in various disciplines and everyday life: texture of fabric, texture of food, texture in music, texture in image processing, texture in...
- Texture doesn't have well established definition. It spans over multiple scales and over space.
- Texture is something we perceive based on geometrical, physical and chemical properties of objects



- Metal
- Plastic
- Glass
- Wood
- Fabric
- Stone
- Clouds
- Water
- Tree bark
- Leafs
- Plaster
- Paper
- Leather
- Sky
- Etc.

https://www.xomacenna.com/blog/a-guide-to-organic-modern-design

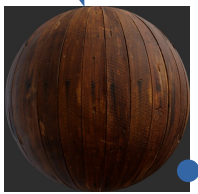
Big picture



1. Scene modeling

3. Image

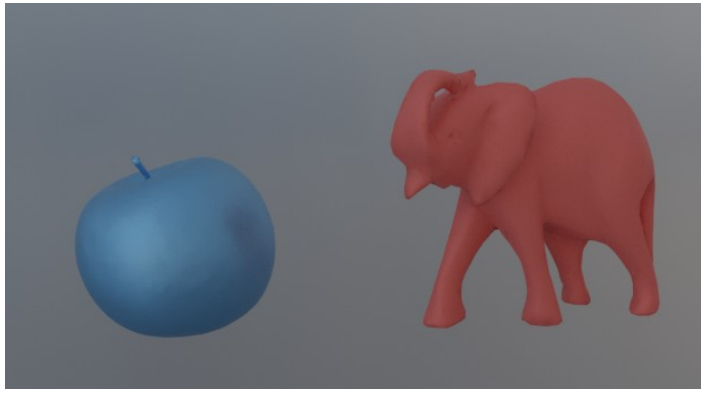
Texture defines look and feel of objects



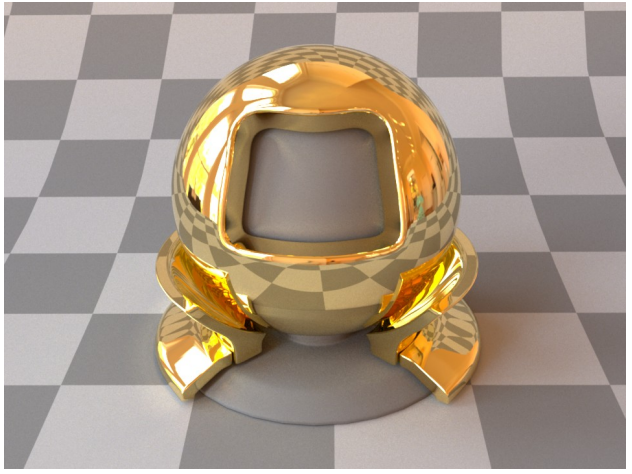
Material characteristics

- **Characteristics that needed to be modeled** in order to obtain required appearance. We can classify any material by following variations:

Color



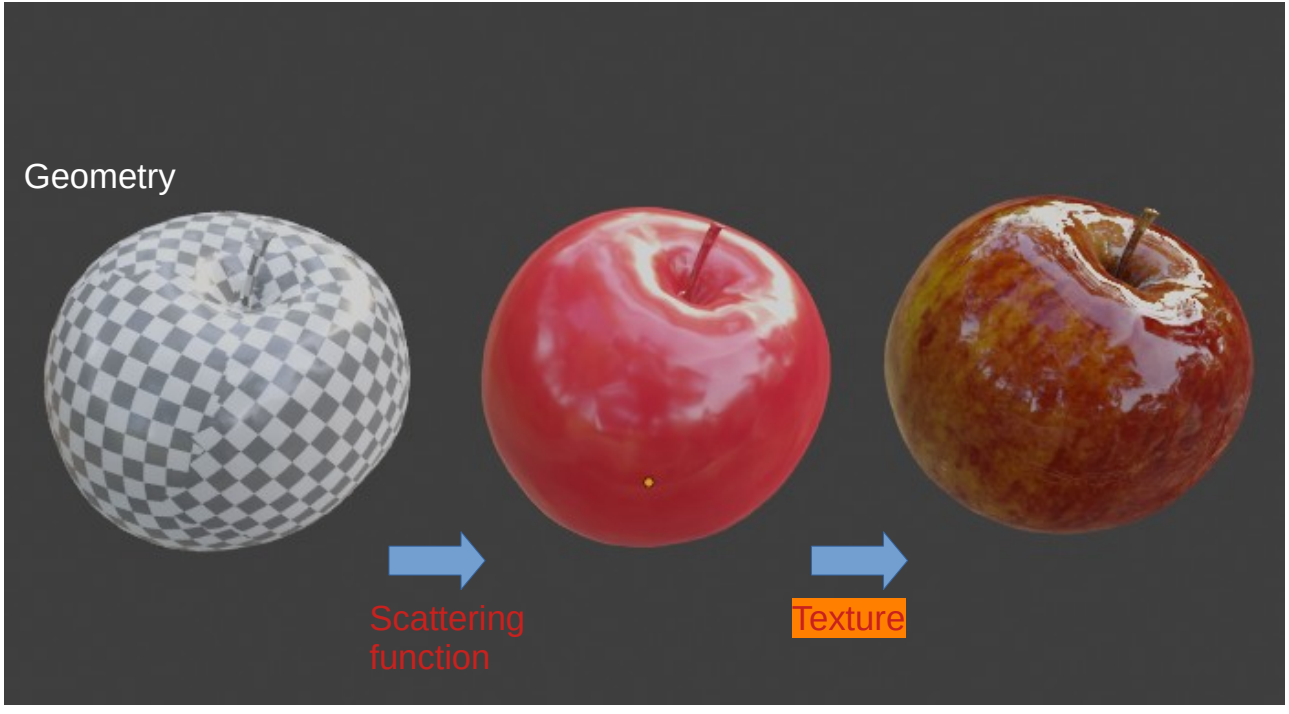
Directional (e.g., reflectivity)



Texture and patterns

Texture in computer graphics

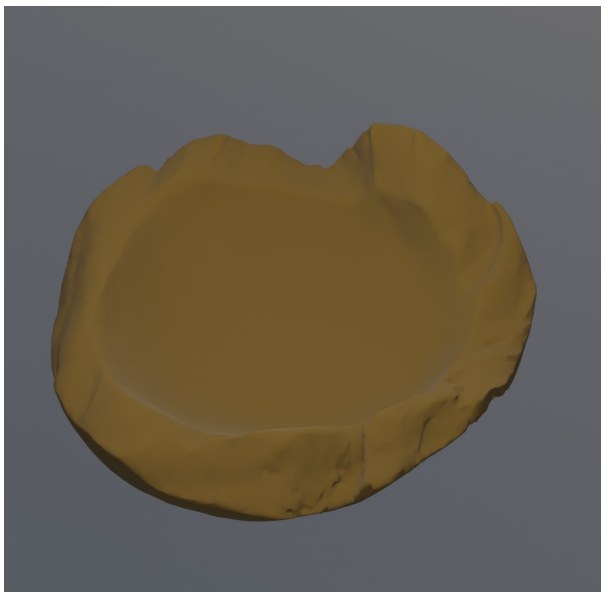
- For 3D modeling, we can try to define texture as function which **varies scattering function properties over surface**.



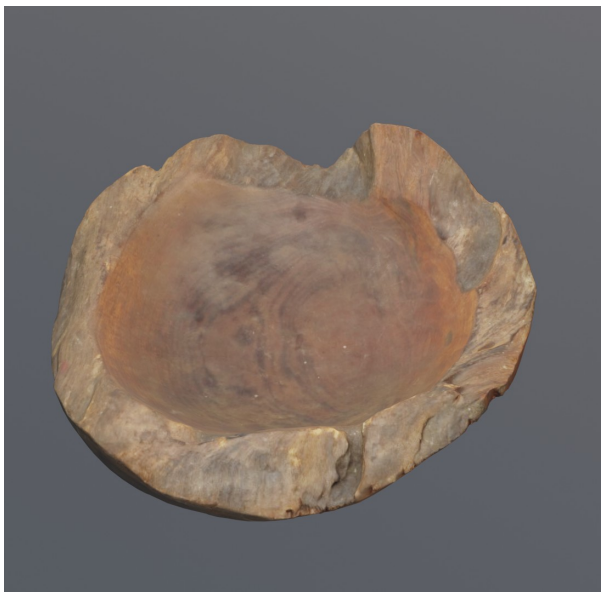
Variation of material over surface

- Real objects rarely have only one material or same material properties over the surface → **homogeneous material**
  - Example: 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 materials look too smooth and perfect. Missing details, variations and imperfections.



Diffuse scattering model with constant color



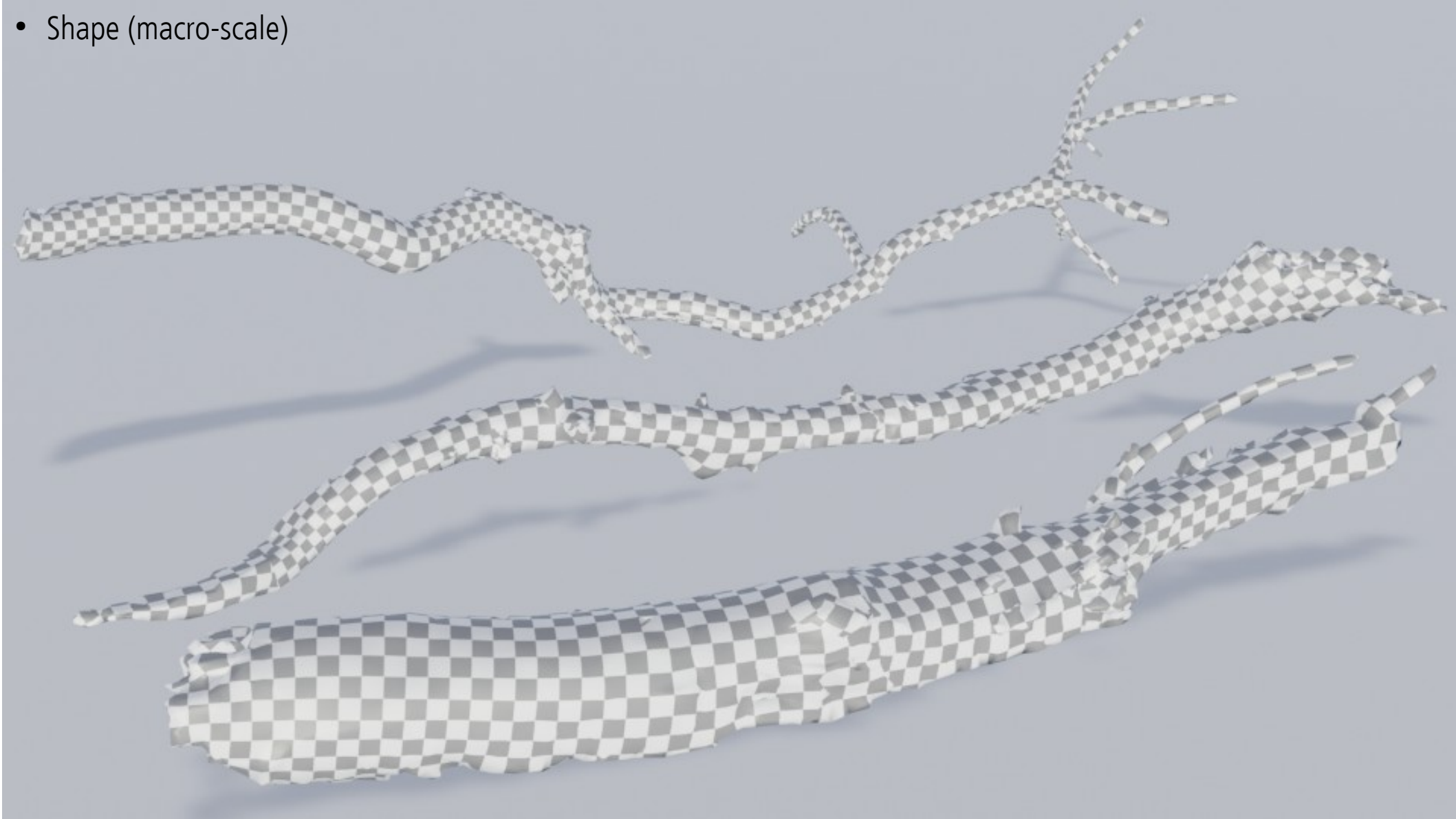
Diffuse scattering model with texture varying color

Texture

- Visual variation on the object surface – **spatial variation → details and richness of surface**
  - **Directional or color characteristics → scattering function parameters**
  - **Small-scale geometrical characteristics**: bumps, pores, imperfections, etc. → **surface normals**
- In computer graphics, **texturing** is process that takes surface and modifies its appearance at each location using image, function or other data source.

* 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.

- Shape (macro-scale)



- Shape (macro-scale)
- Scattering function: diffuse (microscale)
  - Albedo parameter (color) → constant

- Shape (macro-scale)
- Scattering function: diffuse (microscale)
  - Albedo parameter (color) → texture variation

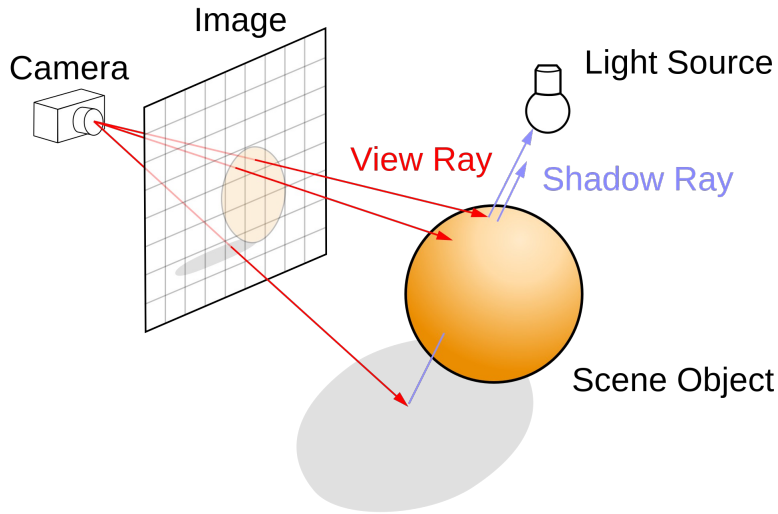
- Shape (macro-scale)
- Scattering function: diffuse (microscale)
  - Albedo parameter (color) → texture variation
- Small-scale geometry: surface normals (mesoscale)
  - Normal perturbation → texture variation

Texturing: variation

- Textures introduce
  - **Scattering function parameters (e.g., color) variation**
  - **Small-scale geometry variation** → Shading normal variation
- Texturing scale
  - **Microscale** variation → **scattering function parameters**
  - **Mesoscale** variation → **shading normal perturbation**
  - **Macroscale** is given by the object shape
- Texturing algorithms: trade-off between complexity and quality.

Texture and rendering

- Rendering: computing color of pixels in an image
- Color of pixels → color of objects visible from camera
  - For each pixel, viewing ray is generated, traced into scene for closest intersection.
  - Color calculated at intersection → pixel color.
- **Shading**: calculating color at viewing ray intersection with object → **shading point.**



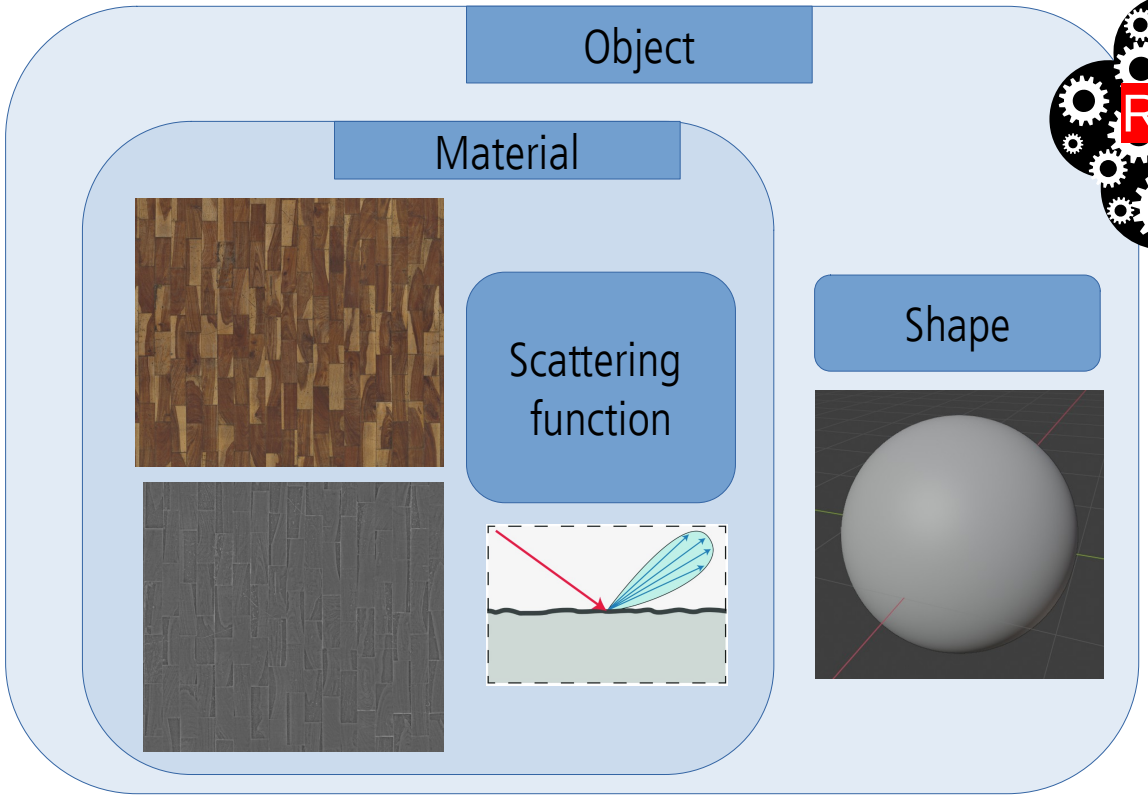
Texture and shading

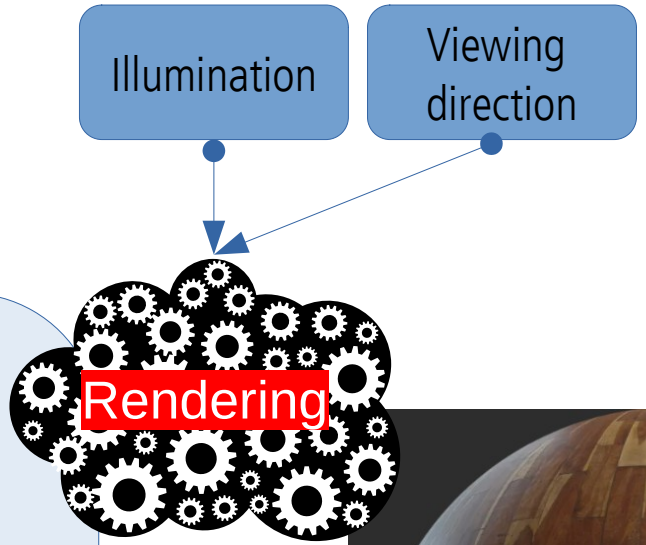
- Shading uses **material**, **shape**, **viewing** and **light** to calculate the color in shading point
- **Material** describes interaction of light with surface:
  - **Scattering model**
  - **Texture model**
- For each shading point, **scattering model is evaluated using texture information** at that point
- Final rendered image, due to variation of colors and intensities over object, the surface will contain pattern
  - Surface with pattern is called textured surface

Note on “texture”

Texture in modeling vs texture on rendered surface

- Texture defines shading parameters over object surface → something that we model
- Resulting pattern of colors on rendered surface → generated texture

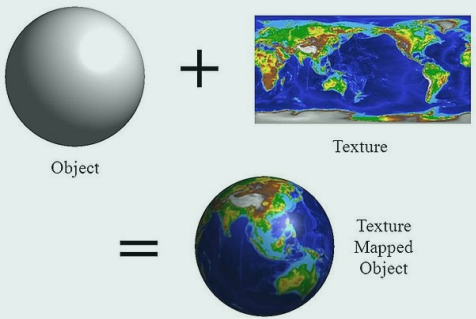




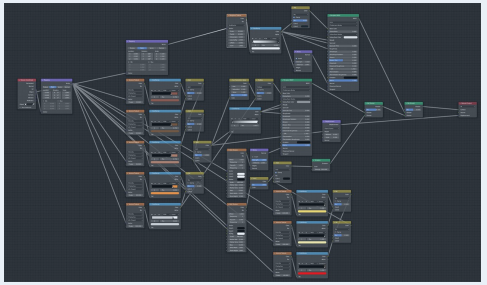
Note on “texture”

- Texture is generally and in computer graphics overloaded term. Meaning depends on the context:
  - Texture can be used for modeling object appearance (material)
  - On GPU rendering, texture can be used as memory buffer to send arbitrary data (e.g., geometry) from CPU to GPU

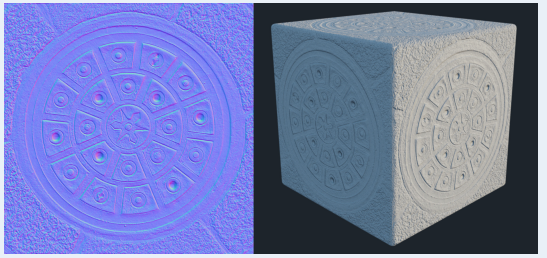
Learning objectives



How to apply textures on objects → **texturing pipeline**



Texturing approaches → **image and procedural**

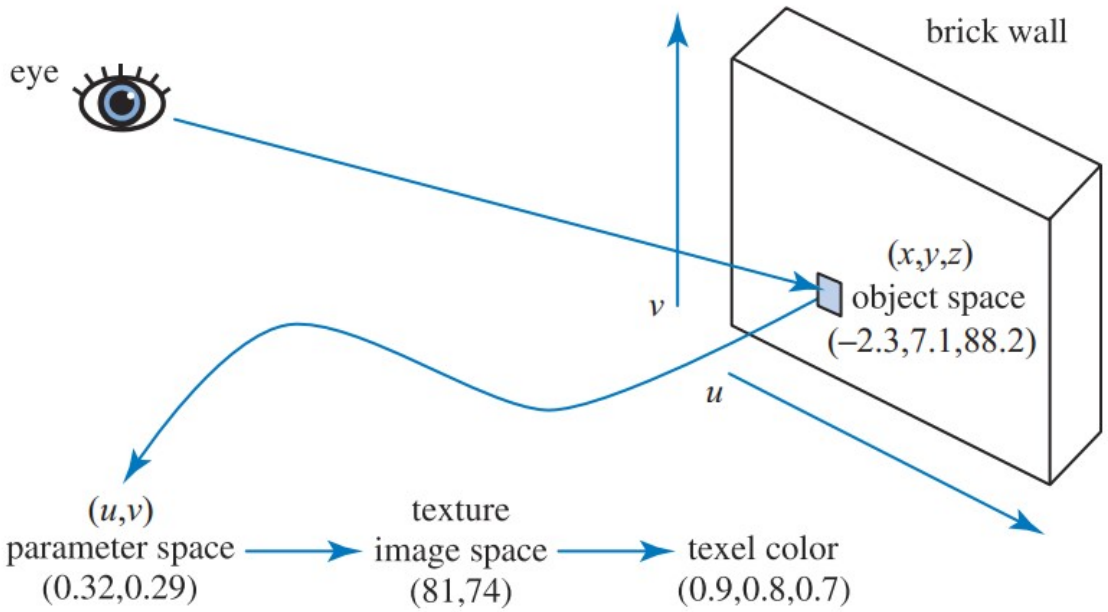


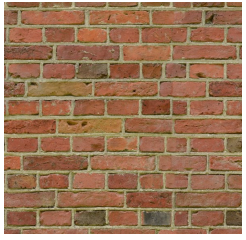
How to use texture for **material modeling** → variation of scattering function parameters and small scale geometry

Texturing pipeline

Texturing pipeline: example

- Brick wall
  - **Camera ray → object position** (x,y,z)
  - **Projector function:** mapping from (x,y,z) to (u,v) → **texture coordinates**
  - **Corresponder function:** texture coordinates → pixel in the texture image.
  - **Texture reading**
  - Optional **transformation on values**

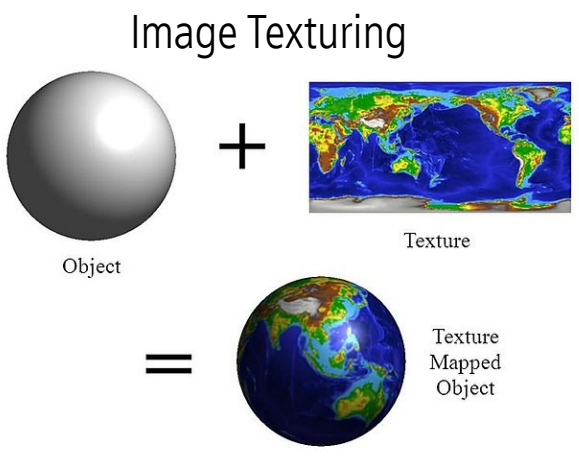


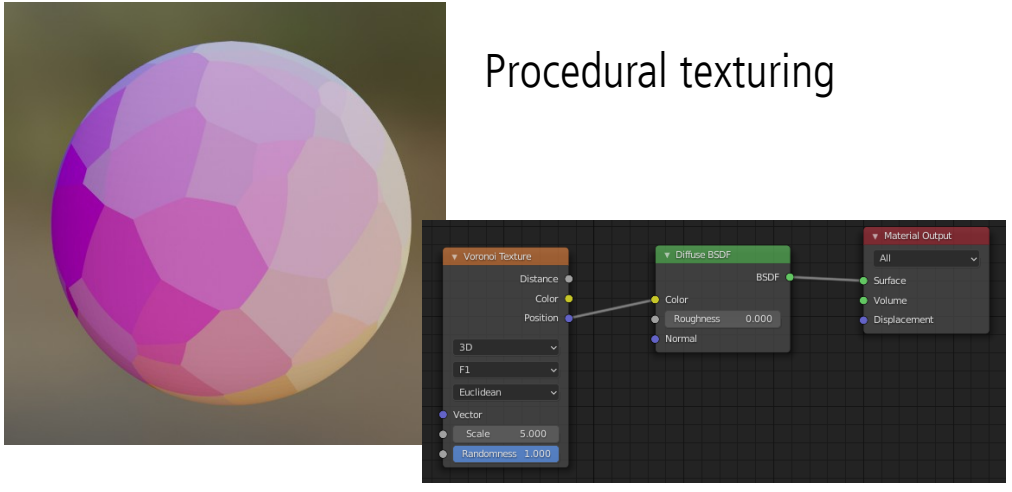


https://www.realtimerendering.com/

Texturing pipeline

- Texturing pipeline is performed by renderer and can be applied on two main types of texturing:
  - **Image texturing**: e.g., 2D image is "pasted" on 3D object
  - **Procedural texturing**: e.g., an algorithm generates pattern on 3D object
- Some steps of texturing pipeline can be omitted depending on texturing method
  - Example: procedural texturing can directly use object space location
- For start, we will explain texturing pipeline for **image textures**

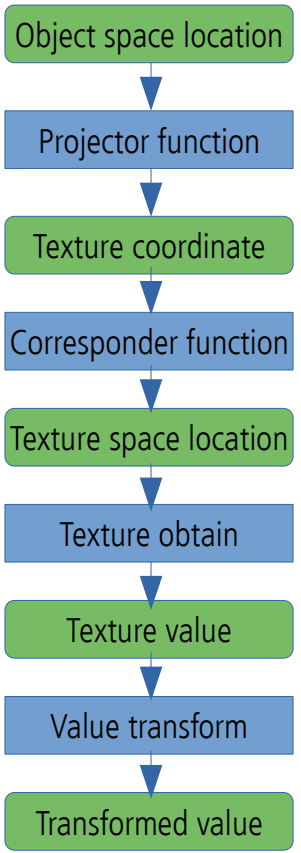




Procedural texturing

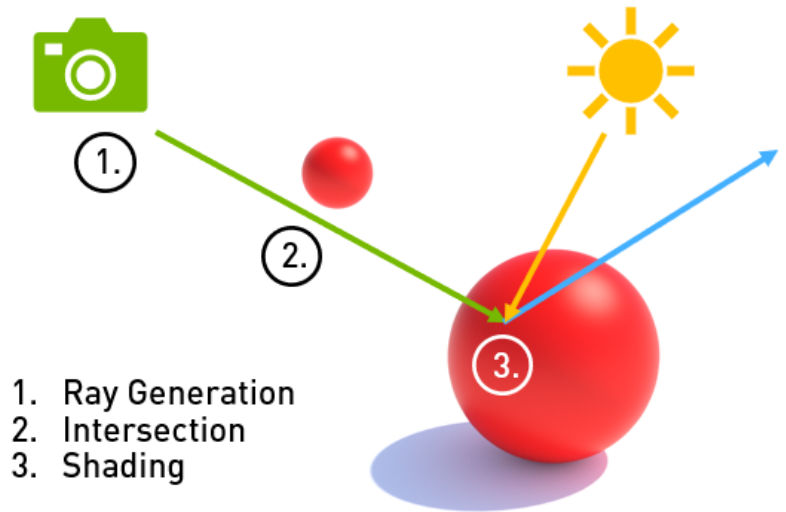
Texturing pipeline

- **Texturing**: variation of shading parameters over surface – **position dependency**
- **Object space location**
  - Shading point → object-space coordinates `(x,y,z)`, e.g., `(1.2, 3.3, -3.4)`
- **Projection function**
  - Mapping process: object space coordinates → texture space coordinates `(u,v)` in `[0,1]`, e.g., `(0.2, 0.5)`
- **Corresponder function**
  - Transform: texture space coordinates → texture locations `(s,t)` e.g., `[image_width, image_height]`, e.g., `[128, 44]`
- **Obtaining texture values (sampling)**
  - Texture space location is used for obtaining texture value
- **Value transform**
  - Optional additional transformations are done on texture value
- **Usage**
  - Final value is used to modify property of surface at that point: scattering function parameter or normal.

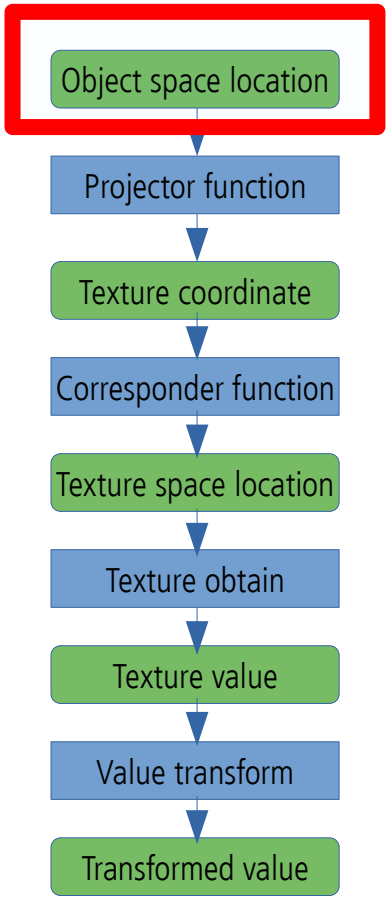


Object space location

- **Intersection** (shading) point
  - **Object space location** `(x,y,z)`
  - World space object location can also be used
    - In this case, as object moves, the coordinates change!

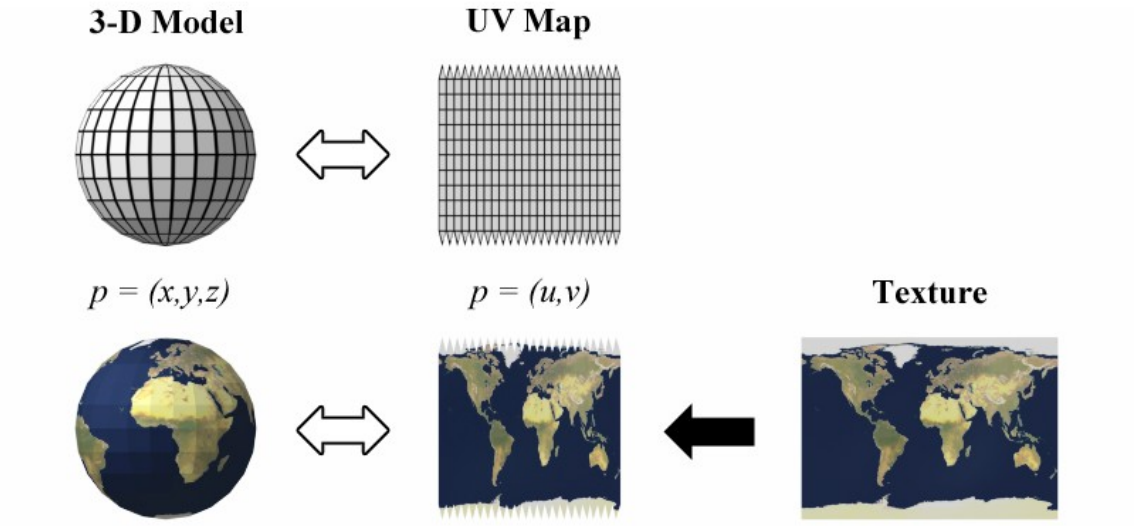


https://developer.nvidia.com/blog/rtx-best-practices/

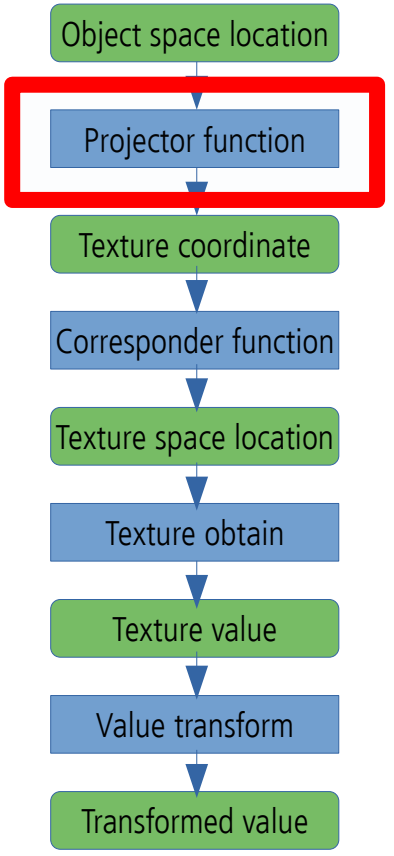


Projector function

- Projector function maps **object space location** into **texture coordinate space**
  - Thus "texture mapping" term
- For image texture: 3D object point `(x,y,z)` to 2D space `(u,v)` in `[0,1]`

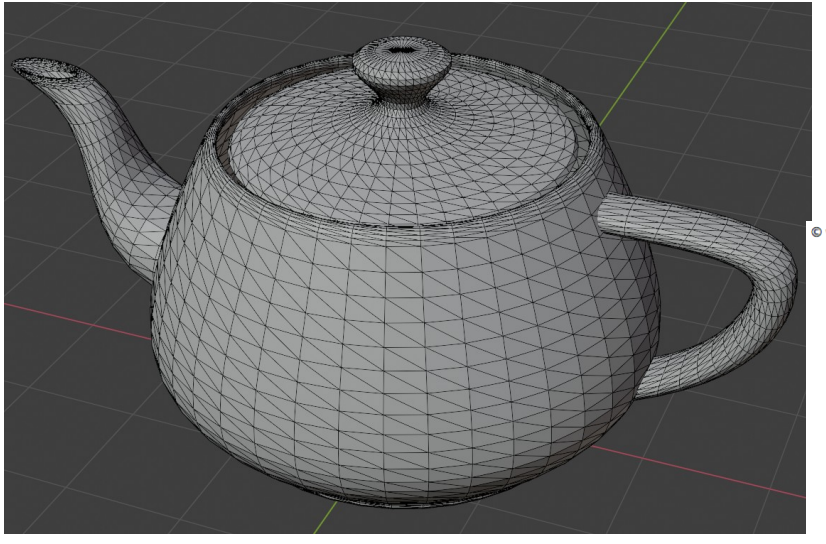


https://users.csc.calpoly.edu/~zwood/teaching/csc572/final10/wdhamilt/

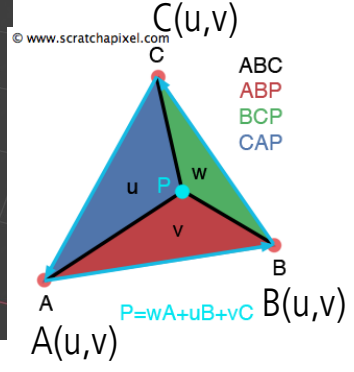


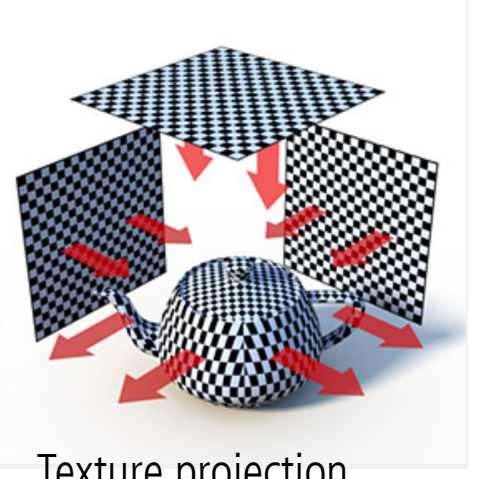
Projector function

- Object space location is used to:
  - **Interpolate texture coordinates stored per vertex**
  - **Calculate texture coordinates using texture projection**



Texture coordinates stored per vertex

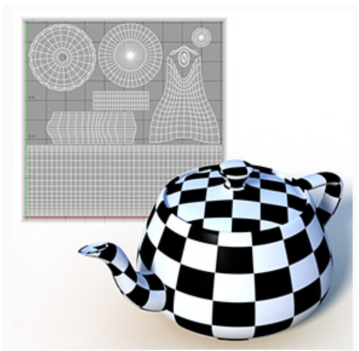




Texture projection

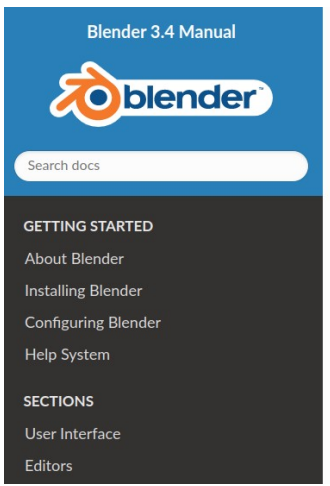
Vertex texture coordinates

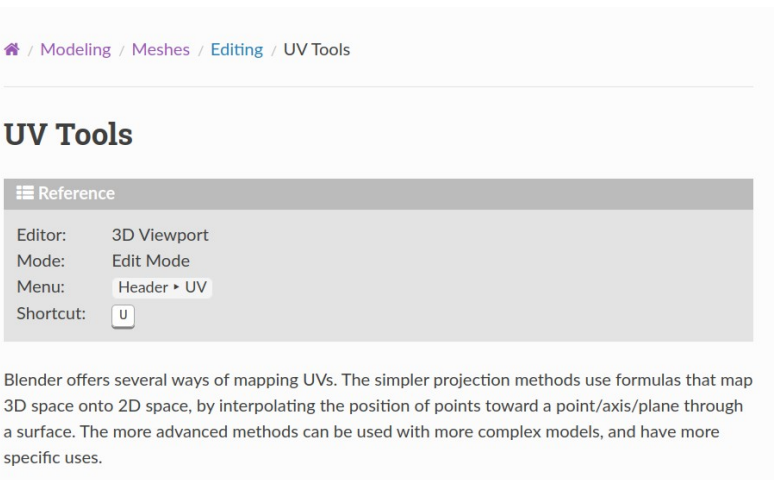
- **Mesh unwrapping or texture projection → generates texture coordinates per triangulated mesh vertex** (performed prior rendering)
- Modeling packages implement different methods for generating texture coordinates per vertex and enable artist to edit texture coordinates



Unwrapping

Texture projection

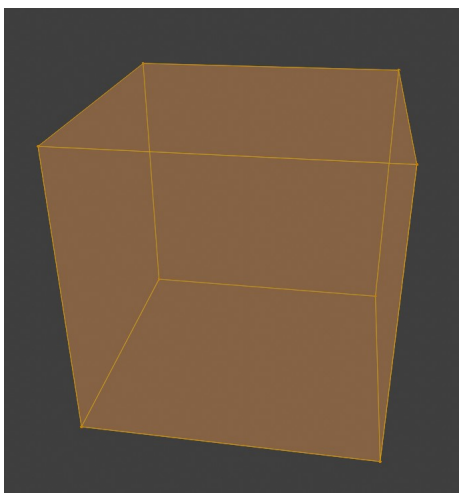


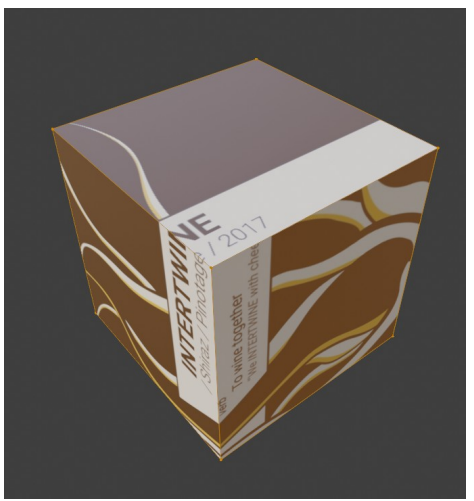


https://docs.blender.org/manual/en/latest/modeling/meshes/editing/uv.html

Projector function: mesh unwrapping

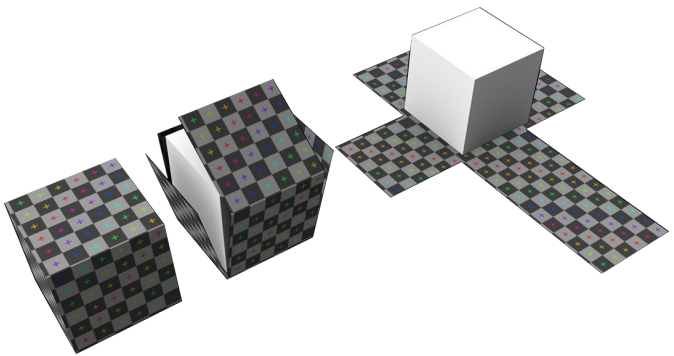
- **Mesh (UV) unwrapping**: each vertex is assigned with 2D texture coordinate `(u,v)` in `[0,1]`
  - Unwrapping methods belongs to larger field called **mesh parametrization**
- Goal is that each polygon gets fair share of texture area (e.g., evade stretching) but maintaining connectivity – which determines where separate parts of texture meet and visible seams.

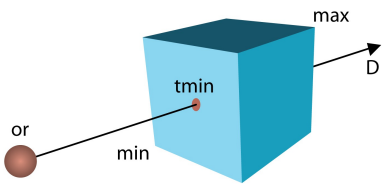


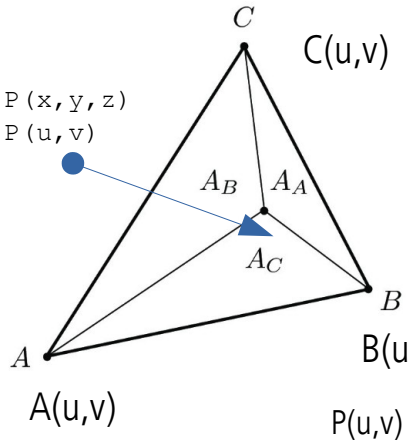


Vertex texture coordinates

- For any intersected/shading point P(x,y,z) on triangle of mesh, **barycentric interpolation** is used to calculate texture coordinate from ones stored per vertex







$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$
$$\gamma = \frac{A_C}{A_A + A_B + A_C}$$

$P(u,v) = \alpha A(u,v) + \beta B(u,v) + \gamma C(u,v)$

$\alpha + \beta + \gamma = 1$

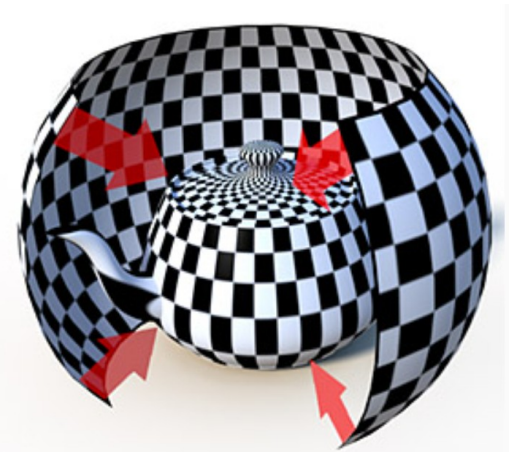
https://help.disguise.one/en/Content/3D-Workflow/UV-Mapping/What-is-UV-mapping.html

Texture projection

- Transform 3D point in space into texture coordinate using projections:
  - Spherical
  - Cylindrical
  - Planar
  - Cubic
  - Box
  - Front
- Found texture coordinate is used for evaluating texture (image or procedure)

Spherical texture projection

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



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

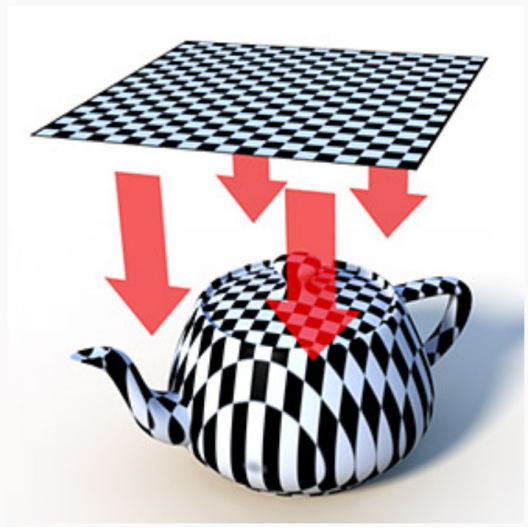
Cylindrical texture projection

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

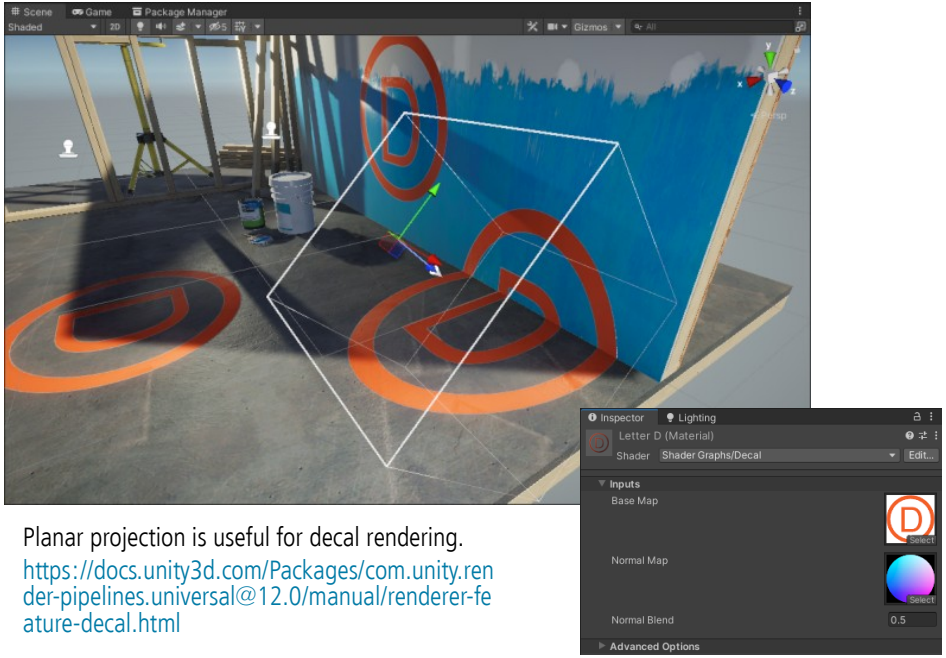
https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

Planar texture projection

- Similar to concept of movie projector but image is projected orhographically.
  - Any vector defining projection plane can be used.
- Great for flat or nearly flat surfaces.
- Other surfaces might cause stretching of texture.



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

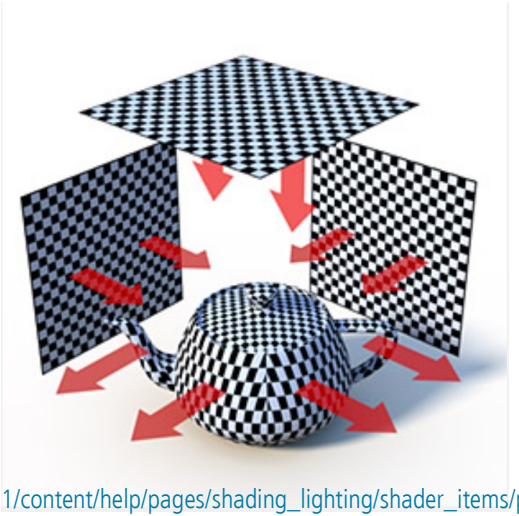


Planar projection is useful for decal rendering.

https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.0/manual/renderer-feature-decal.html

Cubic texture projection

- 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

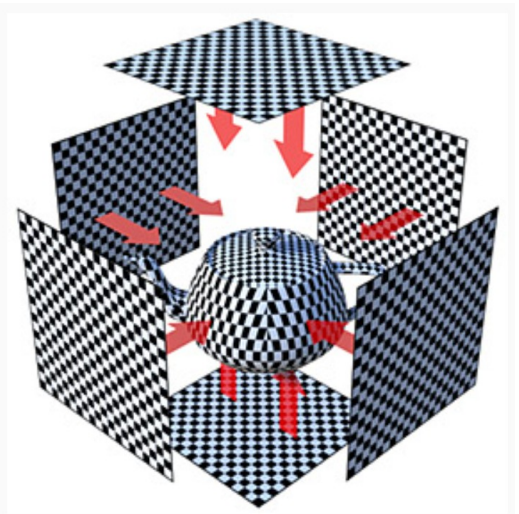


https://learn.foundry.com/modo/901/content/help/pages/shading_lighting/shader_items/projection_type_samples.html

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 (aka **triplanar mapping**)
- Best on cube-shaped objects, and occasionally on detailed surfaces where texture seams are not of great concern
  - For curved and tilted surfaces efficient **blending** of multiple projection planes must be used.

https://ryandowlingsoka.com/unreal/triplanar-dither-biplanar/



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

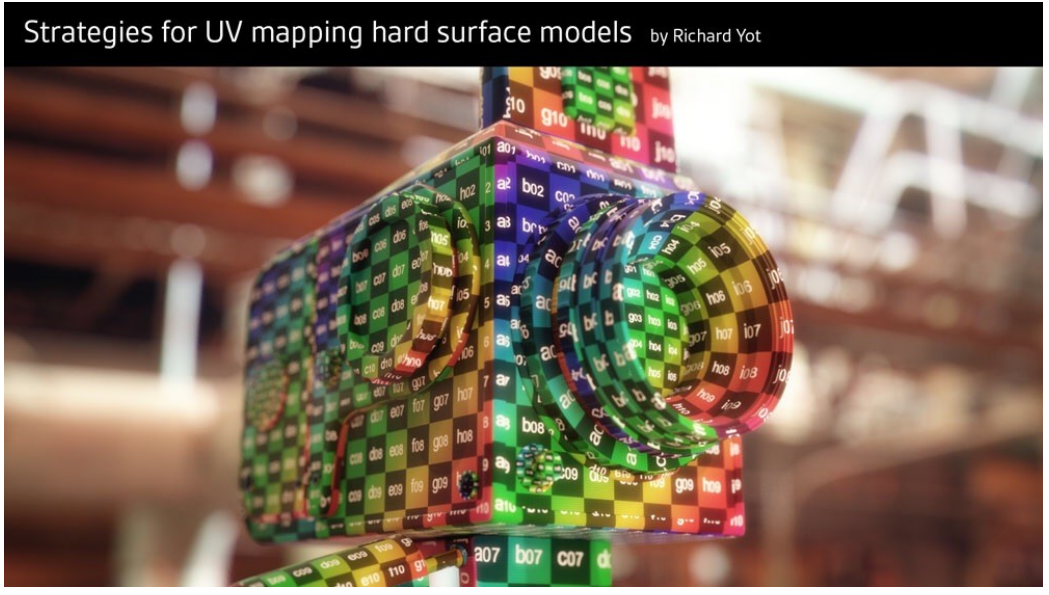
Front texture projection

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

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

Projector function: complex shapes

- For highly complex shapes, separating into simpler shapes can be done and then texture projection can be done separately.



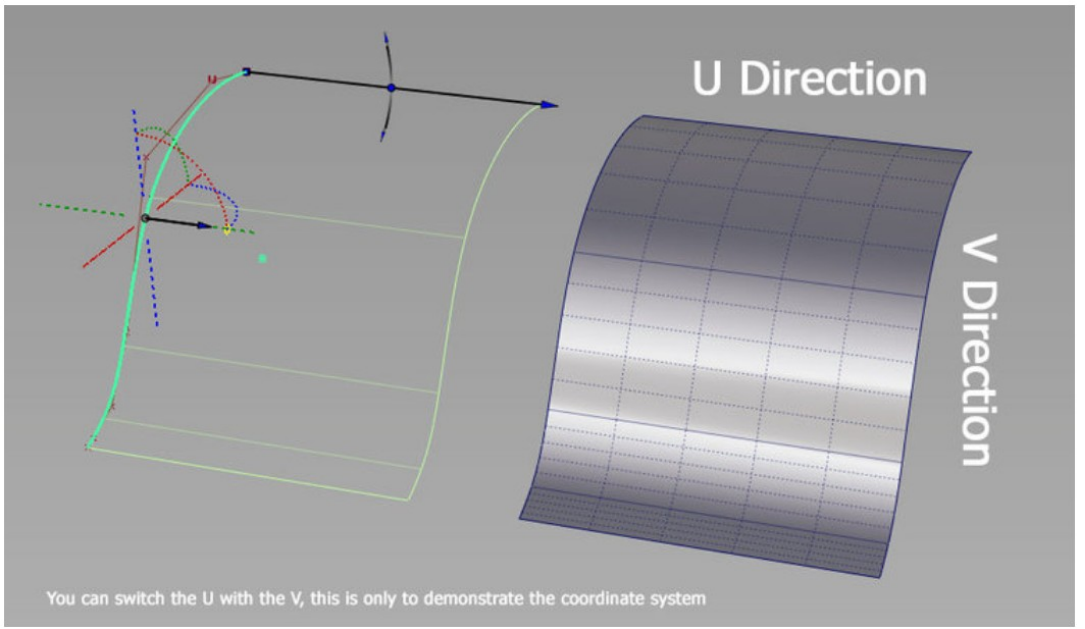
[983] Pagán, Tito, "Efficient UV Mapping of Complex Models," Game Developer, vol. 8, no. 8, pp. 28-34, August 2001

Projector function: different approaches

- If object is supplied with any kind of additional information (e.g., curvature or temperature) it can be also used for texture coordinate generation
- **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

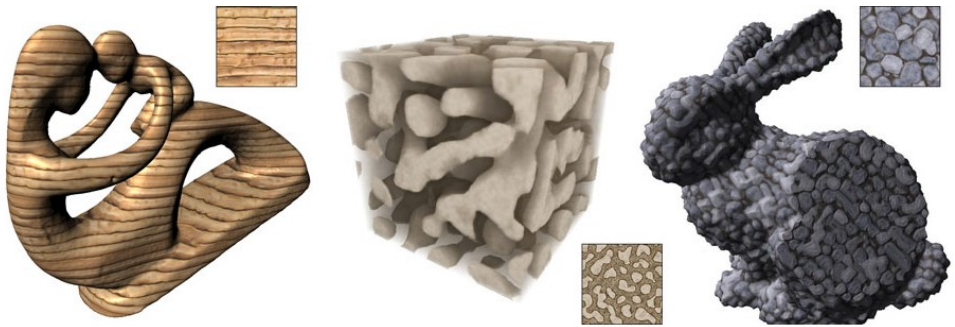
- Parametric surfaces (e.g., NURBS) have natural set of (u,v) texture coordinates which defines any (x,y,z) point on surface
- In this case, we can say that surface representation already has inherent texture coordinates and projector function is not needed.

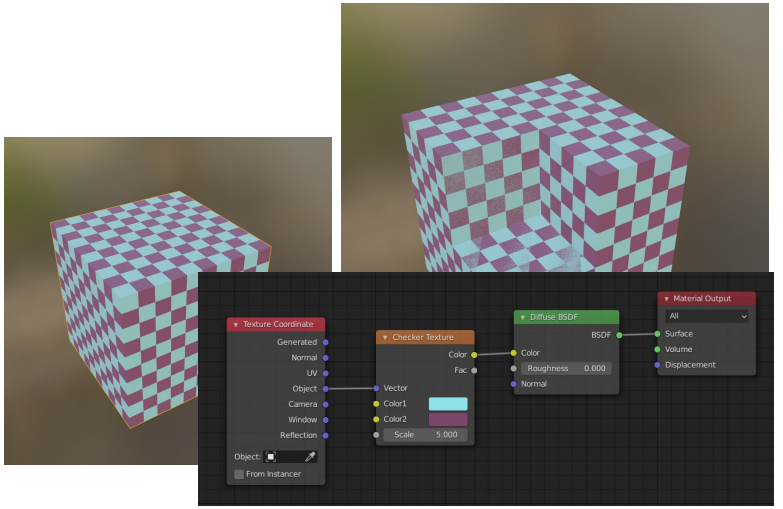


https://www.ebalstudios.com/blog/introduction-nurbs

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)
  - procedural texturing can directly rely on object locations (x,y,z) – solid texturing.
- For animation, texture coordinates can be (u,v,w,t) where t is time and determines change in texture



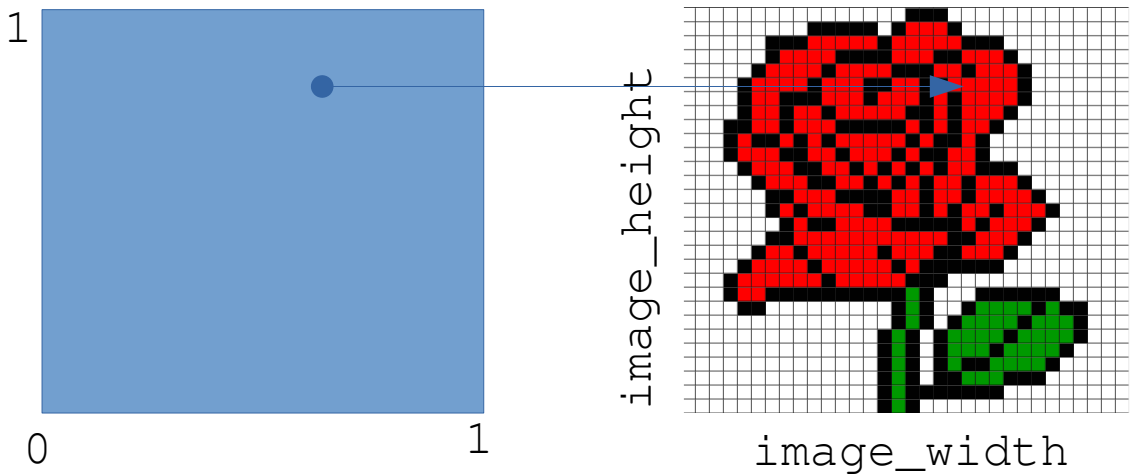


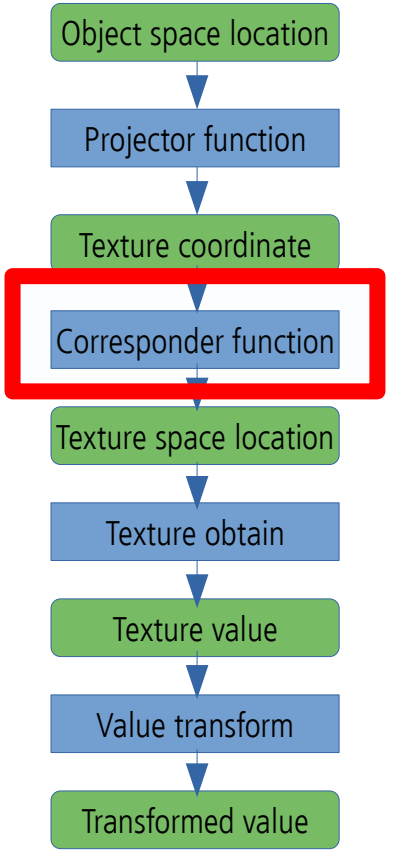
Projector function: multiple texture coordinates

- 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

Corresponder function

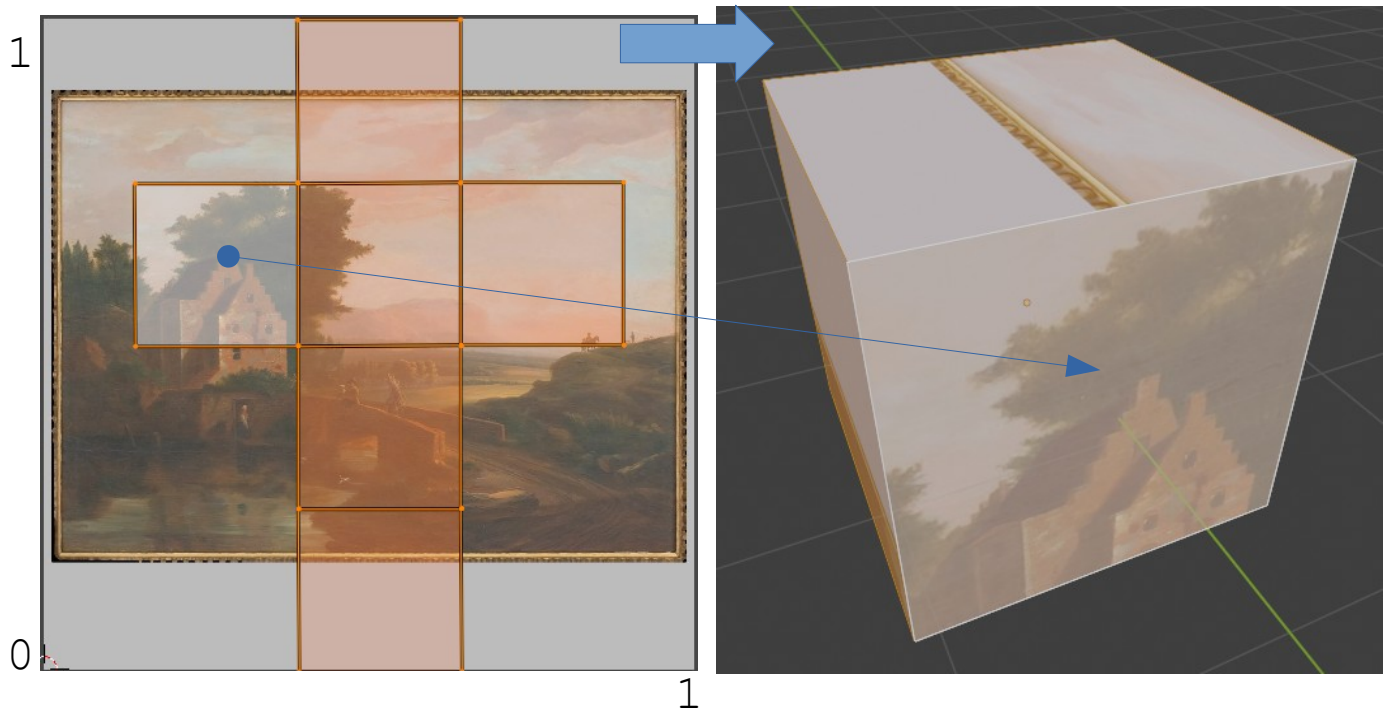
- Transform: **texture coordinates → texture locations**
- For image texture, image indices to retrieve pixel values.
  - Transform: `(u,v)` in `[0,1]` → `(s,t)` in `[image_width, image_height]`





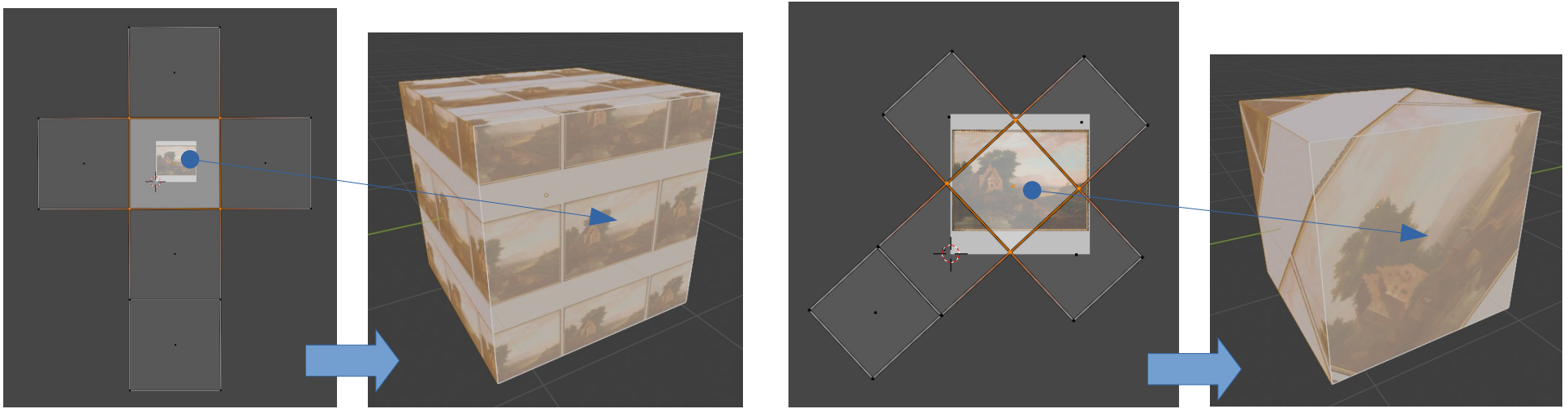
Corresponder function

- Corresponder function:
  - Determine which part of texture will be used



Corresponder function

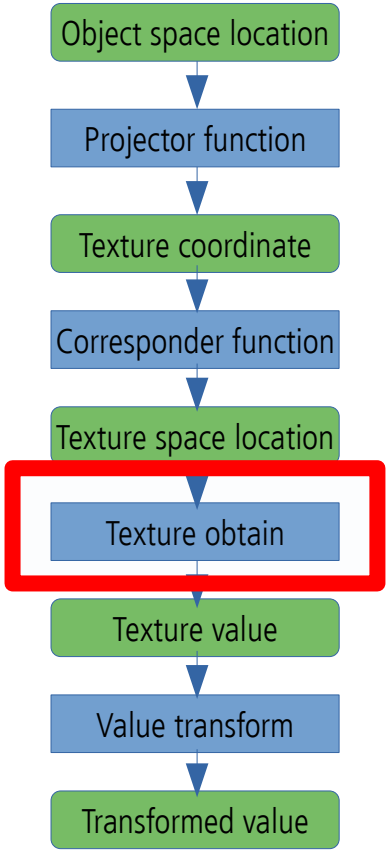
- **Transformations: rotate, scale, translate, shear or project**
- **Originally (u,v) is in [0,1]. Scaling can move (u,v) to be larger than one**
  - Image wrapping: **repeat** (other: wrap, mirror, clamp, border, wang tiles, etc.)



Demo: https://math.hws.edu/graphicsbook/demos/c4/texture-transform.html

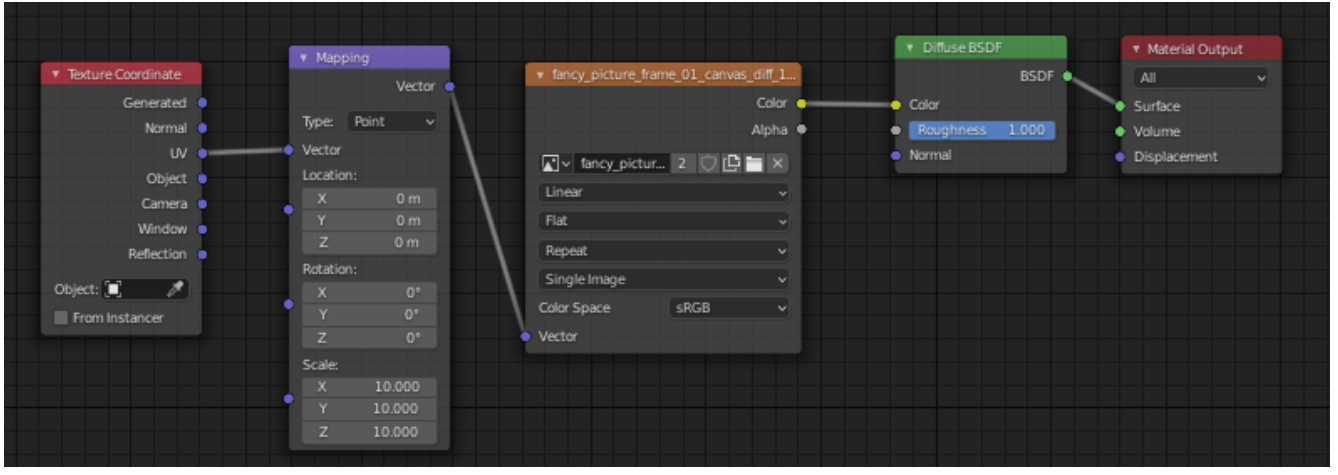
Obtaining texture values

- Texture space location is used for obtaining texture value:
  - Reading image texture
  - Evaluating procedural texture



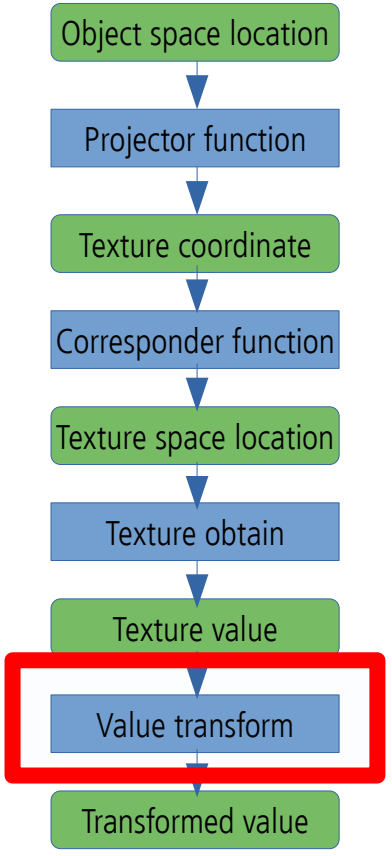
Obtaining texture values

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



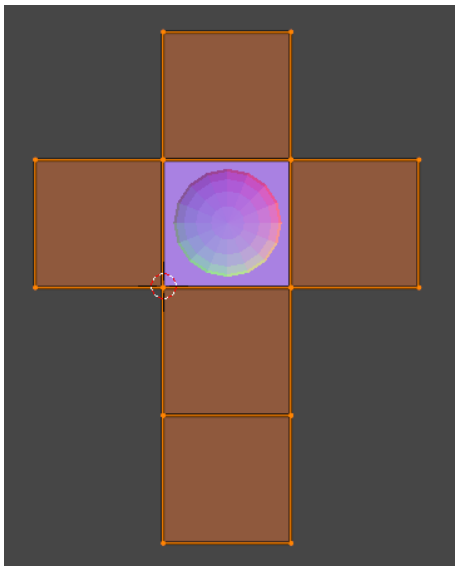
Transforming texture values

- Optional additional transformations on texture value
- 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 then color.

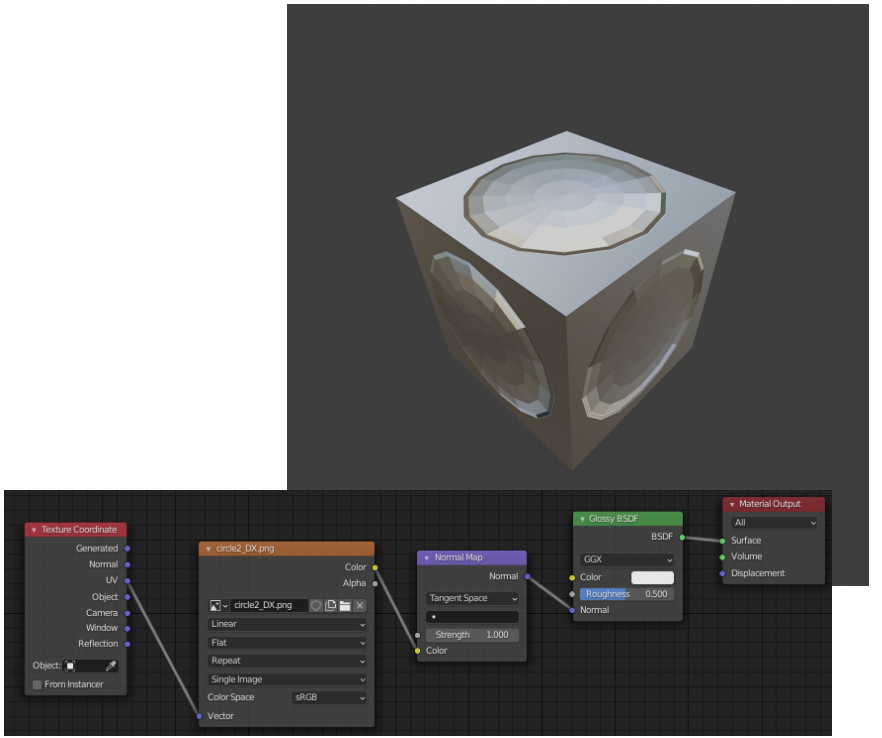


Transforming texture values

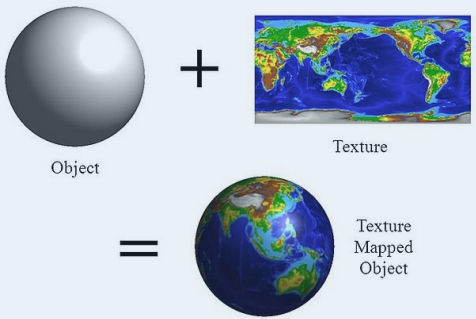
- Example: RGB values can be interpreted as vectors.



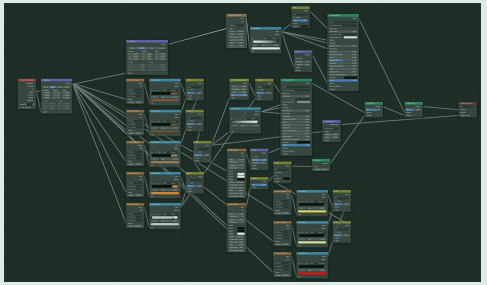
Example: encoding normal vectors into image: **normal maps**



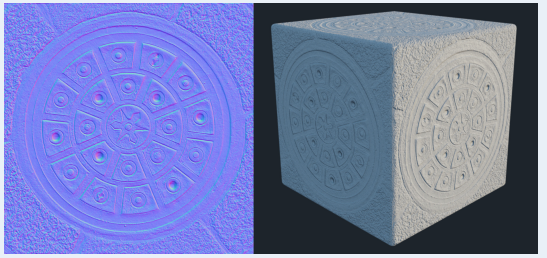
Learning objectives



How to apply textures on objects → **texturing pipeline**



Texturing approaches → **image and procedural**

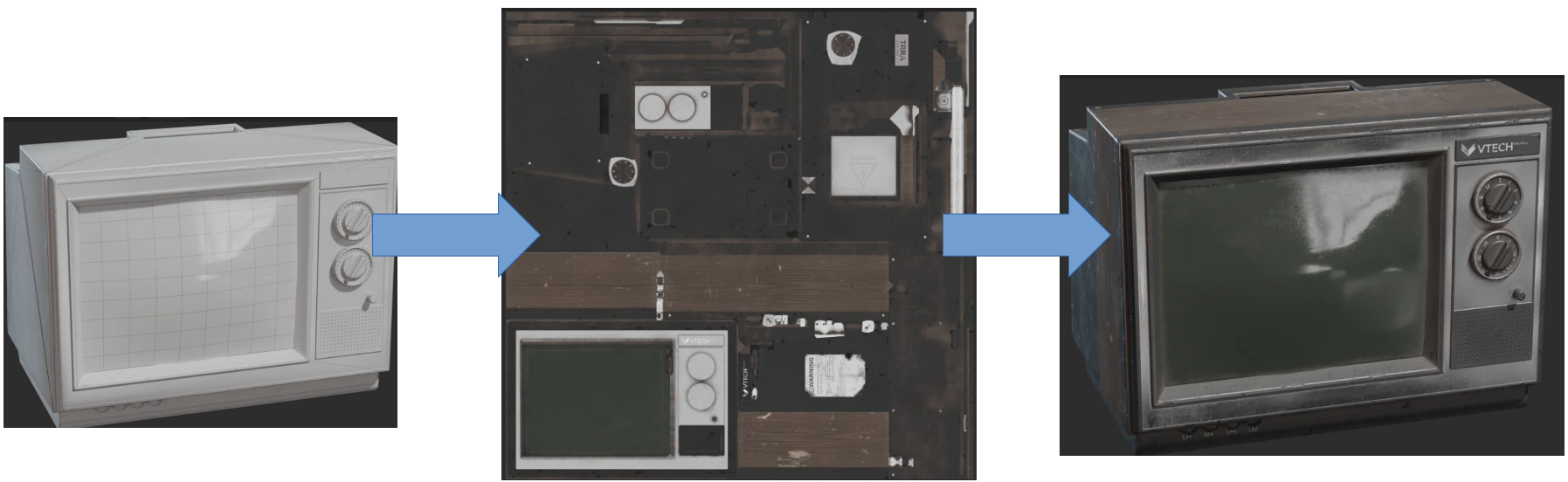


How to use texture for **material modeling** → variation of scattering function parameters and small scale geometry

Image textures

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 x 2^n texels. Thus called power-of-two textures (POT)*

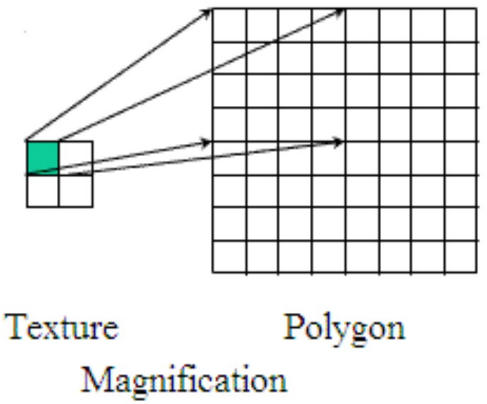


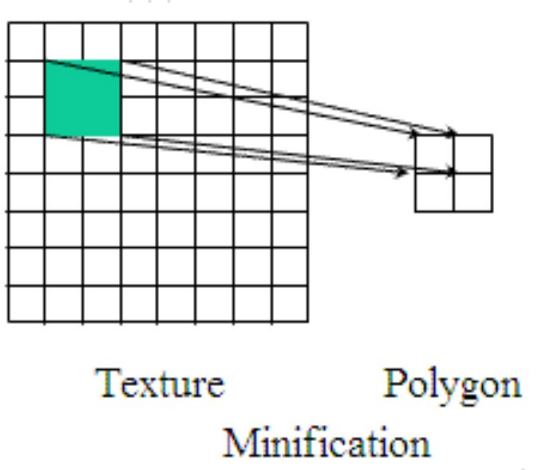
https://polyhaven.com/a/Television_01

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

Magnification and minification

- If more than one pixel cover a texel: **magnification**.
- If more then on texels cover pixel: **minification**

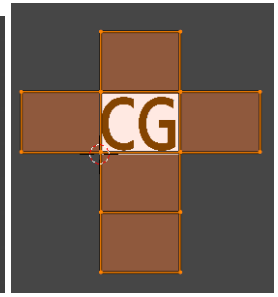


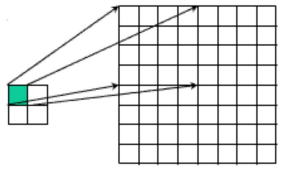


https://jcsites.juniata.edu/faculty/rhodes/graphics/texturemap.htm

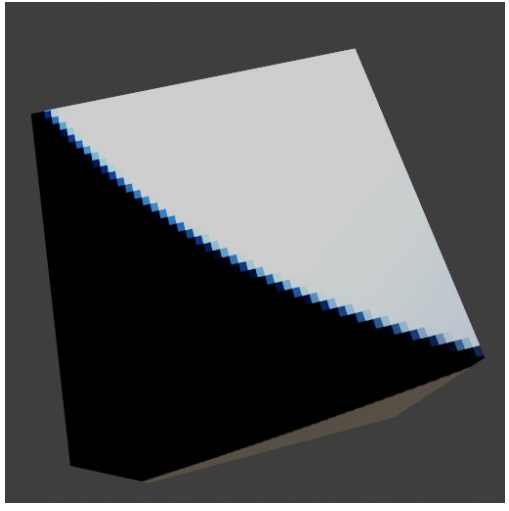
Texture magnification

- Texture system must magnify the texture. Magnification types:

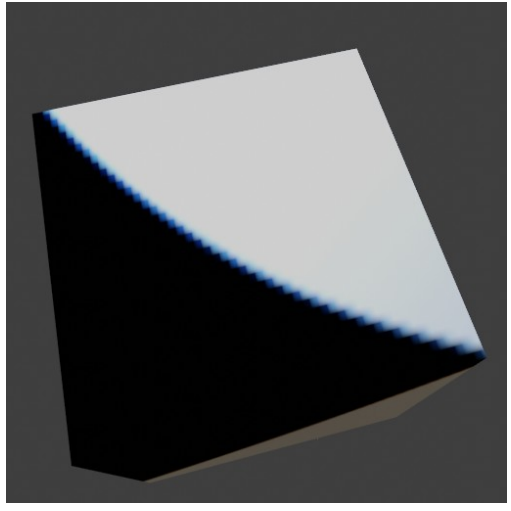




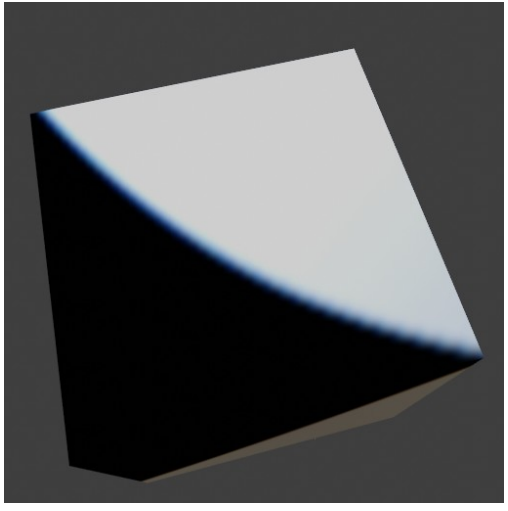
Texture Polygon
Magnification



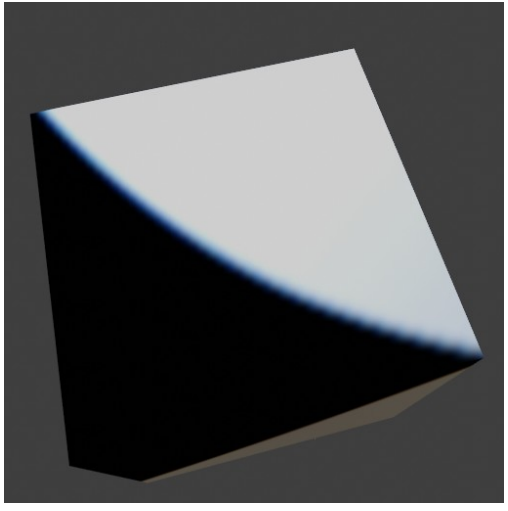
Nearest neighbor (closest).
Problem: pixelization



Linear



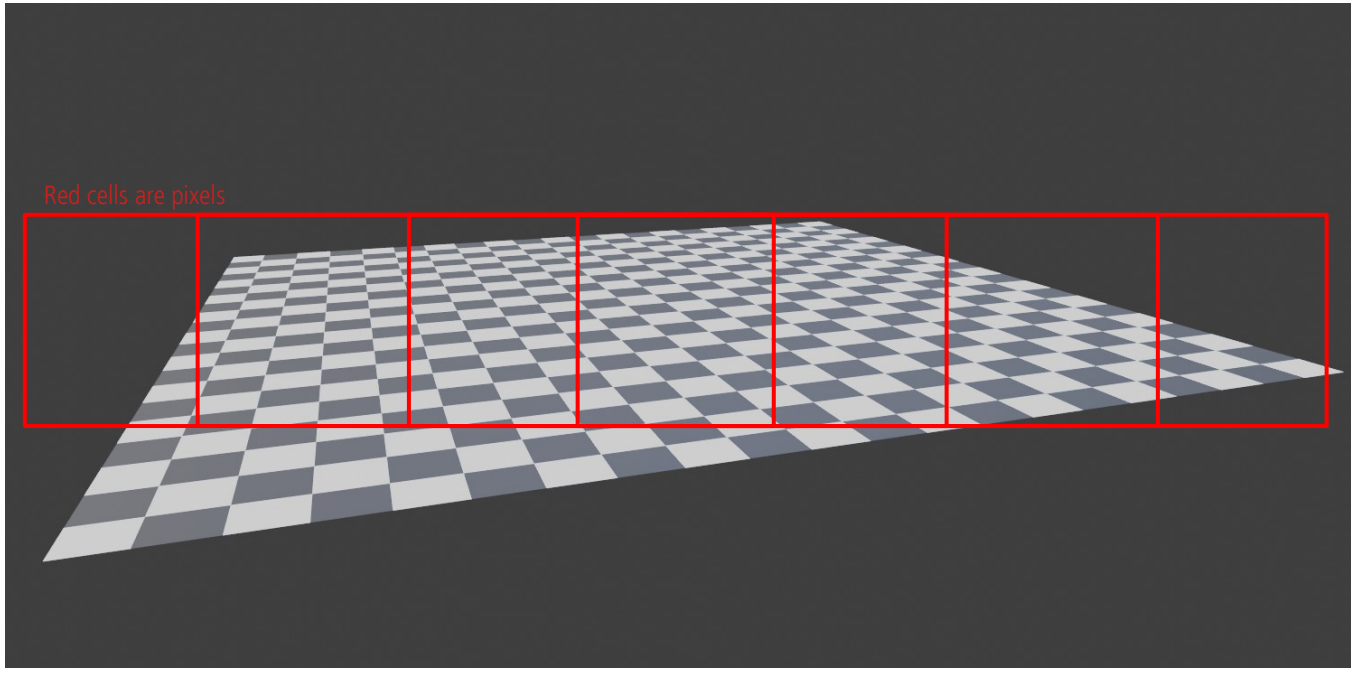
Cubic filtering

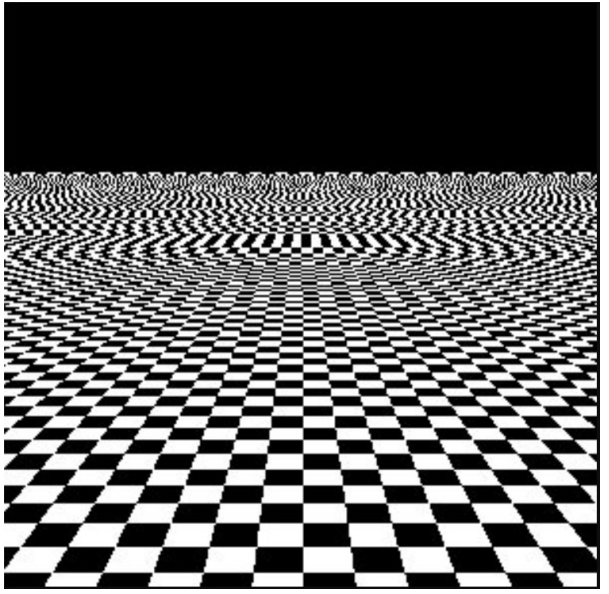


Bilinear interpolation. Problem: blur

Texture minification

- If several texels cover pixel cell, **minification** takes place, 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

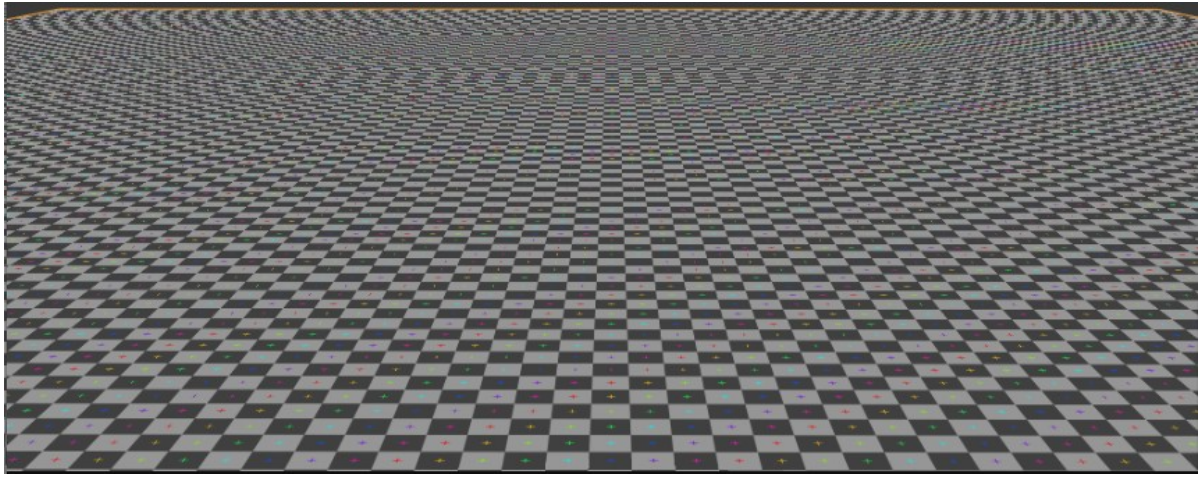




https://textureingraphics.wordpress.com/what-is-texture-mapping/anti-aliasing-problem-and-mipmapping/

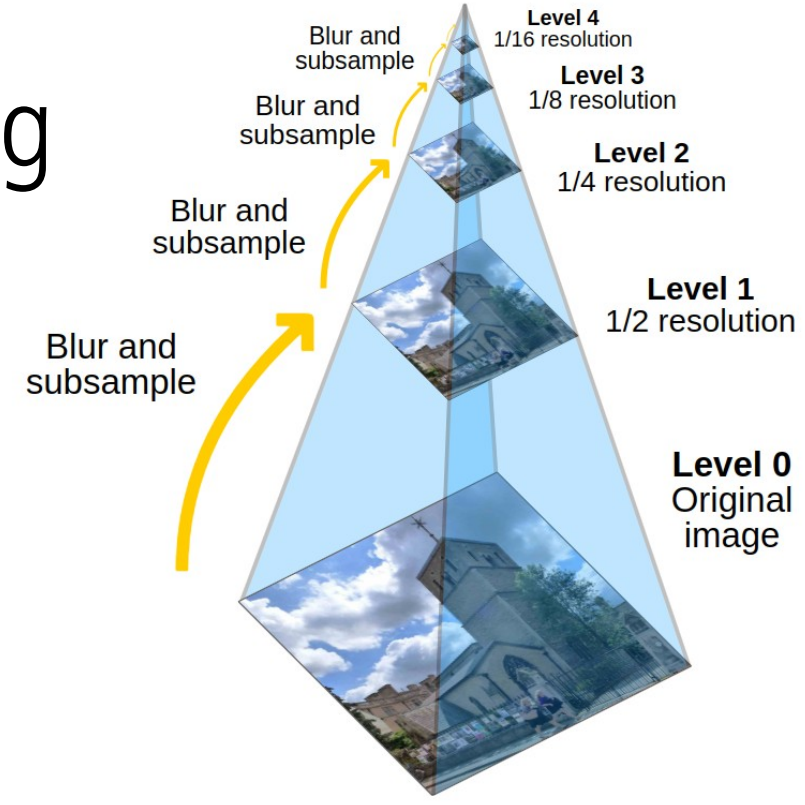
Aliasing

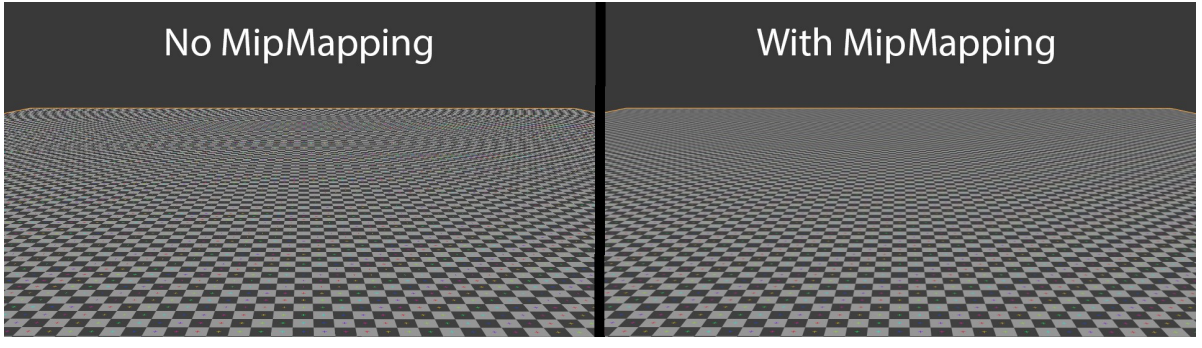
- Frequency of a texture depends how closely are spaced texels on the screen.
- **Nyquist limit** – texture frequency must be not larger than half the sample frequency.
  - To properly display a texture on a screen, we need at least one texel per pixel
- **Anti-aliasing techniques:**
  - Texture frequency must decrease: **mip-mapping** - most popular anti-aliasing method
  - Pixel sampling frequency must increase: **multiple samples per pixel**



Anti-aliasing: mipmapping

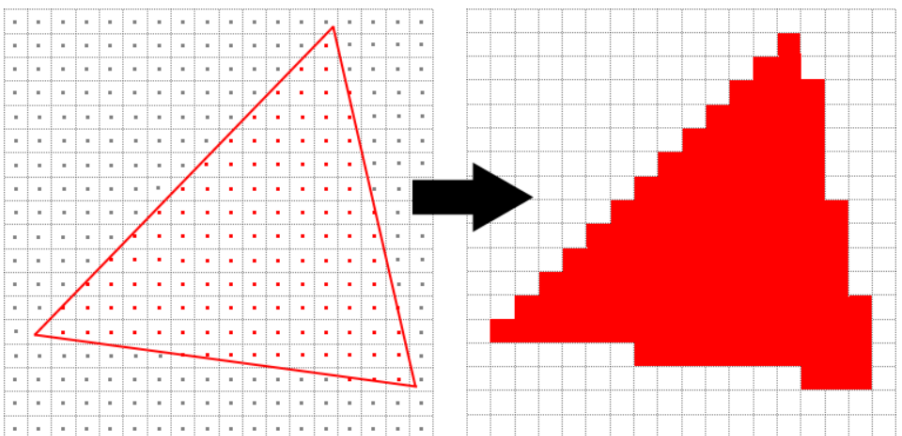
- MIP – multum in parvo (latin) – many things in small place – minimization filter
  - Original image is filtered down repeatedly into smaller images.
  - Before rendering, original image is augmented with sets of smaller versions of this image
  - Using mipmaps to achieve 1:1 pixel to texel ratio to achieve Nyquist rate.
- Image pyramid, Mipmap chain:
  - Original texture is downsampled by quarter of original area. New texel is calculated as average of four neighbor 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: averaging new texel value from 2x2 of texels it is recommended to use Gaussian, Lanczos or Kaiser filter
  - Images must be in linear colorspace



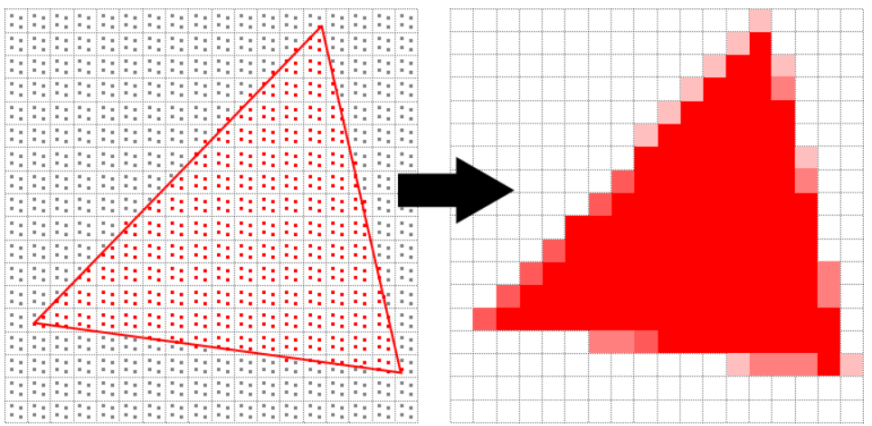


Antialiasing: sampling

- With image minification we introduced the problem of aliasing
- Aliasing appears always when there are a lot of details under a pixel footprint in the scene
- To solve this, multiple sample per pixels can be used to "catch" high frequency details.
  - This method is called Multisample anti-aliasing (MSAA)



https://vulkan-tutorial.com/Multisampling



MSAAx4

Alpha texture values

- Image textures have (R,G,B) and alpha channels.
- Alpha values are used for blending or alpha testing for transparency.

Decals/cutouts: rendering only parts of the image on object surface.

https://xoio-air.de/2017/10-free-cutout-trees-from-tony-textures/

Billboarding method is used for standalone flat polygons with image texture, always facing the camera.

https://80.lv/articles/how-to-fake-a-large-scale-forest-in-blender/

Alpha texture values

- Alpha value is extensively used when adding VFX to rendered or real images - **compositing**.

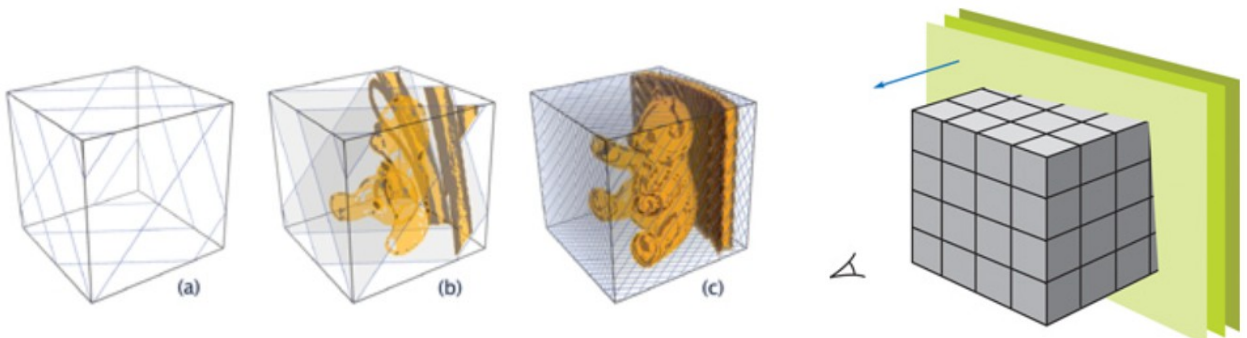
https://www.soundstripe.com/blogs/the-best-techniques-tools-for-compositing

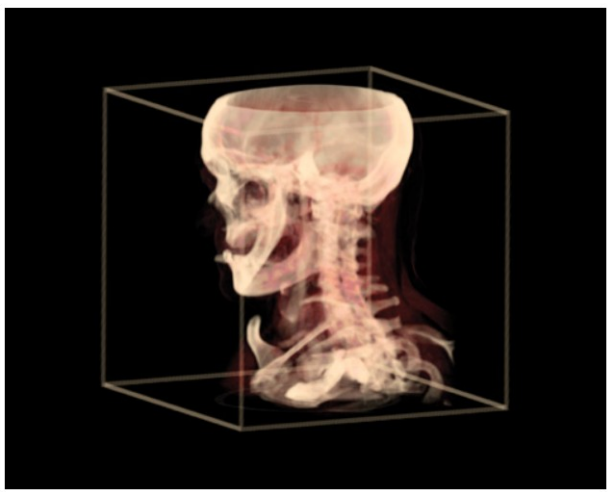
Sources of image textures

- Texture image can be created:
  - By drawing on canvas or 3D object directly
  - Taking photographs
  - Performing measurements of real surfaces
  - Creating from existing geometry or surfaces (baking)
- Library of textures:
  - PolyHaven: https://polyhaven.com/textures
  - Substance library: https://substance3d.adobe.com/assets/allassets?assetType=substanceMaterial&assetType=substanceAtlas&assetType=substanceDecal

Volume textures

- Extension of 2D image is 3D image accessed with **(u,v,w) coordinates**
- Object locations can be used as texture coordinates.
  - Complex problem of finding good 2D parameterize for 3D mesh is not needed
- Volume texture can represent volumetric structure of material such as wood or marble.
- Example: medical imaging data
  - Three dimensional grid is visualized as slices by moving polygon through it or using ray-casting.
- Problem: volumetric image textures are very space demanding



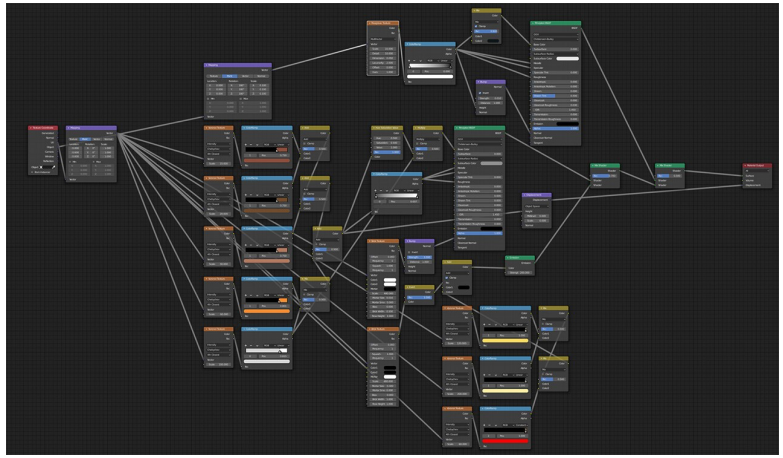


https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch40.html
https://developer.nvidia.com/gpugems/gpugems/part-vi-beyond-triangles/chapter-39-volume-rendering-techniques

Procedural textures

Procedural textures

Umbrella term for a number of techniques in computer graphics to create 3D models and textures from sets of rules - code segments or algorithms that specify some characteristic of a computer-generated model or effect[1]



1. Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (1994). Texturing and modeling: a procedural approach. Academic Press Professional, Inc.

https://www.reddit.com/r/blender/comments/khkz98/finished_my_first_nodevember_link_in_comments/?utm_source=share&utm_medium=web2x&context=3

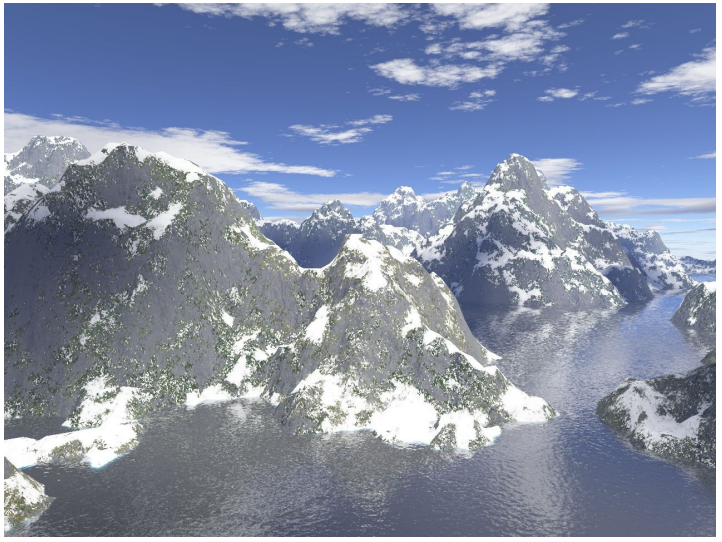
Procedural textures

- **Texture values**, instead of image lookup, are **evaluated from a function**
- Procedural texture defines values for every point in 3D space or 3D surface point.
  - Parametrization of surface is therefore not needed

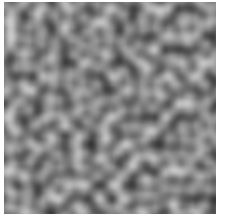
Often based on **noise function**.

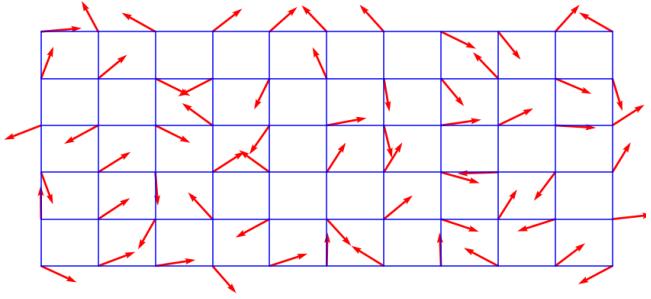
Often used for natural phenomena such as seas, clouds and mountains, etc.

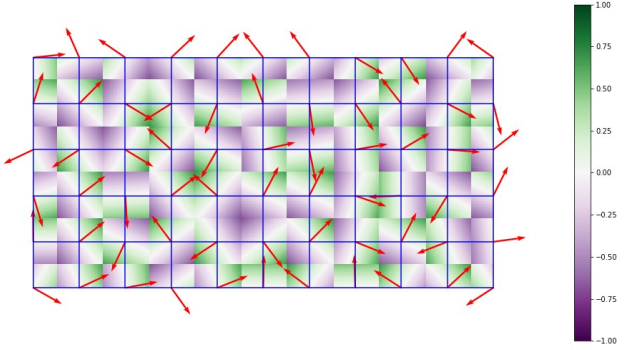
Procedural textures: Perlin noise

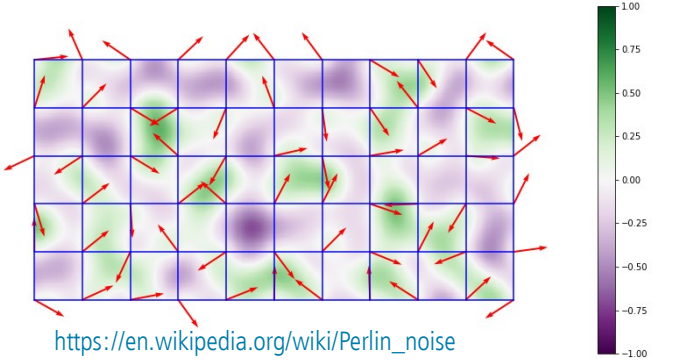


- Property: **local changes are gradual, while global changes can be large**
- Most popular noise function upon which wide range of methods is built
- Type of **gradient noise**.
  - N dimensional grid where each grid intersection has associated n dimensional unit length gradient vector
  - To calculate point inside grid:
    - (1) find cell inside which point lies,
    - (2) calculate dot product of vector from this point to cell corner and gradient vector,
    - (3) interpolate dot products





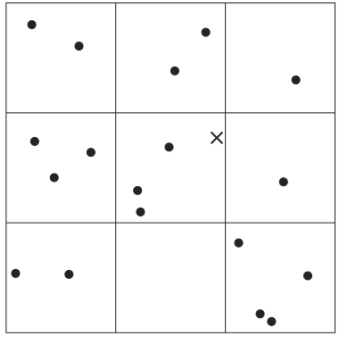


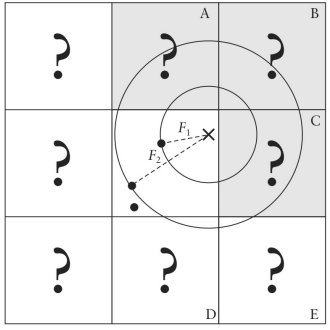


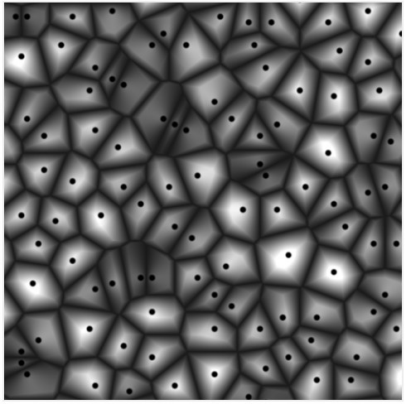
https://en.wikipedia.org/wiki/Perlin_noise

Procedural textures: Cellular texture

- Steven Worley introduced **Voroni-like textures** that can be used for many natural textures: **cells, flagstones, lizard skin, etc.**
  - Another fundamental procedural texture upon which more complex procedural textures are built
- This method is based on **measuring distance from feature points**
  - Space is diced into cubes. Each cube contain random feature points.
  - For current point, find cube inside which lies and check distance to feature points in this and neighboring cubes.
  - First distance is used as value. More combinations with distances are possible.







https://thebookofshaders.com/12/

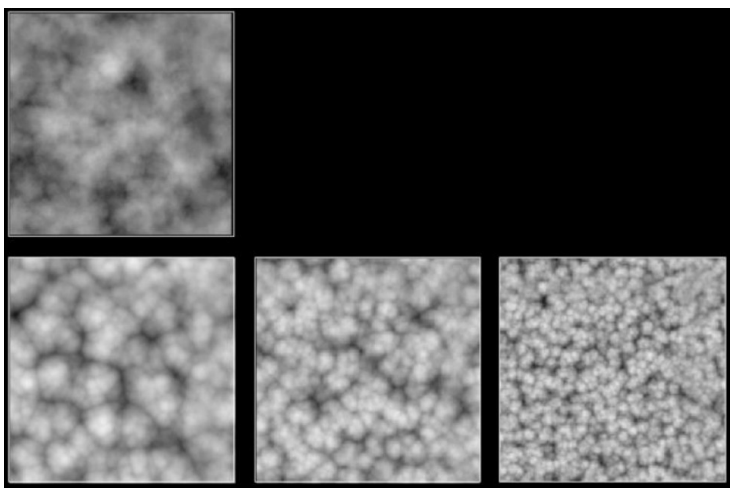
1. Musgrave, F. K., Peachey, D., Perlin, K., & Worley, S. (1994). Texturing and modeling: a procedural approach. Academic Press Professional, Inc.

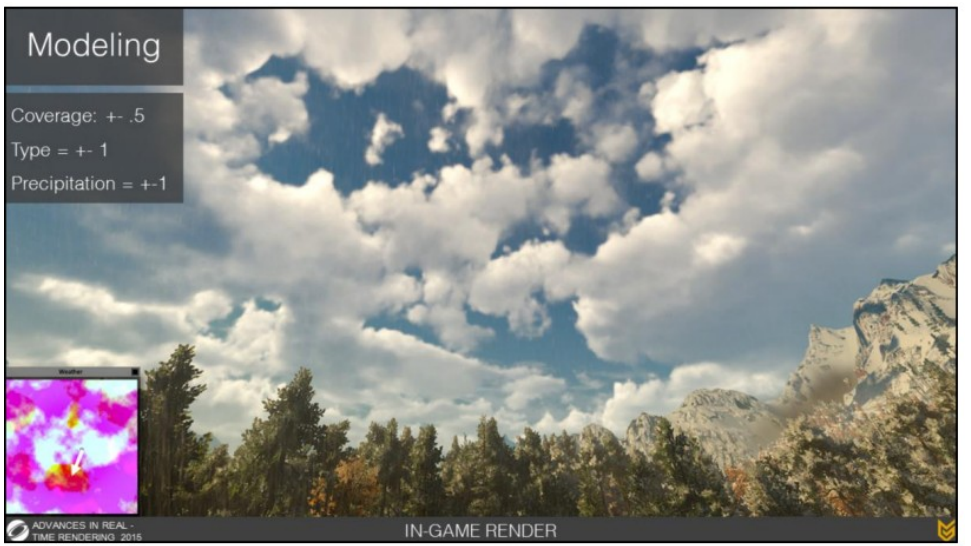
Noise in production

- More complex types of noise are built using Perlin and Worley noise:
  - **Fractal noise**
  - **Simplex noise**
- **Combining Perlin and Worley noise** is often used for more richer details.
- Volume textures are attractive application for procedural texturing since they can be synthesized on the fly

Perlin-Worley

3 layers of Worley

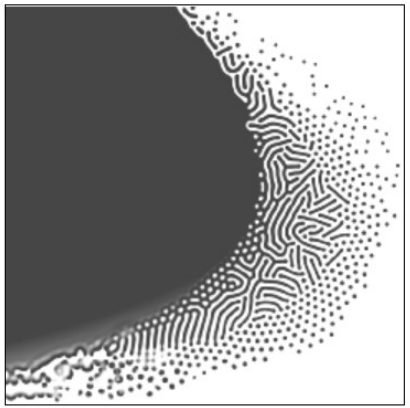


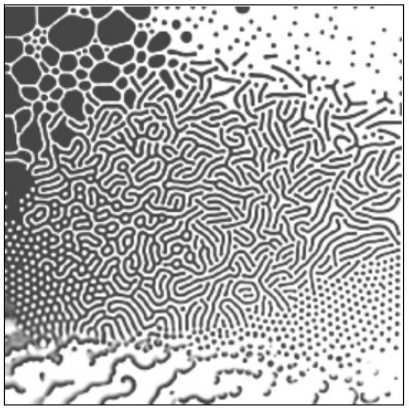


https://www.guerrilla-games.com/read/the-real-time-volumetric-cloudscapes-of-horizon-zero-dawn

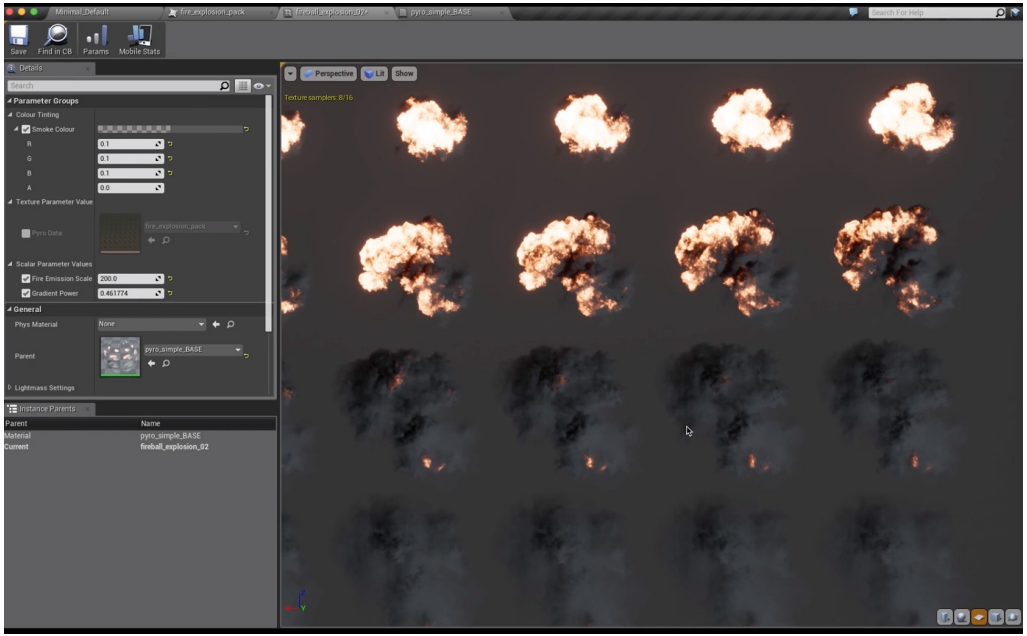
Procedural texture: physical simulation

- Perlin and Cellular textures are mathematical models and phenomenological models for representing natural phenomena
- Phyiscally based simulation can also be used to create textures
  - Reaction-diffusion model
  - Fluid simulation to texture





https://www.karlsims.com/rd.html

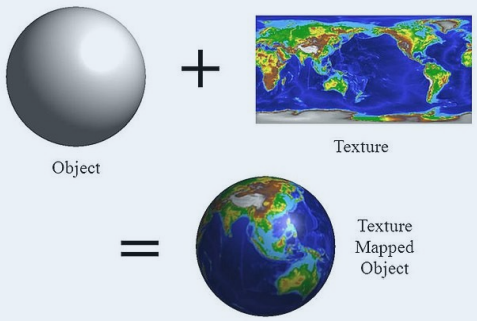


https://www.sidefx.com/ja/tutorials/channel-packing-pyro-data-using-the-texture-sheets-rop/

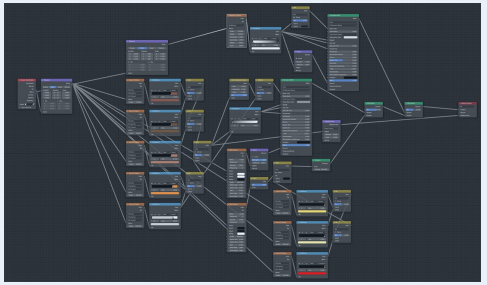
Procedural textures: Antialiasing

- Procedural texturing is much more popular in offline applications while in real-time rendering image textures are far more common
  - GPUs are extremely efficient with image textures for which antialiasing is also easier.
- Antialiasing is hard for procedural textures
- 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

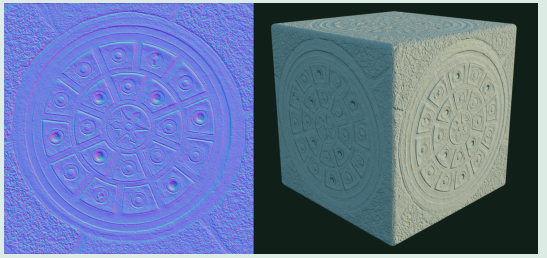
Learning objectives



How to apply textures on objects → **texturing pipeline**



Texturing approaches → **image and procedural**

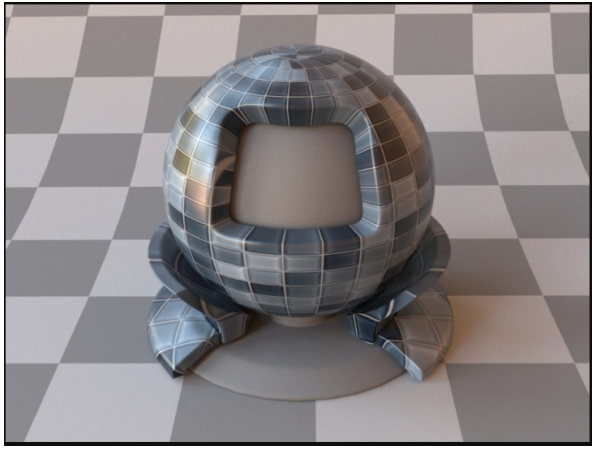


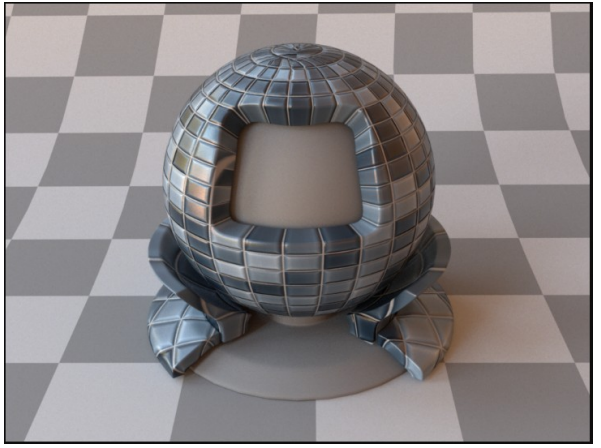
How to use texture for **material modeling** → variation of scattering function parameters and small scale geometry

Textures and material modeling

Usage of texture values

- Texture is used for **varying material parameters** over surface:
  - **Scattering function parameters**: color, reflectance, roughness, etc.
  - **Small scale surface geometry:** bumps, irregularities, etc.

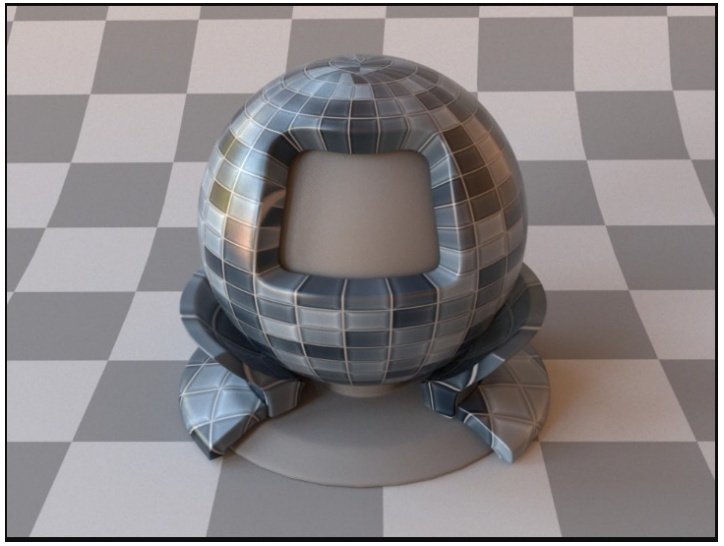


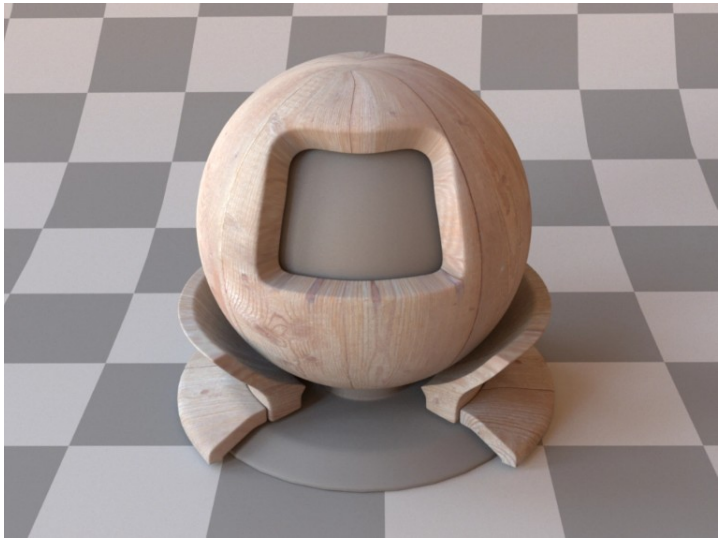


https://mitsuba2.readthedocs.io/en/latest/generated/plugins.html#textures

Scattering function parameters

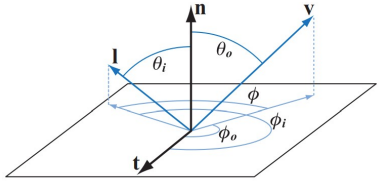
- Color, reflectance, roughness, etc.

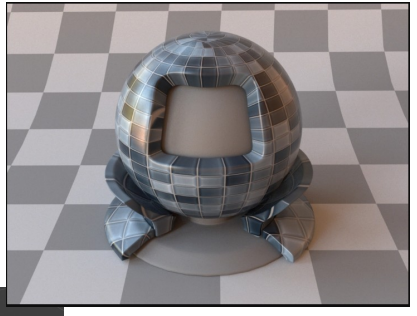


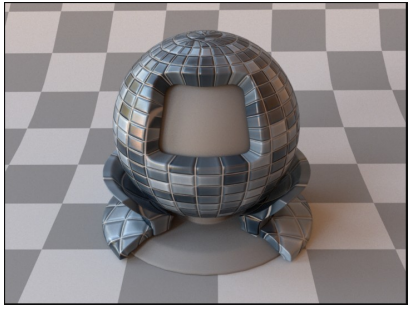


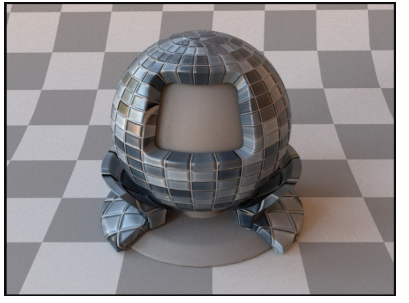
Small scale surface geometry

- **Small scale surface geometry** is represented using texture which values are used for perturbing surface or surface normal
  - Perturbing surface or surface normal causes perturbing of BSDF basis and variation in light reflection

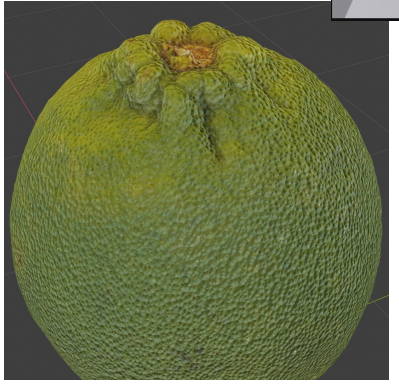








https://mitsuba2.readthedocs.io/en/latest/generated/plugins.html#textures



https://learnopengl.com/Advanced-Lighting/Normal-Mapping

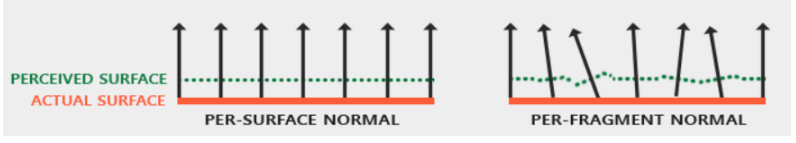
Small-scale geometry: scale of modeling

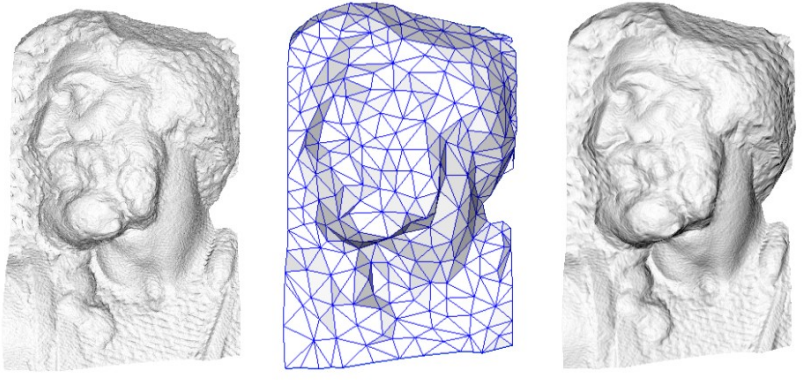
- Details on object can be classified:
  - **Macro-features**: cover many pixels. Represented by **model shape** (e.g., polygonal mesh). For character modeling, limbs are modeled at macroscale
  - **Meso-features**: cover few pixels across. Everything between two scales. It represents detail that is 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.
  - **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

https://www.pbr-book.org/3ed-2018/Materials/Bump_Mapping

Small-scale geometry: models

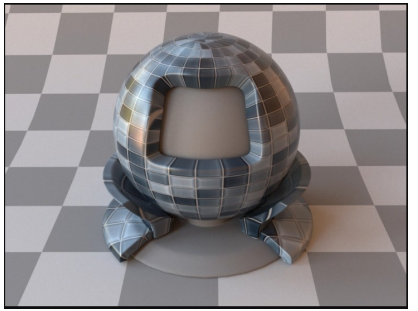
- Texture is employed in shading step during rendering to give more richer 3D appearance than just color
- Methods which do not introduce additional geometrical information. Only approximation to achieve desired surface details:
  - **Bump, normal mapping**
    - Large family of small-scale detail representations
  - **Parallax mapping**
- Methods which require additional geometrical information. Give more realistic surface appearance but are computationally expensive:
  - **Displacement mapping**

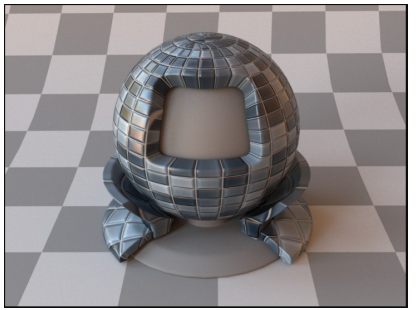




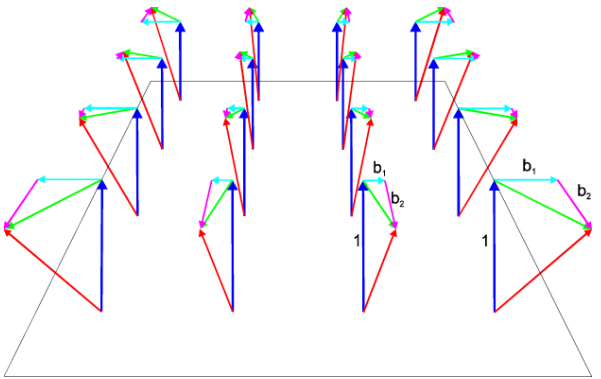
https://helpx.adobe.com/substance-3d-painter/using/baking.html

Bump mapping

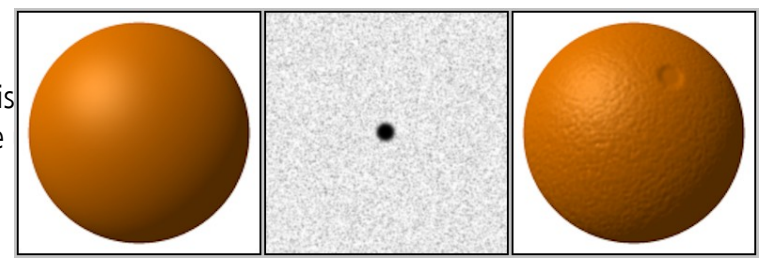




- Main idea (**Blinn method**):
  - Surface will appear to have small scale details is during shading process, if surface normal is slightly perturbed.
- Information for perturbing surface normal can be encoded in image texture as color or procedural texture.
- Original bump mapping – **offset vector 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 an 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
- 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 taking differences between neighboring columns are rows to obtain u and v (Sobel filter an be



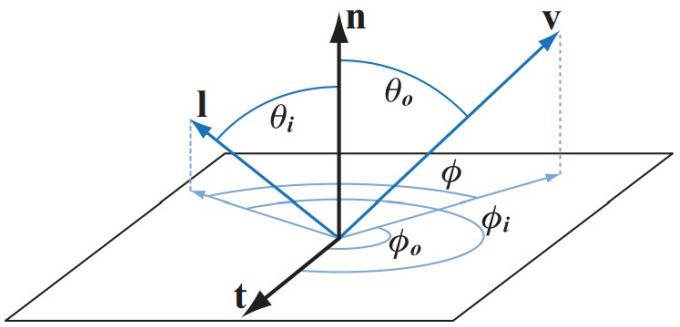
https://www.researchgate.net/publication/228890410_Antialiasing_of_Bump_Maps

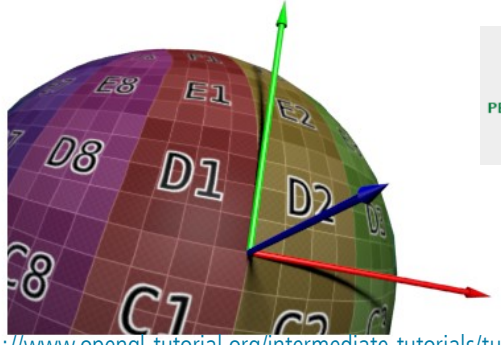


https://en.wikipedia.org/wiki/Bump_mapping

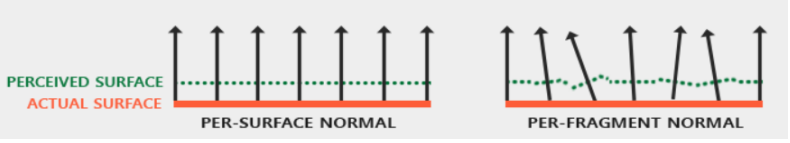
Bump mapping: frame of reference

- Perturbing surface normal must be done with respect to some frame of reference: **tangent frame basis**
- 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





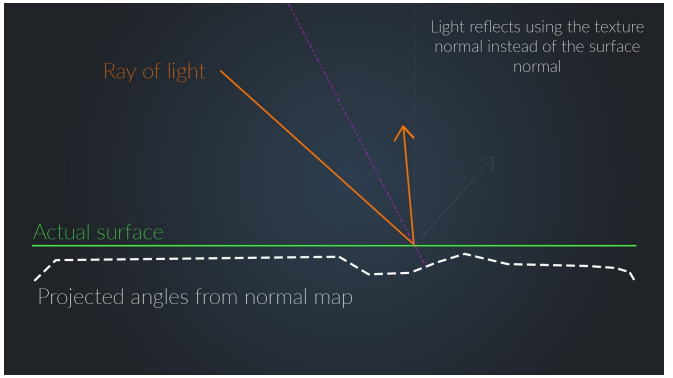
http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/



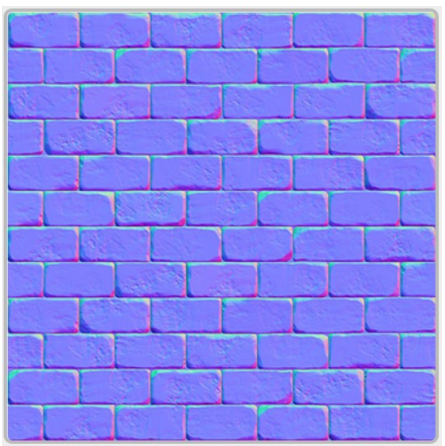
https://learnopengl.com/Advanced-Lighting/Normal-Mapping

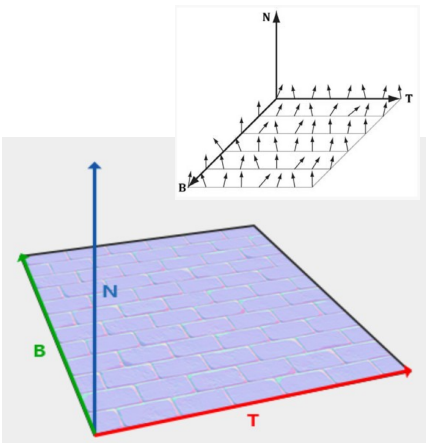
Normal mapping

- 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 (R,G,B) image texture
  - For PNG images, 8bit unsigned int is used. Therefore, [-1,1] → [0,255]
- Normal map are often defined in **tangent space***
  - Color of normal map is determined with x,y and z directions of encoded vectors

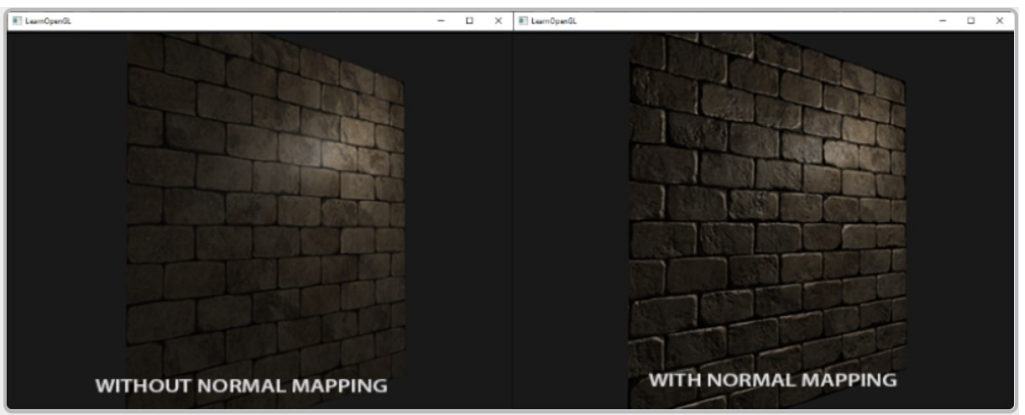


https://cgcookie.com/posts/normal-vs-displacement-mapping-why-games-use-normals





http://ycpcs.github.io/cs470-fall2014/labs/lab12-2.html

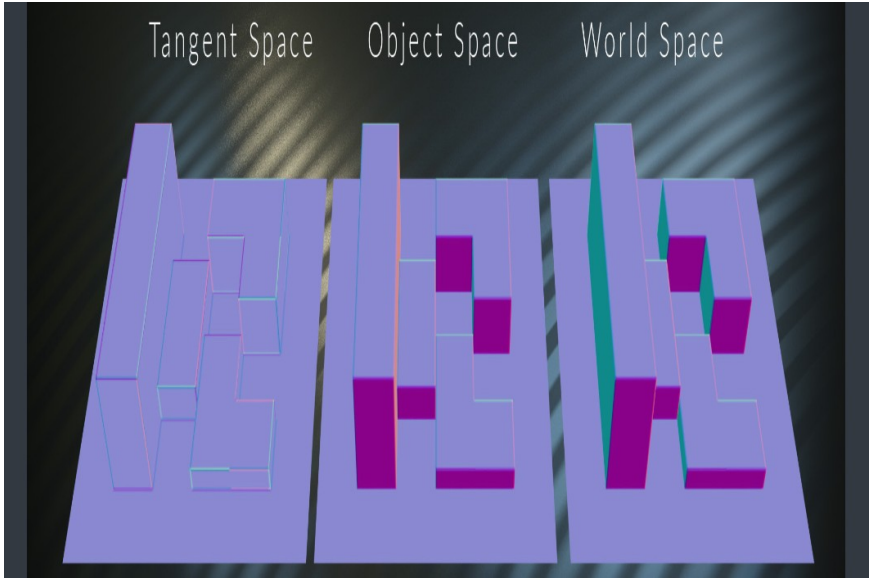


https://learnopengl.com/Advanced-Lighting/Normal-Mapping

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

Normal map types

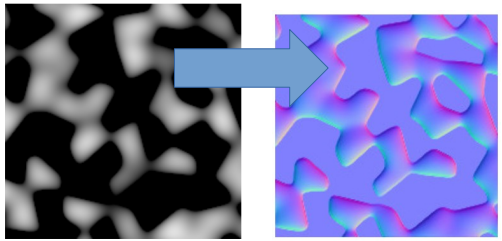
- **Tangent space:** based on tangent direction of face. Normal map can be then maximally reused for different objects, transformations and deformations.
  - Blue: normal direction
  - Red: left/right tangent direction
  - Green: up/down tangent direction
- **Object space**: based on entire object rather than each face individually
  - Faster to compute, tied to to particular model which can not be deformed
- **World space**: based on global coordinates
  - Object must not be rotated
  - Useful for large, static environment objects



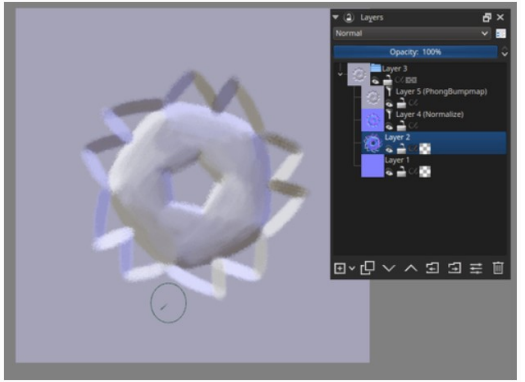
https://cgcookie.com/posts/normal-vs-displacement-mapping-why-games-use-normals

Creating bump/normal maps

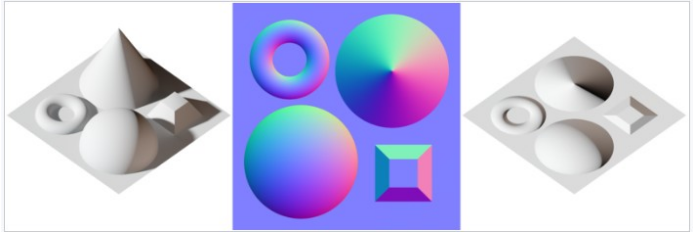
- **Baking** normal maps from high quality 3D models (often done for real-time graphics applications)
- **Image editing/painting software**: converting photo textures into normal maps, procedural normal map images, hand-painting normal maps
- **Scanning** heights from real world and converting them to normal map.
- Normal map can be **derived from height map.**



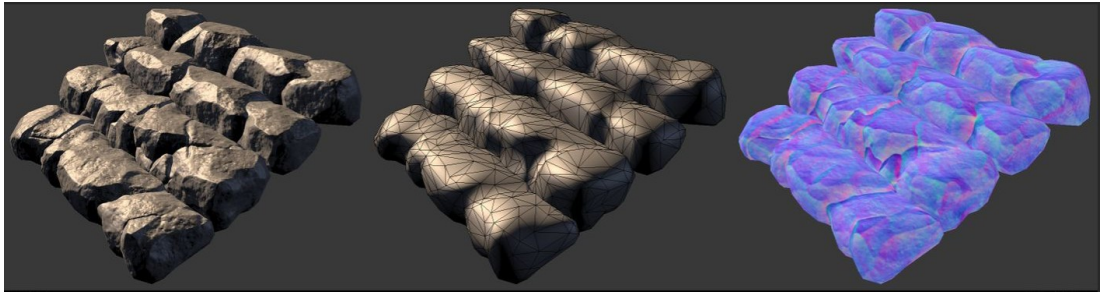
https://docs.gimp.org/2.10/en/gimp-filter-normal-map.html



Hand-painting normal maps.
https://docs.krita.org/en/reference_manual/brushes/brush_engines/tangen_normal_brush_engine.html



Normal map baked from geometry
https://en.wikipedia.org/wiki/Normal_mapping

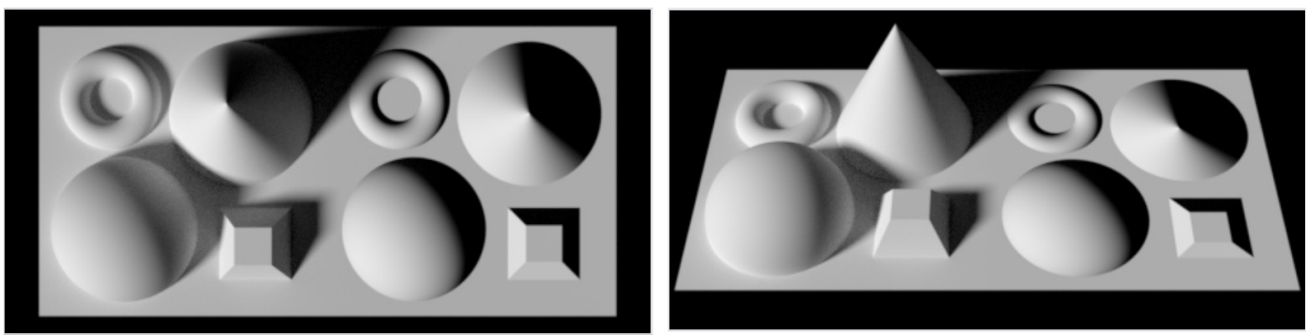


High quality geometry from which normal map is baked and applied on low quality geometry to achieve computationally cheap high quality appearance.

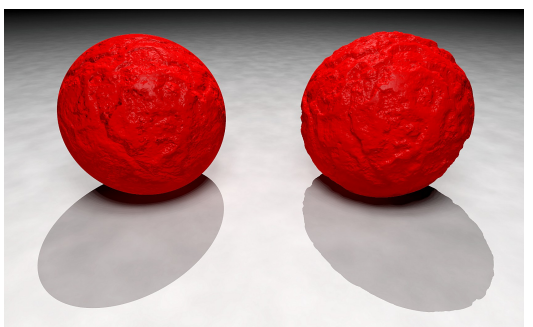
http://wiki.polycount.com/wiki/Normal_map

Bump and Normal mapping: practical notes

- Relationship between normal and shaded color is not linear*. Therefore, filtering of normal maps for **anti-aliasing** requires additional work.
- 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.



Comparison of real geometry and normal mapped geometry https://en.wikipedia.org/wiki/Normal_mapping

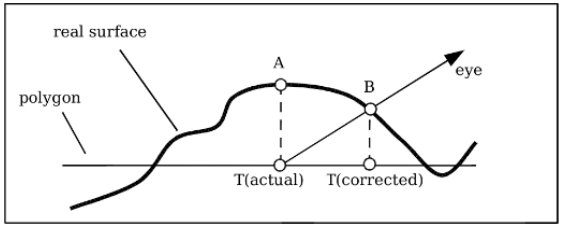


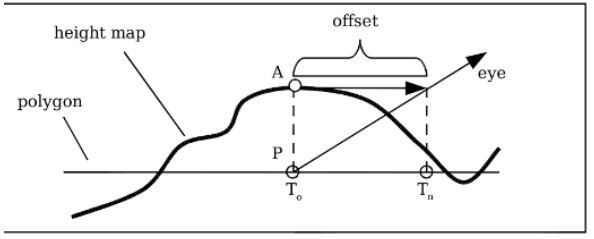
Normal mapped surfaces do not have real geometrical deviation
https://en.wikipedia.org/wiki/Bump_mapping

* 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 lamberitan 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.

Parallax mapping

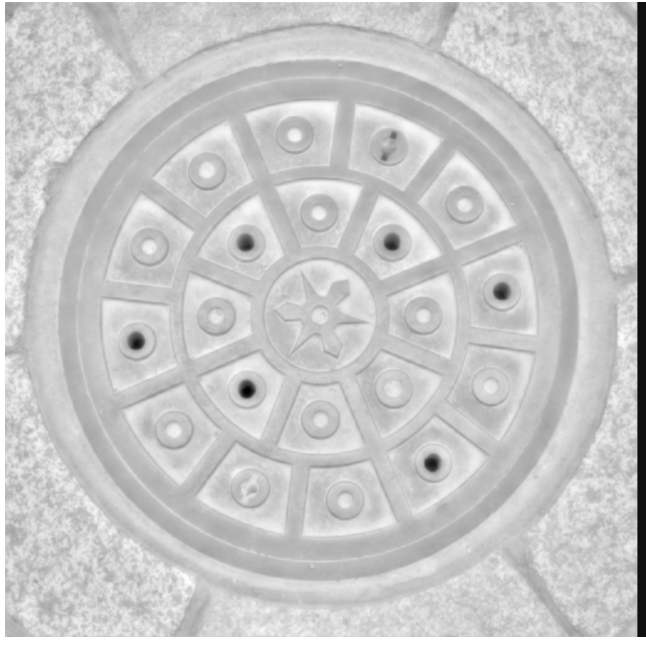


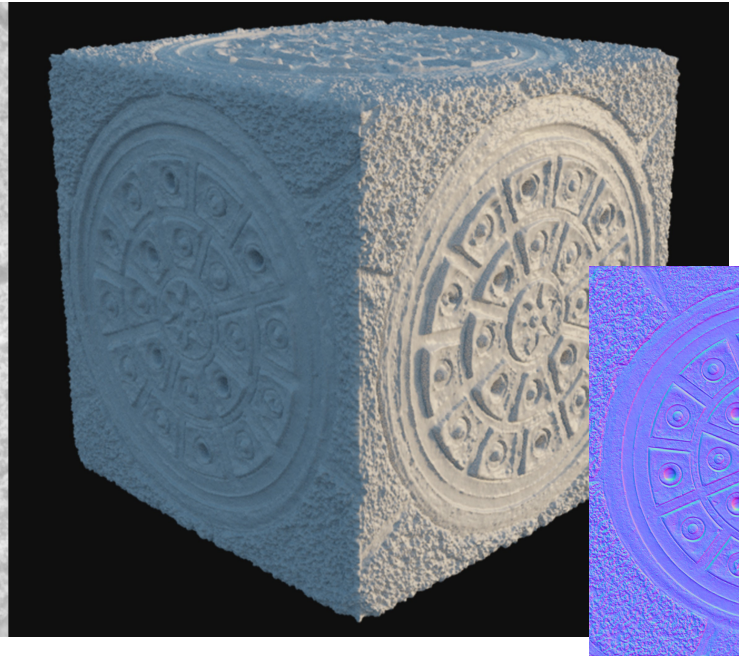


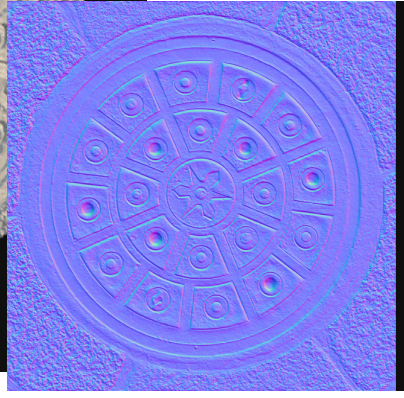
- Bump and normal mapping never shift location with the view angle, nor block each other
- Parallax: positions of objects move relative to one another as the observer moves

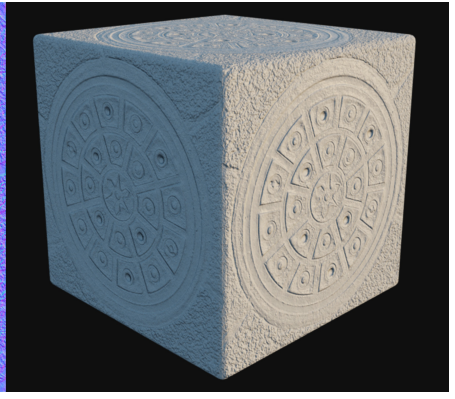
https://github.com/marcusstenbeck/tncg14-parallax-mapping/tree/master/documents

- Parallax mapping: approximation technique which is offsetting texture coordinates based on height: what should have be seen in pixel by examining the height of what was found to be visible.
  - Also known as virtual displacement mapping
- When viewing surface at a given pixel, height value is retrieved at that location and used to shift the texture coordinate to retrieve a different part of the surface. Amount of shift is based on retrieved height and angle of the eye to the surface
- Enhancement: **parallax occlusion mapping or relief mapping**: ray-marching is used along view vector until approximate intersection point on height-field texture



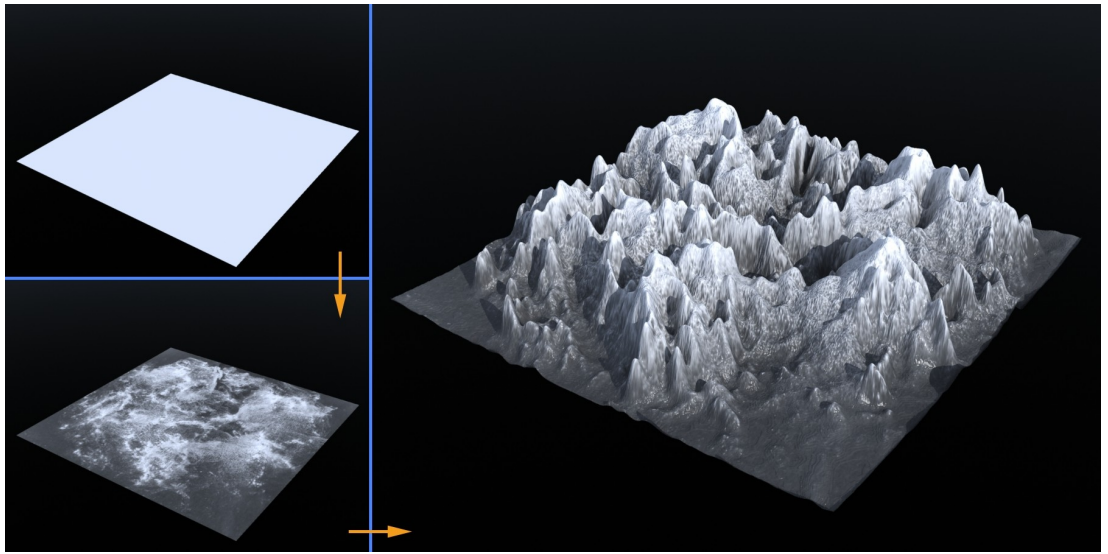




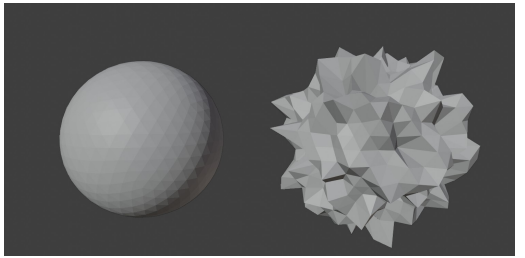


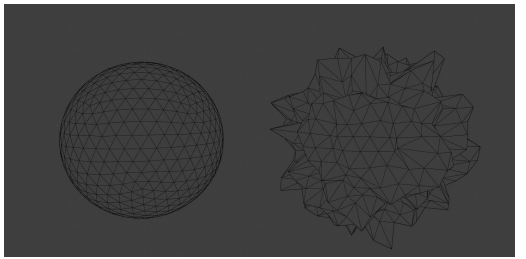
Displacement mapping

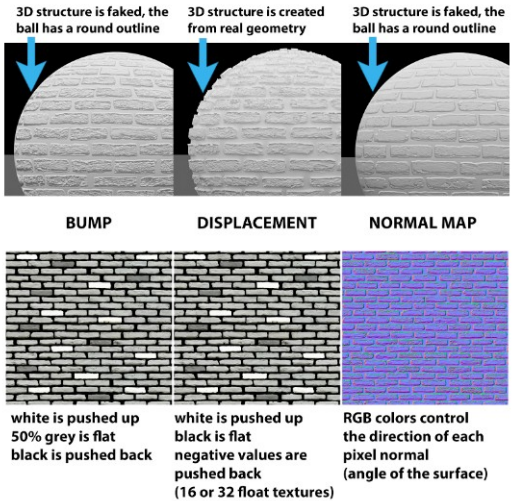
- This method changes the locations of actual geometry vertices in a mesh using height texture information.
  - Height map for displacement mapping can have negative values
- Most computationally expensive technique which requires fine geometry (a lot of vertices which can be displaced).
- Good for modeling larger geometry features such as terrains.



https://ca.wikipedia.org/wiki/Displacement_mapping







https://garagefarm.net/blog/how-displacement-maps-work-and-how-to-optimize-them-in-v-ray-part-1

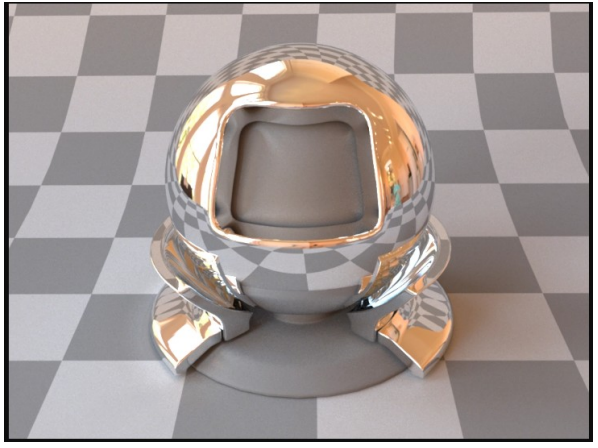
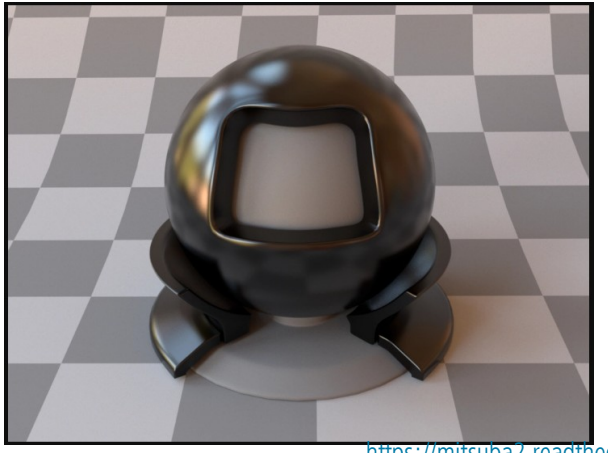
Note on small scale geometry

- Note that texture is often used to represent small scale geometry which is expensive to represent and render directly.
- However, there are rendering/representation techniques which enable modeling and rendering of high quality geometry without need for normal, bump, parallax or displacement mapping.

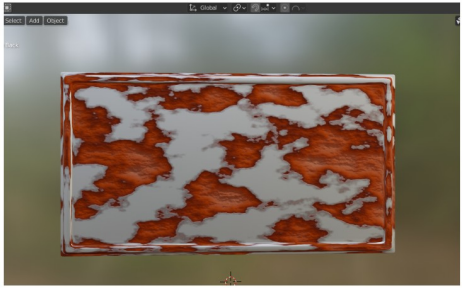
https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/

Distributing materials

- Texture is also used to **distribute different materials** over surface
  - Texture can be used as a "mask" which determines which scattering function and shading calculations will be performed where


https://mitsuba2.readthedocs.io/en/latest/generated/plugins.html#textures




https://blender.stackexchange.com/questions/231205/how-do-i-combine-these-two-textures

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

To remember

- Texture is used to vary material properties over surface
- Texturing pipeline
  - Projection, corresponding, value obtain and transform
- Image textures
  - Aliasing and mip mapping
- Procedural textures
  - Noise functions
- Material variation
  - Scattering function parameters variation
  - Small scale geometry: bump, normal, parallax and displacement mapping

Literature

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