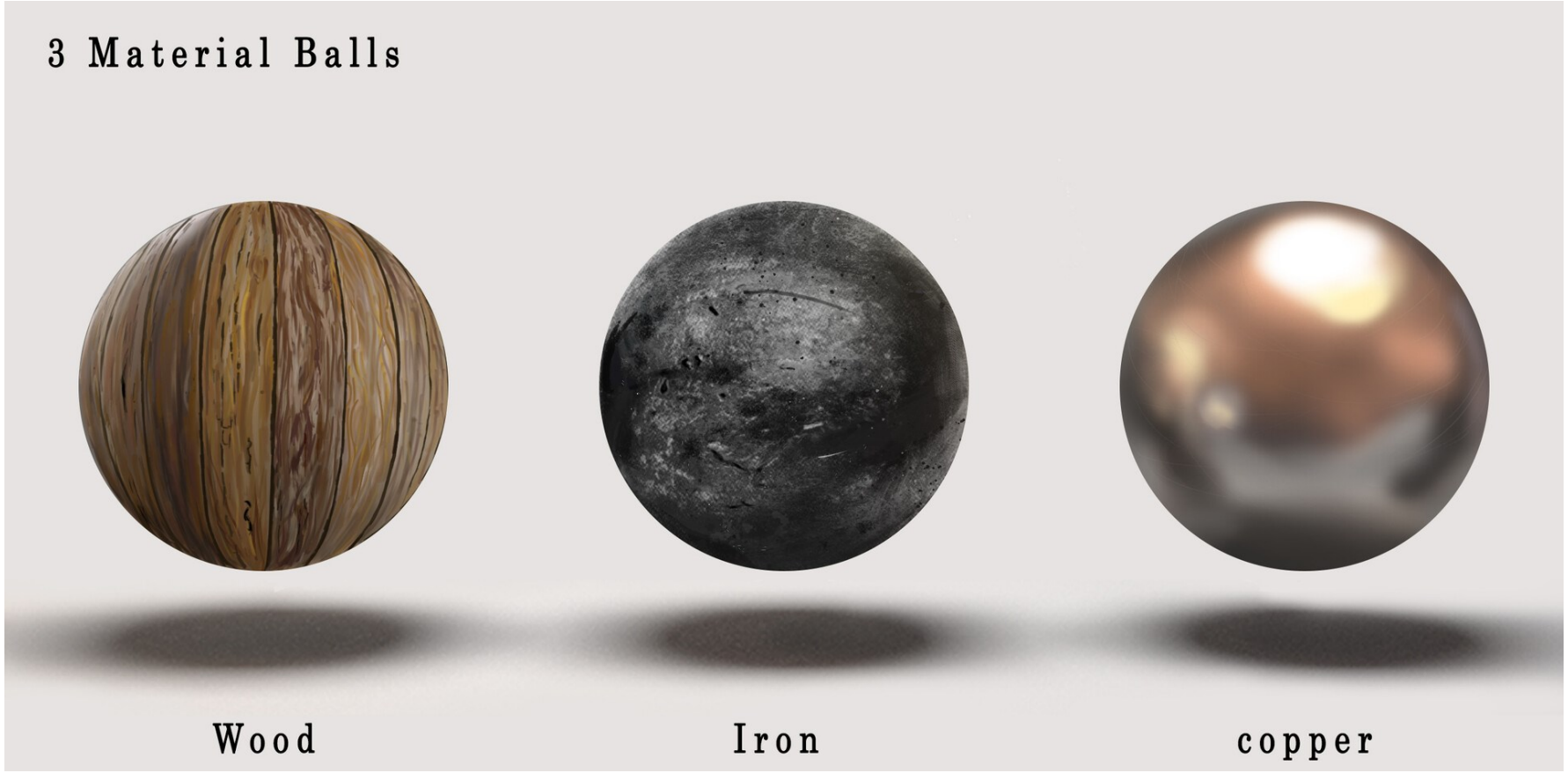
Texture

Texture Mapping

Recall: Shading

Shading in Graphics: applying material to an object, with light response

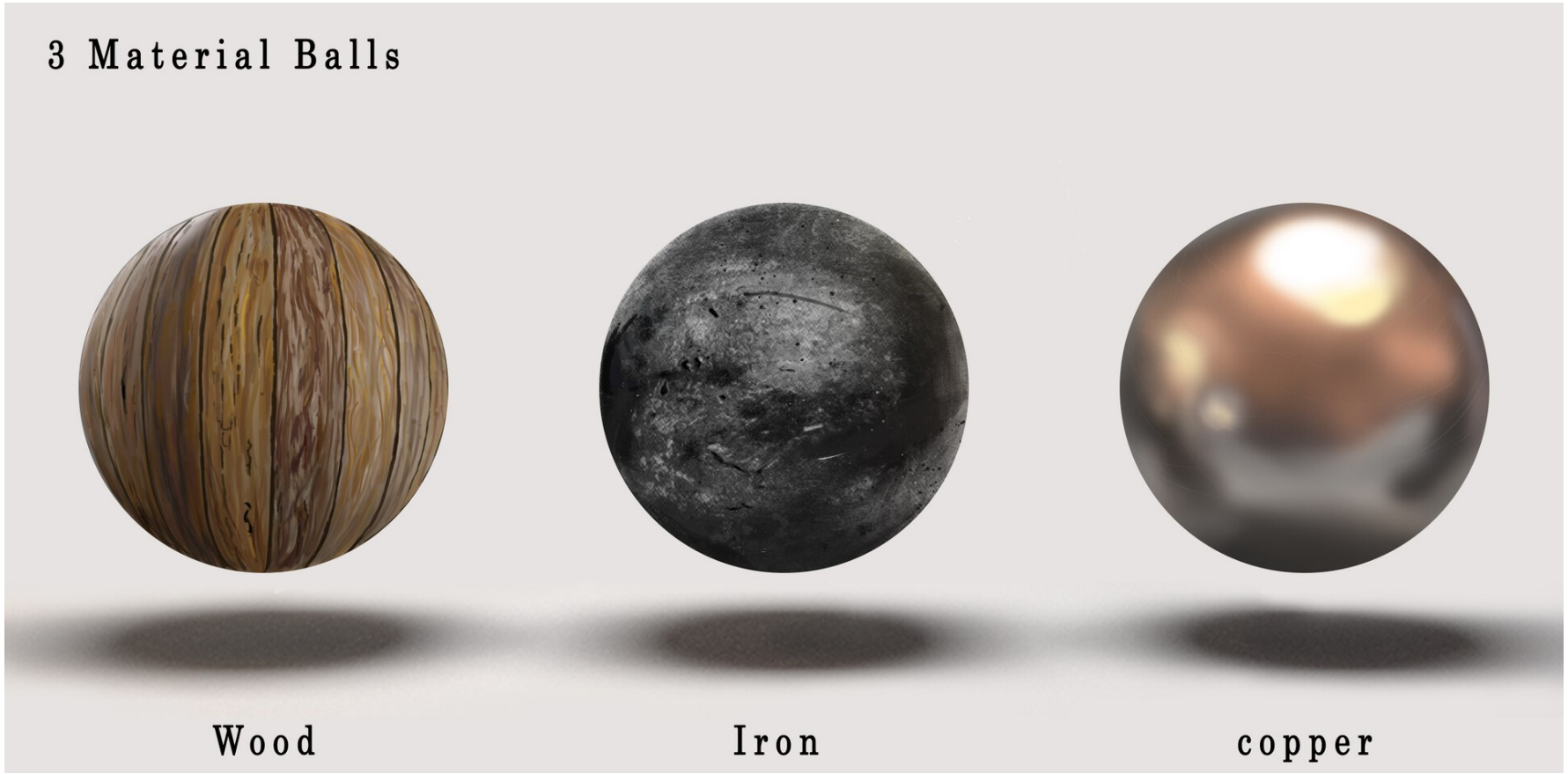
$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$

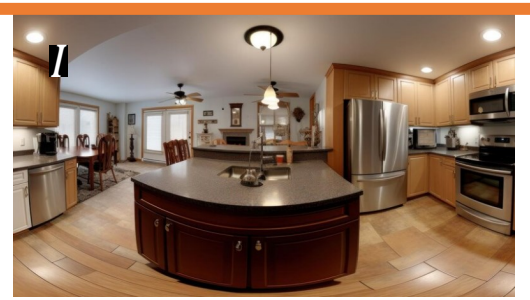


Recall: Shading

Shading in Graphics: applying material to an object, with light response

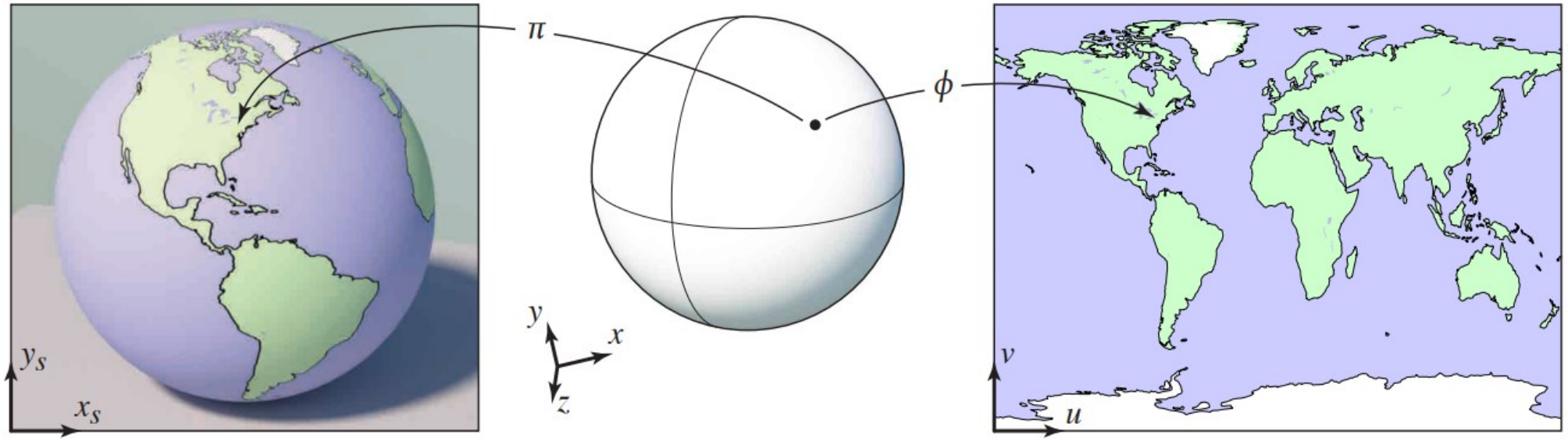
$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$





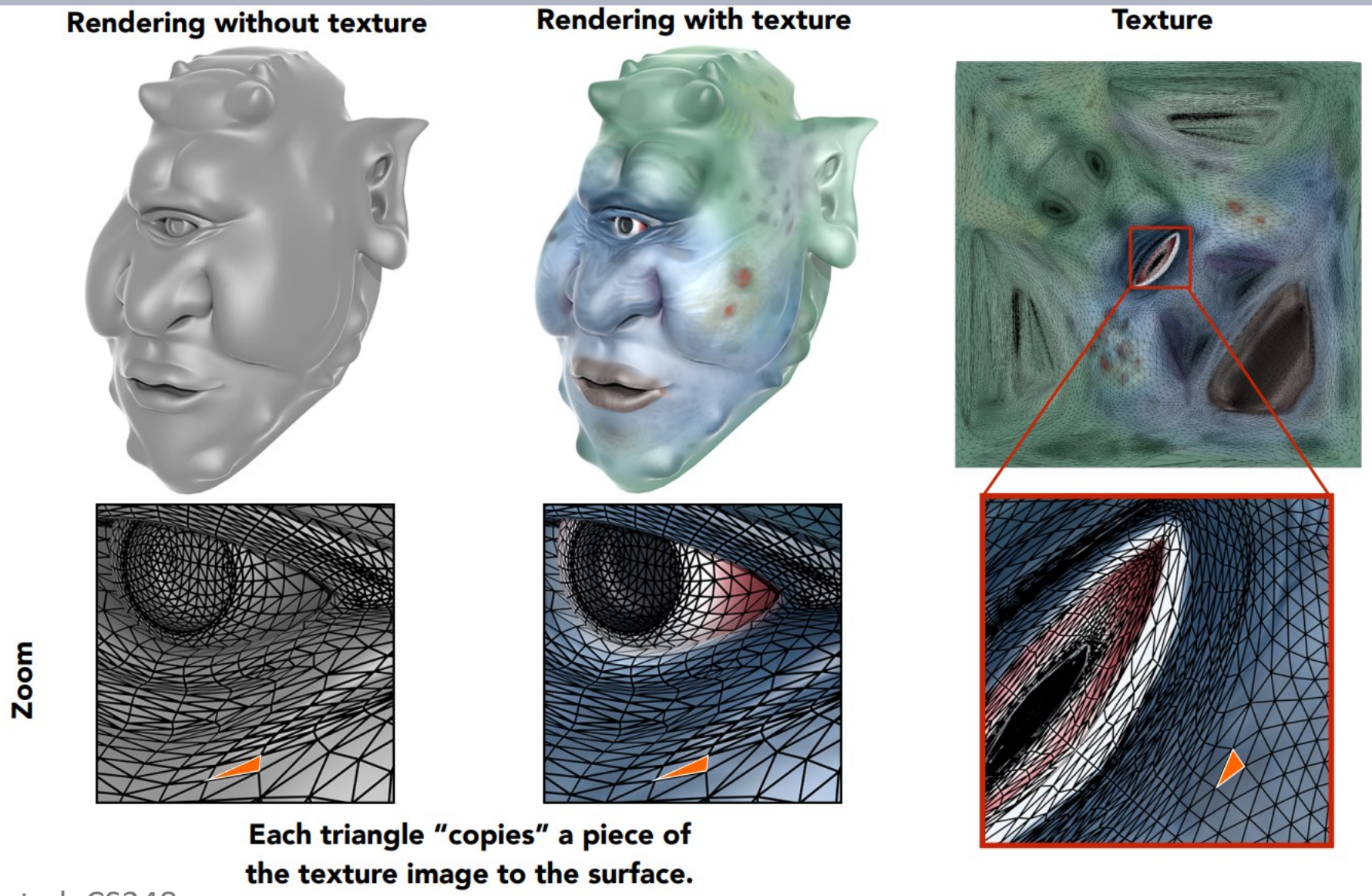
Texture Mapping

- Texture mapping: apply 2D texture onto 3D surface
- Through texture coordinate (uv coordinate) mapping



From Lingqi Yan, GAMES 101

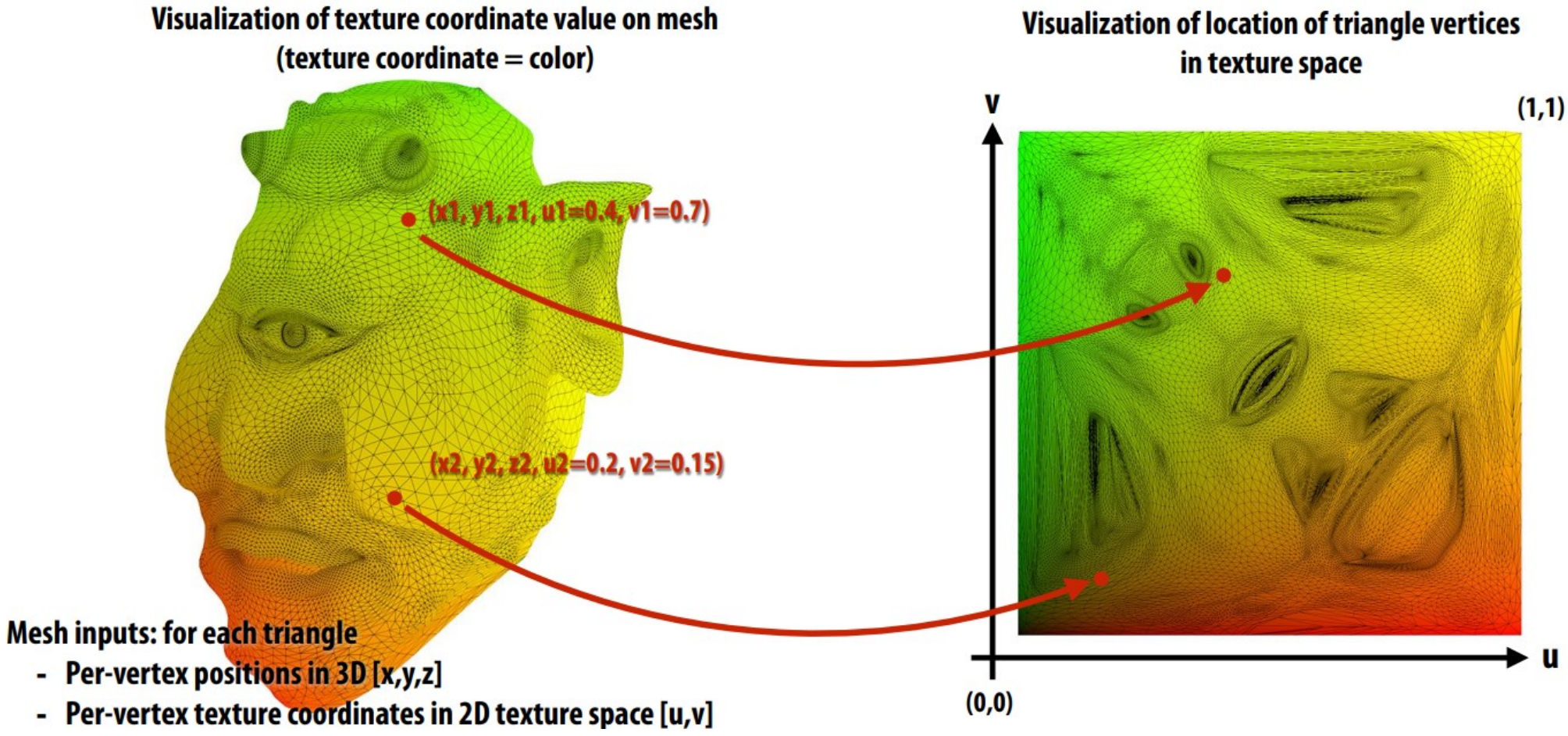
Texture Mapping



From Fatahalian et.al, CS248

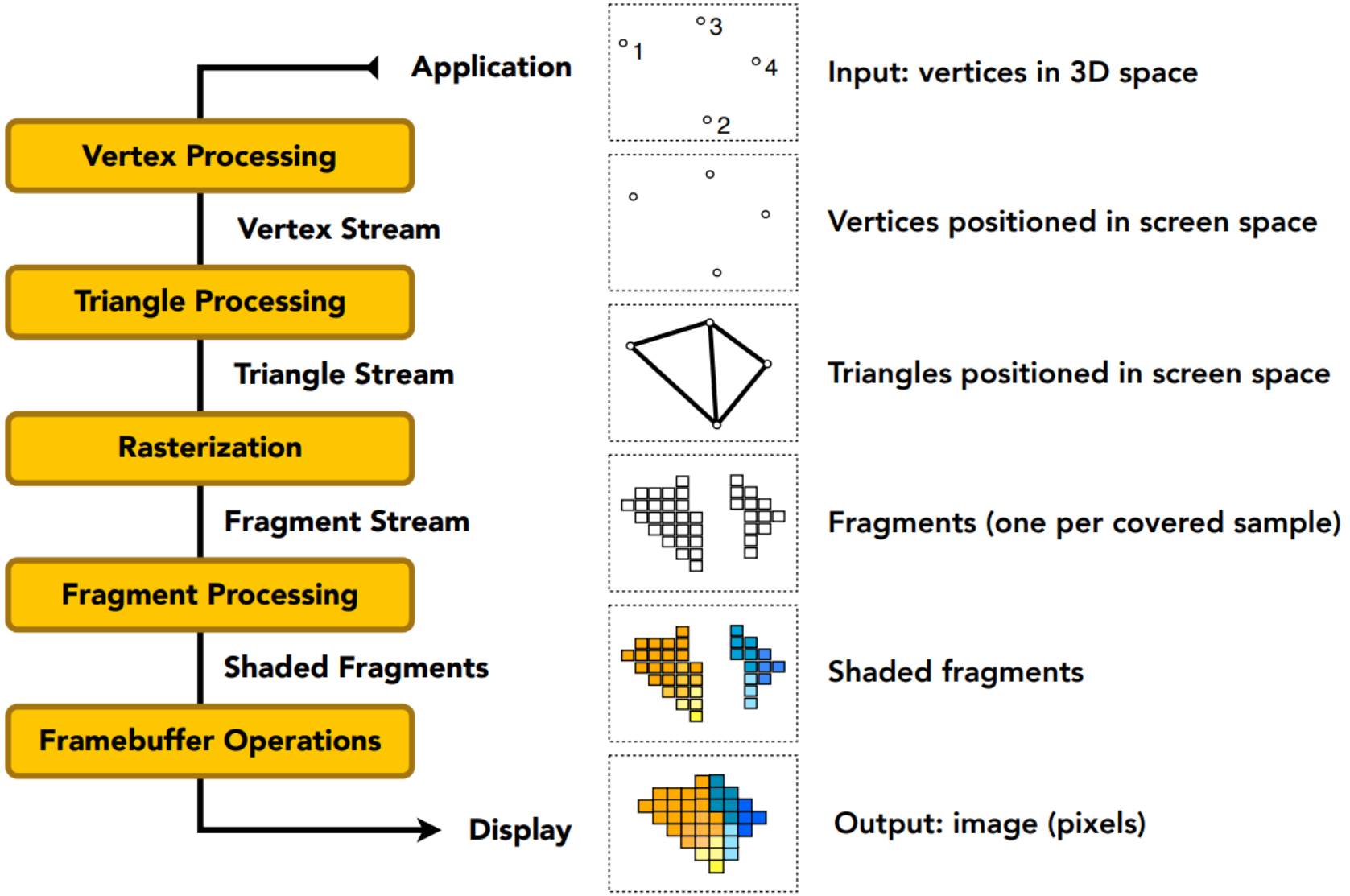
Texture Coordinate

- How to compute texture coordinates is a big subject! Mesh parametrization!



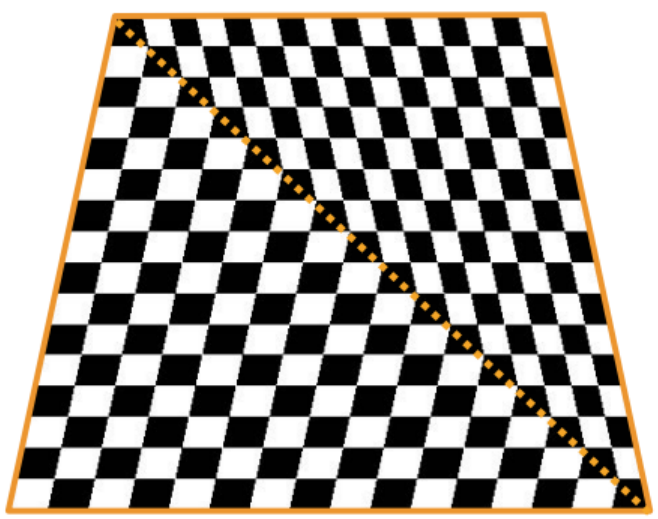
From Fatahalian et.al, CS248

Rendering Pipeline

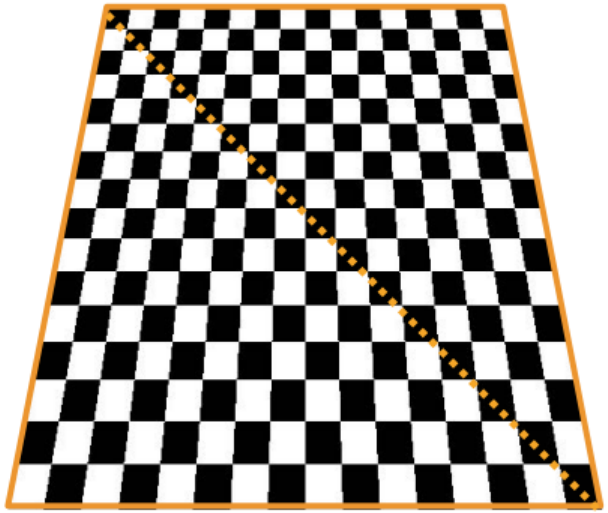


Compared with 2D

- Texture Mapping ≈ 2D image warping
- Bilinear interpolation, MIPMAP … is the same as 2D
- However directly warping image in 2D can lead to artifacts



Wrong mapping



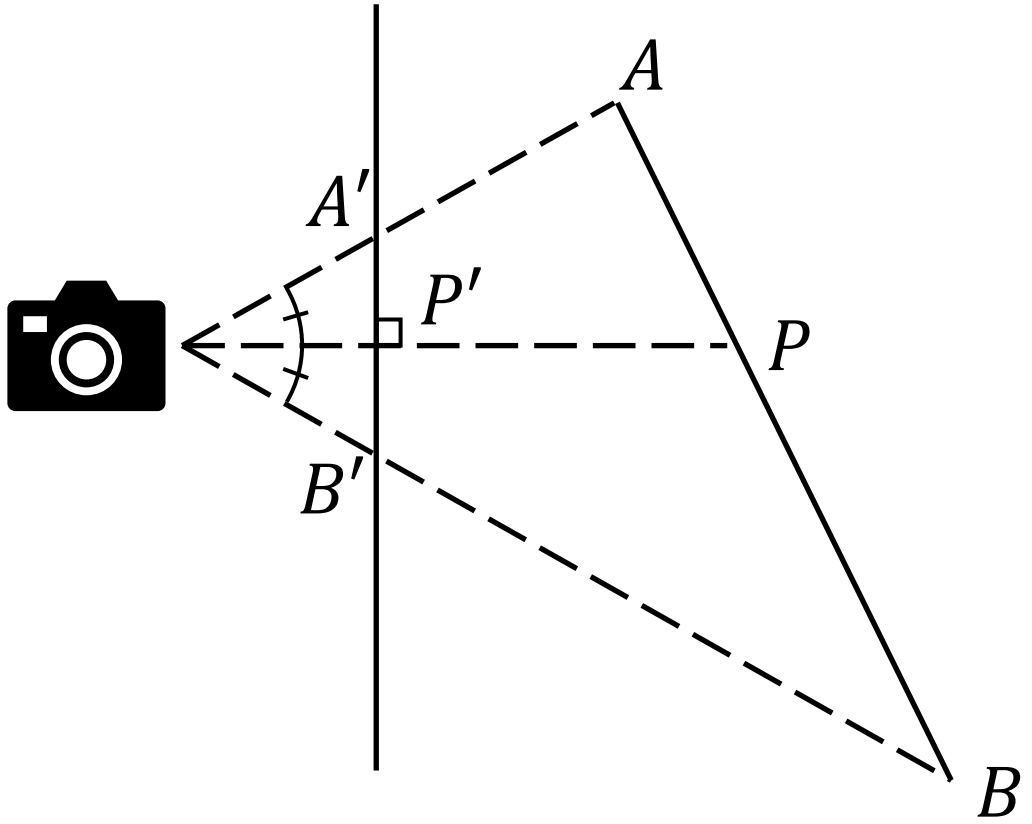
Perspectively correct

From Fatahalian et.al, CS248

Perspective Transformation

- $\overline{A'P'} = \overline{B'P'}$, but $\overline{AP} \neq \overline{BP}$

- Screen space interpolation ≠ world space interpolation

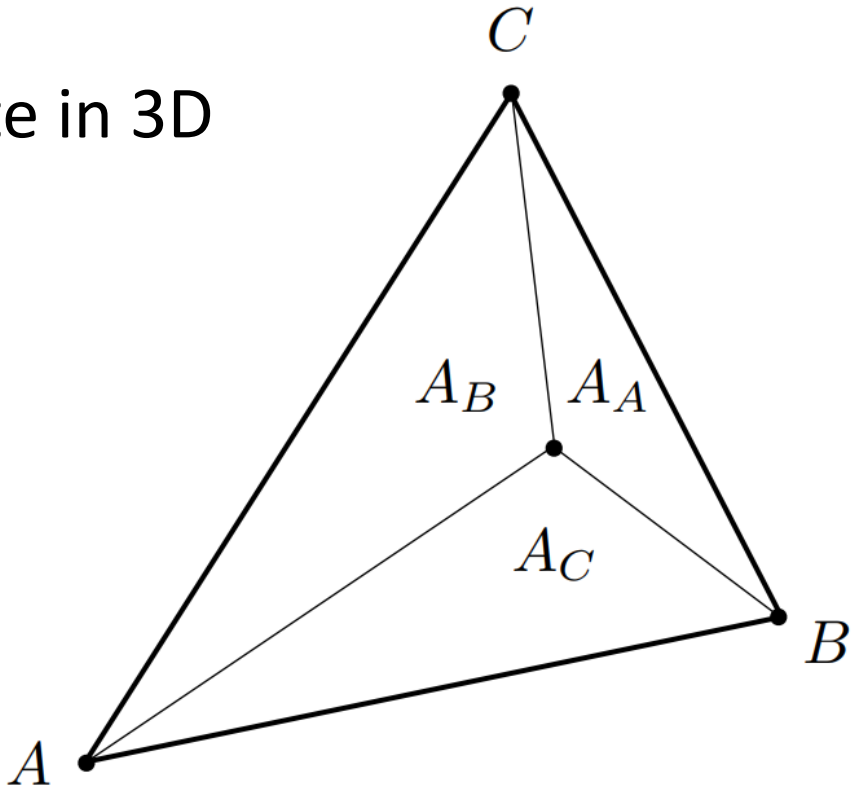


World Space Interpolation

- Barycentric coordinates again!

$$\boldsymbol{x} = \alpha \boldsymbol{x}_A + \beta \boldsymbol{x}_B + \gamma \boldsymbol{x}_C, \alpha + \beta + \gamma = 1$$

- $\boldsymbol{x}$ is coordinate in 3D



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

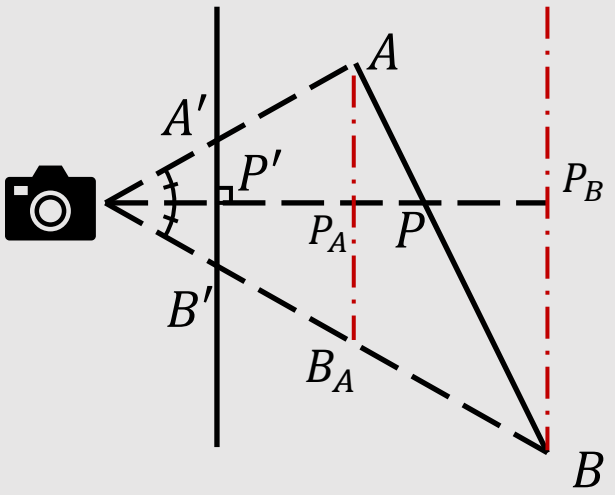
$$\beta = \frac{A_B}{A_A + A_B + A_C}$$

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

From Fatahalian et.al, CS248

Perspective-correct Interpolation

$$f = \frac{\frac{\alpha}{w_A} f_A + \frac{\beta}{w_B} f_B + \frac{\gamma}{w_C} f_C}{\frac{\alpha}{w_A} + \frac{\beta}{w_B} + \frac{\gamma}{w_C}}$$



Given $A'P' : P'B' = \beta : \alpha$

$\Rightarrow$

$$AP : PB = AP_A : P_B B = \frac{A'P'}{P'B'} P_A B_A : P_B B$$
$$= \frac{\beta}{\alpha} * w_A : w_B = \frac{\beta}{w_B} : \frac{\alpha}{w_A}$$

- f_A, f_B, f_C: the value to be interpolated
- α, β, γ: barycentric coordinates in screen coord.
- w_A, w_B, w_C: depth of each vertex before perspective transformation
- More details in: Perspective-Correct Interpolation

Perspective-correct Interpolation

- Perspective-correct interpolation is used not only for texture coordinates, also for surface normals, depth values…
- Perspective-correct interpolation is default for modern graphics APIs, like OpenGL (implemented as fixed pipeline functions)
- Interestingly, if interpolating depth value in screen space, we get

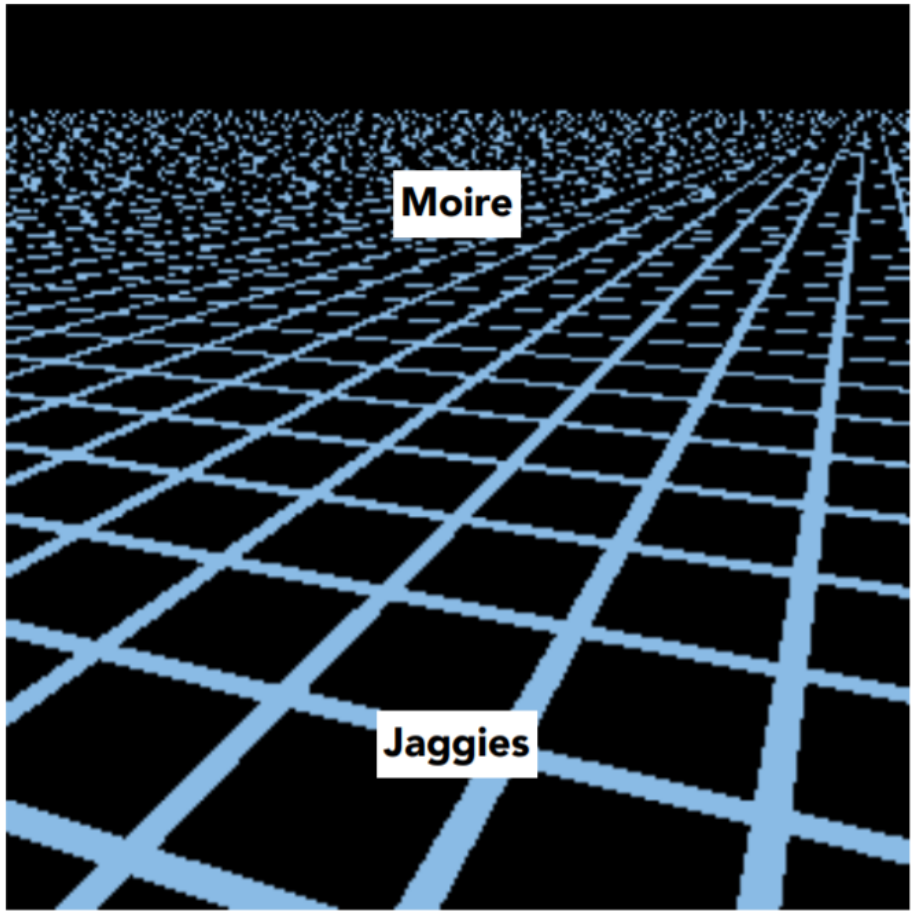
$$\frac{1}{w_p} = \frac{\alpha}{w_A} + \frac{\beta}{w_B} + \frac{\gamma}{w_C}$$

which is linearly interpolating the inverse of depth

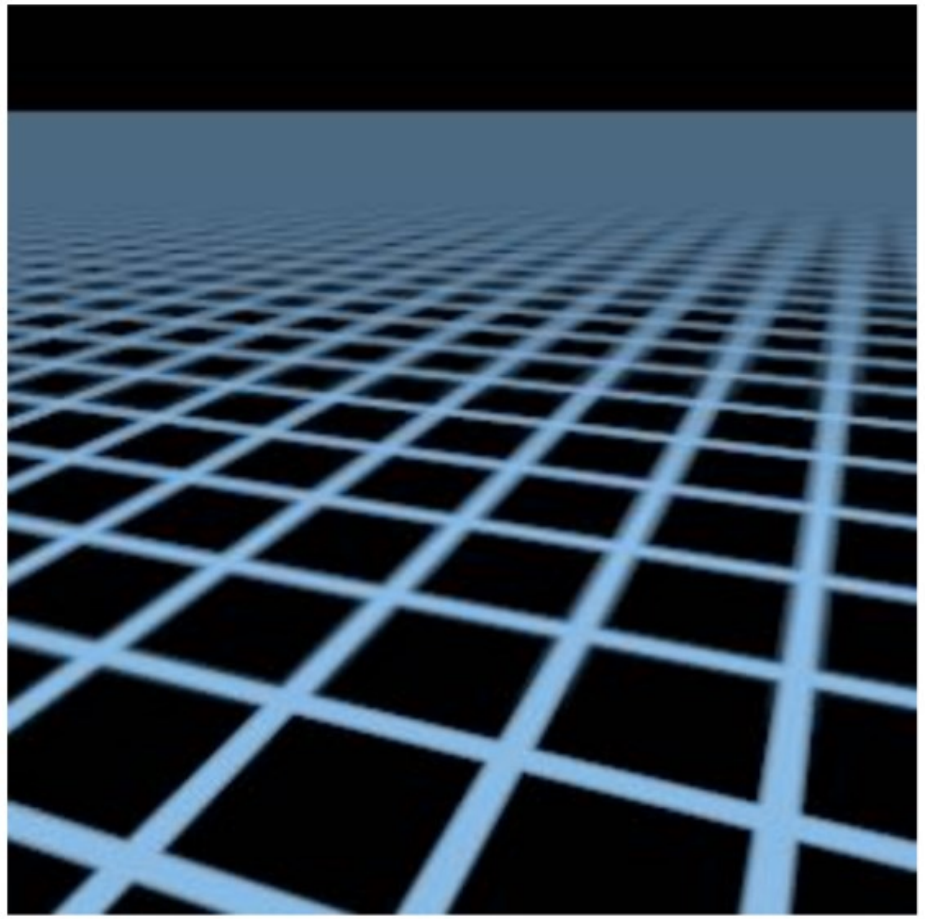
hint: when f=z, i.e, when $f_A = w_A$, $f_B = w_B$, $f_C = w_C$, $f_P = w_P$

$$f = \frac{\frac{\alpha}{w_A} f_A + \frac{\beta}{w_B} f_B + \frac{\gamma}{w_C} f_C}{\frac{\alpha}{w_A} + \frac{\beta}{w_B} + \frac{\gamma}{w_C}}$$

Anti-Aliasing



Aliasing

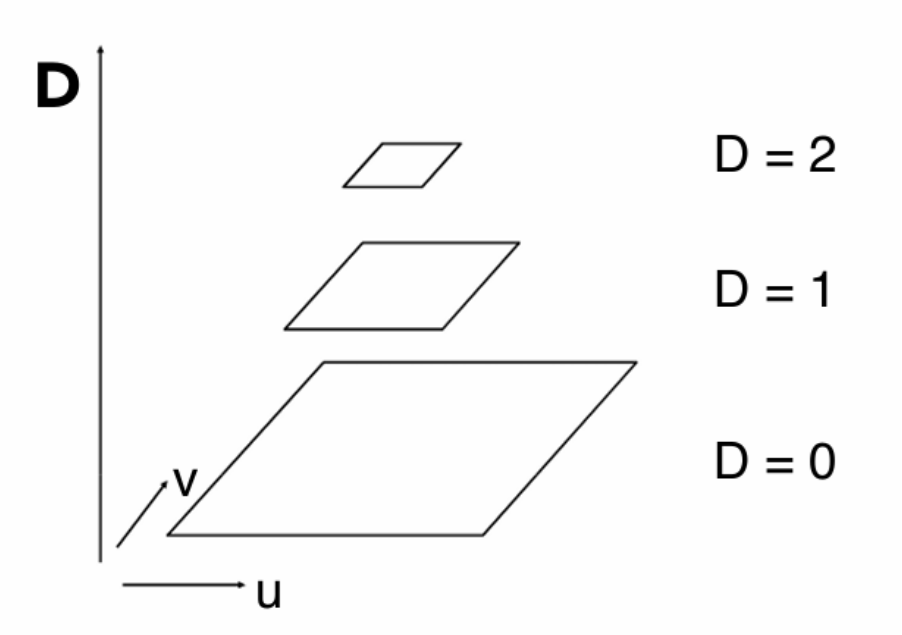


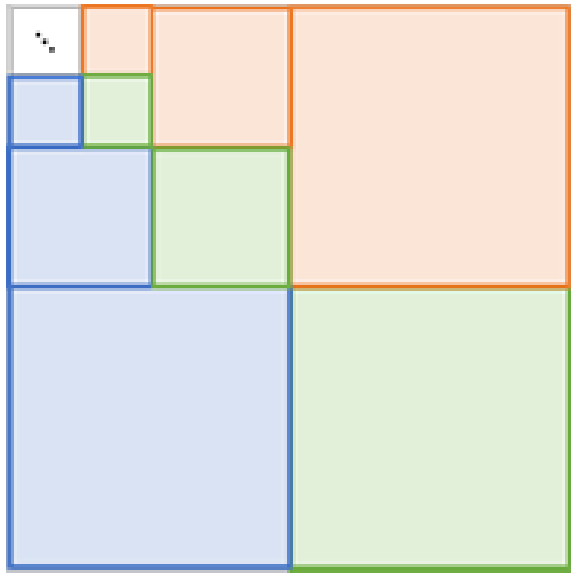
MIPMAP result

From Fatahalian et.al, CS248

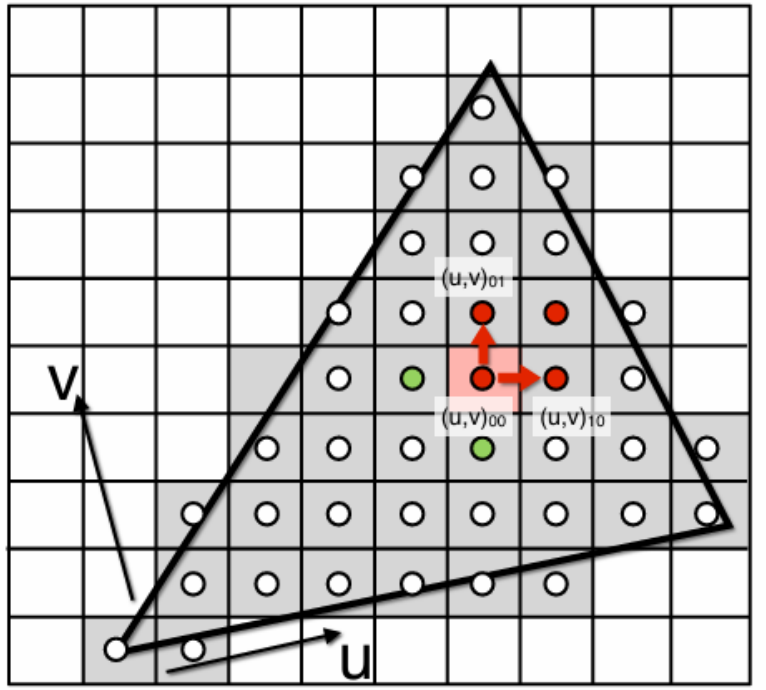
Anti-Aliasing

MIPMAP





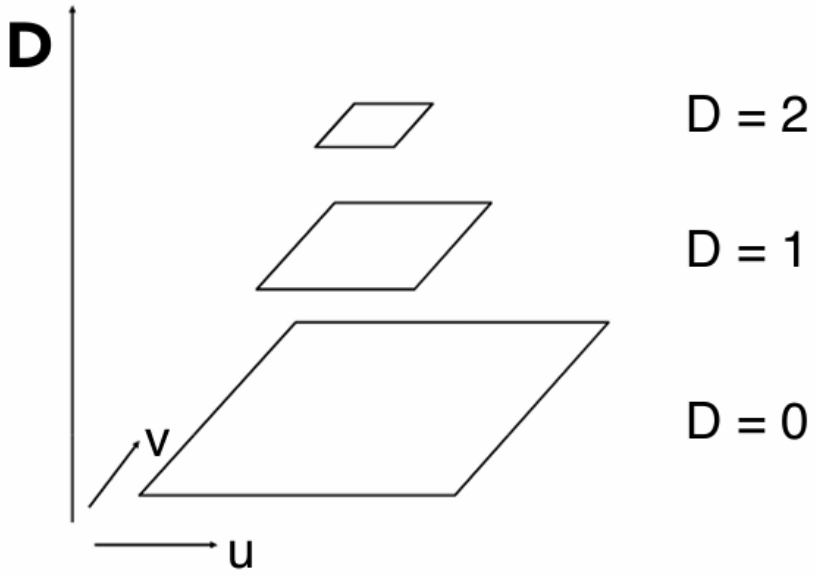
Which level ?

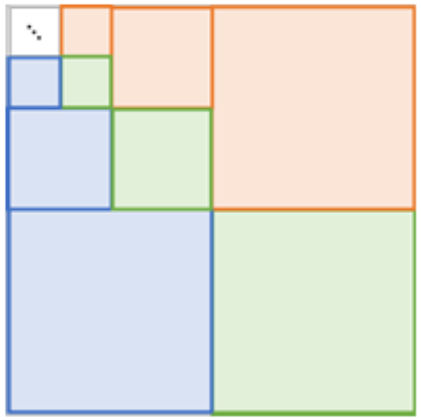


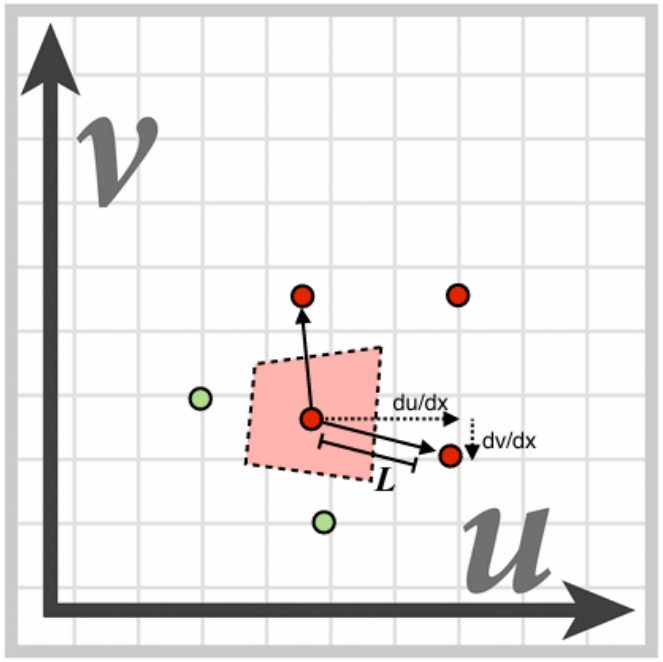
Anti-Aliasing

MIPMAP

$$D = \log_2 L$$

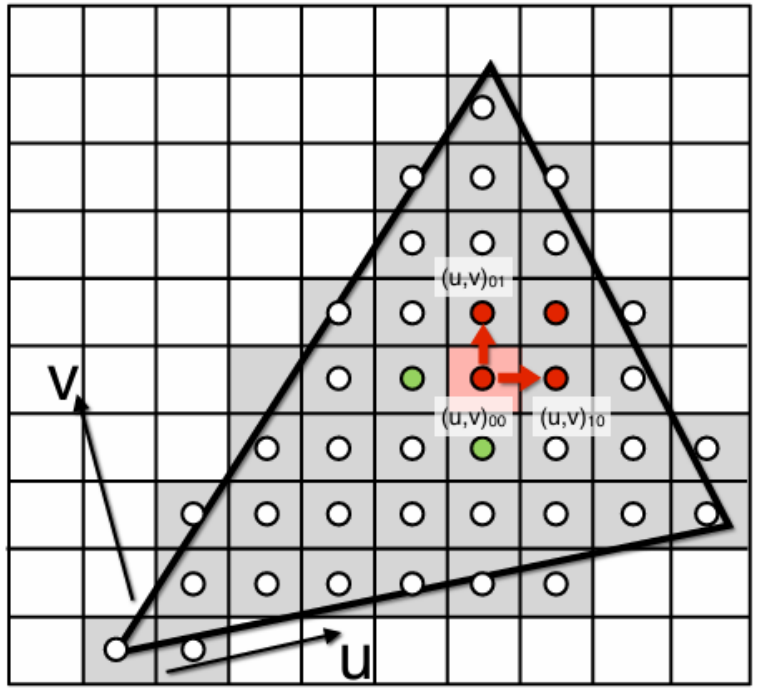






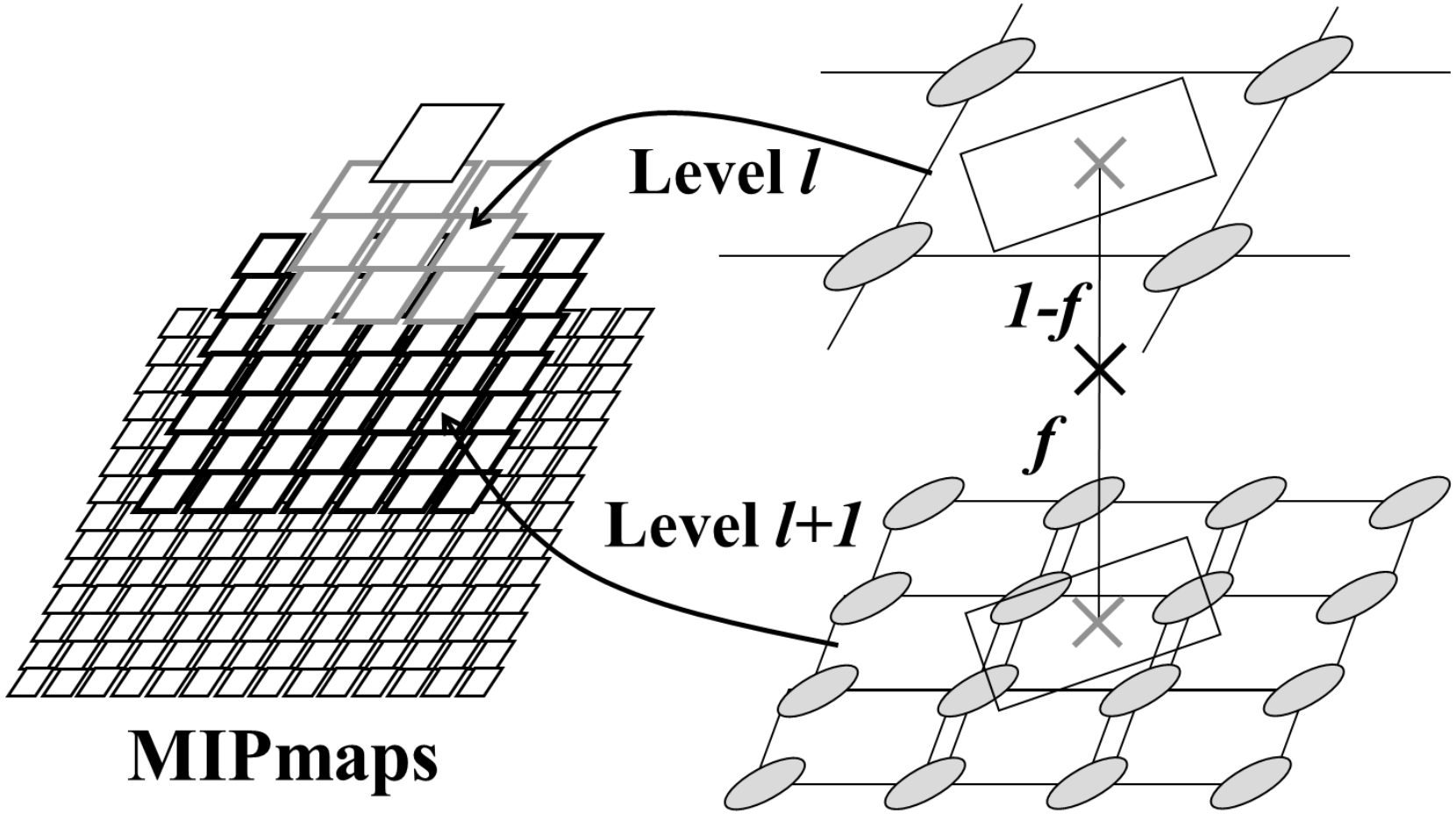
$$L = \max\left(\sqrt{\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2}, \sqrt{\left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2}\right)$$

Which level ?



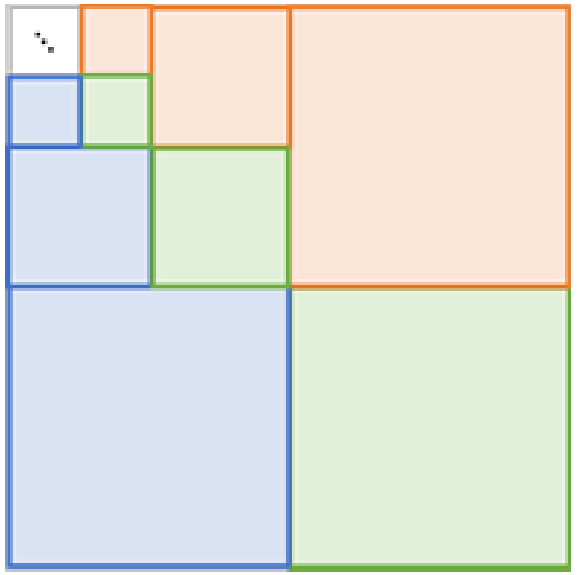
Anti-Aliasing

MIPMAP

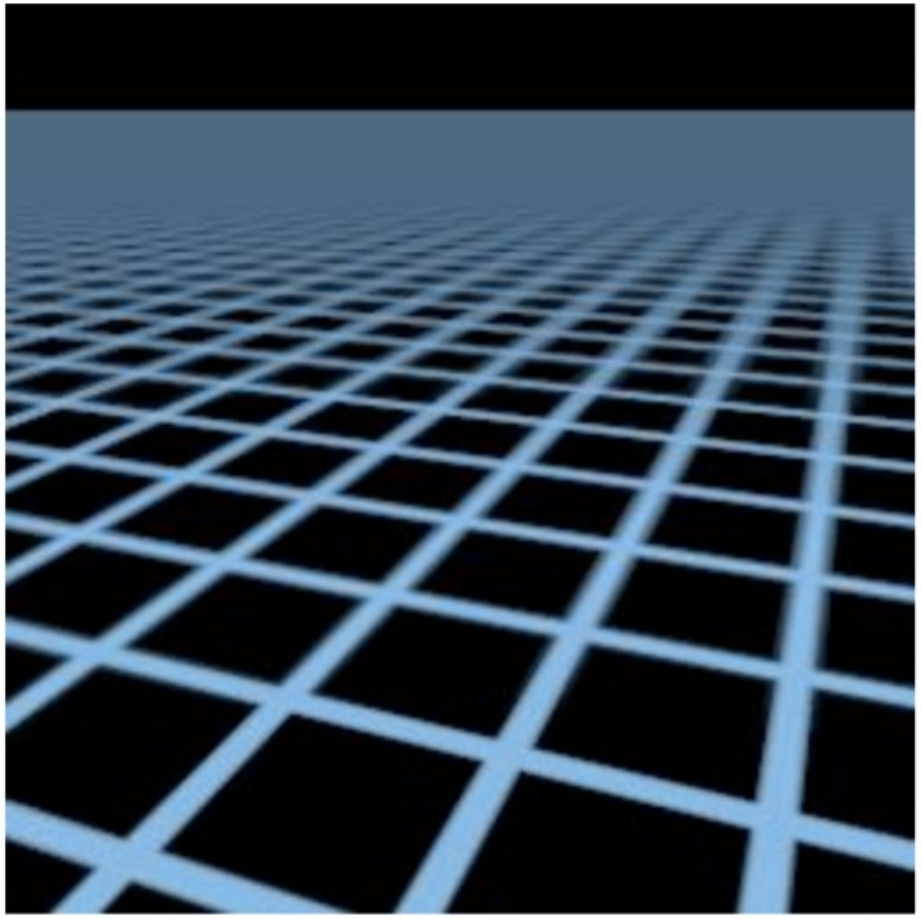


$$D = \log_2 L$$

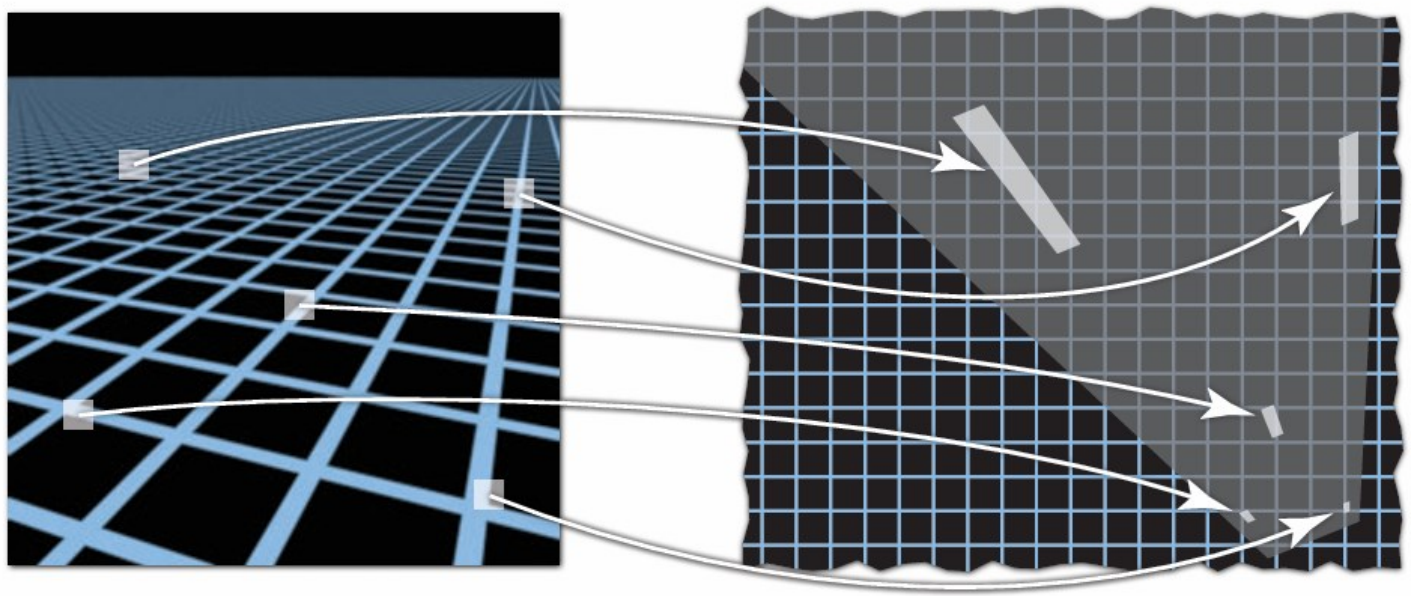
$$l = int(D)$$



Anti-Aliasing

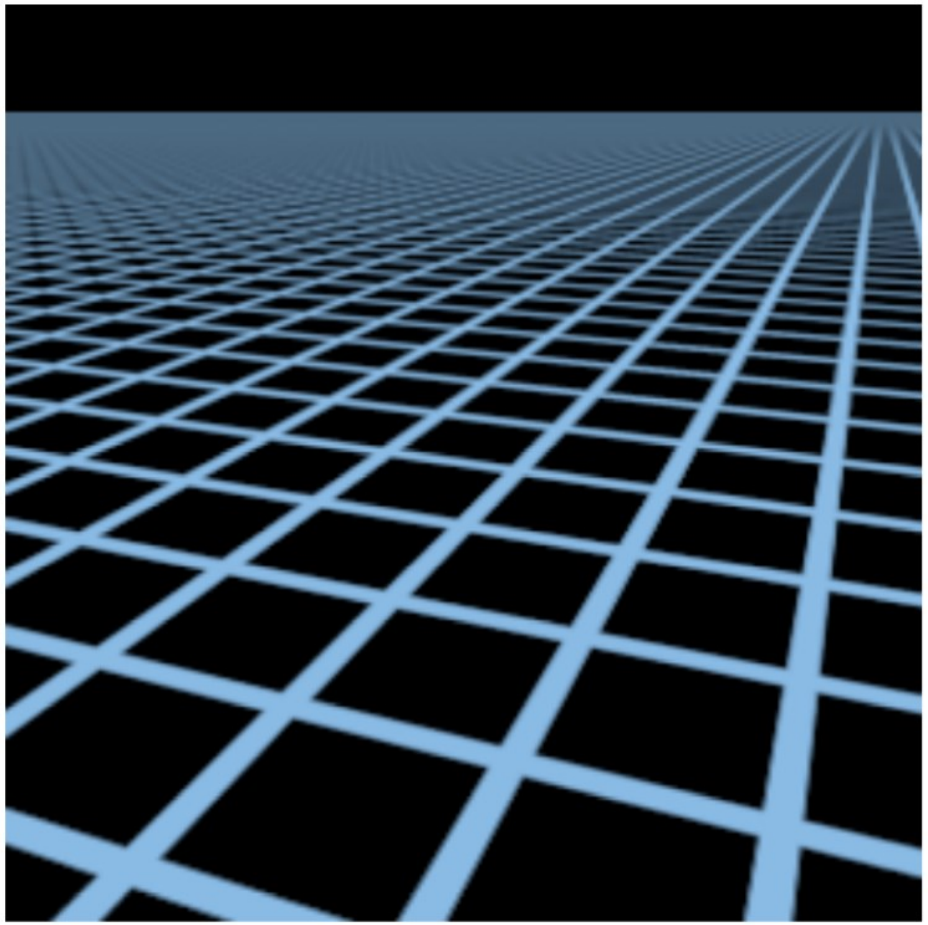


MIPMAP result



From Fatahalian et.al, CS248

Anisotropic Anti-Aliasing



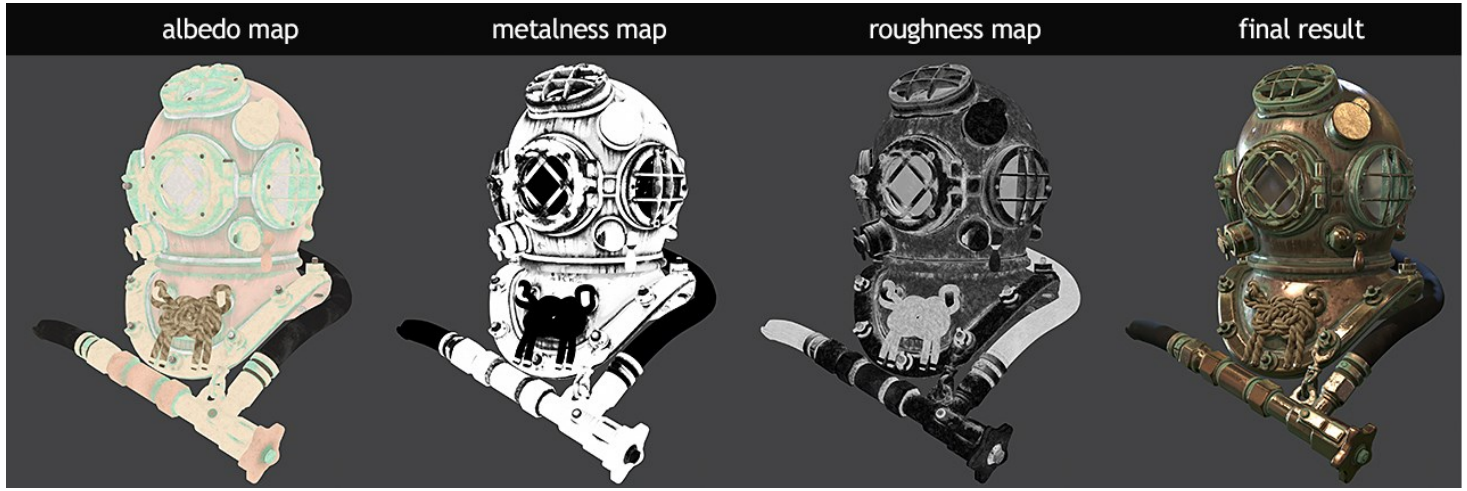
From Fatahalian et.al, CS248

Applications

For Appearance

- Describe colors, roughness, metalness…

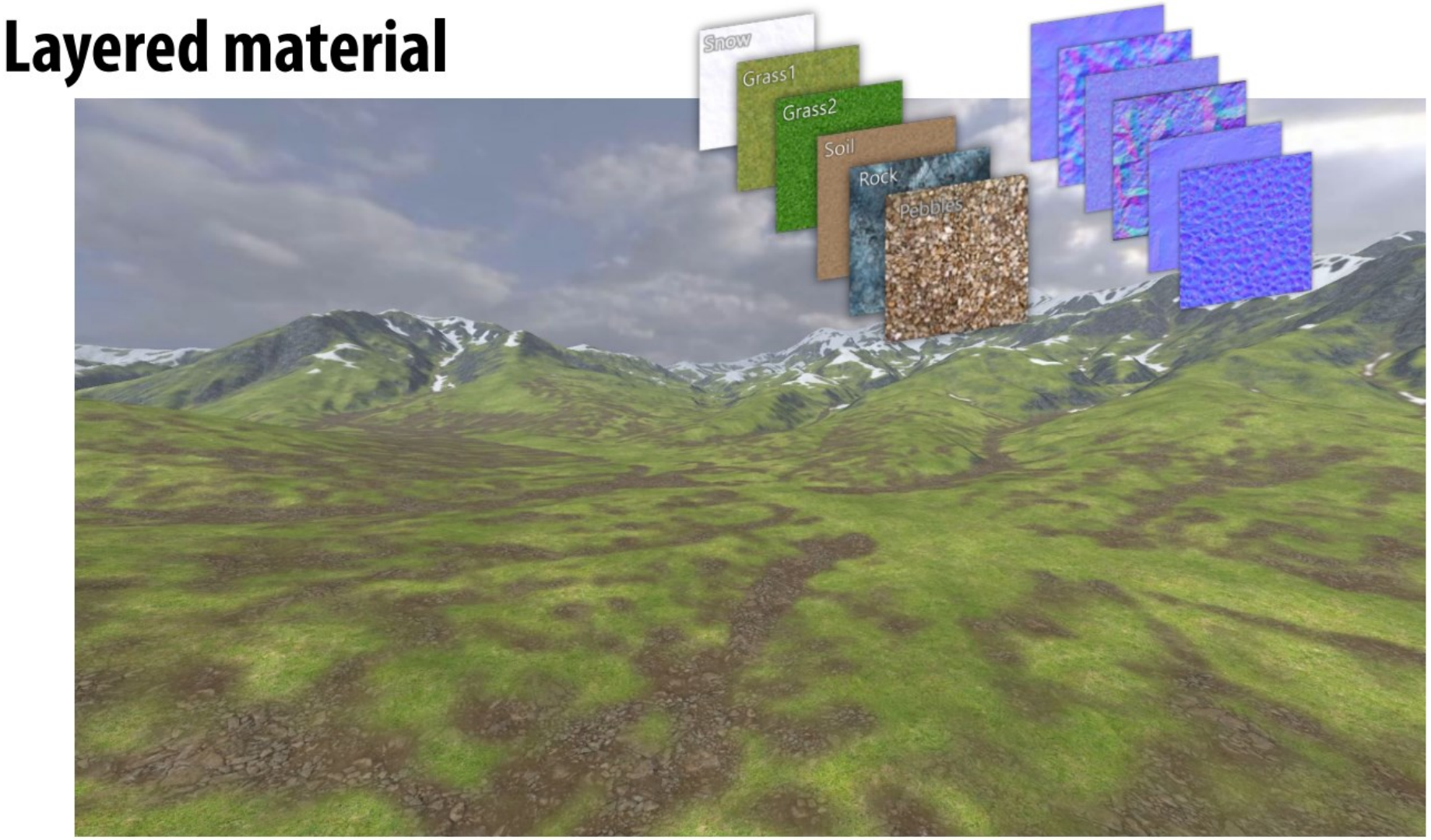
$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$



PBR Texture Conversion

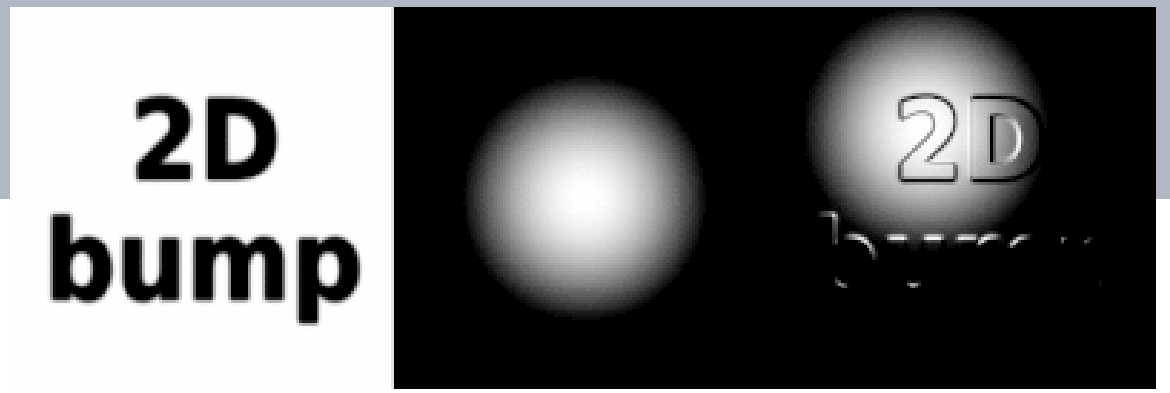
Making Of 'Barack'

For Appearance



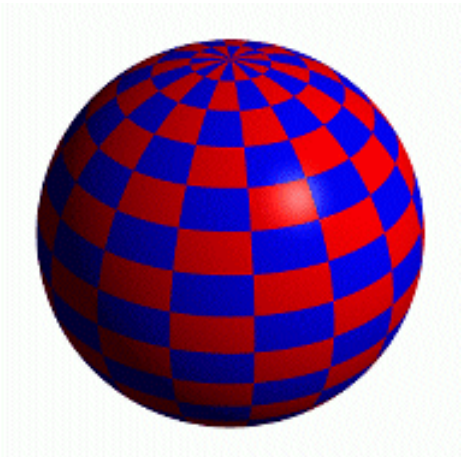
From Fatahalian et.al, CS248

For Geometry



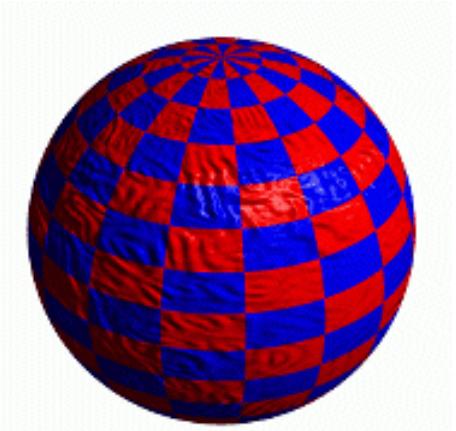
- Bump map

The bump map is treated as a single-valued height function, whose partial derivatives tell how to alter the true surface normal at each point on the surface. Bump Mapping assumes that the Illumination model is applied at every pixel (as in Phong Shading or ray tracing).



Sphere w/Diffuse Texture

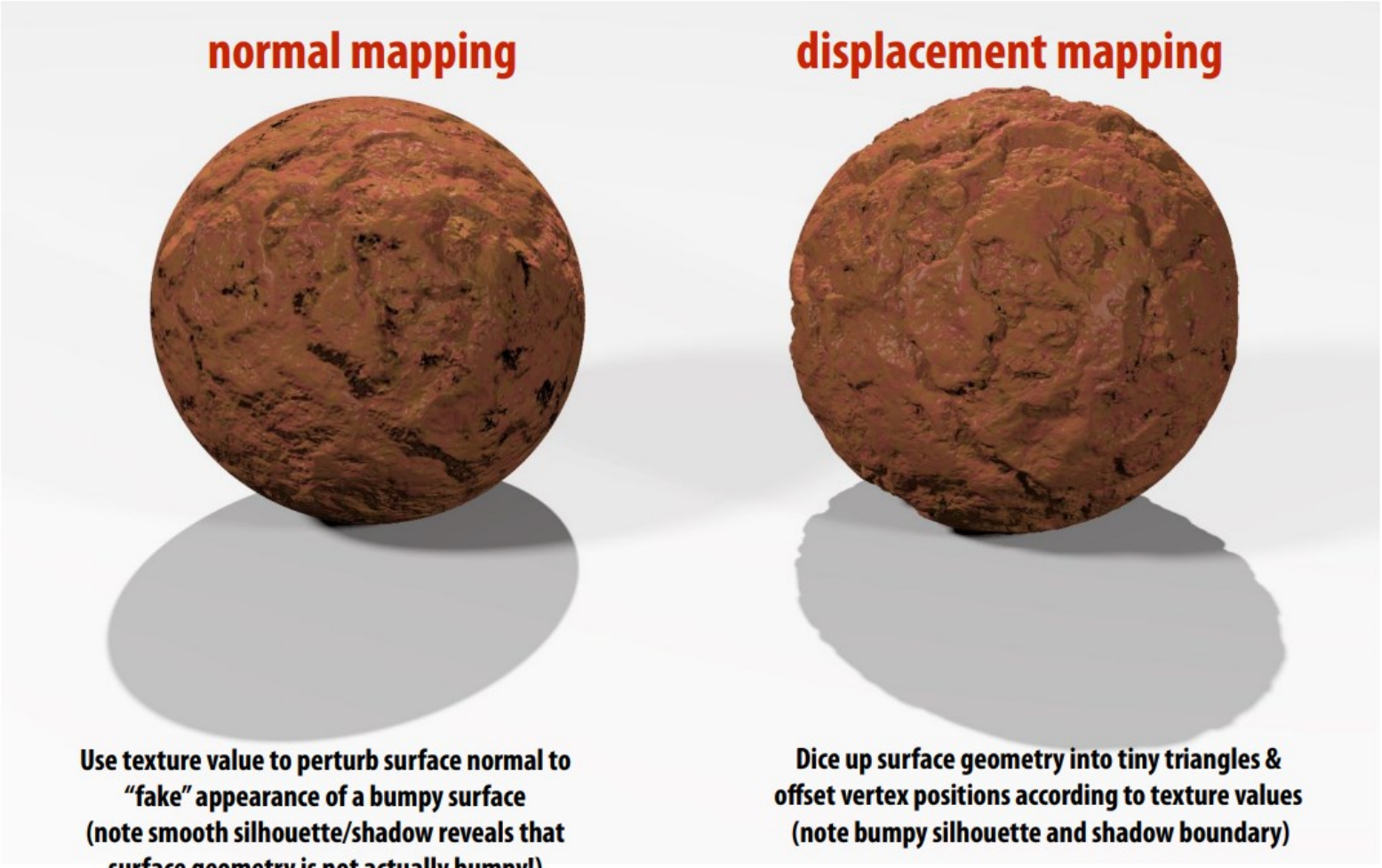
Swirly Bump Map



Sphere w/Diffuse Texture & Bump Map

For Geometry

- Normal map, displacement map

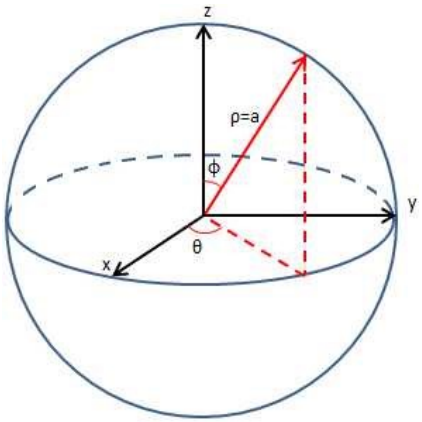


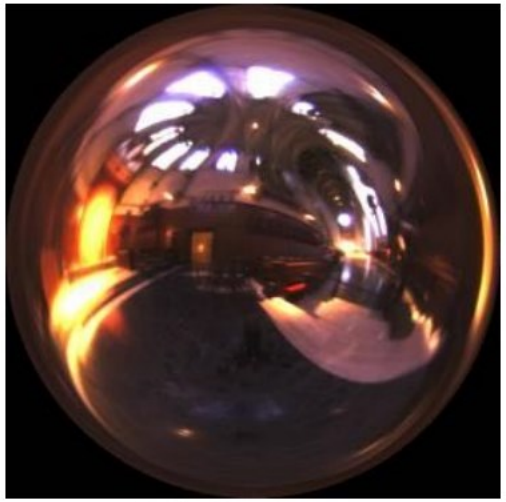
From Fatahalian et.al, CS248

For Lights and Shadows

$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$

Original model | With ambient occlusion | Extracted ambient occlusion map





Grace Cathedral environment map

Environment map used in a rendering

ϕ

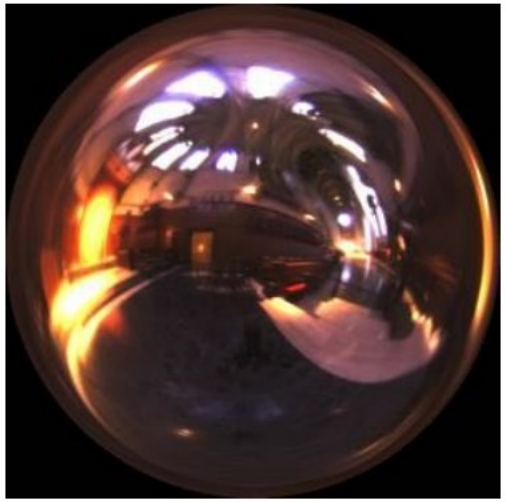
θ

$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$

For Lights and Shadows

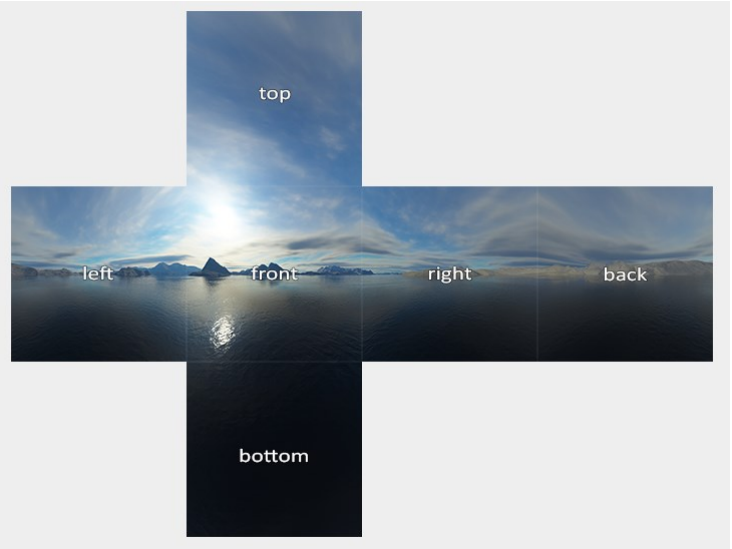
$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$

Original model | With ambient occlusion | Extracted ambient occlusion map



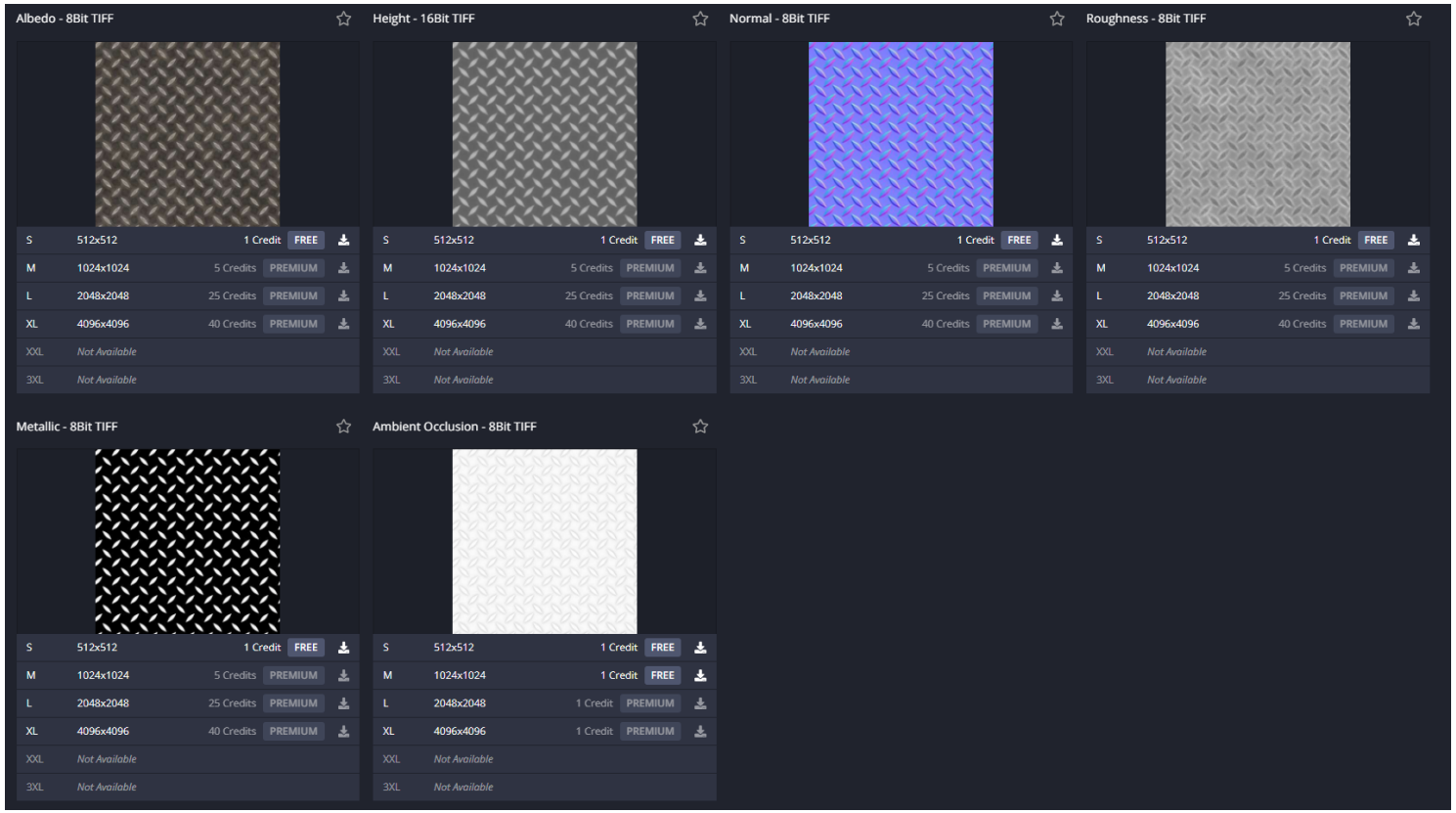
Grace Cathedral environment map

Environment map used in a rendering



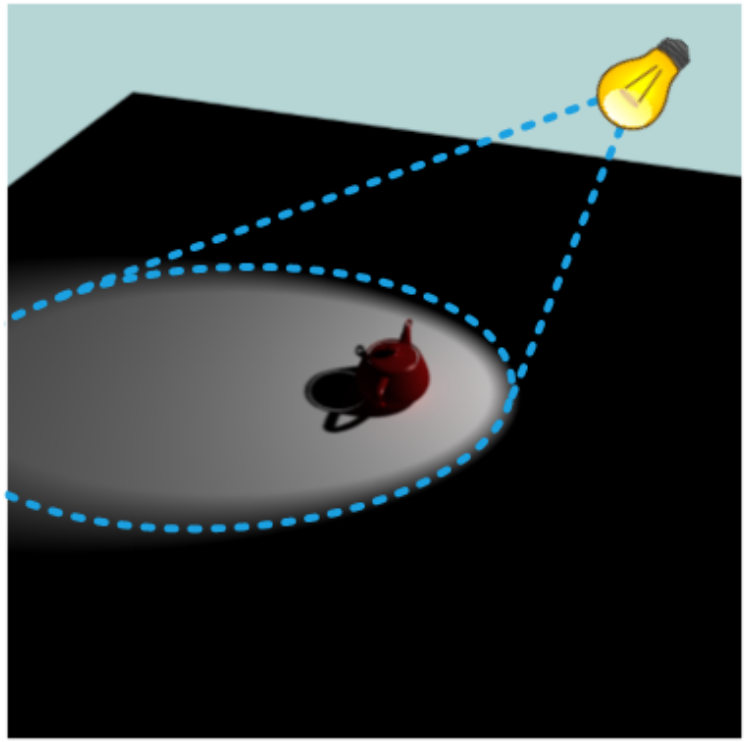
$$I = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$$

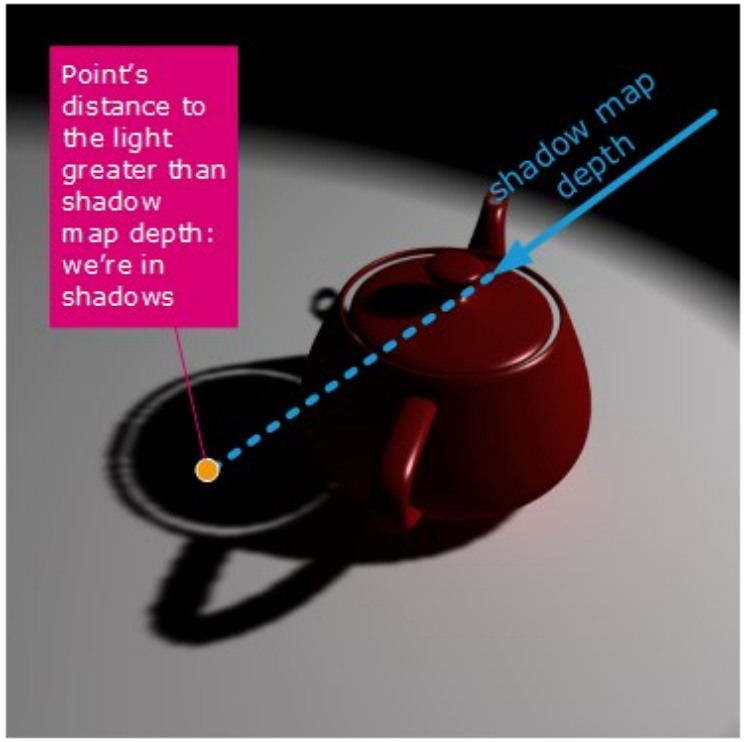
Color + Geometry + Ambient Occlusion



The material you can get online, from textures.com

Shadow Map



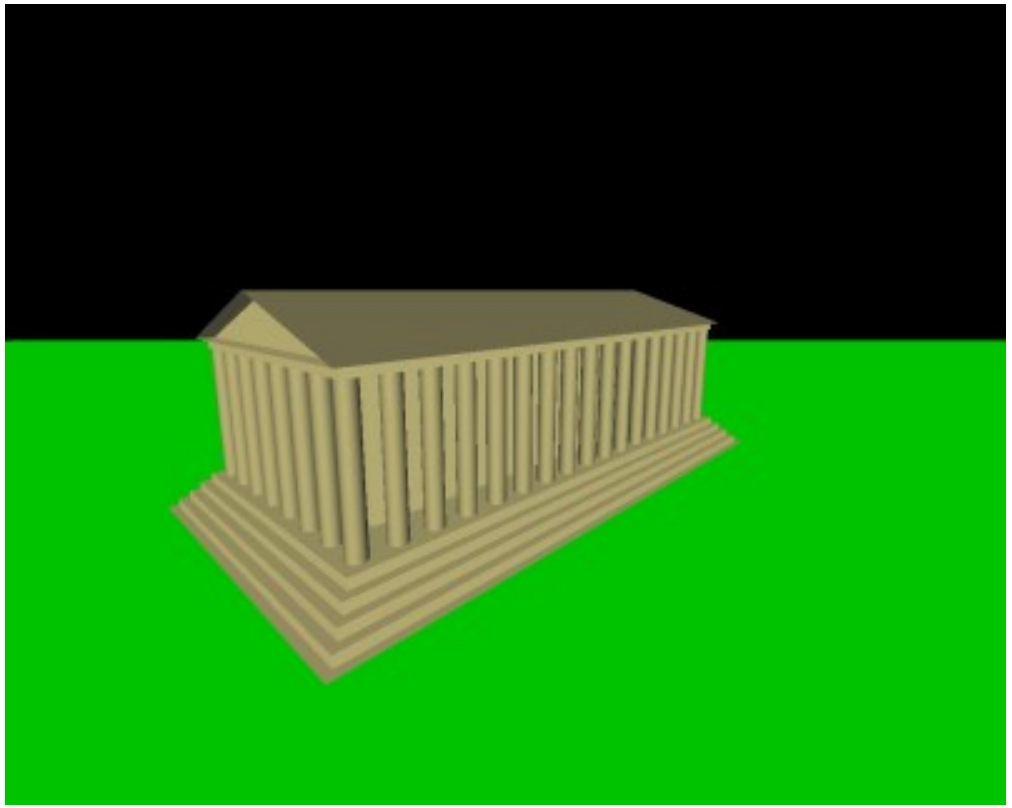


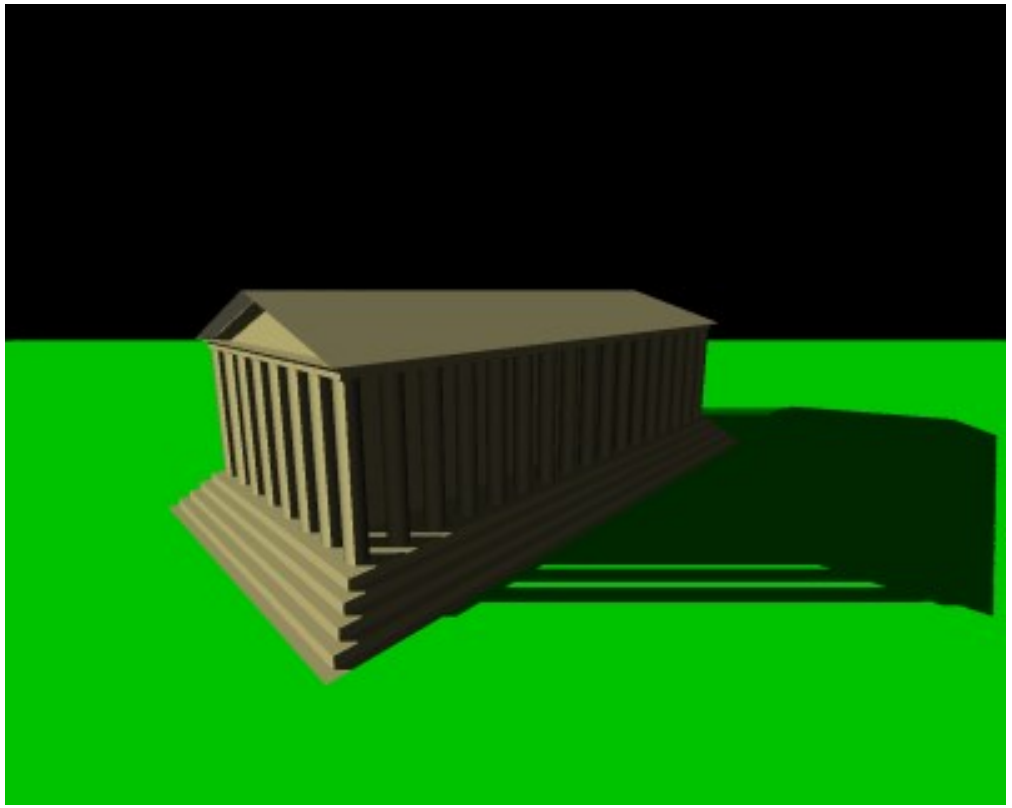
Rendered from light source

If p's distance to the light is greater than the shadow map depth: $I_p = k_a I_a$

Else: $I_p = k_a I_a + f_{att} I_l (k_d \cos\theta + k_s (\cos\phi)^{n_s})$

Shadow Map





Texture Generation

From Capture

From medialoot.com

From forums.sketchup.com

How to make it seamless?

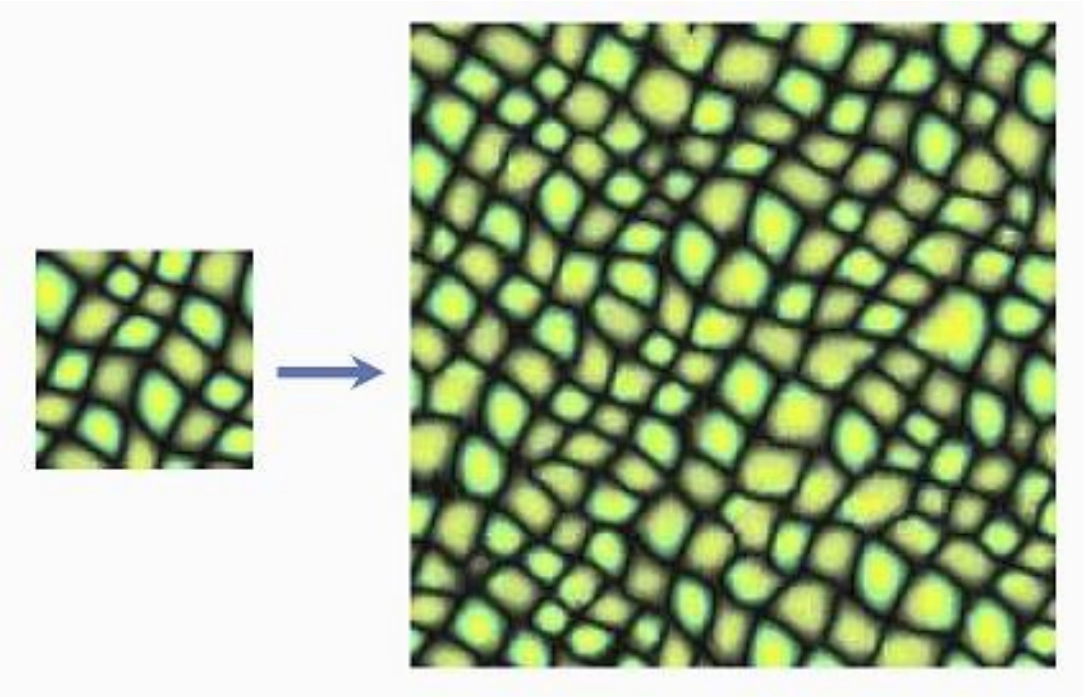
From medialoot.com

From imgonline.com

How to make it seamless?

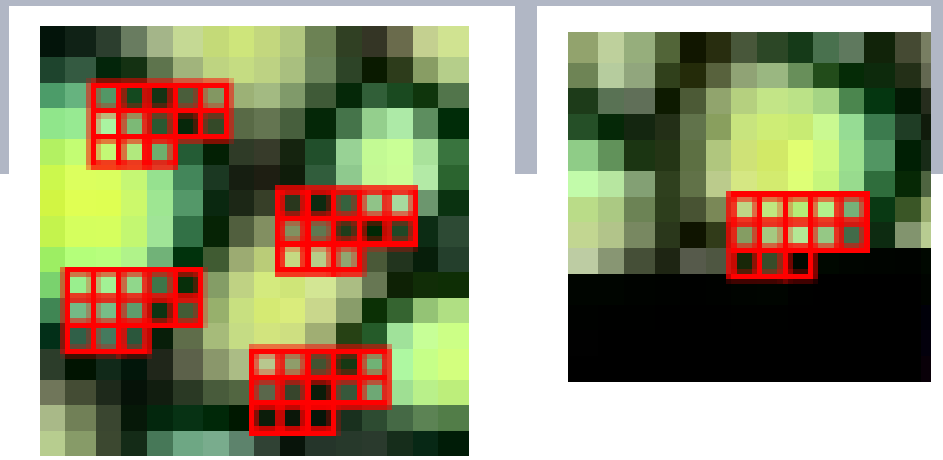
- Manually take care of the edges, or flip it!
- Texture synthesis (big research subject!)

From imgonline.com



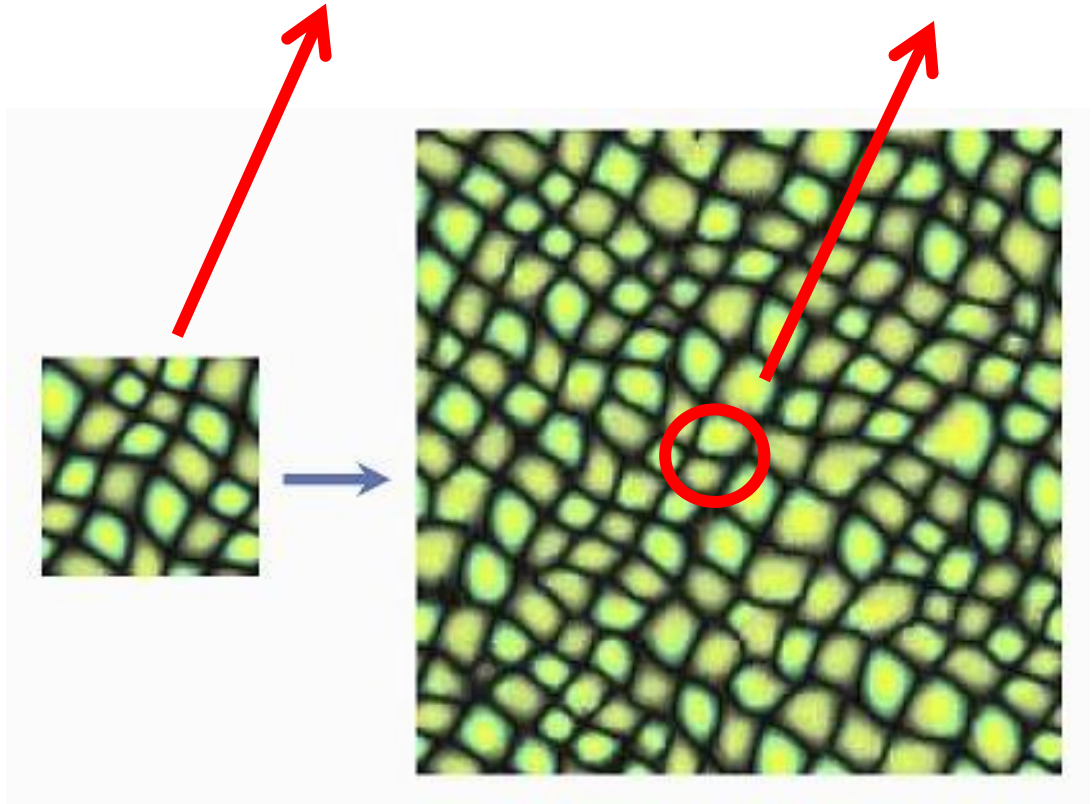
Texture synthesis

How to make it seamless?



- Manually take care of the edges, or flip it!
- Texture synthesis (big research subject!)

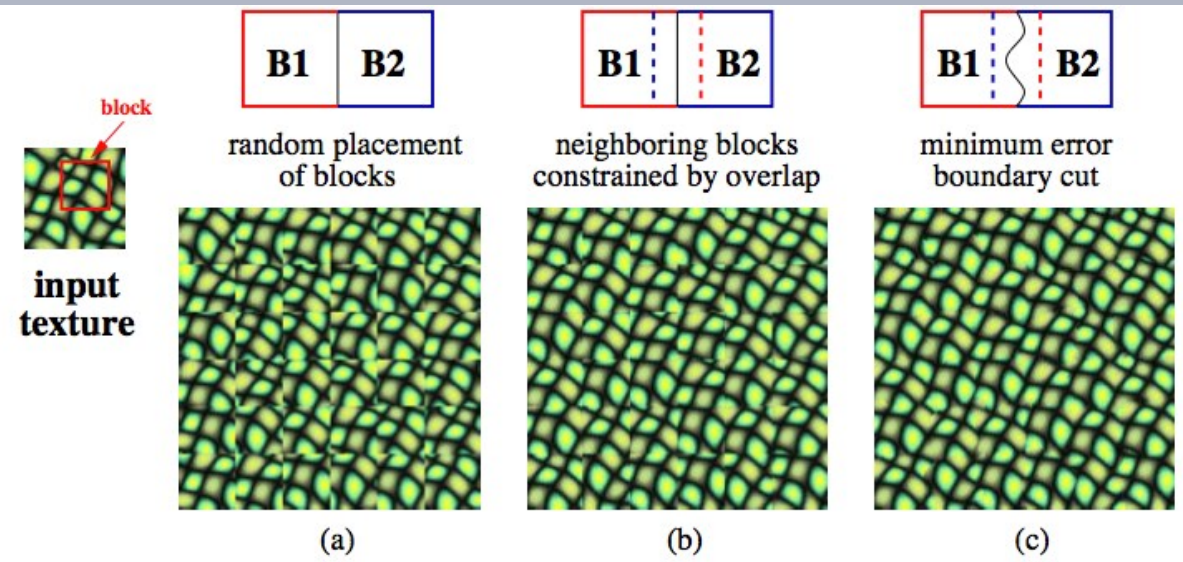
From imgonline.com

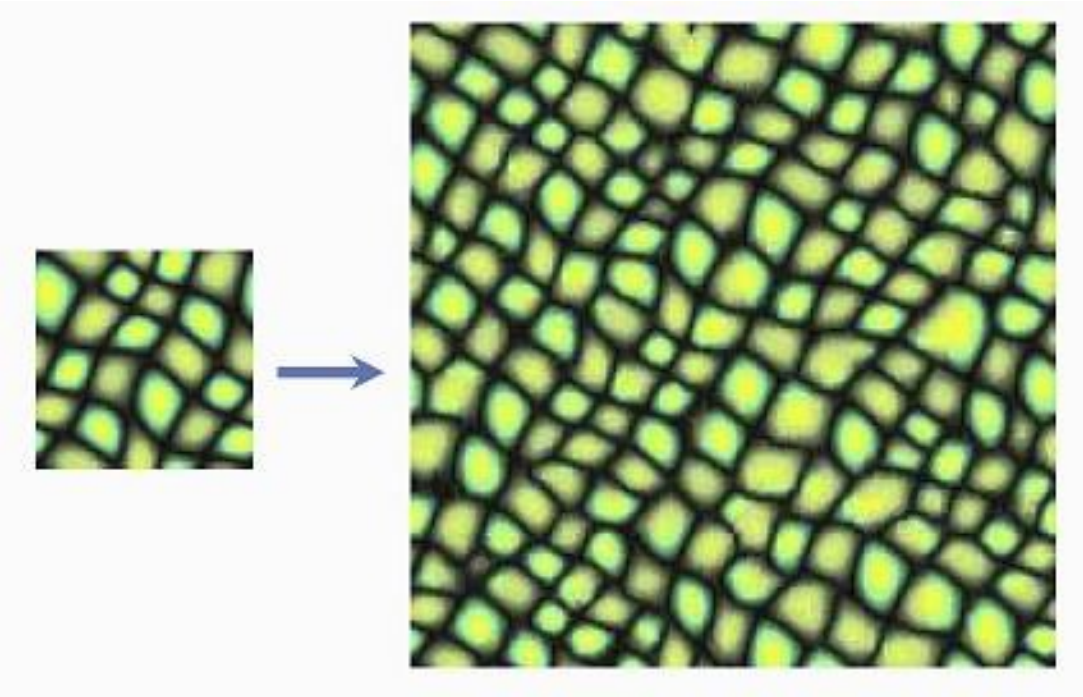


Texture synthesis

How to make it seamless?

- Manually take care of the edges, or flip it!
- Texture synthesis (big research subject!)



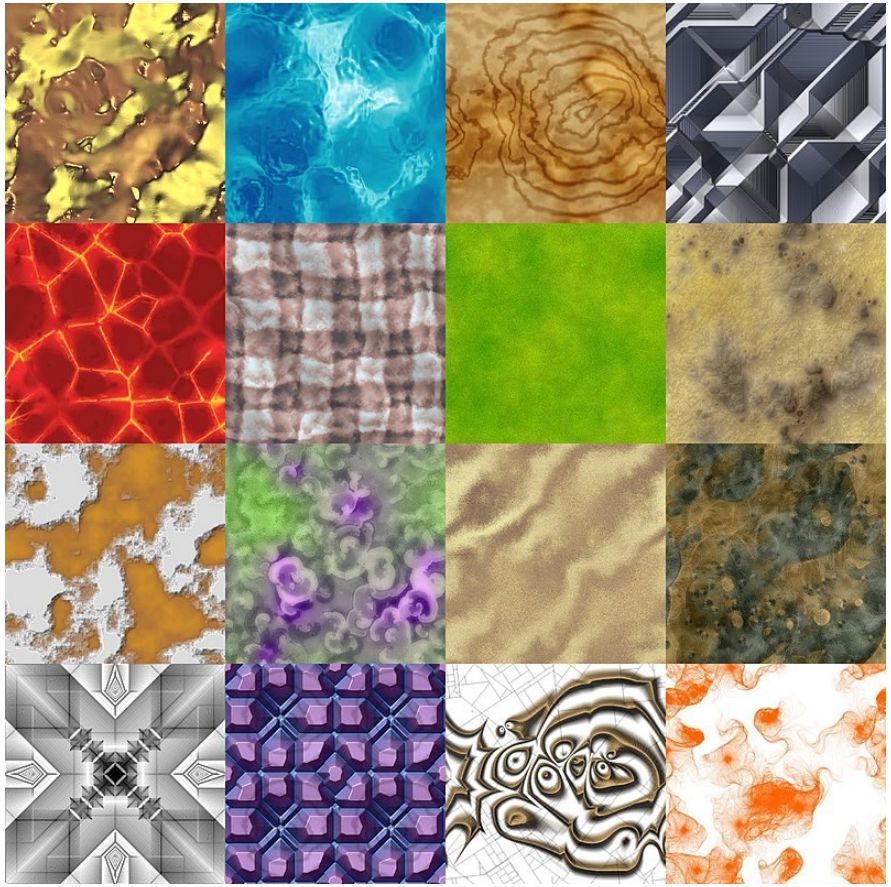


From imgonline.com

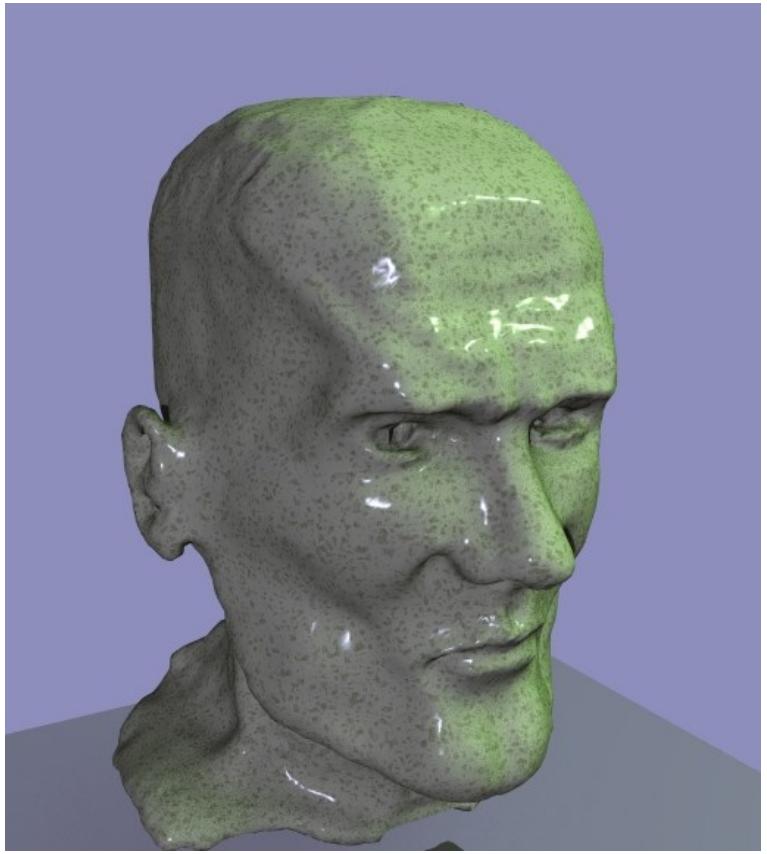
Texture synthesis

Procedural Texture

- Created using algorithms (from noise, or cellular automata, or …)

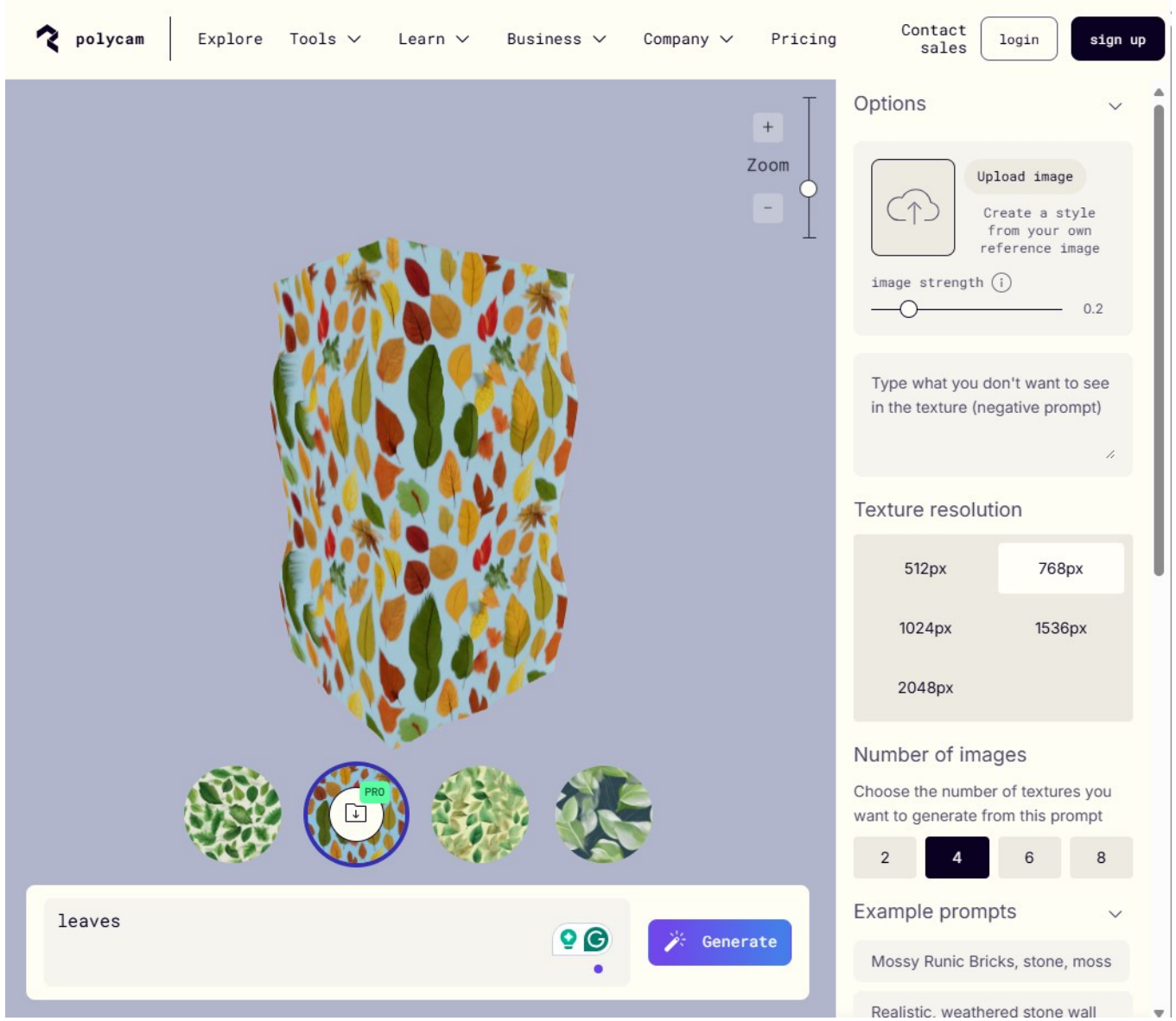


Perlin noise (naturally for 3D!)



Andreas Bærentzen's self sculpture

AI Texture Generator



https://poly.cam/tools/ai-texture-generator

Thanks