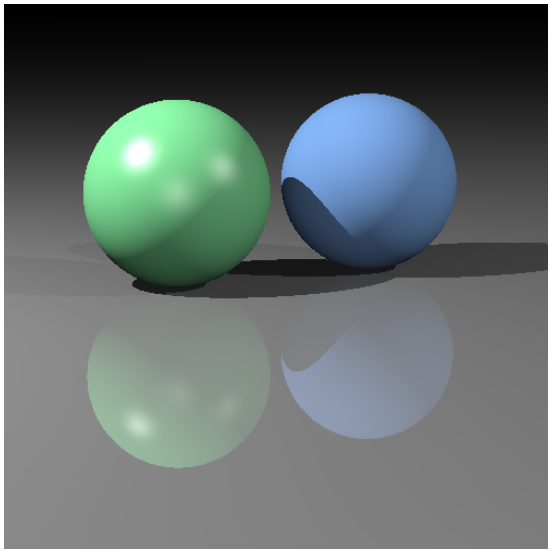
CENG 477
Introduction to Computer Graphics

Texture Mapping

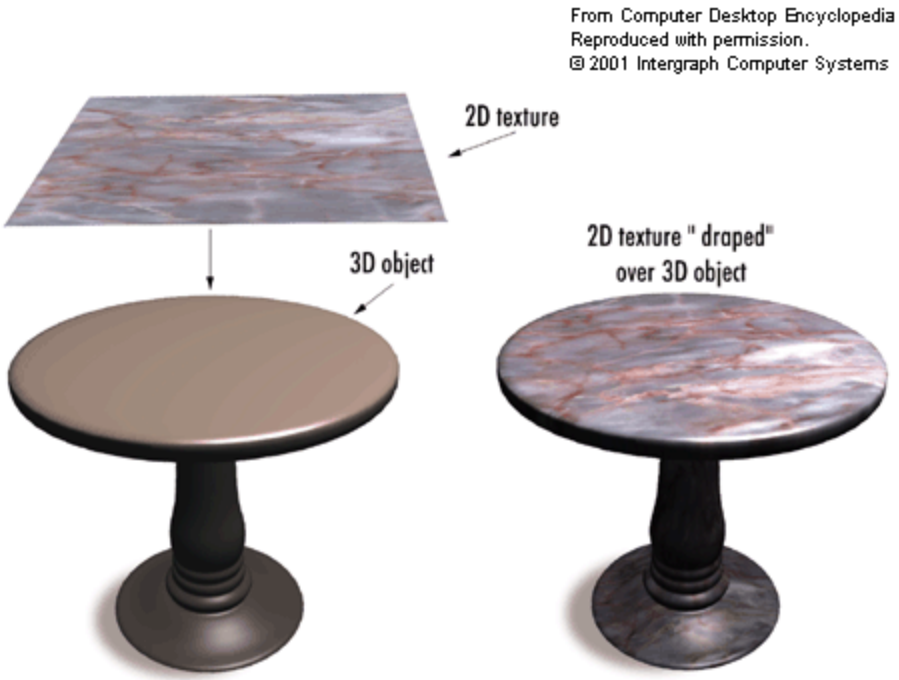
Until Now

- We assumed that objects have fixed R, G, B reflectance values
- Surface orientation and light direction determined the shading of objects
- Our best image so far is something like:



Texture Mapping

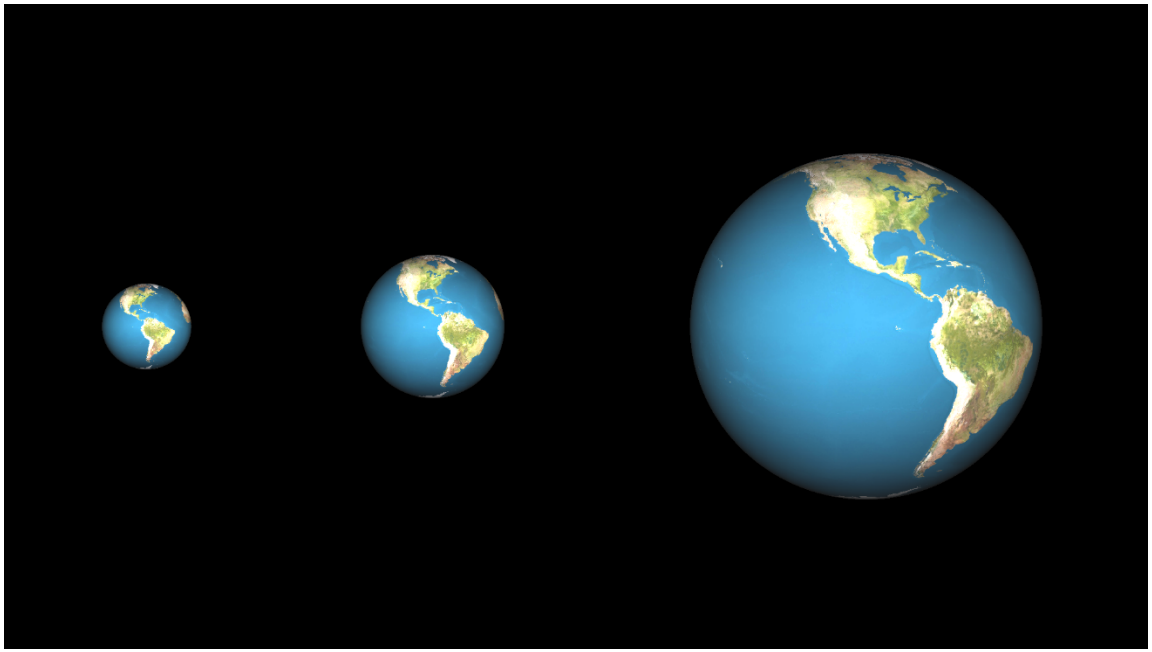
- **Goal:** Increase visual realism by using images to simulate reflectance characteristics of objects.
- A cheap and effective way to spatially vary surface reflectance.



From http://img.tfd.com

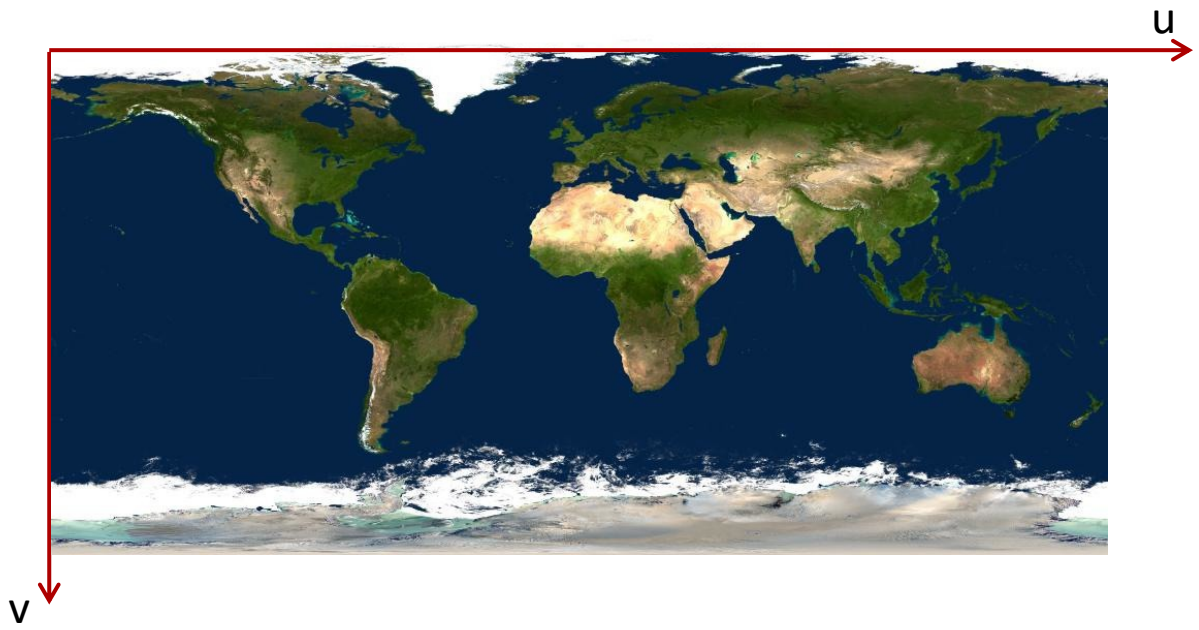
Texture Mapping

- Image (texture) sizes and object sizes may vary
- We need a uniform way so that any object can be mapped by any texture



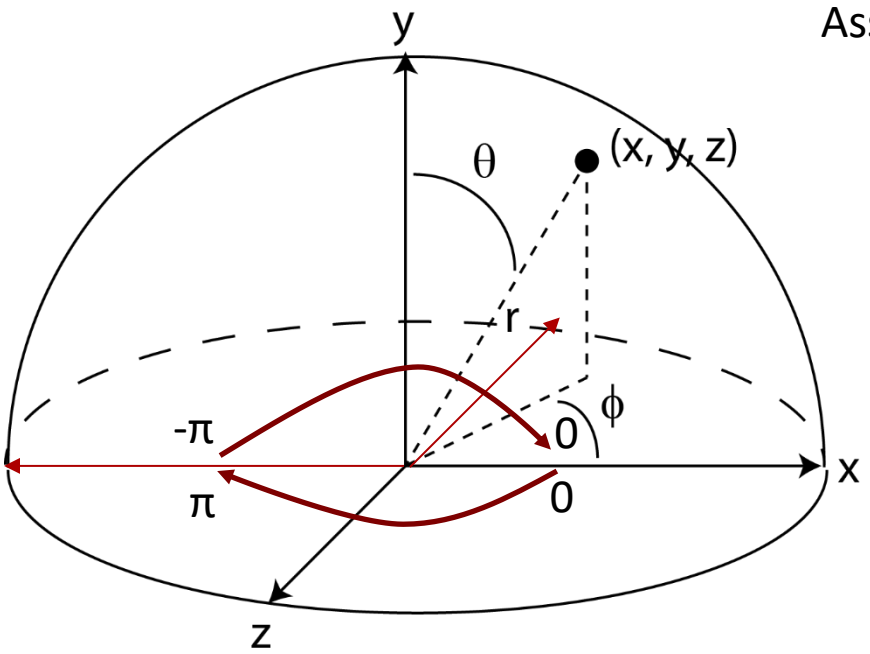
Texture Mapping

- **Step 1:** Associate a *(u, v)* coordinate system with the texture image where *(u, v)* $\in [0,1]x[0,1]$



Texture Mapping

- **Step 2:** Parameterize the surface to be texture mapped using two coordinates. For instance, a sphere:



Assuming that the center is at (0, 0, 0):

$x = r\sin\Theta\cos\varphi$

$y = r\cos\Theta$

$z = r\sin\Theta\sin\varphi$

$\Theta = \arccos(y / r)$

$\varphi = \arctan(z / x)$

In practice, $\varphi = \text{atan2}(z, x)$

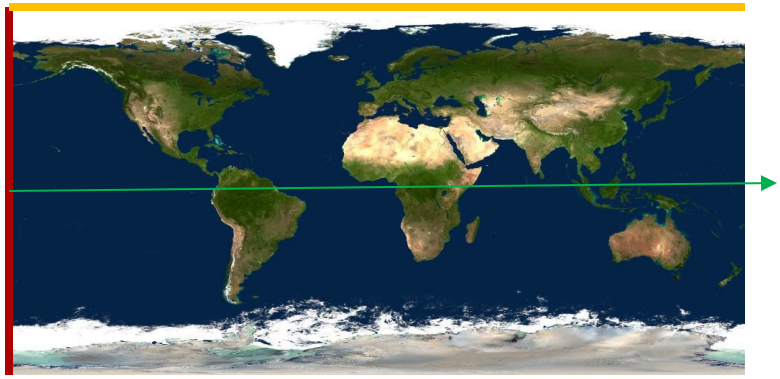
$(\Theta, \varphi) \in [0, \pi] \times [-\pi, \pi]$

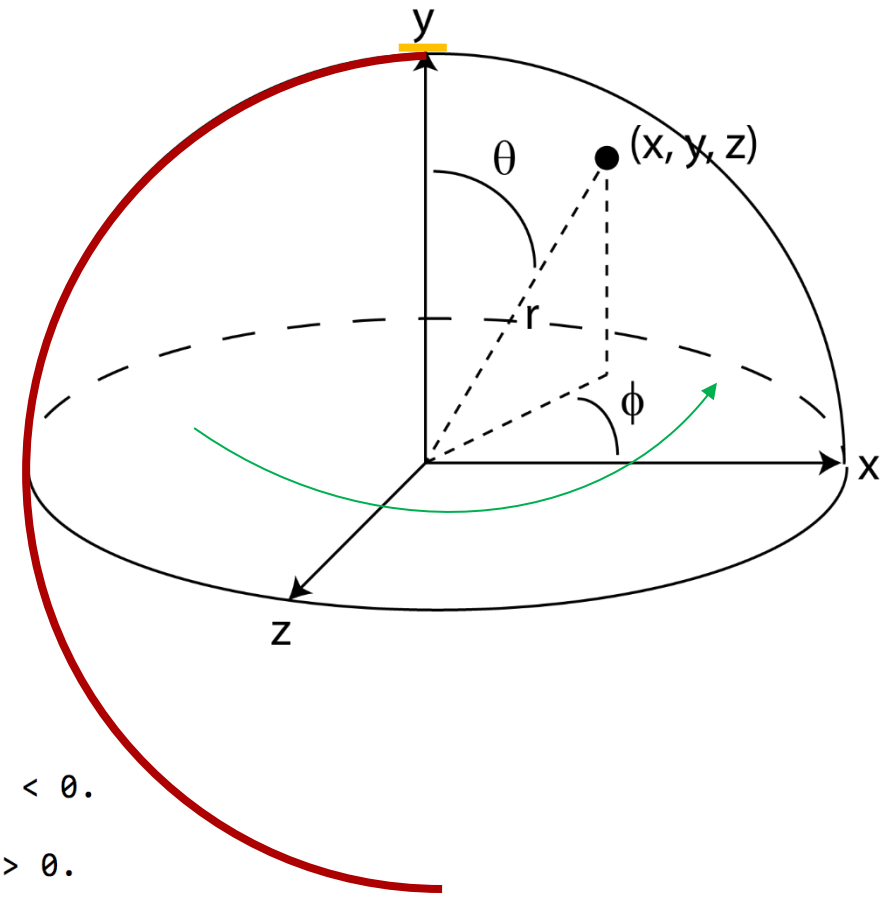
atan2(+−0, x) returns +−pi for x < 0.

atan2(+−0, x) returns +−0 for x > 0.

Texture Mapping

- Assume you want to wrap the image such that:





$u = (-\varphi + \pi) / (2\pi)$

$v = \Theta / \pi$

atan2(+−0, x) returns +−pi for x < 0.

atan2(+−0, x) returns +−0 for x > 0.

Texture Mapping

- **Step 3:** Compute a (u, v) value for every surface point. For instance, a sphere:

$$u = (-\varphi + \pi) / (2\pi)$$

$$v = \Theta / \pi$$

- **Step 4:** Find the texture image coordinate (i, j) at the given (u, v) coordinate:

$$i = u.n_x$$

$$j = v.n_y$$

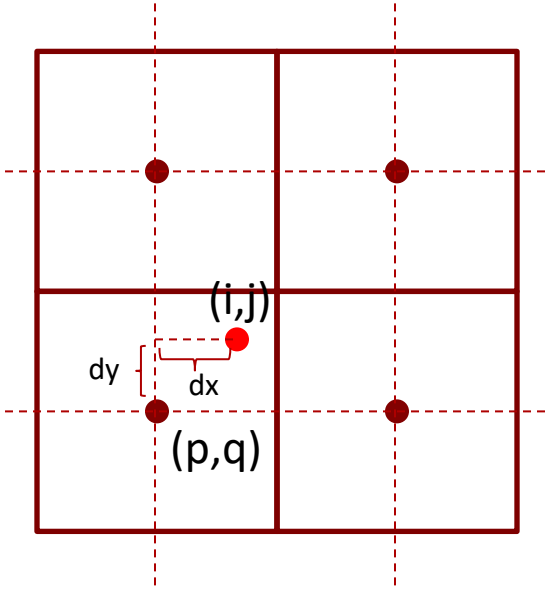
Note that i, j can be fractional!

n_x = texture image width
n_y = texture image height

Texture Mapping

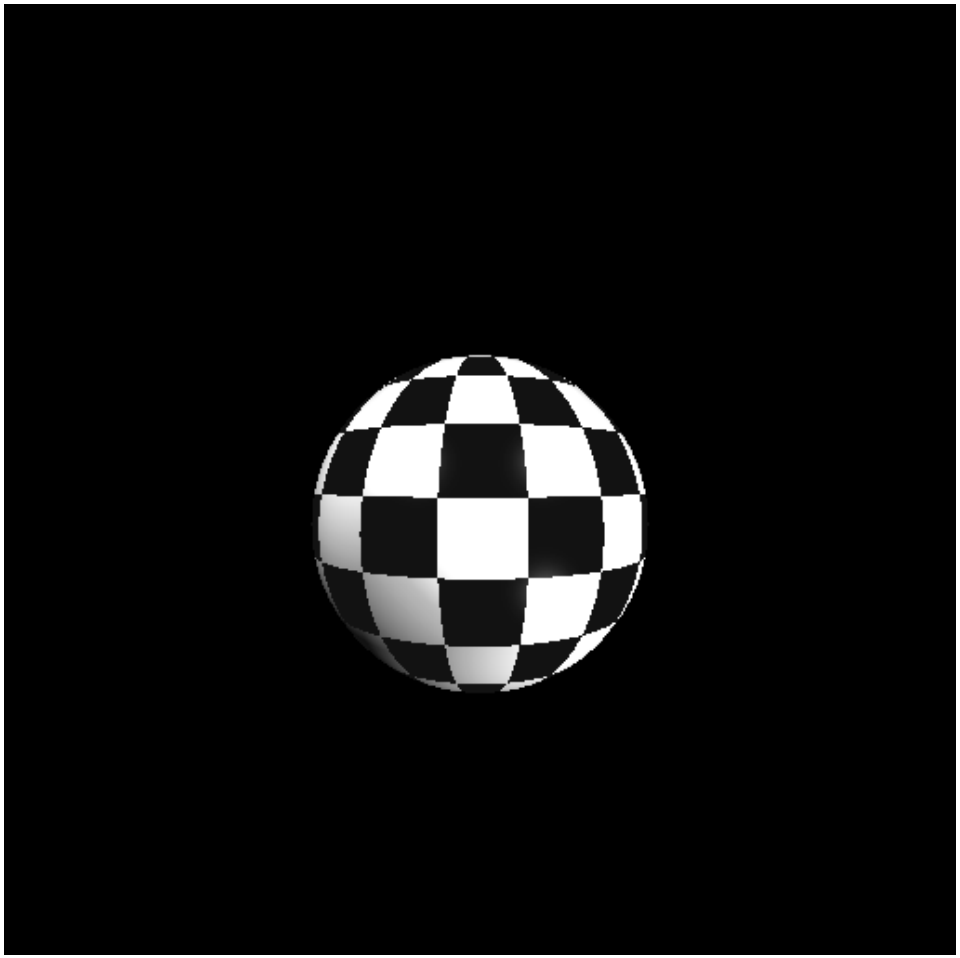
- **Step 5:** Choose the texel color using a suitable interpolation strategy
  - **Nearest Neighbor:** fetch texel at nearest coordinate (faster)

    Color(x, y, z) = fetch(**round(i, j)**)
  - **Bilinear Interpolation:** Average four closest neighbors (smoother):



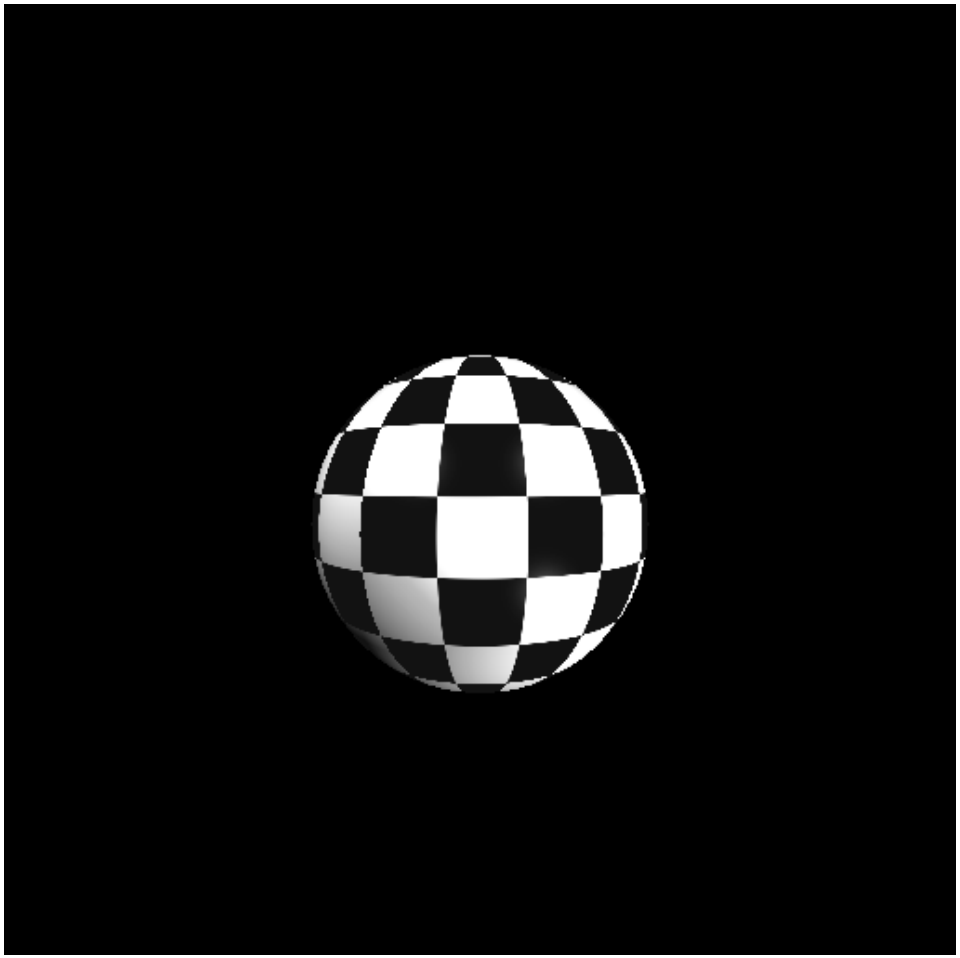
p = floor(i)
q = floor(j)
dx = i – p
dy = j - q
Color(x, y, z) = fetch(p, q).(1 – dx).(1 – dy) +
fetch(p+1, q).(dx).(1 – dy) +
fetch(p, q+1).(1 – dx).(dy) +
fetch(p+1, q+1).(dx).(dy)

NN vs Bilinear Interpolation



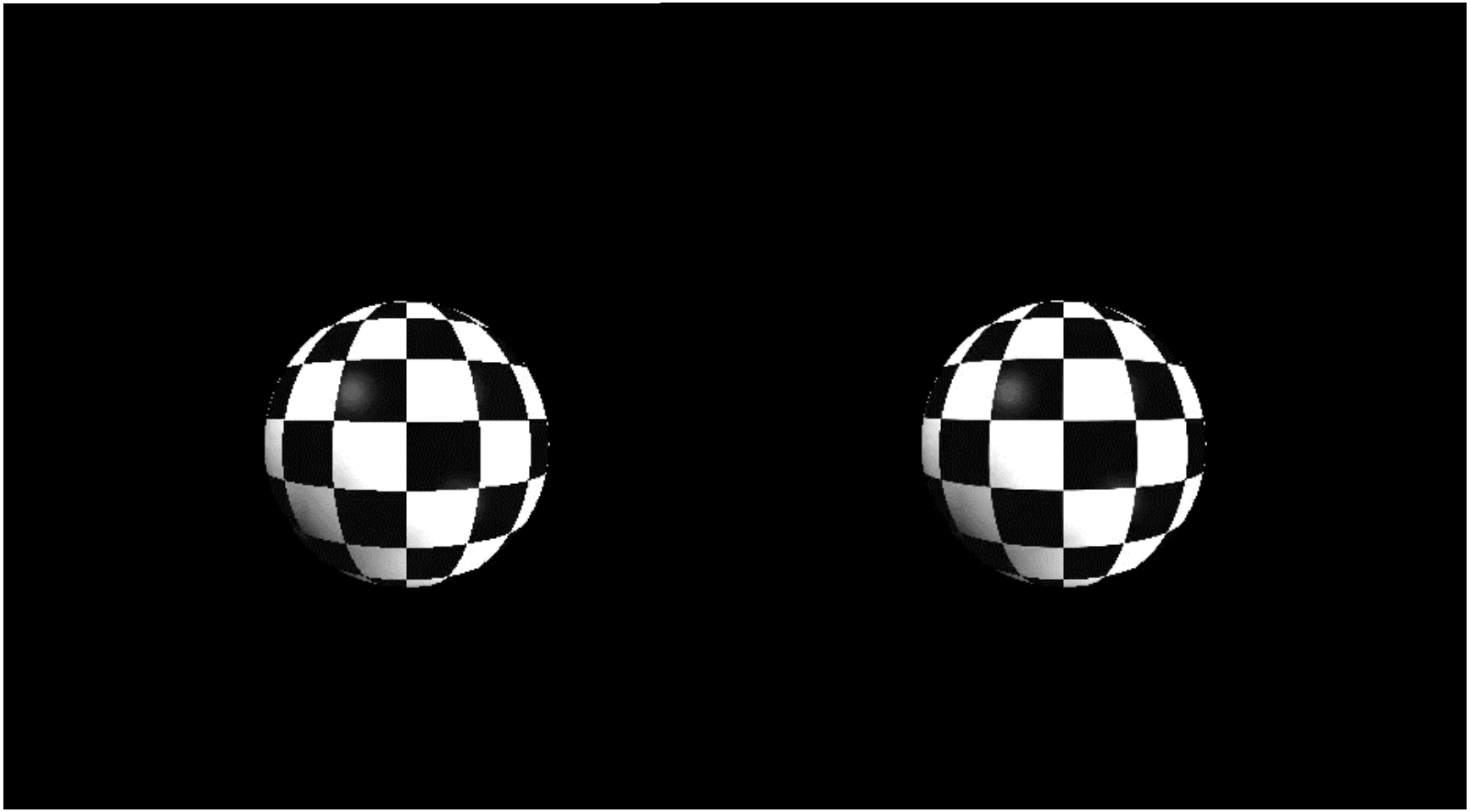
Nearest-neighbor

NN vs Bilinear Interpolation



Bilinear

NN vs Bilinear Interpolation

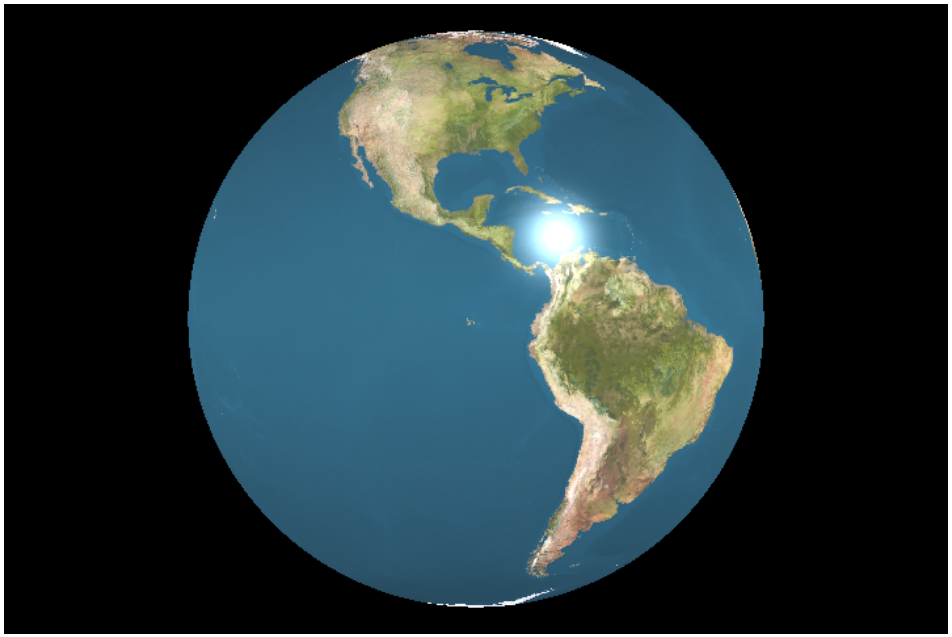


Nearest-neighbor

Bilinear

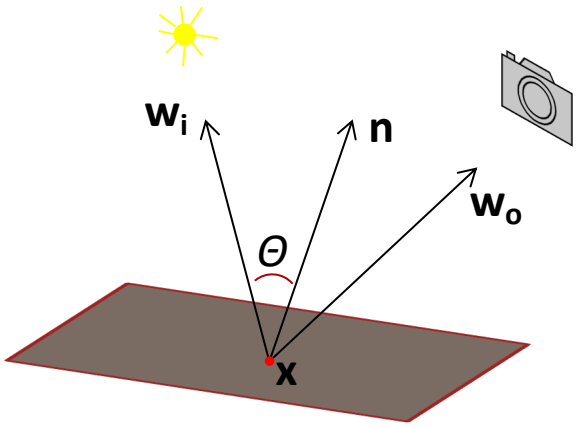
Texture Color

- Once we have the texture color, what to do with it?
- Several options possible: replace shading color, blend with shading color, replace k_d component, etc.



Texture Color

- For instance, to replace the k_d component:



$$\boldsymbol{L_o^d(x, wo) = k_d cos\theta' E_i(x, w_i)}$$

Outgoing radiance

Received irradiance

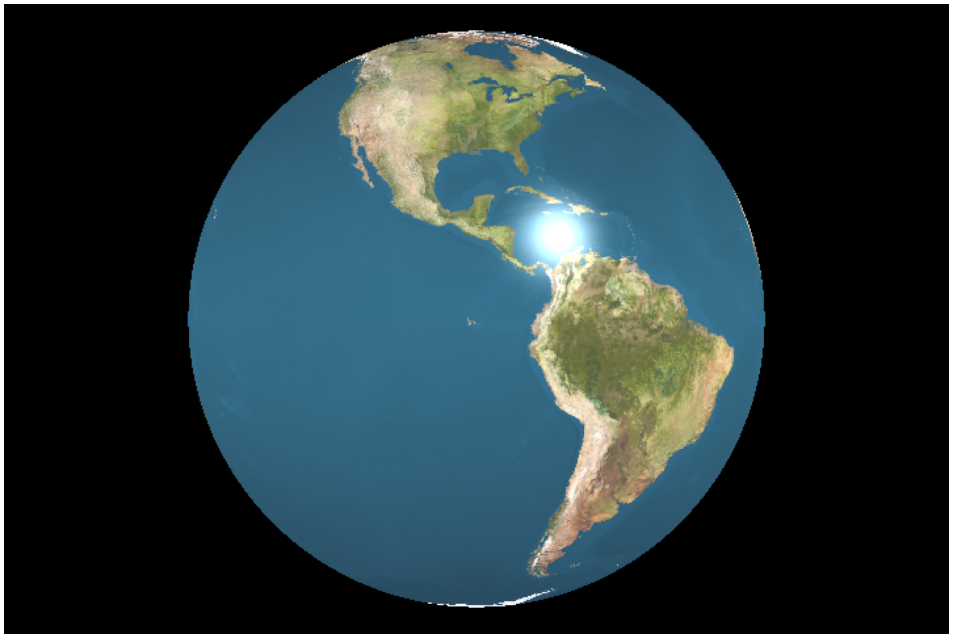
$$cos\theta' = \max(0, \boldsymbol{w_i . n})$$

where

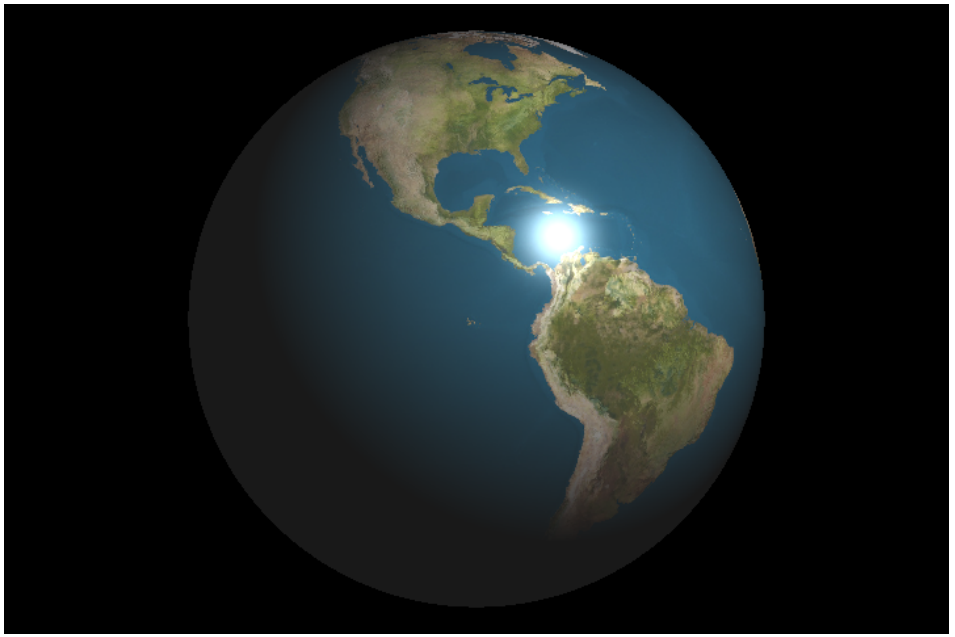
$\boldsymbol{k_d}$ = textureColor / 255

Texture Color

- Replacing k_d typically improves realism over replacing the entire diffuse color:



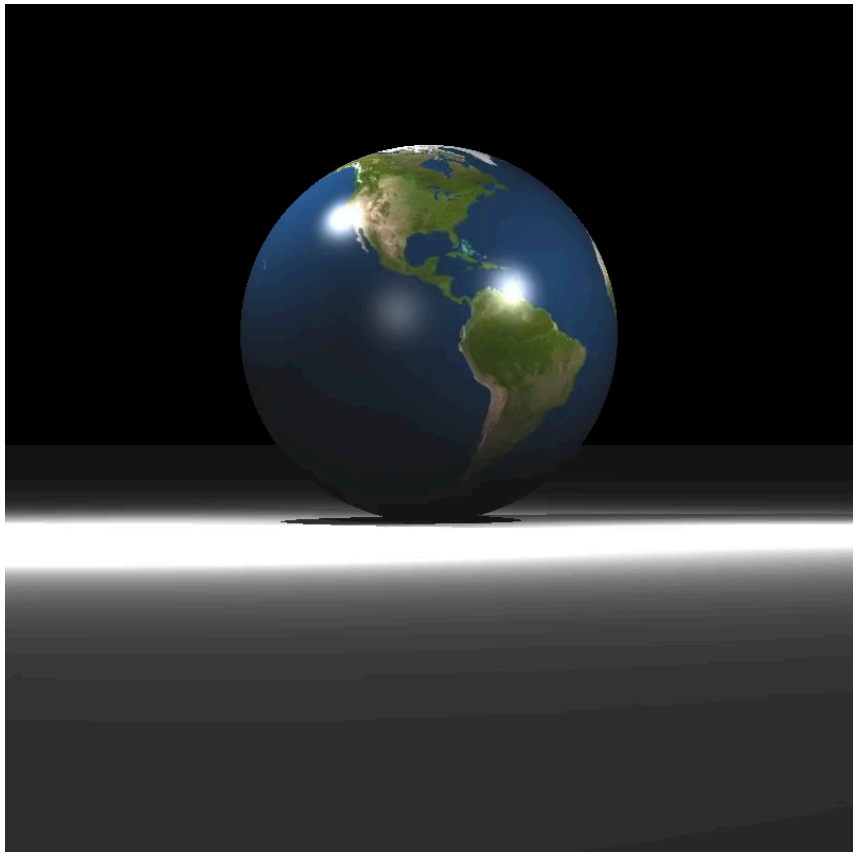
Diffuse color replaced



k_d component replaced

Animation

- Animation, shading, and texture mapping enhances realism:



Texture Mapping Triangles

- First, we must associate (u, v) coordinates for each vertex
- The (u, v) coordinates inside the triangle can be found using the barycentric coordinates
- Recall that the position of point p at barycentric coordinates (β, γ) is equal to:

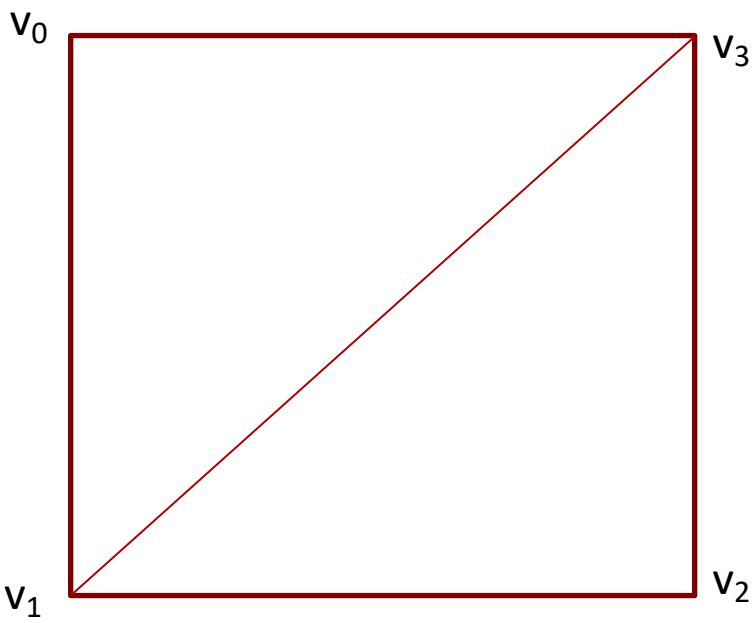
$$p(\beta, \gamma) = a + \beta(b - a) + \gamma(c - a)$$

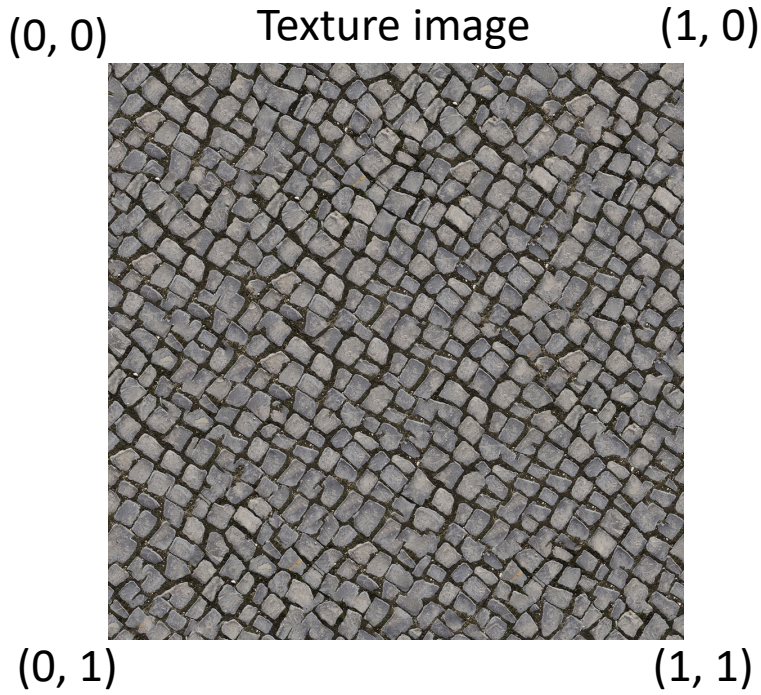
- Texture coordinates can be interpolated similarly:

$$u(\beta, \gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a)$$
$$v(\beta, \gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)$$

Texture Mapping Triangles

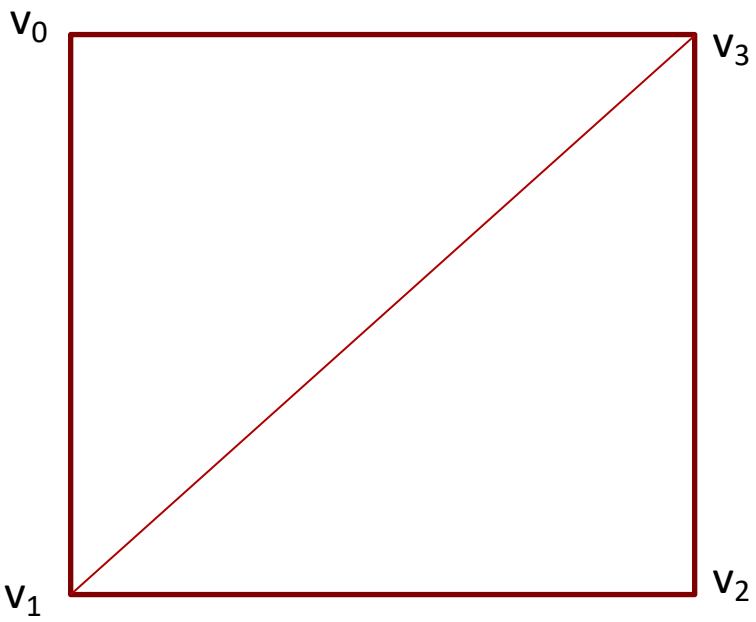
- Assume we want the texture map a rectangle made of two triangles:
  - v_3, v_1, v_2
  - v_1, v_3, v_0

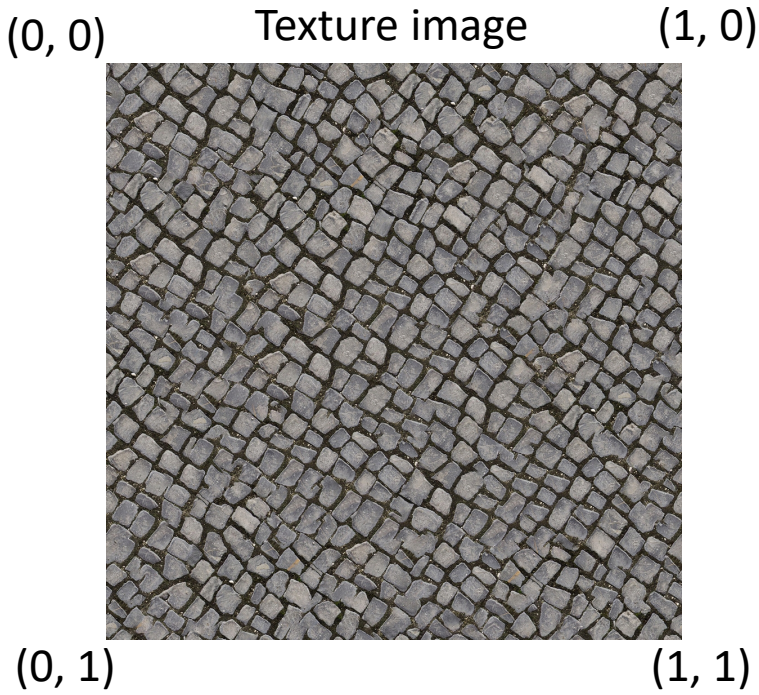




Texture Mapping Triangles

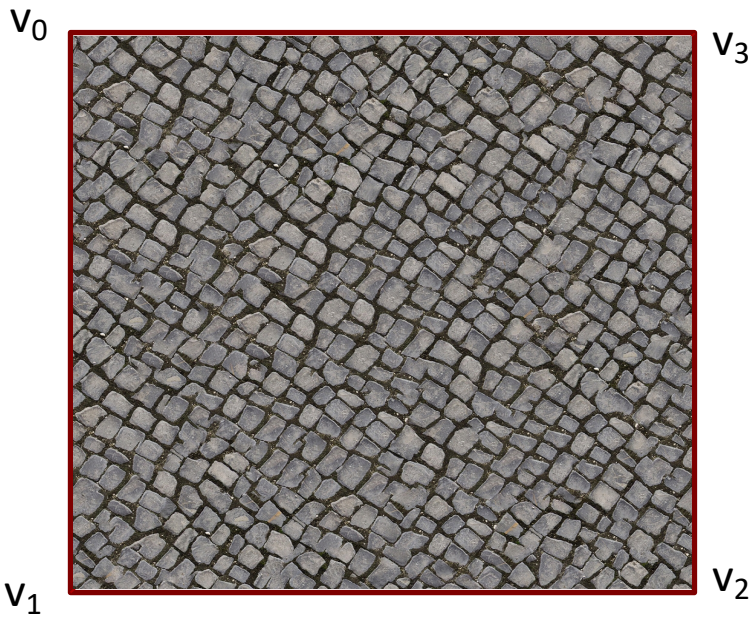
- We associate a texture coordinate with each vertex of each triangle:
  - v_3, v_1, v_2: (1, 0), (0, 1), (1, 1)
  - v_1, v_3, v_0: (0, 1), (1, 0), (0, 0)





Texture Mapping Triangles

- This way, the texture image gets stretched to match the triangles' positions
  - v_3, v_1, v_2: (1, 0), (0, 1), (1, 1)
  - v_1, v_3, v_0: (0, 1), (1, 0), (0, 0)



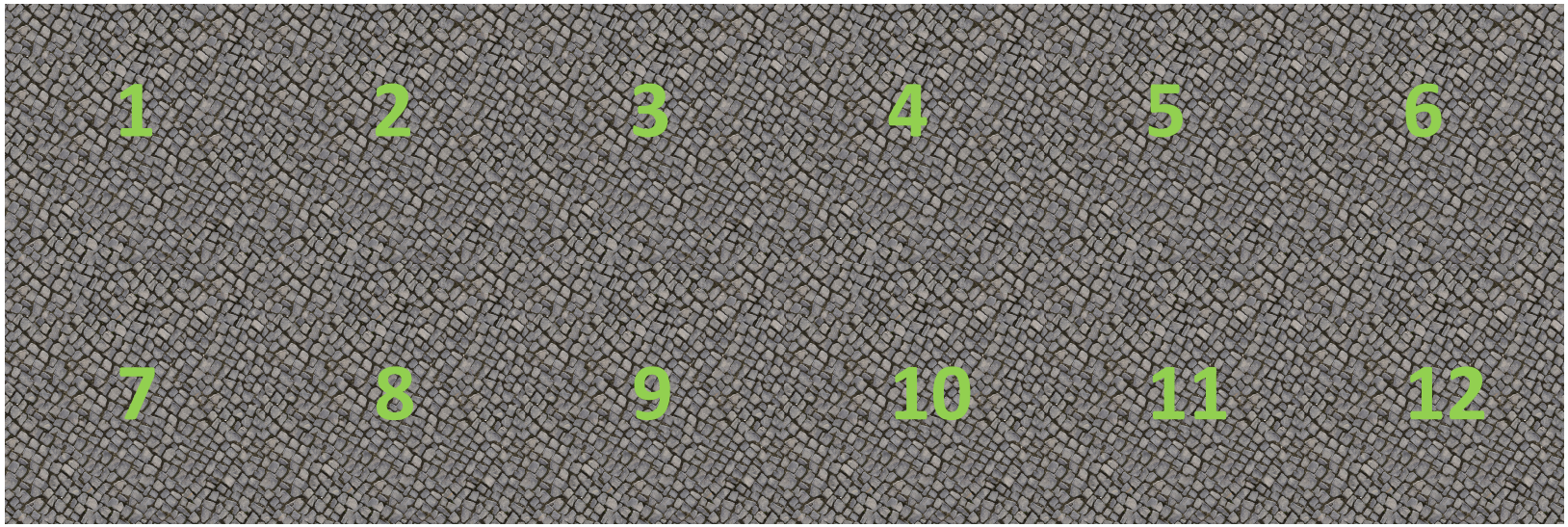
Combined Result

Tiling

- For large surfaces, stretching may excessively distort the texture image making it unrealistic

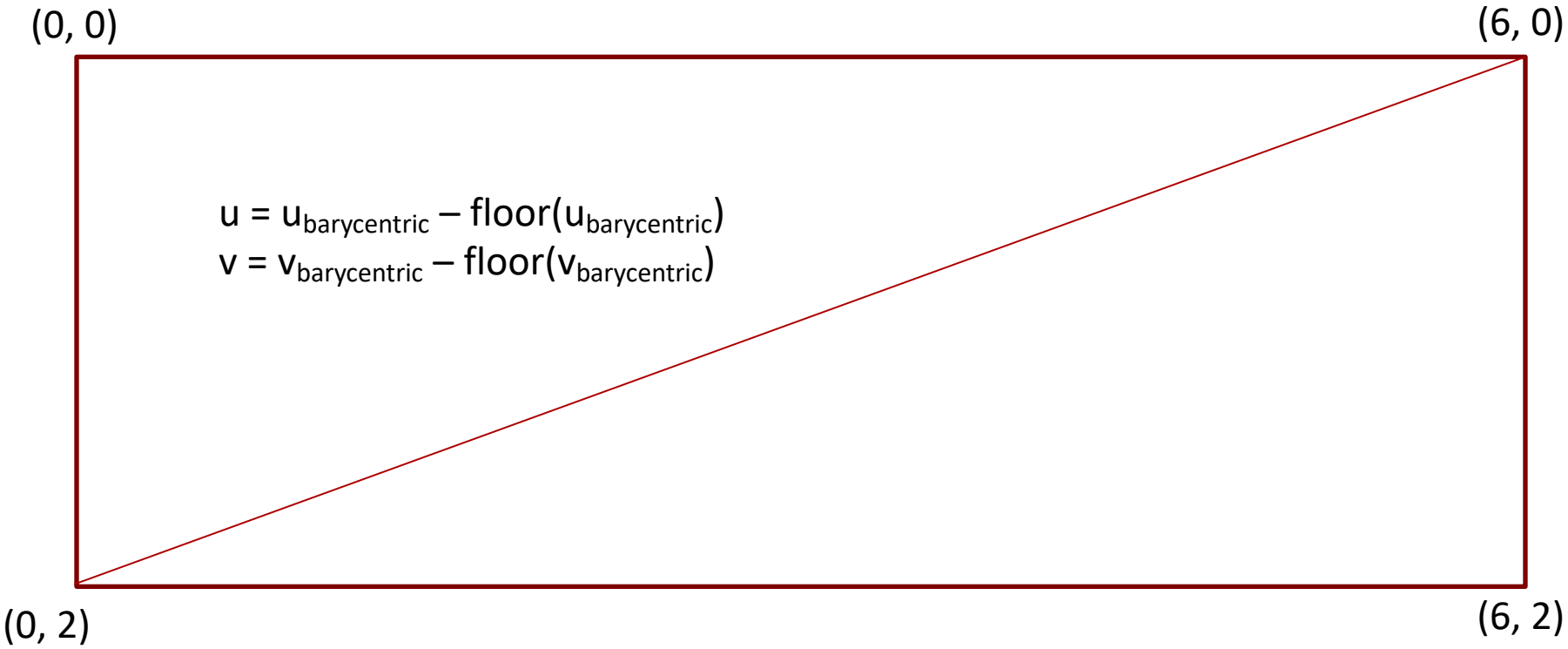
Tiling

- Tiling is the process of repeating a texture instead of stretching
- Multiple copies of the texture fill the surface
- Tiling may also look unrealistic if repetition is clear



Tiling

- To support tiling, tex. coords. are not limited to [0, 1] range



Textures Atlases

- Most objects in CG will have a texture applied to it
- This requires using multiple textures, sometimes thousands or more, for a moderately complex environment
- Many times text is also retrieved from texture images
- Instead of using multiple texture, a texture atlas combines textures of many objects into a single image

Textures Atlases

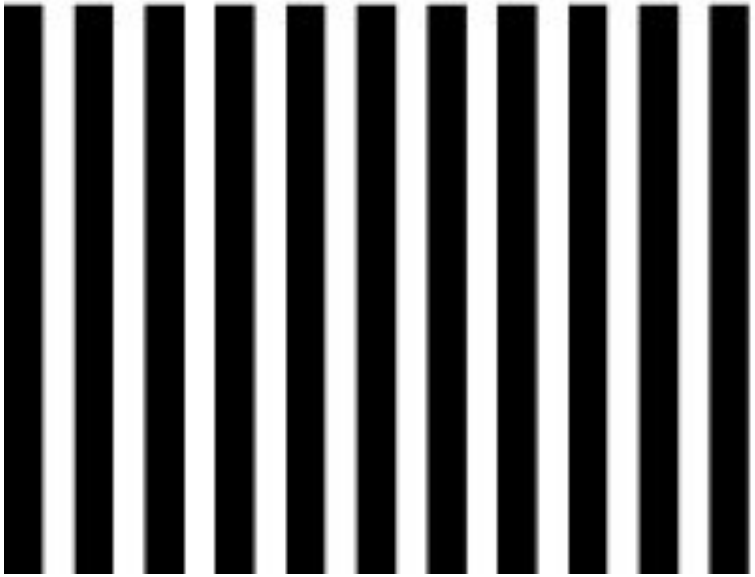
http://jeapostrophe.github.io/2013-05-06-texture--post.html

Procedural Textures

- The previous approach requires us to use an image as the texture to be mapped
- We can also achieve spatial variation without using any texture image at all
- **Solution:**
  - Generate textures procedurally for each surface point
- **Problem:**
  - How to make them look natural?

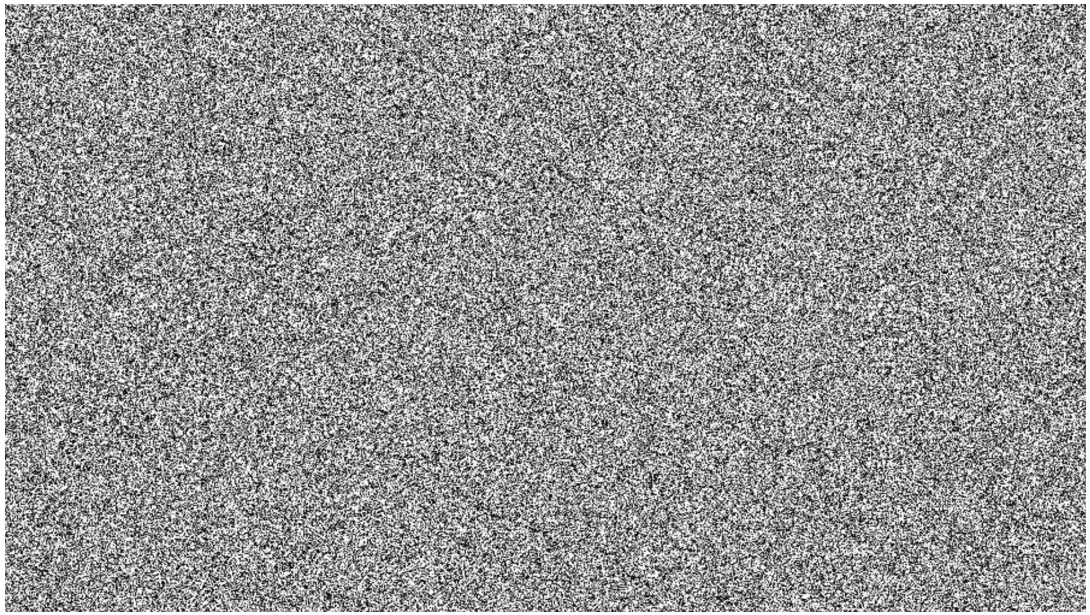
Procedural Textures

- Periodic patterns are easy to generate but they do not look natural (e.g. evaluate a sine function at every point):



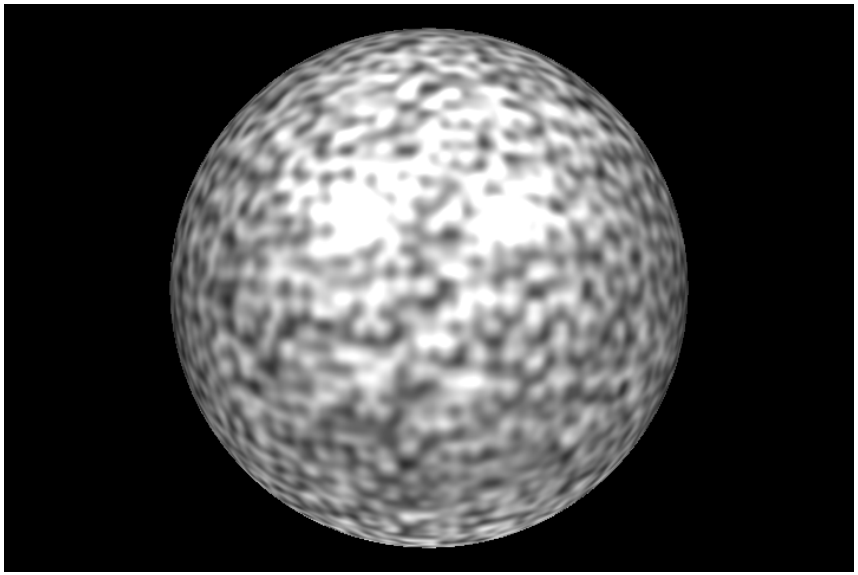
Procedural Textures

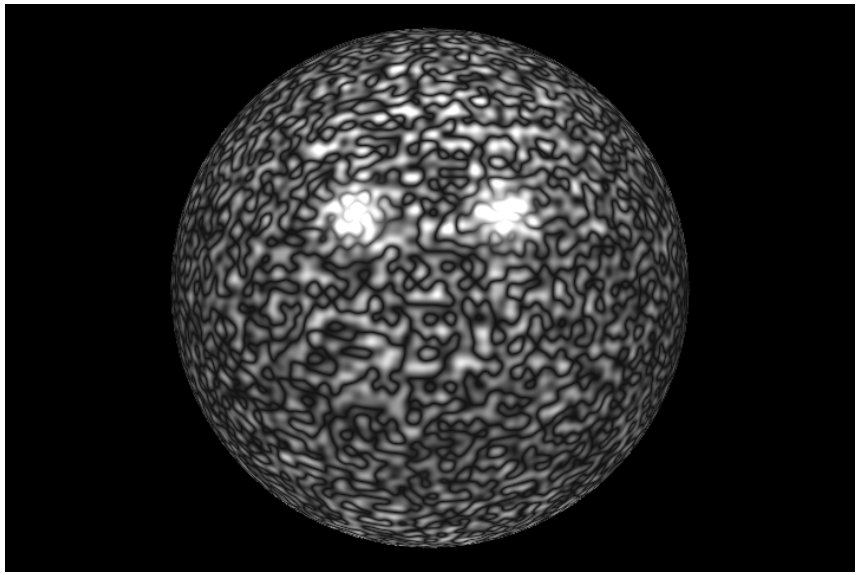
- We need a function that returns a random value for each (x, y, z) point in space
- However, too random is not good as it will appear as noise:



Procedural Textures

- There are various techniques such as Perlin noise that allow on-the-fly computation of controlled random patterns





Procedural Textures

- Image and procedural textures can be used together: