Texture Mapping

Texture mapping

- Objects have properties that vary across the surface

Texture Mapping

- So we make the shading parameters vary across the surface

[Foley et al. / Perlin]

Texture mapping

- Adds visual complexity; makes appealing images

[Pixar / *Toy Story*]

Texture mapping

- Color is not the same everywhere on a surface
  - one solution: multiple primitives
- Want a function that assigns a color to each point
  - the surface is a 2D domain, so that is essentially an image
  - can represent using any image representation
  - raster texture images are very popular

A definition

Texture mapping: a technique of defining surface properties (especially shading parameters) in such a way that they vary as a function of position on the surface.

- This is very simple!
  - but it produces complex-looking effects

Examples

- Wood gym floor with smooth finish
  - diffuse color k_D varies with position
  - specular properties k_S, n are constant
- Glazed pot with finger prints
  - diffuse and specular colors k_D, k_S are constant
  - specular exponent n varies with position
- Adding dirt to painted surfaces
- Simulating stone, fabric, …
  - to approximate effects of small-scale geometry
    - they look flat but are a lot better than nothing

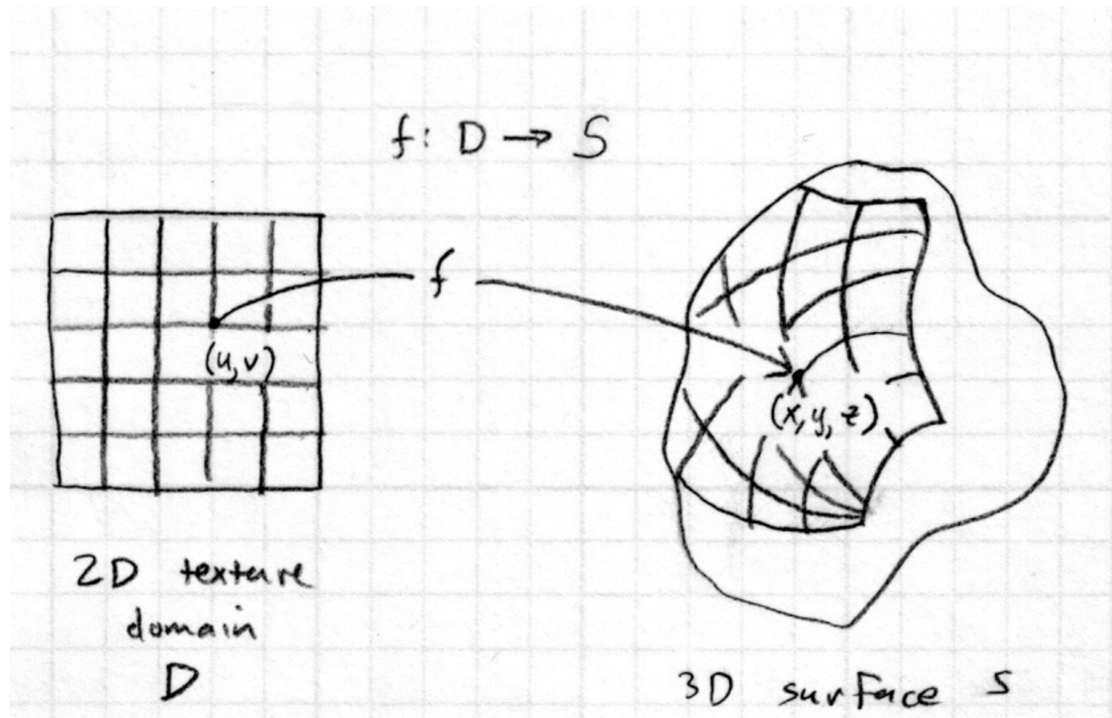
Mapping textures to surfaces

- Usually the texture is an image (function of *u*, v)
  - the big question of texture mapping: where on the surface does the image go?
  - obvious only for a flat rectangle the same shape as the image
  - otherwise more interesting
- Note that *3D textures* also exist
  - texture is a function of (*u*, *v*, *w*)
  - can just evaluate texture at 3D surface point
  - good for solid materials
  - often defined procedurally

[Wolfe / SG97 Slide set]

Mapping textures to surfaces

- "Putting the image on the surface"
  - this means we need a function *f* that tells where each point on the image goes
  - this looks a lot like a parametric surface function
  - for parametric surfaces you get *f* for free

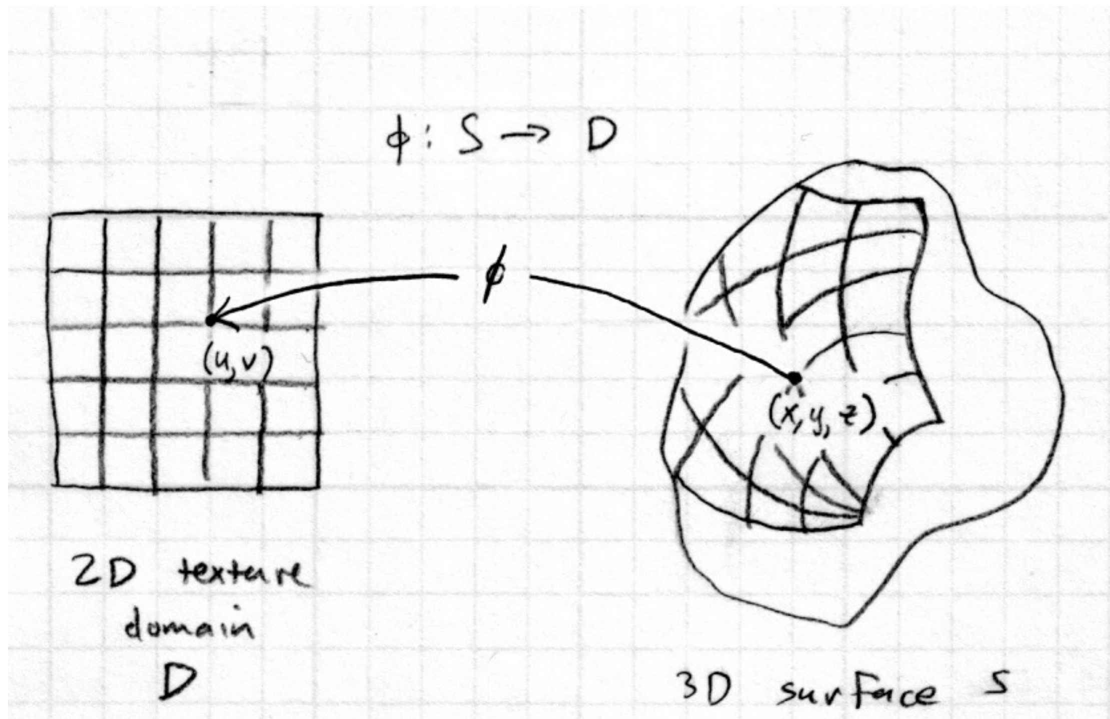


Texture coordinate functions

- Non-parametrically defined surfaces: more to do
  - can't assign texture coordinates as we generate the surface
  - need to have the *inverse* of the function *f*
- Texture coordinate fn.

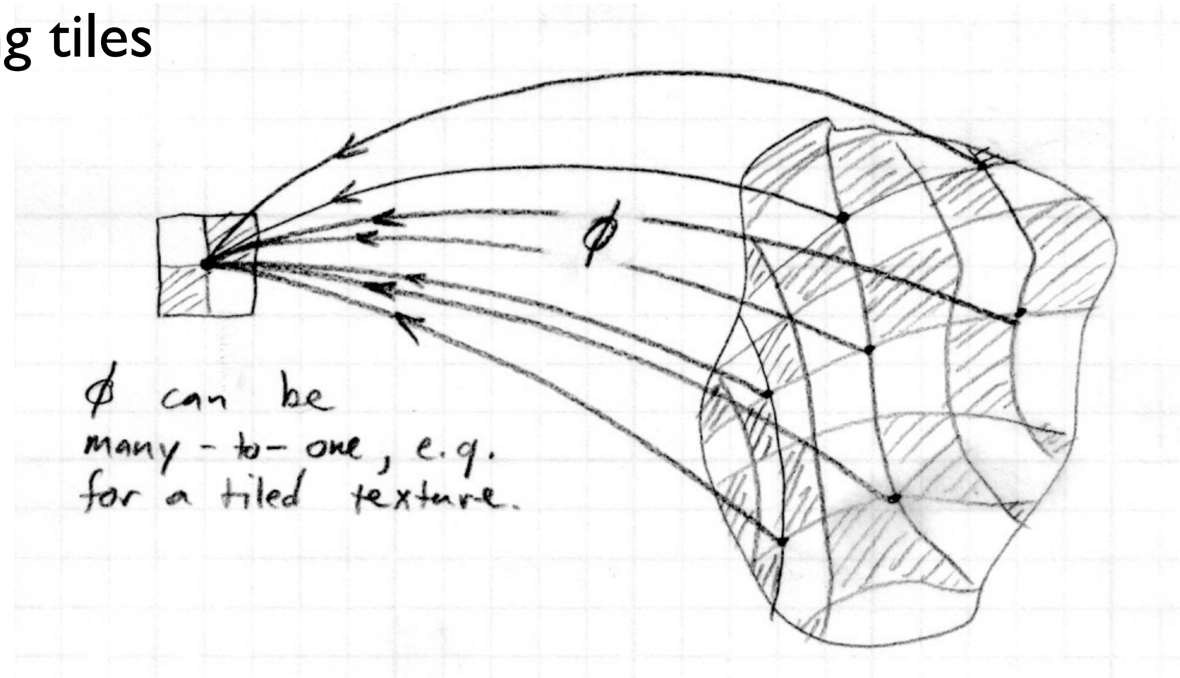
  $$\phi : S \to \mathbb{R}^2$$

  - for a vtx. at **p** get texture at φ(**p**)



Texture coordinate functions

- Mapping from *S* to *D* can be many-to-one
  - that is, every surface point gets only one color assigned
  - but it is OK (and in fact useful) for multiple surface points to be mapped to the same texture point
    - e.g. repeating tiles



Texture coordinate functions

- Define texture image as a function

$$T : D \to C$$

  – where *C* is the set of colors for the diffuse component

- Diffuse color (for example) at point **p** is then

$$k_D(\mathbf{p}) = T(\phi(\mathbf{p}))$$

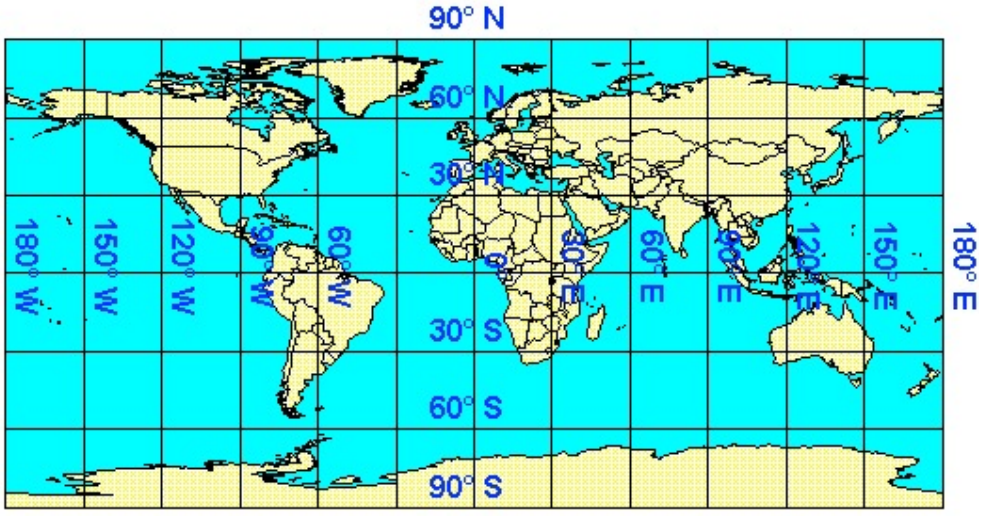
Examples of coordinate functions

- A rectangle
  - image can be mapped directly, unchanged

Examples of coordinate functions

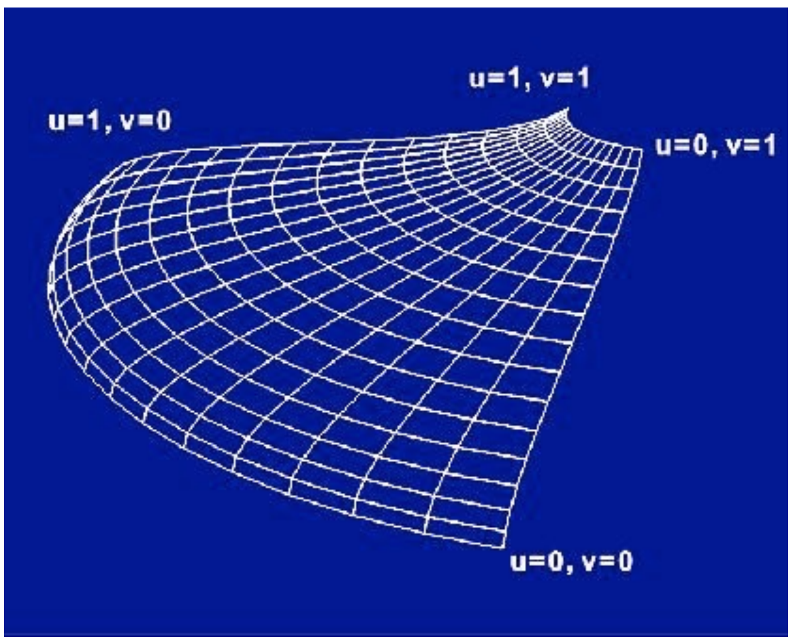
- For a sphere: latitude-longitude coordinates
  - φ maps point to its latitude and longitude

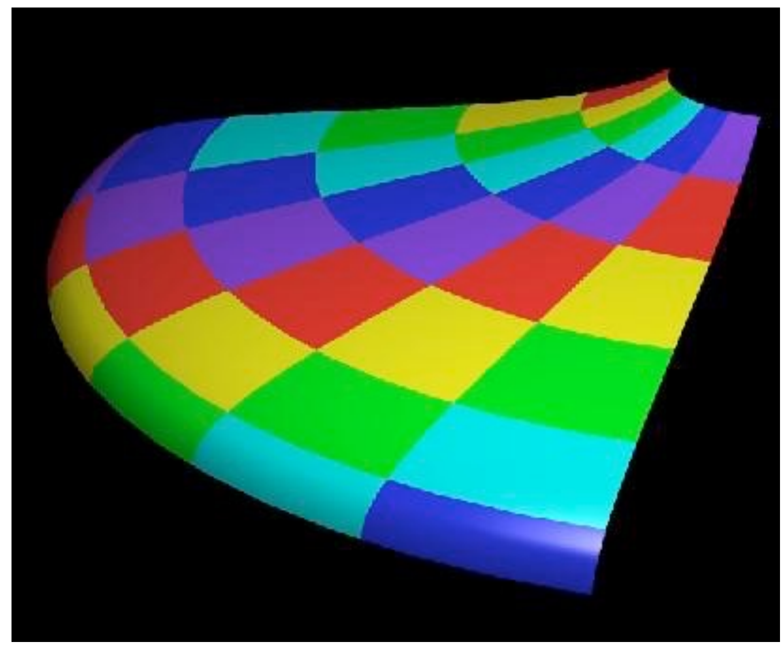
[map: Peter H. Dana]



Examples of coordinate functions

- A parametric surface (e.g. spline patch)
  - surface parameterization gives mapping function directly (well, the inverse of the parameterization)

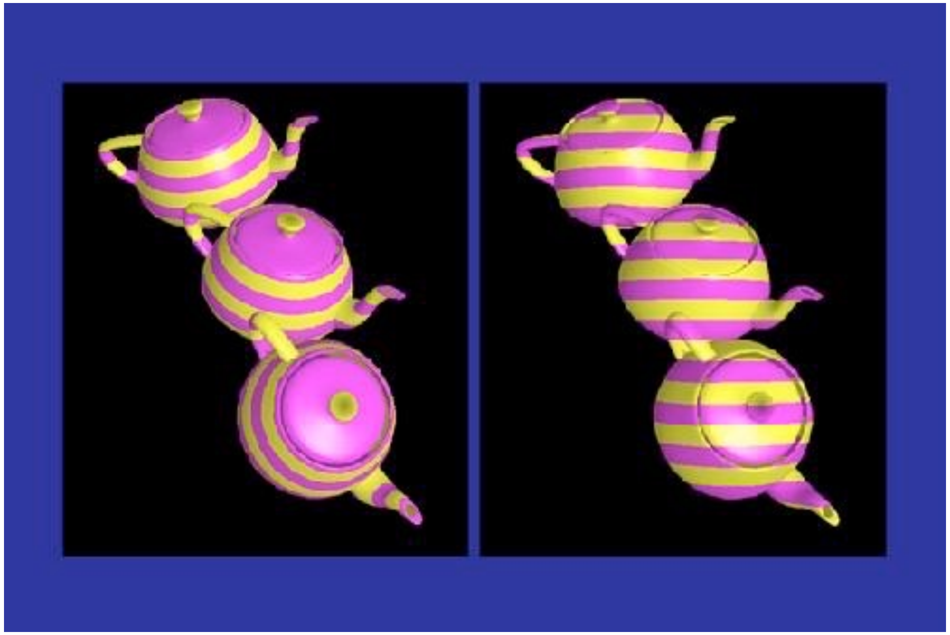




[Wolfe / SG97 Slide set]

Examples of coordinate functions

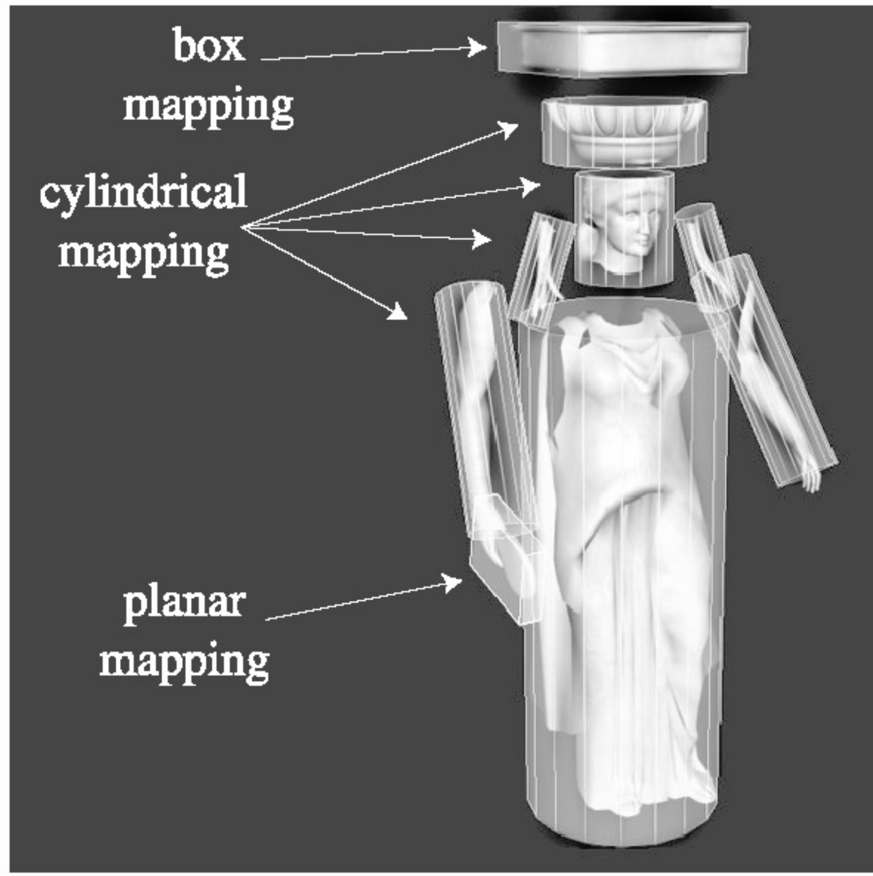
- For non-parametric surfaces it is trickier
  - directly use world coordinates
    - need to project one out



[Wolfe / SG97 Slide set]

Examples of coordinate functions

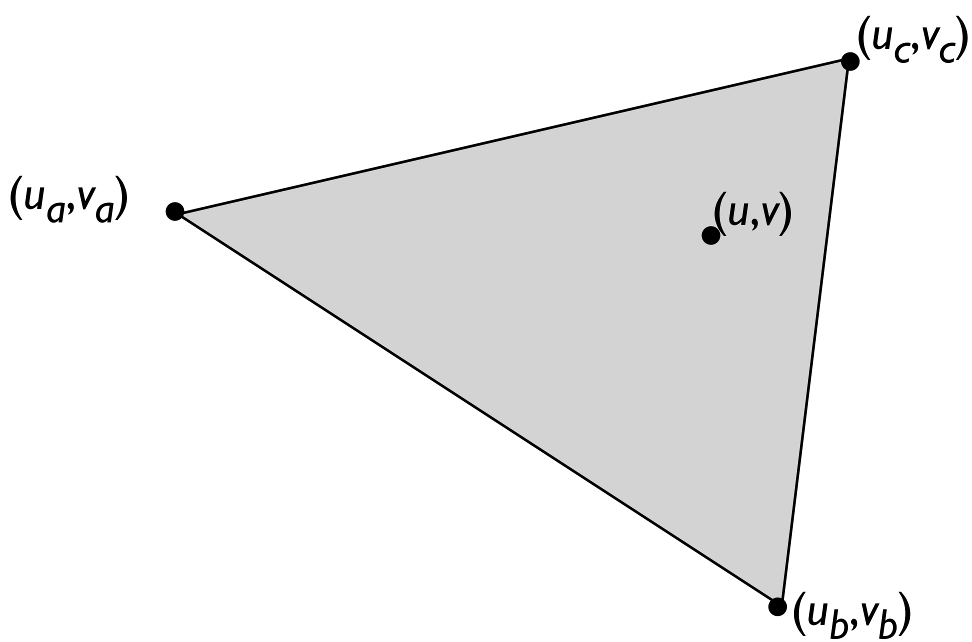
- Non-parametric surfaces: project to parametric surface



[Moller & Haines 2002]

Examples of coordinate functions

- Triangles
  - specify (u,v) for each vertex
  - define (u,v) for interior by linear interpolation



Interpolation Across Triangles: Barycentric Coordinates

Interpolation Across Triangles

Why do we want to interpolate?

- Specify values **at vertices**
- Obtain smoothly varying values **across triangles**

What do we want to interpolate?

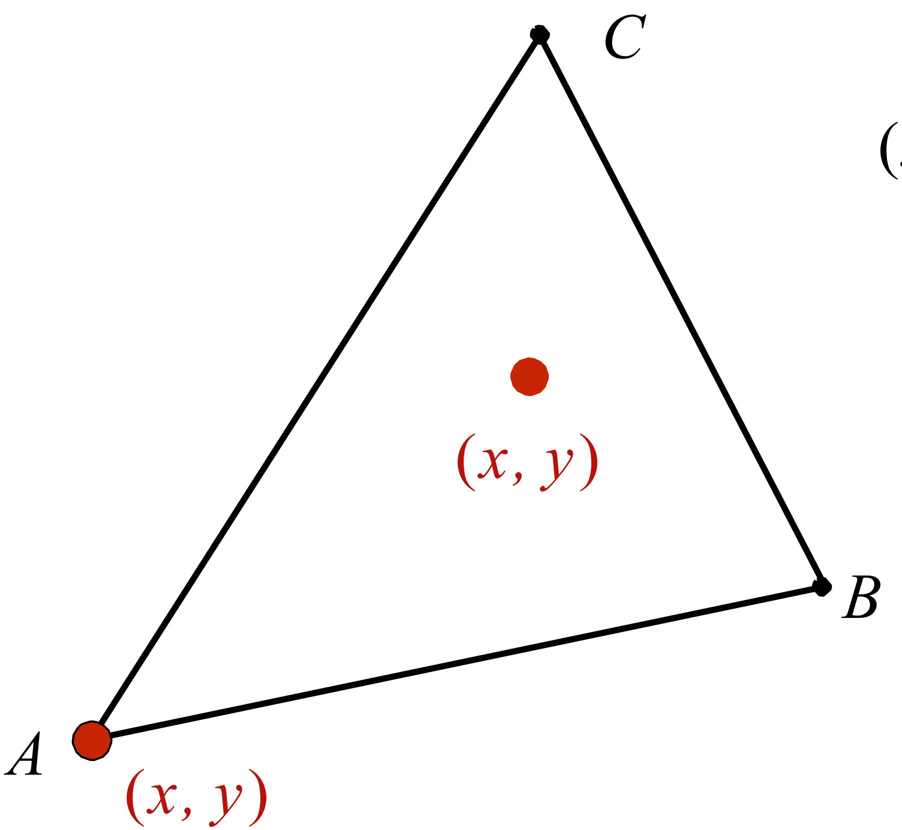
- Texture coordinates, colors, normal vectors, …

How do we interpolate?

- **Barycentric coordinates**

Barycentric Coordinates

A coordinate system for triangles $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$



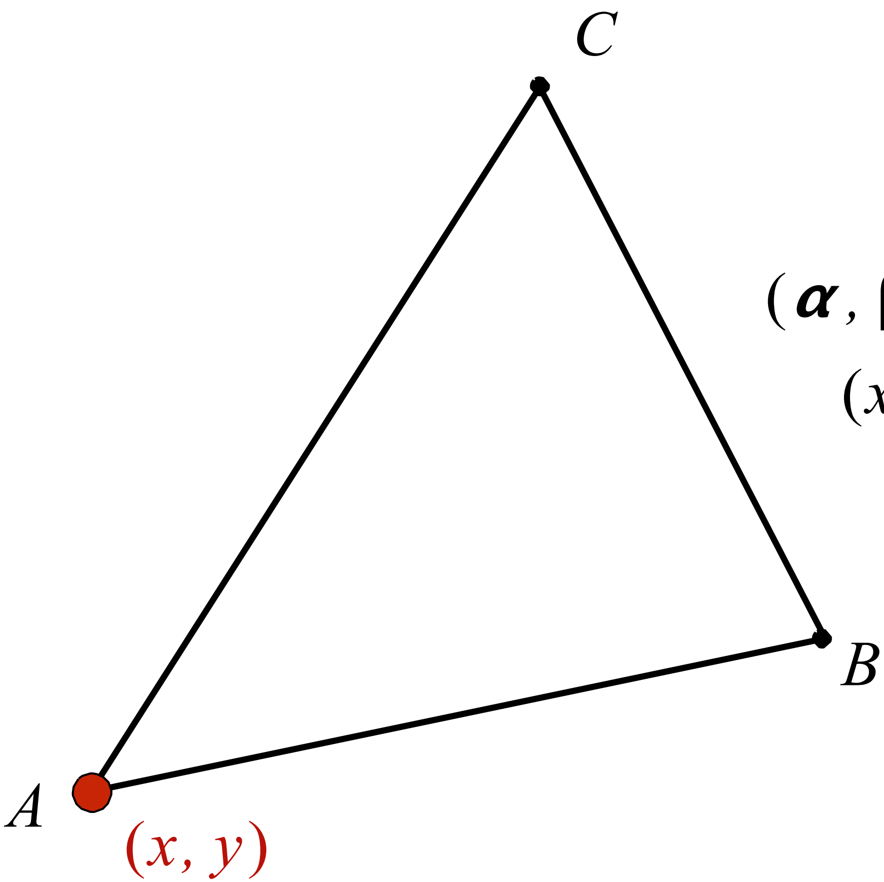
$$(x, y) = \boldsymbol{\alpha} A + \boldsymbol{\beta} B + \boldsymbol{\gamma} C$$

$$\boldsymbol{\alpha} + \boldsymbol{\beta} + \boldsymbol{\gamma} = 1$$

Inside the triangle if all three coordinates are non-negative

Barycentric Coordinates

- What's the barycentric coordinate of A?

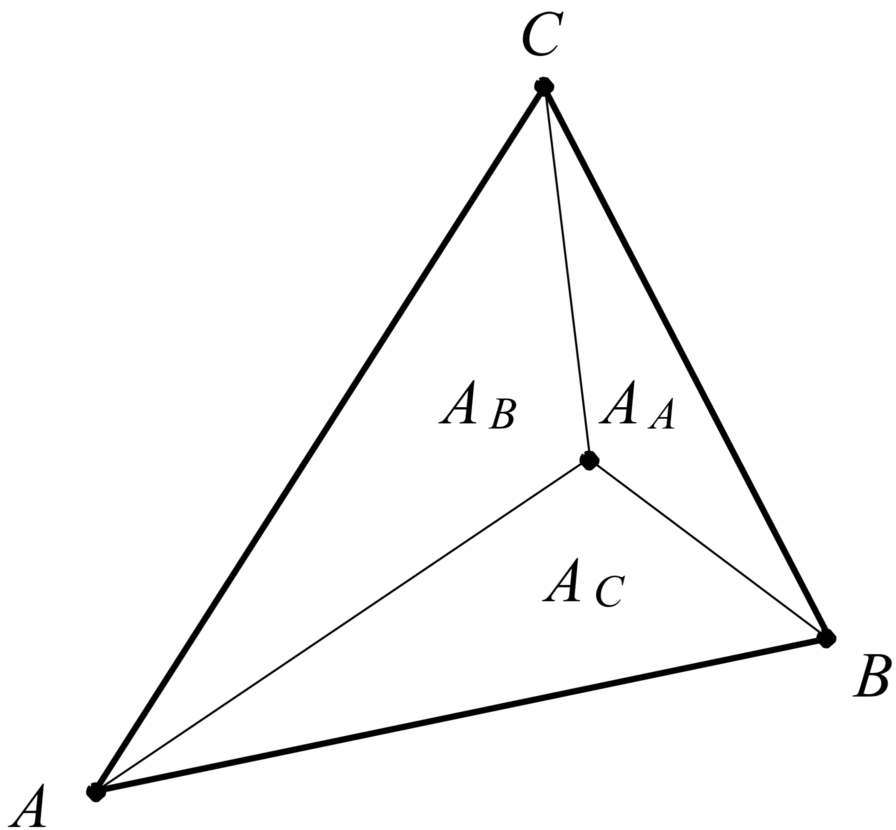


$$(\alpha, \beta, \gamma) = (1, 0, 0)$$

$$\begin{aligned}(x, y) &= \alpha A + \beta B + \gamma C \\ &= A\end{aligned}$$

Barycentric Coordinates

Geometric viewpoint — proportional areas



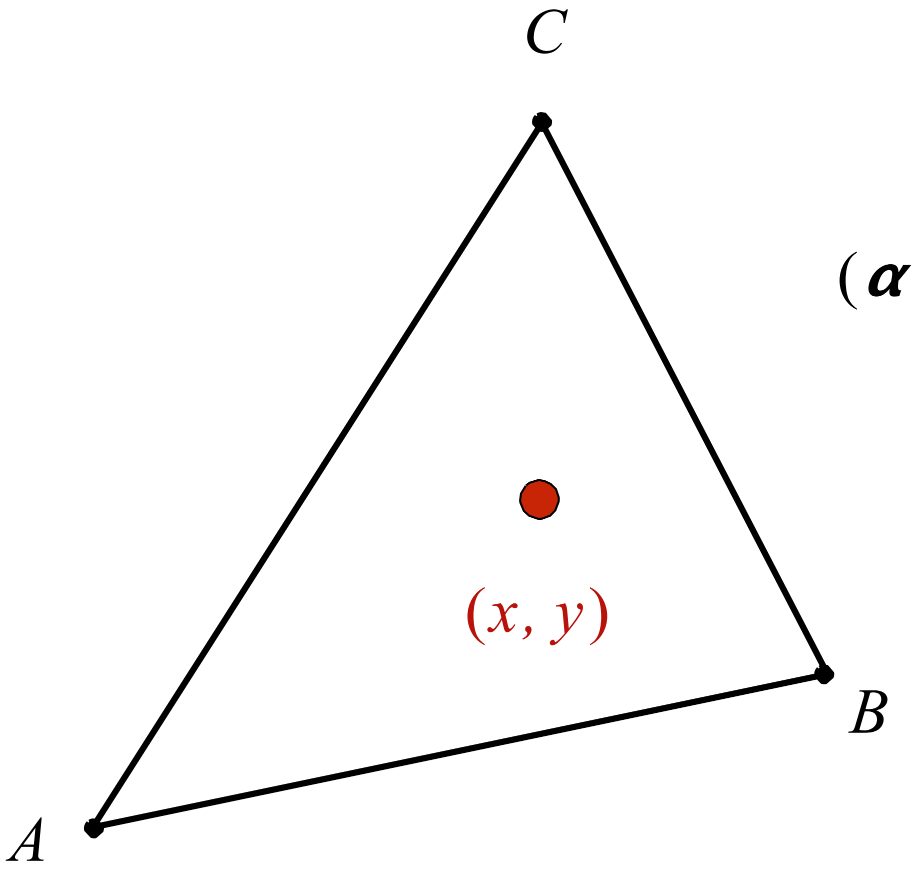
$$\alpha = \frac{A_A}{A_A + A_B + A_C}$$

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

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

Barycentric Coordinates

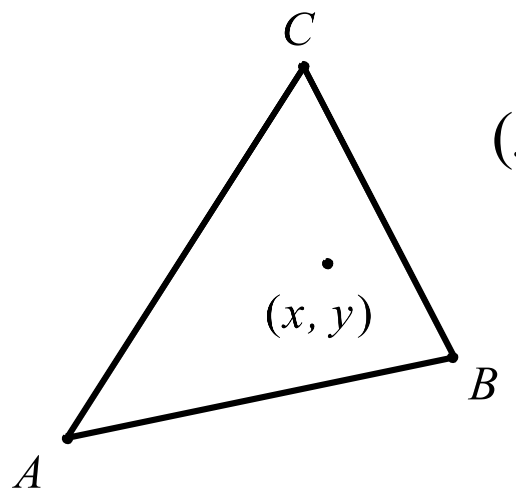
What's the barycentric coordinate of the centroid?



$$(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$$

$$(x, y) = \frac{1}{3} A + \frac{1}{3} B + \frac{1}{3} C$$

Barycentric Coordinates: Formulas



$$(x, y) = \alpha A + \beta B + \gamma C$$

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

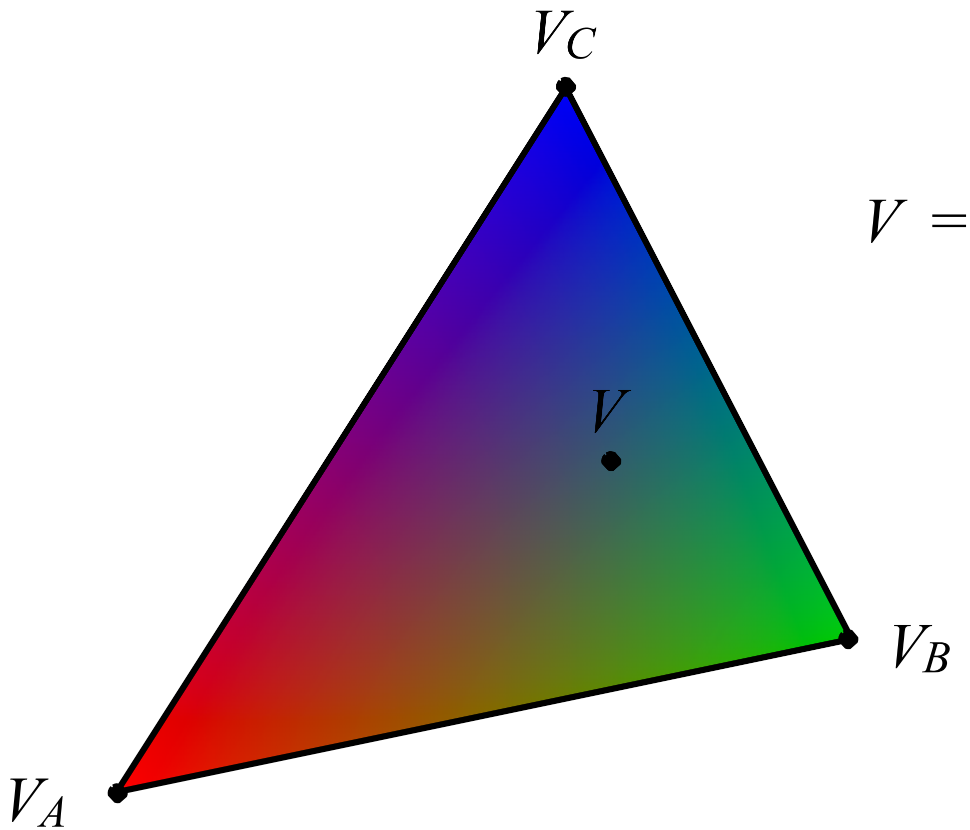
$$\alpha = \frac{-(x - x_B)(y_C - y_B) + (y - y_B)(x_C - x_B)}{-(x_A - x_B)(y_C - y_B) + (y_A - y_B)(x_C - x_B)}$$

$$\beta = \frac{-(x - x_C)(y_A - y_C) + (y - y_C)(x_A - x_C)}{-(x_B - x_C)(y_A - y_C) + (y_B - y_C)(x_A - x_C)}$$

$$\gamma = 1 - \alpha - \beta$$

Using Barycentric Coordinates

Linearly interpolate values at vertices



$$V = \boldsymbol{\alpha} V_A + \boldsymbol{\beta} V_B + \boldsymbol{\gamma} V_C$$

V_A, V_B, V_C can be positions, texture coordinates, color, normal, depth, material attributes…

However, barycentric coordinates are not invariant under projection!

Applying Textures

Simple Texture Mapping: Diffuse Color

Usually a pixel's center

for each rasterized screen sample (x,y)

(u,v) = evaluate texture coordinate at (x,y)

texcolor = texture.sample(u,v);

set sample's color to texcolor;

Using barycentric coordinates!

Usually the diffuse albedo Kd
(recall the Blinn-Phong reflectance model)