Image Texture Fundamentals

Texturing

Allows higher-frequency color variation

- Not just interpolated from vertex colors

May be 2D (surface-based) or 3D (volume-based)

May be strictly image-based or procedural

- Today we’ll talk about simple image-based

2D Texture Mapping

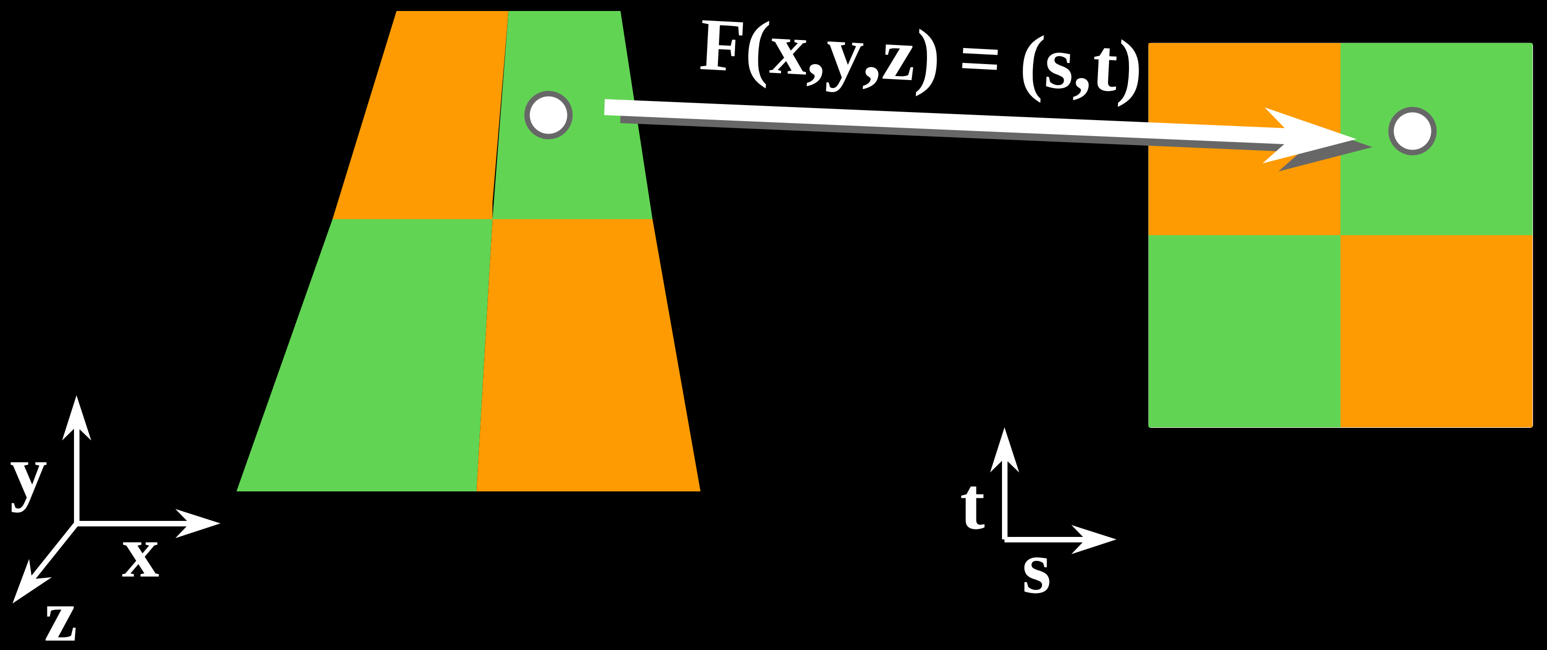
Requires surface parameterization

- Mapping from 3D surface to 2D parametric domain

Colors defined in 2D parameter space

Parameterization (texture coordinates) used to determine material color at point on surface

2D Texture Diagram



2D Texture Applications

Most useful for colors that are sitting on the surface, rather than running through the material

- Pictures on the wall
- Printed/painted logos, text, etc.
- Fake wood grain

Other Types of 2D Maps

Bump/normal maps

- Modify or define surface normals

Displacement maps

- Modify surface itself

Environment/reflection maps

- Define environment seen in specular reflections

3D Texture Maps

Colors defined in 3D space

3D coordinates of surface used for mapping

Usually convenient to define 3D texture in object space

3D Texture Applications

More like carving object out of material than pasting a picture on the surface

- **wood, marble, etc.**
- **clouds, fog, fire (hypertextures, using additional density information)**

Image-based Texture Mapping (2D)

2D texel array (image) determines colors in texture domain

Given texture coordinates on surface, look up color in image

Lookup may be return nearest texel (*point sampled*) or bilinear interpolation of 4 surrounding texels

Acquiring Texture Images

Photograph

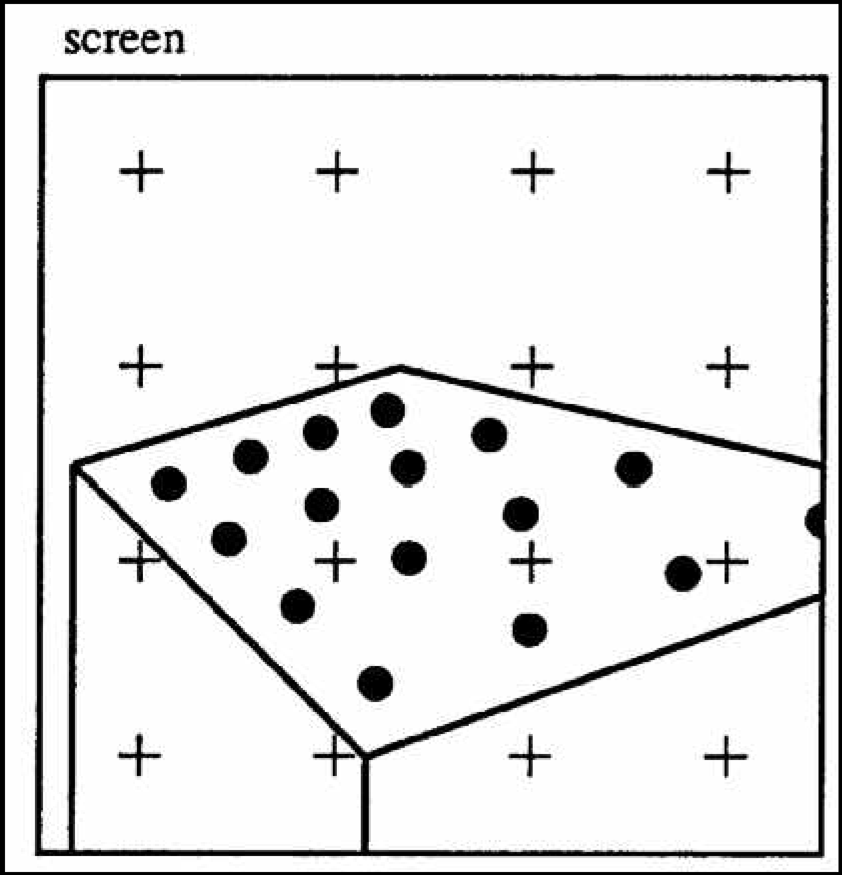
- flat surface
- even lighting (no specularity)

3D Rendering

Procedural synthesis

- Sample a procedural texture

Texture Sampling



from Heckbert, Paul. *Fundamentals of Texture Mapping and Image Warping*. Masters Thesis. UC Berkeley. 1989. page 7.

Sampling Approaches

Point Sampling

- Pick closest texel

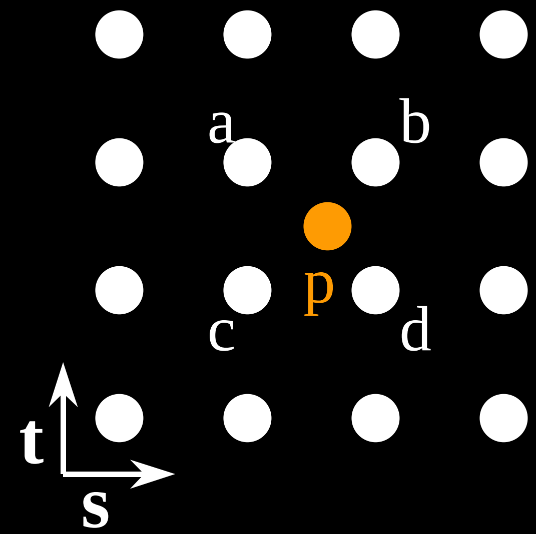
Interpolation

- Blend closest texels

Area Sampling

- Blend all covered texels

Bilinear Interpolation



$$p = (p_s, p_t)$$

$$p' = ((p_s - a_s) / (b_s - a_s), (p_t - a_t) / (c_t - a_t))$$

$$p_{color} = lerp(lerp(a_{color}, b_{color}, p_s'), lerp(c_{color}, d_{color}, p_s'), p_t')$$

$$lerp(k_1, k_2, t) = (1-t)*k_1 + t*k_2$$

Texture Area Sampling

If frequency of texture content is higher than sampling rate, may want better filtering

Pixel-sized area on surface covers some area in texture domain

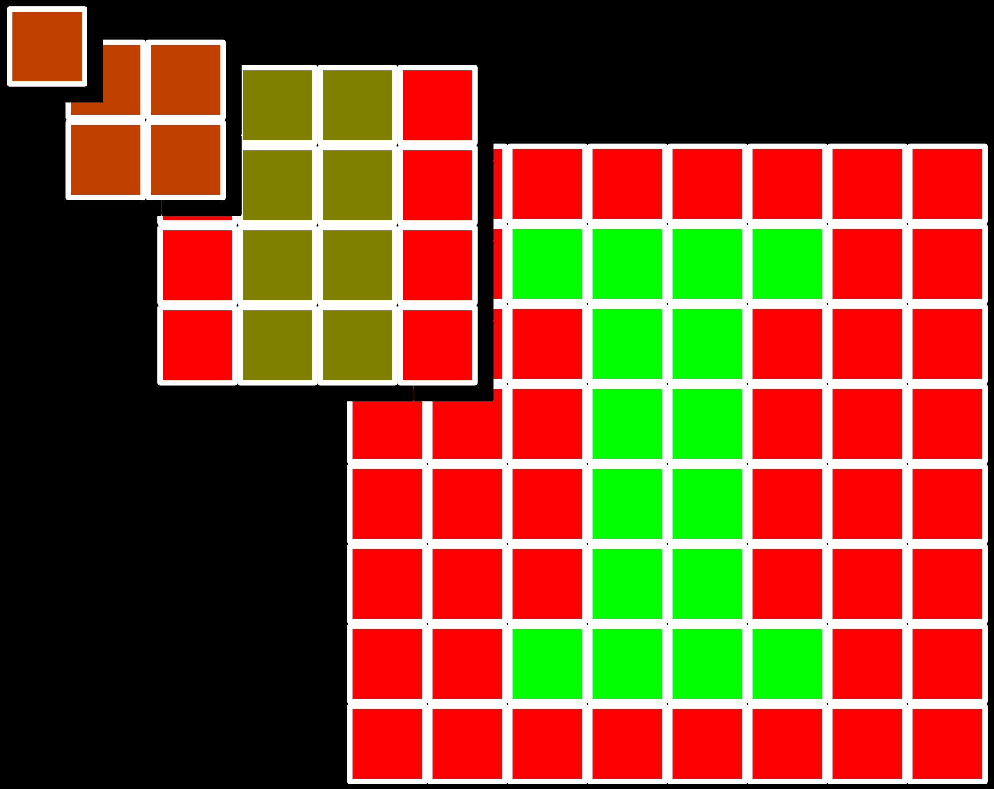
- Curvilinear quadrilateral or ellipse

Perform weighted average of texels covered by pixel-sized piece of surface

Mip-mapped Texture Filtering

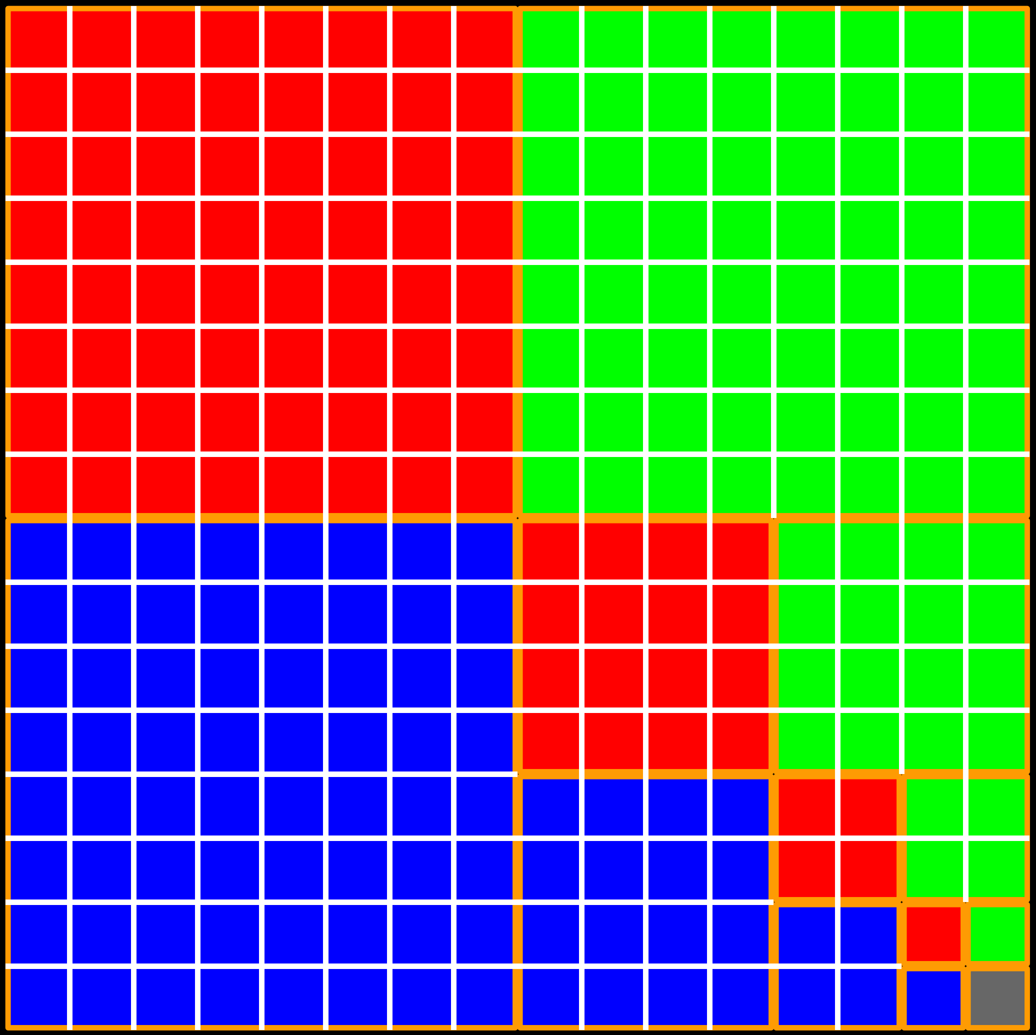
- **M**ultim **i**m **p**arvo (many things in a small place)
- Pre-compute *image pyramid* to filter texture to various resolutions
- Look up colors from the appropriate level(s) of the image pyramid
- Approximation to accurate area sampling

Image Pyramid



parent color = average(4 children colors)

Mip-map Organization



Mip-map Filtering Methods

- Compute d, the parameter along level space
- Sample texture
  - Option 1: Point sample nearest level
  - Option 2: Point sample each adjacent level, then linearly interpolate between them
  - Option 3: Choose nearest level, then bilinearly interpolate within that level
  - Option 4: Trilinearly interpolate between the 8 samples of two adjacent mip-map levels (2 bilinear interps + 1 linear)

Computing d

Somewhat tricky, because a circular footprint on the screen is elliptical in the texture domain

Typically either over-filter or under-filter

One possible formulation:

$$d = \max\left(\mathrm{sqrt}\left((du/dx)^2 + (dv/dx)^2 \right),\ \mathrm{sqrt}\left((du/dy)^2 + (dv/dy)^2 \right) \right)$$

(i.e. use the larger of the ellipse dimensions)

3D Image-based Texture Mapping

- Store data in a 3D image (voxel grid)
- Point sample using nearest voxel
- Linearly interpolate using 8 nearest voxels
- Pre-filtering possible using 3D analog to mip-mapping

Acquiring 3D images

Slice and photograph real materials

- **e.g. - The Visible Human**

Measure density volume using CT scan or MRI, then map densities to colors

Sample a procedurally-generated volume

Canonical Parameterizations

Three common primitives:

- Plane
- Cylinder
- Sphere

Plane Parameterization

Suppose we have a plane with origin O and non-colinear axes, i and j

- $(x,y,z) = (O_x+si_x+tj_x,\ O_y+si_y+tj_y,\ O_z+si_z+tj_z)$
- $(u,v) = (s,t)$

Cylinder Parameterization

Suppose we have a circular cylinder of height h about z-axis (with base at z=0)

- $(x,y,z) = (r\cos\theta, r\sin\theta, z)$
- $(u,v) = (\theta/2\pi, z/h)$

Or we can choose to cover only a portion of the cylinder:

- $(u,v) = (a(\theta-\theta_0)/2\pi, b(z-z_0)/h)$

Sphere Parameterization

We can similarly parameterize the sphere:

- $(x,y,z) = (r\cos\theta\sin\phi, r\sin\theta\sin\phi, r\cos\phi)$
- $(u,v) = (\theta/2\pi, \phi/\pi)$

Note: parameterization degenerate at poles

- **"you can't comb the hair on a sphere"**

Cover portion of sphere with texture:

- $(u,v) = (a*(\theta - \theta_0)/2\pi, b*(\phi - \phi_0)/\pi)$

Two-stage Mapping

1. Map texture onto canonical primitive (the *intermediate surface)*

2. Map intermediate surface to arbitrary object

   - Position objects with respect to each other
   - Project along normal direction (of either one)

Two-stage Example

Atlas Approaches

Break complex surface into patches

Parameterize / texture each patch

- Parameterizations optimized to minimize distortions

Atlas describes mapping between texture domains and surface domain

Atlas Example

from Pederson, "Decorating Implicit Surfaces", *Proceedings of SIGGRAPH 95.*

Other Texturing Options

- Application Modes: relationship between texture colors and surface colors
  - Decal - texture color replaces surface color
  - Blend - colors are combined (e.g. multiplied)
- Wrap modes: what to do with parameters outside of [0,1]
  - Clamp
  - Repeat