Texture Mapping in Practice

Dinesh K. Pai

Textbook Appendix A4, Chapter 15

Some slides courtesy of M. Kim, KAIST

1

Today

- Announcements
  - Assignment 2 spotlights

- Texture mapping, continued

2

Assignment 2 Spotlights
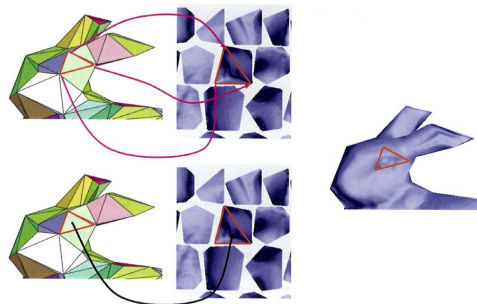
- Soohyun Kim
- Tim Straubinger
- Hoda Hashemi
- Fan Wu
- Sean Vanbergen
- Jean Chen

3

Texture mapping

- In basic texturing, we simply 'glue' part of an image onto a triangle by specifying texture coordinates at the three vertices.


4

Texture mapping

- Bunch of OpenGL/WebGL functions to load a texture and set various parameters (lin/const, mipmap, wrapping rules).
- A uniform variable is used to point to the desired texture unit
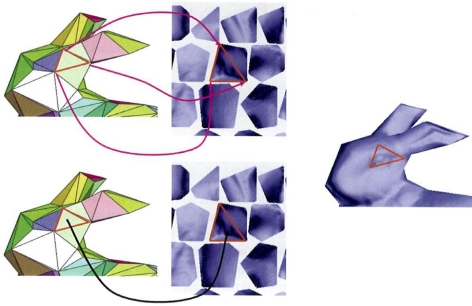
5

Texture mapping

- Varying variables are used to store texture coordinates.
- In this simplest incarnation, we just fetch r,g,b values from the texture and send them directly to the frame buffer.
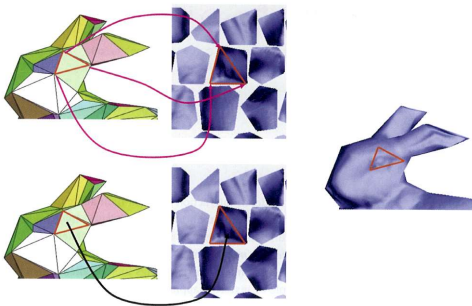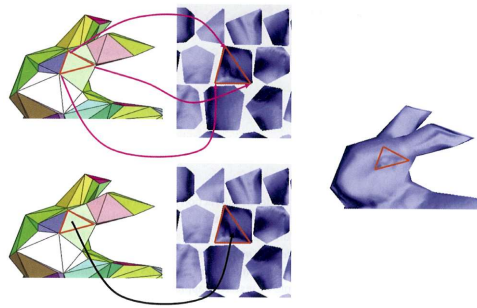

6

Texture mapping

- Alternatively, the texture data could be interpreted as, say, the diffuse material color of the surface point, which would then be followed by the diffuse material computation described earlier.

7

Steps for Texture Mapping

1. Create a *texture object* and load texels into it
2. Include *texture coordinates* with your vertices
3. Associate a *texture sampler* with each texture map used in shader
4. Retrieve texel values

(Reference: Red Book)

8

Recap
Understanding Texture Mapping

- Better view: An efficient way to model surface detail using discrete (sampled) data
- Need to understand two surprisingly subtle concepts
  - "Coordinates"
    Parameterization of surfaces
  - "Images"
    Sampled representations of continuous functions
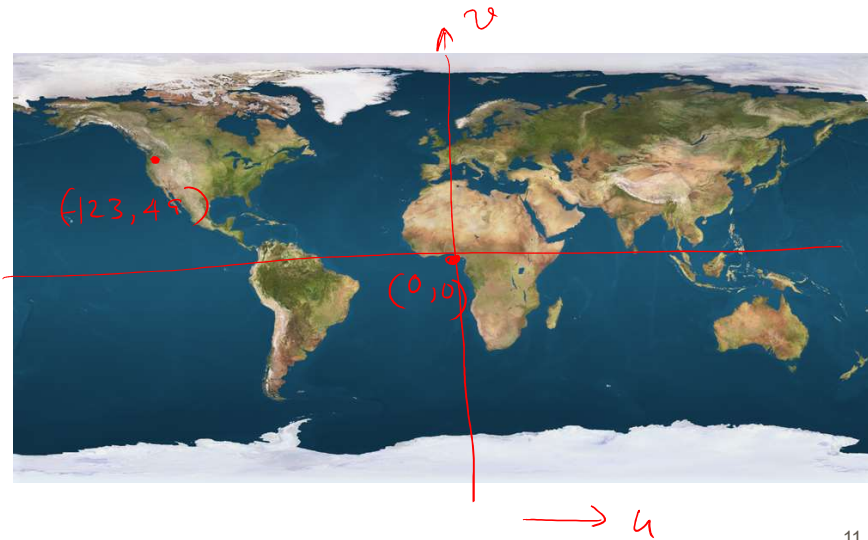   More details in Chapters 16-18. We'll be covering this at a high level.
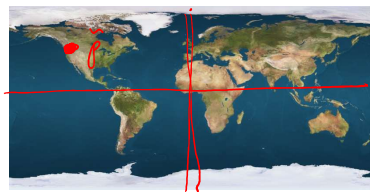
9

An intuitive example
How to model the earth?

longitude, latitude
(123W 49N)
You are here
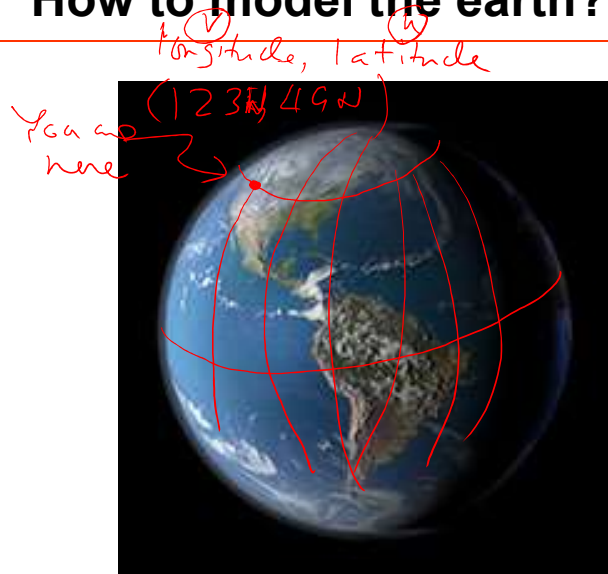
10

Earth (texture) Map

Texture coordinates, simple case

Texture Mapping

3D

2D

2D

1D

$\vec{3}$

$f(p)$

p

$\vec{2}$

$-\pi$

π

f

f

$\cdot f(p)$

$\cdot p$
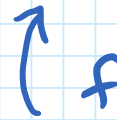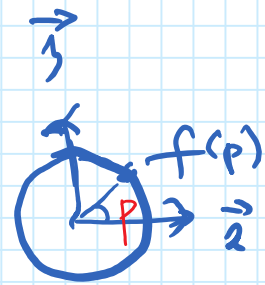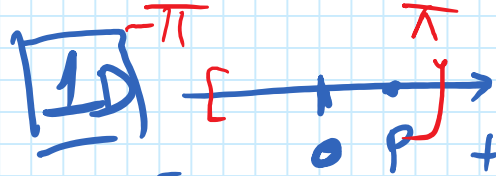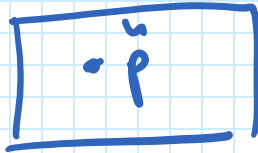
Construct a map f which covers the higher dimensional object.

$$f(p) = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \cos p \\ \sin p \\ 1 \end{pmatrix}$$

In general, this simple technique doesn't work. (No good "global" function exists) But can do this locally!

Per-vertex data

$\check{P}_1$ u_1

$\check{P}_2$ u_2

f

f can be a simple linear interpolation of vertex values. Using Varying Variables

P_1 u_2

u_1 P_2

0 1

0 1