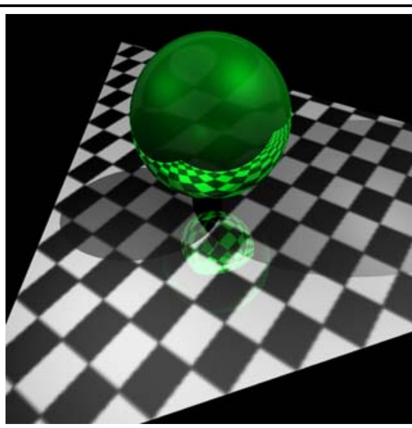
cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- depth of field
- lens effects
- jittering
- soft shadows

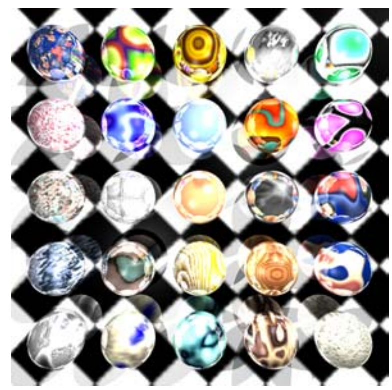
texture mapping

"glue" image to surface

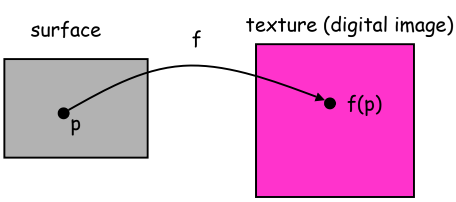
steve yan,
fall 2001



drew levin,
fall 2001



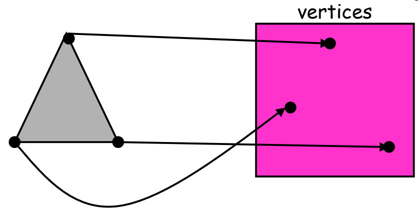
texture mapping



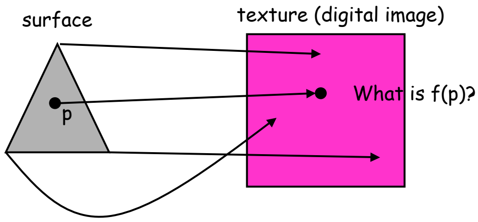
f(p): coordinates in the texture map corresponding to surface point p

texture mapping triangle

Input specifies texture coordinates of triangle vertices

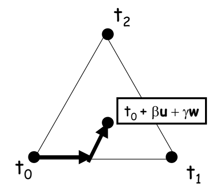


texture mapping triangle

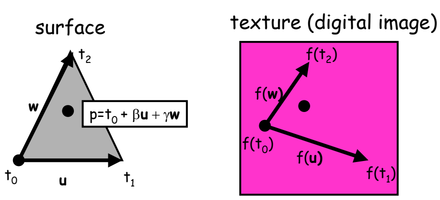


triangle: parametric form

a point q on the triangle T can be uniquely represented as
$q = t_0 + \beta\mathbf{u} + \gamma\mathbf{w}$ where $\beta \geq 0$, $\gamma \geq 0$, $\beta + \gamma \leq 1$

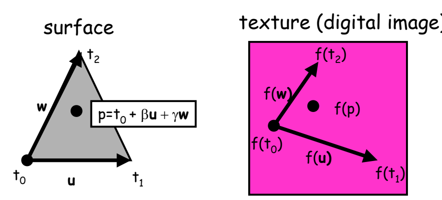


computing f(p)



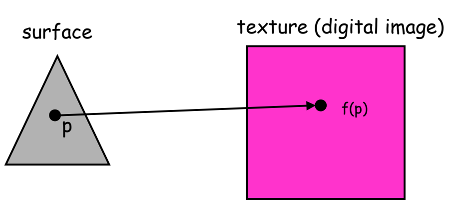
compute $f(\mathbf{u}) = f(t_1) - f(t_0)$ and $f(\mathbf{w}) = f(t_2) - f(t_0)$

computing f(p)



compute $f(p) = f(t_0) + \beta f(\mathbf{u}) + \gamma f(\mathbf{w})$

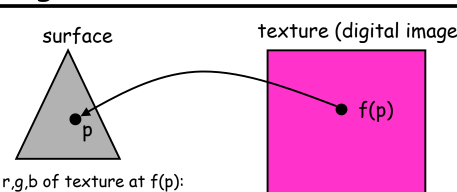
texture mapping triangle



What is image color at f(p)?

Need to resample! For your ray tracer use bilinear interpolation.

using the color

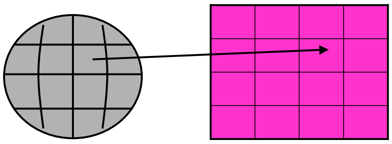


r,g,b of texture at f(p):

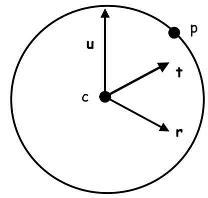
1. use as pixel color
2. use as diffuse and specular coefficient of surface at p
3. use as diffuse coefficient of surface at p

texture mapping sphere

e.g. latitude/longitude

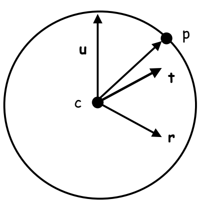


parameterization



a point p on the surface can be represented relative c and **r**, **t**, **u**

parameterization



$p - c = \alpha \mathbf{r} + \beta \mathbf{t} + \gamma \mathbf{u}$

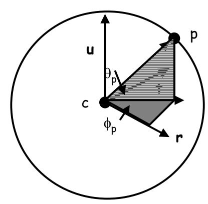
where

$\alpha =$

$\beta =$

$\gamma =$

parameterization



$p - c = \alpha \mathbf{r} + \beta \mathbf{t} + \gamma \mathbf{u}$

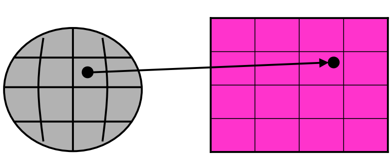
where

$\alpha = r \cos(\theta_p)\cos(\phi_p)$

$\beta = r \cos(\theta_p)\sin(\phi_p)$

$\gamma = r \sin \theta_p$

texture mapping sphere



$f(p) = (w\phi_p /360, h\theta_p /360)$ where the the angles are in degrees and w, h are the image width and height in pixels

as before

- resample image
- use color as ...

problems

- given p compute ϕ, θ

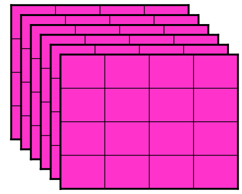
you can do that!

- poles

test for pole and use default texture coordinate

- seams
  - use good textures
  - overlap & blend or mix
  - don't look there
  - 3d textures

3d textures



use stack of images

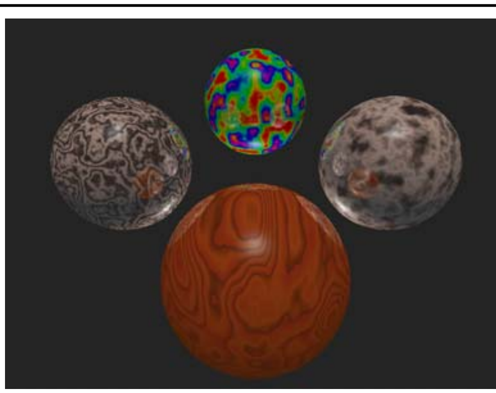
how do we generate these images?

cheap tricks

- texture mapping
- **procedural texture mapping**
- bump mapping
- transparency mapping
- depth of field
- lens effects
- jittering
- soft shadows

procedural texture mapping

- procedure returns a texture color for any point in 3d space (note this is not an image stack)
- sample to find texture for surface



adrian mettler, spring 2003

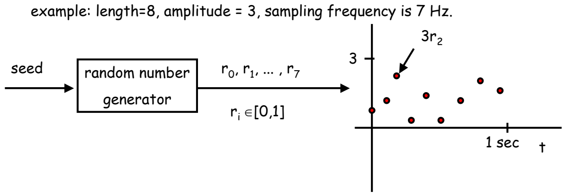
procedural textures

- advantages
  - don't need to find a mapping from a (complex) 3d surface to a 2d texture image
  - concise representation of texture
- disadvantages
  - ad hoc techniques cannot duplicate photographs

perlin noise – 1D example

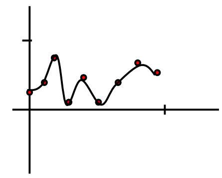
step 1: generate discrete noise function with specified length, amplitude, sampling frequency

example: length=8, amplitude = 3, sampling frequency is 7 Hz.



perlin noise – 1D example

step 2: interpolate with smoothing

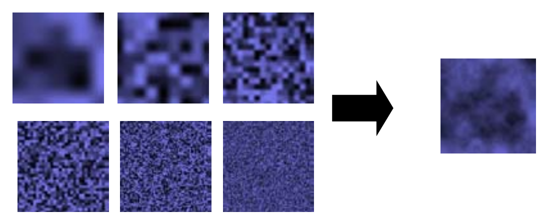


perlin noise – 1D example

step 3: repeat with various amplitudes/frequencies

step 4: add together

perlin noise – 2D example



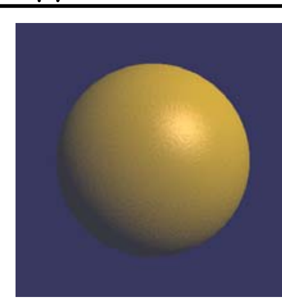
for more info see Perlin Noise link on proj2 web site

cheap tricks

- texture mapping
- procedural texture mapping
- **bump mapping**
- transparency mapping
- depth of field
- lens effects
- jittering
- soft shadows

creating bumpy surfaces

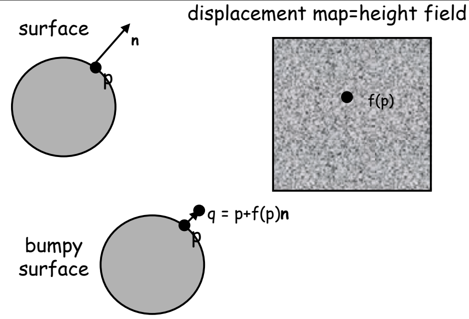
adrian mettler,
spring 2003



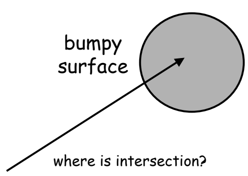
bump mapping vs texture mapping

- bump mapping effects change with lighting changes
- texture mapping is computationally easier

displacement mapping



displacement mapping



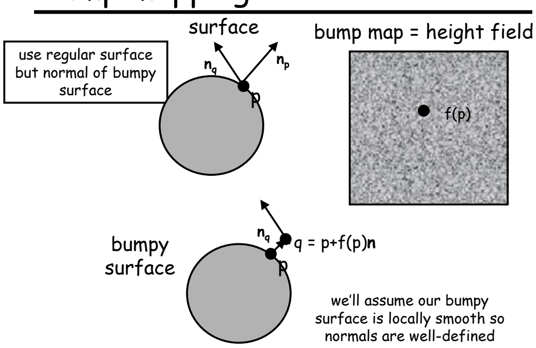
bump mapping intuition

a surface appears to have a bump surface

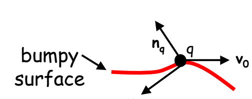
- jagged silhouette
- surface normals fluctuate across surface

simulate this by perturbing normals in the lighting calculations

bump mapping



bump mapping



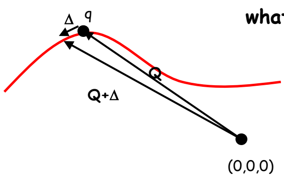
computing $\mathbf{n}_q$:

1. find vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ in plane tangent to bumpy surface at point q
2. $\mathbf{n}_q = (\mathbf{v}_0 \times \mathbf{v}_1)/||\mathbf{v}_0 \times \mathbf{v}_1||$

find vectors in tangent plane

take partial derivatives of **Q** in two directions

what directions?

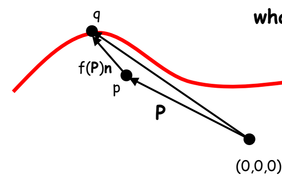


find vectors in tangent plane

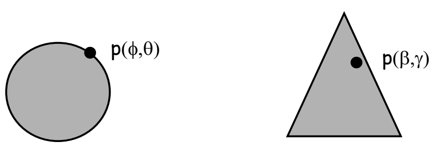
take partial derivatives of **Q**=**P**+f(**P**)**n** in two directions

d**Q**/du = d**P**/du + d[f(**P**)**n**]/du

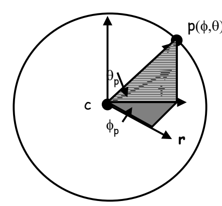
what directions?



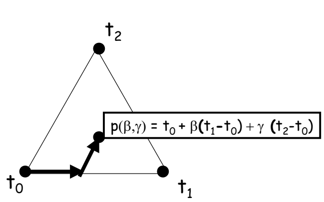
2d parameterization of original surface



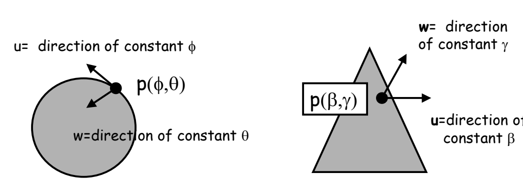
parameterization



parameterization



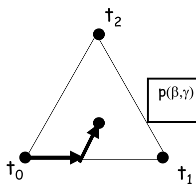
2d parameterization of surface



note: by direction we mean a unit vector

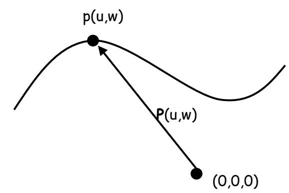
parameterization

$\mathbf{u} = (t_1 - t_0)/||t_q - t_0||$ -- this is a little different than our parameterization for triangle intersection



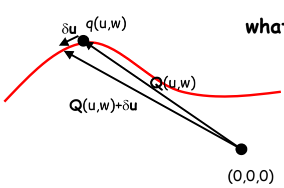
$p(\beta,\gamma) = t_0 + \beta(t_1 - t_0) + \gamma\ (t_2 - t_0) = t_0 + \beta'\mathbf{u} + \gamma'\mathbf{w}$

2d parameterization of surface



find vectors in tangent plane

take partial derivatives of **Q** in two directions

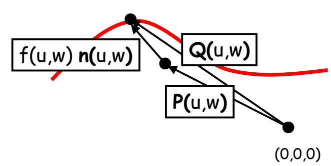


what directions?

ans: u,w

find vectors in tangent plane

take partial derivatives of **Q**(u,w)=**P**(u,w)+f(u,w)**n**(u,w) with respect to u,w



find vectors in tangent plane

take partial derivatives of **Q**(u,w)=**P**(u,w)+f(u,w)**n**(u,w) with respect to u,w

$\mathbf{Q}_u = \mathbf{P}_u + [df(u,w)/d\mathbf{u}]\mathbf{n}(u,w) + f(u,w)d\mathbf{n}(u,w)/d\mathbf{u}$

$\mathbf{Q}_w = \mathbf{P}_w + [df(u,w)/d\mathbf{w}]\mathbf{n}(u,w) + f(u,w)d\mathbf{n}(u,w)/d\mathbf{w}$

find vectors in tangent plane

take partial derivatives of **Q**(u,w)=**P**(u,w)+f(u,w)**n**(u,w) with respect to u,w

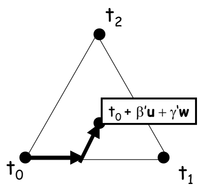
$\mathbf{Q}_u = \boxed{\mathbf{P}_u} + \boxed{[df(u,w)/d\mathbf{u}]}\ \mathbf{n}(u,w) + f(u,w)\ \boxed{d\mathbf{n}(u,w)/d\mathbf{u}}$

$\mathbf{Q}_w = \boxed{\mathbf{P}_w} + \boxed{[df(u,w)/d\mathbf{w}]}\ \mathbf{n}(u,w) + f(u,w)\ \boxed{d\mathbf{n}(u,w)/d\mathbf{w}}$

We can compute $\mathbf{P}_u$ and $\mathbf{P}_w$ for our surfaces.

triangle: parametric form

$P_u = \lim_{||\delta u|| \to 0} [\ (t_0 + \beta'(\delta u + u) + \gamma w) - (t_0 + \beta' u + \gamma w)\ ]/||\delta u|| = \beta' u$



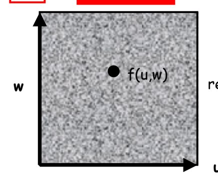
reminder: $u = (t_1 - t_0)/||t_1 - t_0||$ and $w = (t_2 - t_0)/||t_2 - t_0||$

find vectors in tangent plane

take partial derivatives of $Q(u,w)=P(u,w)+f(u,w)n(u,w)$ with respect to u,w

$Q_u = P_u + [df(u,w)/du]\ n(u,w) + f(u,w)\ dn(u,w)/du$

$Q_w = P_w + [df(u,w)/dw]\ n(u,w) + f(u,w)\ dn(u,w)/dw$



how does bump change with respect to changes in u and w?

bump map derivative

convolution kernel

-1	0	1
-1	0	1
-1	0	1

change in u

1	1	1
0	0	0
-1	-1	-1

change in w

find vectors in tangent plane

take partial derivatives of $Q(u,w)=P(u,w)+f(u,w)n(u,w)$ with respect to u,w

$Q_u = P_u + [df(u,w)/du]\ n(u,w) + f(u,w)\ dn(u,w)/du$

$Q_w = P_w + [df(u,w)/dw]\ n(u,w) + f(u,w)\ dn(u,w)/dw$

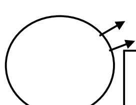
we'll call these scalars $b_u(u,w)$ and $b_w(u,w)$

find vectors in tangent plane

take partial derivatives of $Q(u,w)=P(u,w)+f(u,w)n(u,w)$ with respect to u,w

$Q_u = P_u + b_u(u,w)\ n(u,w) + f(u,w)\ dn(u,w)/du$

$Q_w = P_w + b_w(u,w)\ n(u,w) + f(u,w)\ dn(u,w)/dw$



how does normal change with respect to changes in u and w

we'll ignore this because
(a) it is small,
(b) it is computationally difficult, and
(c) results look ok if we do.

find vectors in tangent plane

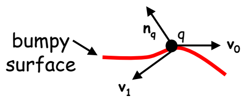
take partial derivatives of $Q(u,w)=P(u,w)+f(u,w)n(u,w)$ with respect to u,w

$Q_u = P_u + b_u(u,w)\ n(u,w)$

$Q_w = P_w + b_w(u,w)\ n(u,w)$

we can do this!

bump mapping



computing $\mathbf{n}_q$:

1. find vectors $\mathbf{v}_0$ and $\mathbf{v}_1$ in plane tangent to bumpy surface at point q
2. $\mathbf{n}_q = (\mathbf{v}_0 \times \mathbf{v}_1)/||\mathbf{v}_0 \times \mathbf{v}_1||$

take the cross product

take partial derivatives of $\mathbf{Q}(u,w)=\mathbf{P}(u,w)+f(u,w)\mathbf{n}(u,w)$ with respect to u,w

$$\mathbf{Q}_u = \mathbf{P}_u + b_u(u,w)\ \mathbf{n}(u,w)$$

$$\mathbf{Q}_w = \mathbf{P}_w + b_w(u,w)\ \mathbf{n}(u,w)$$

take cross product

$$\mathbf{Q}_u \times \mathbf{Q}_v = \mathbf{P}_u \times \mathbf{P}_v + b_u(u,w)\,\mathbf{P}_u \times \mathbf{n}(u,w) + b_w(u,w)\,\mathbf{P}_w \times \mathbf{n}(u,w) + \mathbf{n}(u,w) \times \mathbf{n}(u,w)$$

take the cross product

take partial derivatives of $\mathbf{Q}(u,w)=\mathbf{P}(u,w)+f(u,w)\mathbf{n}(u,w)$ with respect to u,w

$$\mathbf{Q}_u = \mathbf{P}_u + b_u(u,w)\ \mathbf{n}(u,w)$$

$$\mathbf{Q}_w = \mathbf{P}_w + b_w(u,w)\ \mathbf{n}(u,w)$$

take cross product

$$\mathbf{Q}_u \times \mathbf{Q}_v = \mathbf{P}_u \times \mathbf{P}_v + b_u(u,w)\,\mathbf{P}_u \times \mathbf{n}(u,w) + b_w(u,w)\,\mathbf{P}_w \times \mathbf{n}(u,w) + \boxed{\mathbf{n}(u,w) \times \mathbf{n}(u,w)}$$

this is 0

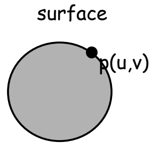
Computation

$$\mathbf{Q}_u \times \mathbf{Q}_v = (\mathbf{P}_u \times \mathbf{P}_v) + b_u(u,w)\,\mathbf{P}_u \times \mathbf{n}(u,v) + b_w(u,w)\,\mathbf{P}_w \times \mathbf{n}(u,v)$$

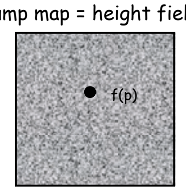
1. Compute derivatives of surface $\mathbf{P}_u$ and $\mathbf{P}_v$
2. Compute derivates of bump map $b_u(u,v)$ and $b_v(u,v)$
3. Take cross products and add

2d parameterization of surface

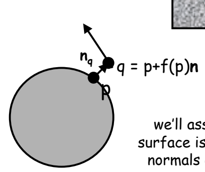
surface



bump map = height field



bumpy surface



we'll assume our bumpy surface is locally smooth so normals are well-defined

cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- **transparency mapping**
- depth of field
- lens effects
- jittering
- soft shadows

Texture specifies transparency of surface

cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- **depth of field**
- lens effects
- jittering
- soft shadows

blur based on distance from viewpoint

cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- depth of field
- **lens effects**
- jittering
- soft shadows

See paper mentioned in assignment

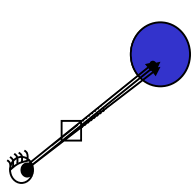
cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- depth of field
- lens effects
- **jittering**
- soft shadows

jittering: antialiasing technique

Run rt for example

jittering: anti-aliasing technique



- cast several ray through pixel neighborhood into scene
- for each:
  - find intersection point (if any) that is closest to eye
  - compute luminance at intersection
- compute average luminance

cheap tricks

- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- depth of field
- lens effects
- jittering
- **soft shadows**

soft shadows

run rt for examples

soft shadows

use jittering in occlusion test

ray tracing

- simple ray casting
- recursive ray tracing
- modeling transforms
- cheap tricks
- **optimizations**