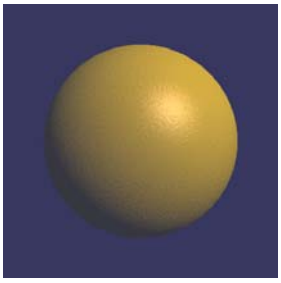
cheap tricks

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

creating bumpy surfaces

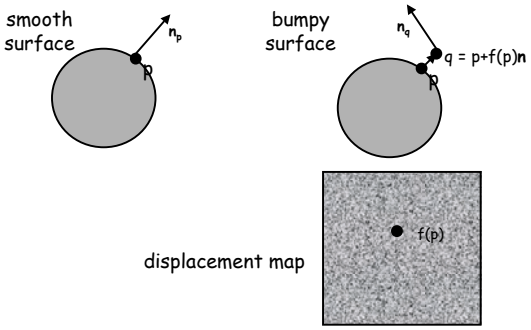
adrian mettler,
spring 2003



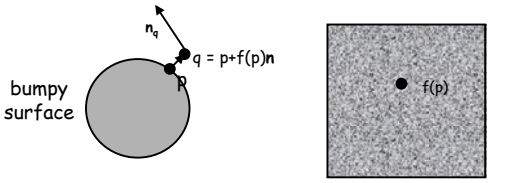
bump mapping vs texture mapping

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

smooth vs. bumpy surface models

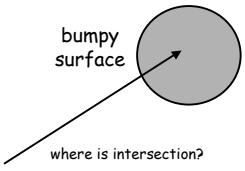


displacement mapping



displacement map=height field

displacement mapping problem



where is intersection?

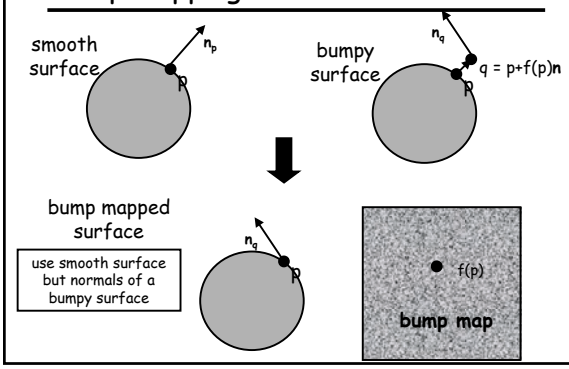
bump mapping intuition

a surface appears to have a bumpy surface

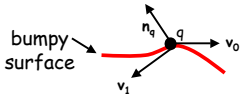
- jagged silhouette
- surface normals fluctuate across surface

simulate this by perturbing normals in the lighting calculations

bump mapping



bump mapping: computing n_q



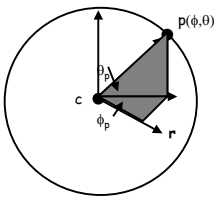
computing n_q:

1. find vectors v_0 and v_1 in plane tangent to bumpy surface at point q
2. $n_q = (v_0 \times v_1)/||v_0 \times v_1||$

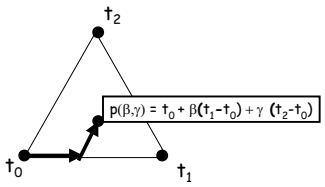
Warning!

2d parameterization of smooth surface

sphere parameterization



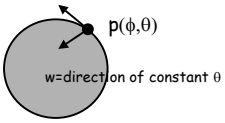
triangle parameterization



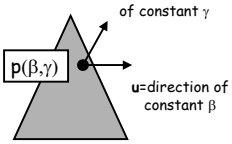
2d parameterization of smooth surface

general parameterization: p(u,w)

u= direction of constant ϕ



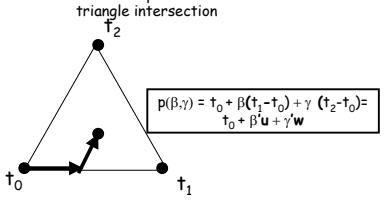
w= direction of constant γ



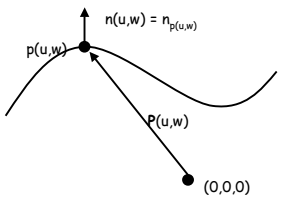
note: by direction we mean a unit vector

parameterization (warning)

u = $(t_1-t_0)/||t_q-t_0||$ -- we are normalizing here – we did not do that for our parameterization in the triangle intersection

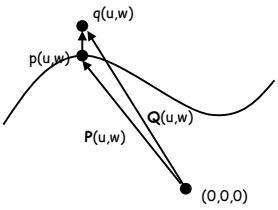


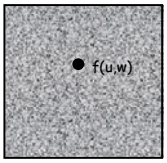
2d parameterization of smooth surface



2d parameterization of bumpy surface

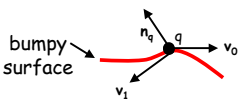
Q(u,w)=**P**(u,w)+f(u,w)**n**(u,w)





f(u,w)=f(p(u,w))

bump mapping: computing n_q



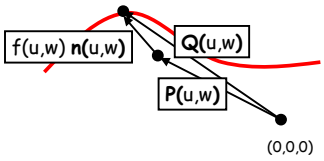
We need to do this!!

computing $\mathbf{n}_q$:

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

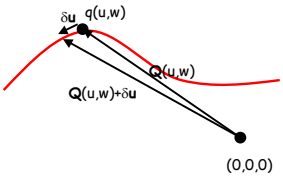
find vectors in tangent plane

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



find vectors in tangent plane

$dQ(u,v)/du = \lim_{\delta u \to 0} Q(u,w) + \delta u$



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

dQ/du = **dP/du** + [df(u,w)/du]**n**(u,w) + f(u,w)d**n**(u,w)/du

dQ/dw = **dP/dw** + [df(u,w)/dw]**n**(u,w) + f(u,w)d**n**(u,w)/dw

find vectors in tangent plane

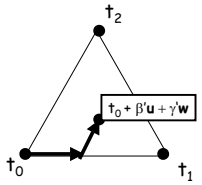
take partial derivatives of **Q**(u,w)=**P**(u,w)+f(u,w)**n**(u,w) with respect to u,w

dQ/du =	**dP/du**	+	[df(u,w)/du]	**n**(u,w)	+	f(u,w) **dn**(u,w)/du
dQ/dw =	**dP/dw**	+	[df(u,w)/dw]	**n**(u,w)	+	f(u,w) **dn**(u,w)/dw

We can compute **dP/du** and **dP/dw** for our surfaces.

triangle: parametric form

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

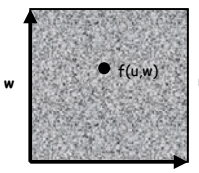


reminder: **u** = $(t_1-t_0)/||t_1-t_0||$ is a unit vector in the t_1-t_0 direction and
w = $(t_2-t_0)/||t_2-t_0||$ is a unit vector in the t_2-t_0 direction

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

dQ/du =	**dP/du**	+	[df(u,w)/du]	**n**(u,w)	+	f(u,w) **dn**(u,w)/du
dQ/dw =	**dP/dw**	+	[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 $\mathbf{Q}(u,w)=\mathbf{P}(u,w)+f(u,w)\mathbf{n}(u,w)$ with respect to u,w

$$\mathbf{dQ/du} = \mathbf{dP/du} + [df(u,w)/du]\ \mathbf{n}(u,w) + f(u,w)\ \mathbf{dn}(u,w)/du$$
$$\mathbf{dQ/dw} = \mathbf{dP/dw} + [df(u,w)/dw]\ \mathbf{n}(u,w) + f(u,w)\ \mathbf{dn}(u,w)/dw$$

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

find vectors in tangent plane

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

$$\mathbf{dQ/du} = \mathbf{dP/du} + [df(u,w)/du]\ \mathbf{n}(u,w) + f(u,w)\ \mathbf{dn}(u,w)/du$$
$$\mathbf{dQ/dw} = \mathbf{dP/dw} + [df(u,w)/dw]\ \mathbf{n}(u,w) + f(u,w)\ \mathbf{dn}(u,w)/dw$$

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.

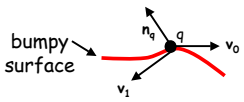
find vectors in tangent plane

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

$$\mathbf{dQ/du} \approx \mathbf{dP/du} + b_u(u,w)\ \mathbf{n}(u,w)$$
$$\mathbf{dQ/dw} \approx \mathbf{dP/dw} + b_w(u,w)\ \mathbf{n}(u,w)$$

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

$$\mathbf{dQ/du} \approx \mathbf{dP/du} + b_u(u,w)\ \mathbf{n}(u,w)$$
$$\mathbf{dQ/dw} \approx \mathbf{dP/dw} + b_w(u,w)\ \mathbf{n}(u,w)$$

take cross product

$$\mathbf{dQ/du} \times \mathbf{dQ/dw} = \mathbf{dP/du} \times \mathbf{dP/dw} + b_u(u,w)\mathbf{dP/dw} \times \mathbf{n}(u,w) + b_w(u,w)\ \mathbf{dP/du} \times \mathbf{n}(u,w) + b_u(u,w)\ \mathbf{n}(u,w) \times b_w(u,w)\ \mathbf{n}(u,w)$$

take the cross product

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

$$\mathbf{dQ/du} \approx \mathbf{dP/du} + b_u(u,w)\ \mathbf{n}(u,w)$$

$$\mathbf{dQ/dw} \approx \mathbf{dP/dw} + b_w(u,w)\ \mathbf{n}(u,w)$$

take cross product

$$\mathbf{dQ/du} \times \mathbf{dQ/dw} = \mathbf{dP/du} \times \mathbf{dP/dw} +$$
$$b_u(u,w)\mathbf{dP/dw} \times \mathbf{n}(u,w) + b_w(u,w)\ \mathbf{dP/du} \times \mathbf{n}(u,w) +$$
$$b_u(u,w)\ \mathbf{n}(u,w) \times b_w(u,w)\ \mathbf{n}(u,w)$$

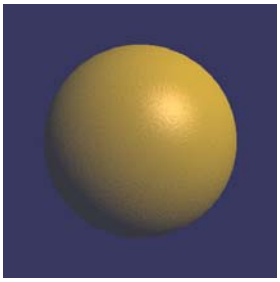
this is 0

Computation

1. Compute derivatives of surface
   dP/du and **dP/dw**
2. Compute derivatives of bump map
   $b_u(u,v)$ and $b_v(u,v)$
3. Take cross products and add:
   $\mathbf{dP/du} \times \mathbf{dP/dw} + b_u(u,w)\mathbf{dP/dw} \times \mathbf{n}(u,w) + b_w(u,w)\ \mathbf{dP/du} \times \mathbf{n}(u,w)$

ta da!

adrian mettler,
spring 2003



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

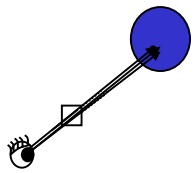
cheap tricks

- 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



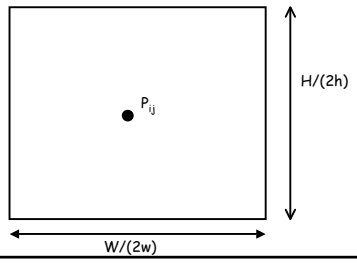
- cast several (random) rays through pixel neighborhood into scene
- for each:
  - find intersection point (if any) that is closest to eye
  - compute color at intersection
- compute average color

random ray

shoot ray i,j through $P_{i,j}$ + (dx,dy) where

dx is chosen uniformly at random over [-W/(2w), W/(2w)]

dy is chosen uniformly at random over [-H/(2h),H/(2h)]



cheap tricks

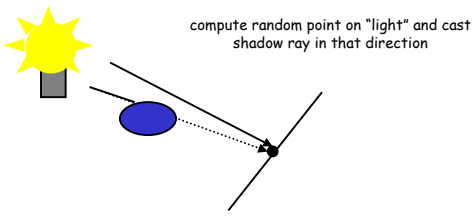
- texture mapping
- procedural texture mapping
- bump mapping
- transparency mapping
- depth of field
- lens effects
- jittering
- **soft shadows**

soft shadows

run rt for examples

soft shadows

In reality, we don't have point lights!!



compute random point on "light" and cast shadow ray in that direction