Bump Maps

Intro to parametric curves

We're all familiar with $f(x)=x^2$ and its derivative $f'(x)=2x$

But I'd prefer parametric $f(t)=(t,t^2)$ and its derivative $f'(t)=(1,2t)$

This gives more freedom:

Speedy $f(t)=(2t,4t^2)$ and $f'(t)=(2,8t)$

More shapes: $f(t)=(\cos t,\sin t)$, and $f'(t)=(-\sin t,\cos t)$

Intro to parametric surfaces

$P: \ \Re^2 \rightarrow \Re^3$

$P(u,v) = (X(u,v),Y(u,v),Z(u,v))$

But we try to avoid the individual coordinate representation when possible

For instance a linear surface (A, B, and C are vectors in $\Re^3$)

$L(u,v) = C + uA + vB = (C_x+uA_x+vB_x, C_y+uA_y+vB_y, C_z+uA_z+vB_z)$

or a general polynomial

$$R(u,v) = A_{00}+uA_{10}+vA_{01}+u^2A_{20}+uvA_{11}+v^2A_{02} = \begin{pmatrix}1 & u & u^2\end{pmatrix}\begin{pmatrix}A_{00} & A_{01} & A_{02}\\ A_{10} & A_{11} & 0\\ A_{20} & 0 & 0\end{pmatrix}\begin{pmatrix}1\\ v\\ v^2\end{pmatrix} = (\ldots)$$

or a sphere

$S(\theta,\phi) = (\sin\phi\cos\theta, \sin\phi\sin\theta, \cos\phi)$

Calculus on a parametric surface

Can be done on individual coordinates

$$\frac{\partial P}{\partial u} = \left(\frac{\partial}{\partial u}X(u,v), \frac{\partial}{\partial u}Y(u,v), \frac{\partial}{\partial u}Z(u,v),\right) \qquad \frac{\partial P}{\partial v} = \ldots$$

But we try to avoid coordinate representations when possible

$$\frac{\partial L}{\partial u} = A = (A_x,A_y,A_z) \qquad \frac{\partial L}{\partial v} = B = (B_x,B_y,B_z)$$

$$\frac{\partial R}{\partial u} = A_{10}+2u\,A_{20}+v\,A_{11} = \begin{pmatrix}0 & 1 & 2u\end{pmatrix}\begin{pmatrix}A_{00} & A_{01} & A_{02}\\ A_{10} & A_{11} & 0\\ A_{20} & 0 & 0\end{pmatrix}\begin{pmatrix}1\\ v\\ v^2\end{pmatrix} \qquad \frac{\partial R}{\partial v} = \ldots$$

$$\frac{\partial S(\theta,\phi)}{\partial\theta} = (-\sin\phi\sin\theta,\ \sin\phi\cos\theta,\ 0)$$

$$\frac{\partial S(\theta,\phi)}{\partial\phi} = (\cos\phi\cos\theta,\ \cos\phi\sin\theta,\ -\sin\phi)$$

$$N = \frac{\partial S}{\partial\theta}\times\frac{\partial S}{\partial\phi}$$

$$= -\sin\phi\ (\sin\phi\cos\theta,\ \sin\phi\sin\theta,\ \cos\phi\sin\theta\sin\theta+\cos\phi\cos\theta\cos\theta)$$

$$= -\sin\phi\ \ S(\theta,\phi)$$

Normal of a parametric surface

$$P_N(u,v) = \frac{\partial P(u,v)}{\partial u} \times \frac{\partial P(u,v)}{\partial v}$$

$$L_N(u,v) = A\times B$$

Bump map derivation

Model a surface and bumps as parametric surfaces:

P(u,v) = (X(u,v), Y(u,v), Z(u,v)) is a surface in 3D

f(u,v) is a vertical bump to be applied onto the surface

Assumptions

f has small values

it's derivative may be large

The *bumped* surface

$$\bar{P}(u,v) = P(u,v) + f(u,v)\frac{N(u,v)}{\|N(u,v)\|}$$

Want

values of $P(u,v)$ for rendering,

normals of $\bar{P}(u,v)$ for lighting calculations

Calculate

$$\frac{\partial \bar{P}}{\partial u} = \frac{\partial}{\partial u}P(u,v) + \frac{\partial}{\partial u}f(u,v)\frac{N(u,v)}{\|N(u,v)\|} + f(u,v)\frac{\partial}{\partial u}\frac{N(u,v)}{\|N(u,v)\|}$$

$$\frac{\partial \bar{P}}{\partial v} = \ldots$$

The last term is assumed to be small enough to ignore. Using short cut notation

$$\bar{P}_u = P_u + f_u\frac{N}{\|N\|}$$

$$\bar{P}_v = \ldots$$

So the normal of $\bar{P}(u,v)$ is

$$\begin{aligned}
\bar{N} &= \bar{P}_u \times \bar{P}_v \\
&= P_u \times P_v + f_u\left(\frac{N}{\|N\|}\times P_v\right) + f_v\left(P_u \times \frac{N}{\|N\|}\right) + f_u f_v\left(\frac{N\times N}{\|N\|\|N\|}\right) \\
&= P_u \times P_v + (f_v P_u - f_u P_v)\times\frac{N}{\|N\|} \\
&= N + D
\end{aligned}$$

where

$$D = (f_v P_u - f_u P_v)\times\frac{N}{\|N\|}$$

is perpendicular to N, and in the tangent plane $\langle P_u, P_v\rangle$

Derivation of f_u and $f_{v.}$

For $f: [0,1] \times [0,1] \rightarrow \Re^3$

Find i,j such that:

$$i \le uw \le (i+1)$$
$$j \le vh \le (j+1)$$

Then with blending basis functions

$$u_0 \;=\; 1 + i - uw \qquad u_1 \;=\; uw - i$$
$$v_0 \;=\; 1 + j - vh \qquad v_1 \;=\; vh - j$$

F(u,v) over that region is

$$f(u,v) \;=\; u_0 v_0 f_{i,j} \;+\; u_1 v_0 f_{i+1,j} \;+\; u_0 v_1 f_{i,j+1} \;+\; u_1 v_1 f_{i+1,j+1}$$

so

$$\frac{\partial f(u,v)}{\partial u} \;=\; -w v_0 f_{i,j} \;+\; w v_0 f_{i+1,j} \;-\; w v_1 f_{i,j+1} \;+\; w v_1 f_{i+1,j+1}$$

$$\;=\; w v_0 (f_{i+1,j} - f_{i,j}) \;+\; w v_1 (f_{i+1,j+1} - f_{i,j+1})$$

$$\frac{\partial f(u,v)}{\partial v} \;=\; -h u_0 f_{i,j} \;+\; h u_1 f_{i+1,j} \;-\; h u_0 f_{i,j+1} \;+\; h u_1 f_{i+1,j+1}$$

$$\;=\; h u_0 (f_{i,j+1} - f_{i,j}) \;+\; h u_1 (f_{i+1,j+1} - f_{i+1,j})$$

But these values of f_u and f_v themselves are in some unspecified range of values

And so must be scaled by what?

Some user specified value to control the intensity of the bump map.

Depends on visual system interpretation of intensity changes as depth changes.

Adjust it until you get the the desired effect.

Derivation of P_u and P_v from texture coordinates

Assume a triangle, P_0, P_1, P_2 with texture coordinates T_0, T_1, T_2
For convenience define $T_{10} = T_1 - T_0$, $T_{20} = T_2 - T_0$, $P_{10} = P_1 - P_0$, $P_{20} = P_2 - P_0$
Will derive a linear map: $P(u,v) = P(t)$ s.t. $P(T_i) = P_i$

If we can find a and b as functions of T such that:

$$T = (1-a-b)T_0 + aT_1 + bT_2 = T_0 + aT_{10} + bT_{20} ,$$

then

$$P(T) = P(T_0) + aP(T_{10}) + bP(T_{20}) = P_0 + aP_{10} + bP_{20}$$

and the derivatives we want are

$$P_u = \frac{\partial P(t)}{\partial u} = \frac{\partial a}{\partial u}(P_{10}) + \frac{\partial b}{\partial u}(P_{20})$$

$$P_v = \frac{\partial P(t)}{\partial v} = \frac{\partial a}{\partial v}(P_{10}) + \frac{\partial b}{\partial v}(P_{20})$$

Now to compute a and b:

$$T = T_0 + aT_{10} + bT_{20}$$

implies

$$u = u_0 + a(u_{10}) + b(u_{20})$$

$$v = v_0 + a(v_{10}) + b(v_{20})$$

Use Cramer's rule to solve for a and b:

$$d = \begin{pmatrix} u_{10} & u_{20} \\ v_{10} & v_{20} \end{pmatrix}$$

$$a = \frac{\begin{pmatrix} u-u_0 & u_{20} \\ v-v_0 & v_{20} \end{pmatrix}}{d} = \frac{(u-u_0)v_{20} - (v-v_0)u_{20}}{d}$$

$$b = \frac{\begin{pmatrix} u_{10} & u-u_0 \\ v_{10} & v-v_0 \end{pmatrix}}{d} = \frac{u_{10}(v-v_0) - v_{10}(u-u_0)}{d}$$

The derivatives are

$$\frac{\partial a}{\partial u} = v_{20}/d \qquad \frac{\partial a}{\partial v} = -u_{20}/d$$

$$\frac{\partial b}{\partial u} = -v_{10}/d \qquad \frac{\partial b}{\partial v} = u_{10}/d$$

and so

$$P_u = (\ \ v_{20} P_{10} - v_{10} P_{20})/d$$

$$P_v = (-u_{20} P_{10} + u_{10} P_{20})/d$$

Bump Map in shaders

Vertex shader

Input: attribute vec3 vertexTangent
Transform: Pu = gl_NormalMatrix*vertexTangent
Output: varying vec3 Pu
as well as the usual:
 gl_Position, and
 normal, light, and eye direction vectors for Phong

Fragment Shader

Input: varying vec3 Pu
Calculate:
 Pv = cross(N,Pu)
 Normalize N, Pu, Pv
 get fu, fv
 N += scaleFudge*cross((fv*Pu-fu*Pv), N);
 normalize N (again)
 Light with normal N