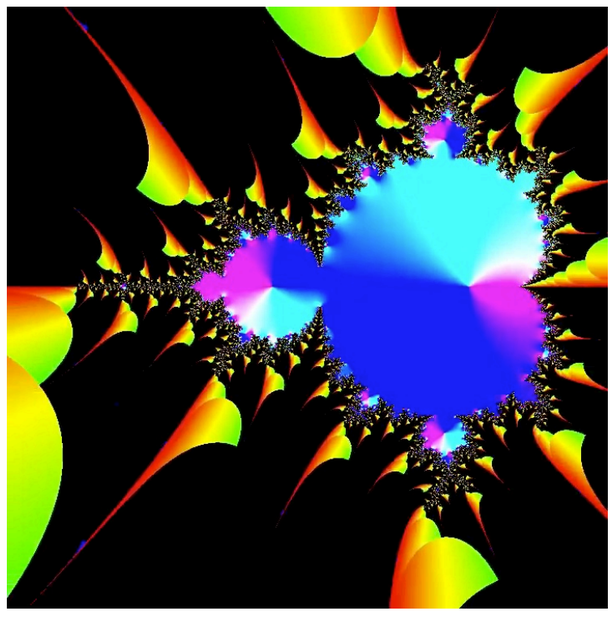
TSBK 07

Computer Graphics

Ingemar Ragnemalm, ISY



Lecture 7

More bump mapping

Light mapping

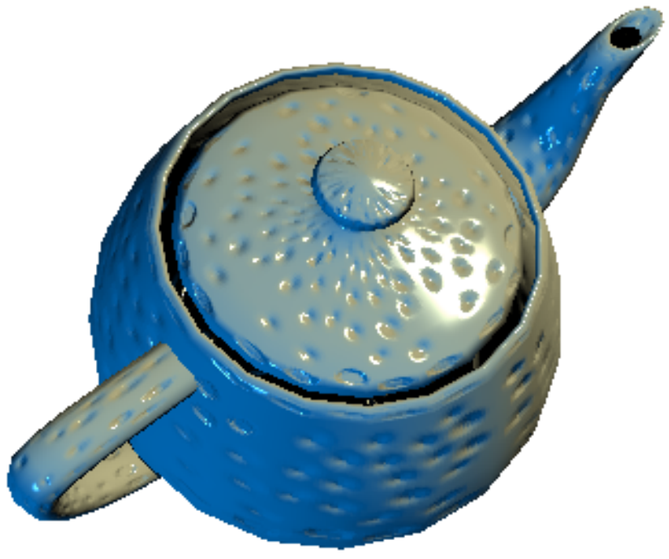
Normal matrix

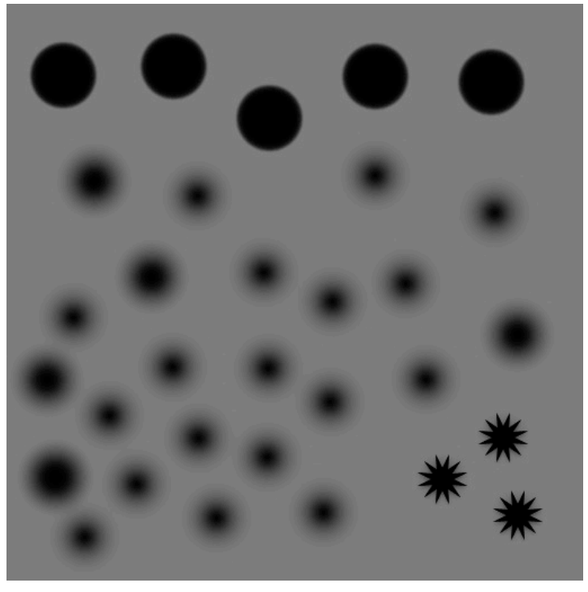
Painter's algorithm

Transparency

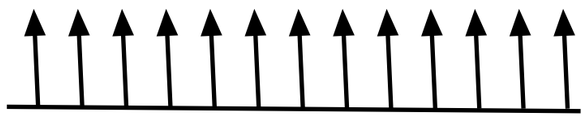
Bump mapping

Simulates surface structure by manipulating the normal vector



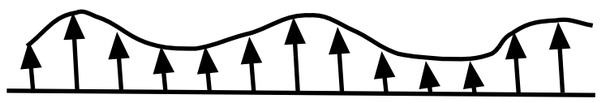


Bump mapping - model

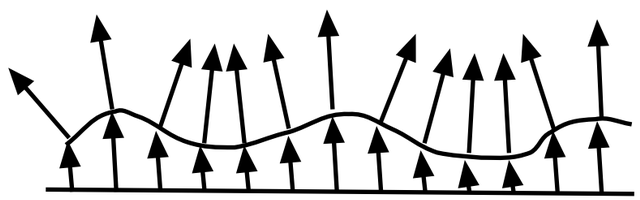


Surface with normal vectors

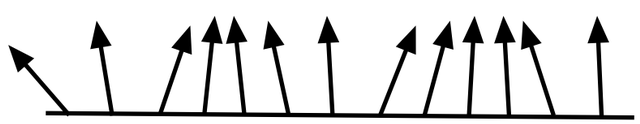
Bump map: scalar function of the texture coordinates



Modulate the surface by the bump function, along normal



Calculate new normals



Resulting normal vectors

Bump mapping - the coordinate systems

Input:
A point **p**, normal vector **n**
Texture coordinates s(**p**), t(**p**)
Directions of texture coordinates **s**, **t**
The bump function b(s,t)

Calculate the partial derivative of the bump function, b_s and b_t

$$\mathbf{n}' = \mathbf{n} + b_t * (\mathbf{s} \times \mathbf{n}) + b_s * (\mathbf{t} \times \mathbf{n})$$

or, if **s**, **t**, **n** are orthogonal

$$\mathbf{n}' = \mathbf{n} + b_s * \mathbf{s} + b_t * \mathbf{t}$$

Texture coordinate system

n = normal vector

s = tangent

t = bitangent

s and t can be calculated from texture coordinate variations (e.g. Lengyel's method)

Light calculations

needs (n, s, t) to create n'
needs light source and surface positions

Must transform to the same coordinate system,
e.g. view coordinates

Transforming directional vectors

Important!

n, s and t are all *directional* vectors

Must be transformed by modified model-to-world/ world-to-view matrices *without translations*

More about this in a moment...

Calc of modified normal vector

$b_s = db/ds$
$b_t = db/dt$

$n' = n + b_s \cdot s + b_t \cdot t$ ("in")

or

$n' = n - b_s \cdot s - b_t \cdot t$ ("out")

Gradients are simply one step differences in s and t!

$-b_s = b[s, t] - b[s+1, t]$
$-b_t = b[s, t] - b[s, t+1]$
1

Normalize!

Variant/optimization of bump mapping: *Normal mapping*

Precalculate b_s och b_t, save as picture!

But it is just a simple difference! Why can this be significant?

Storage in texture

”Scale and bias”:

R = (ds+1)/2
G = (dt+1)/2

(Why?)

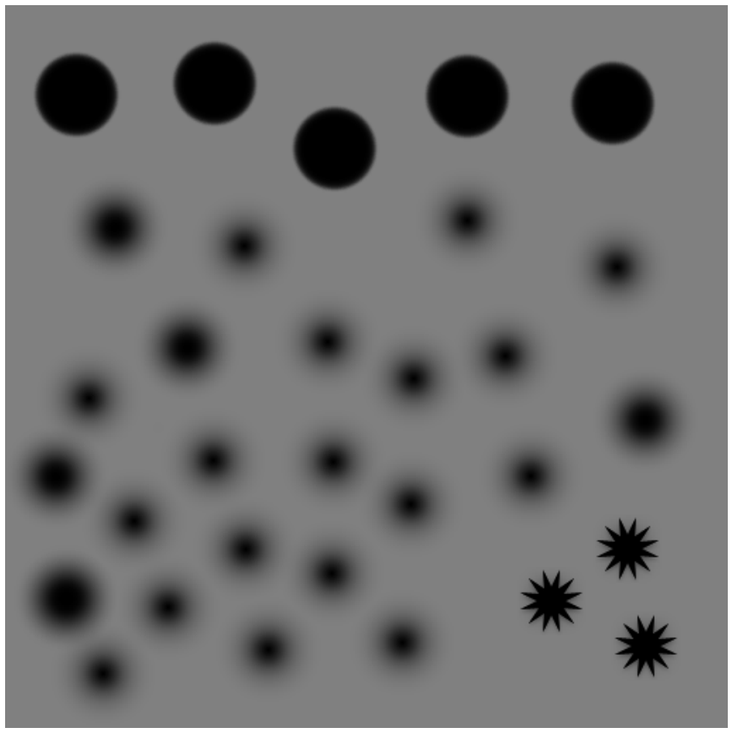
Fetch from texture:

ds = 2R - 1
dt = 2G - 1

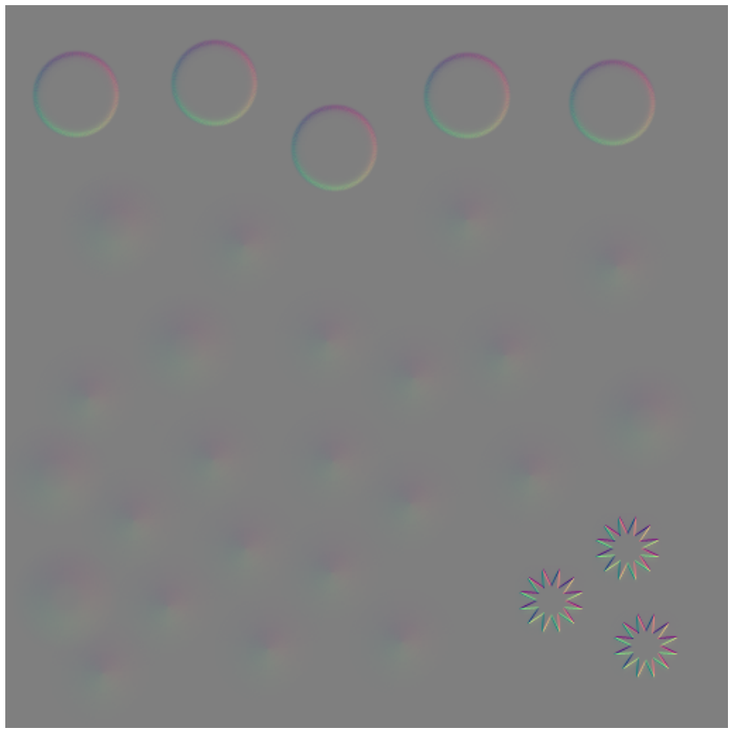
$$n_t = (x, y, z)$$

Example of normal map

Bump map in my example



Bump map



Normal map

Bump mapping or normal mapping?

Normal mapping optimizes memory access

Bump mapping gives more information and is easier to edit

We might want *both* the bump map and the normal map!