CSE 528: Computer Graphics

Bump and Environment Mapping

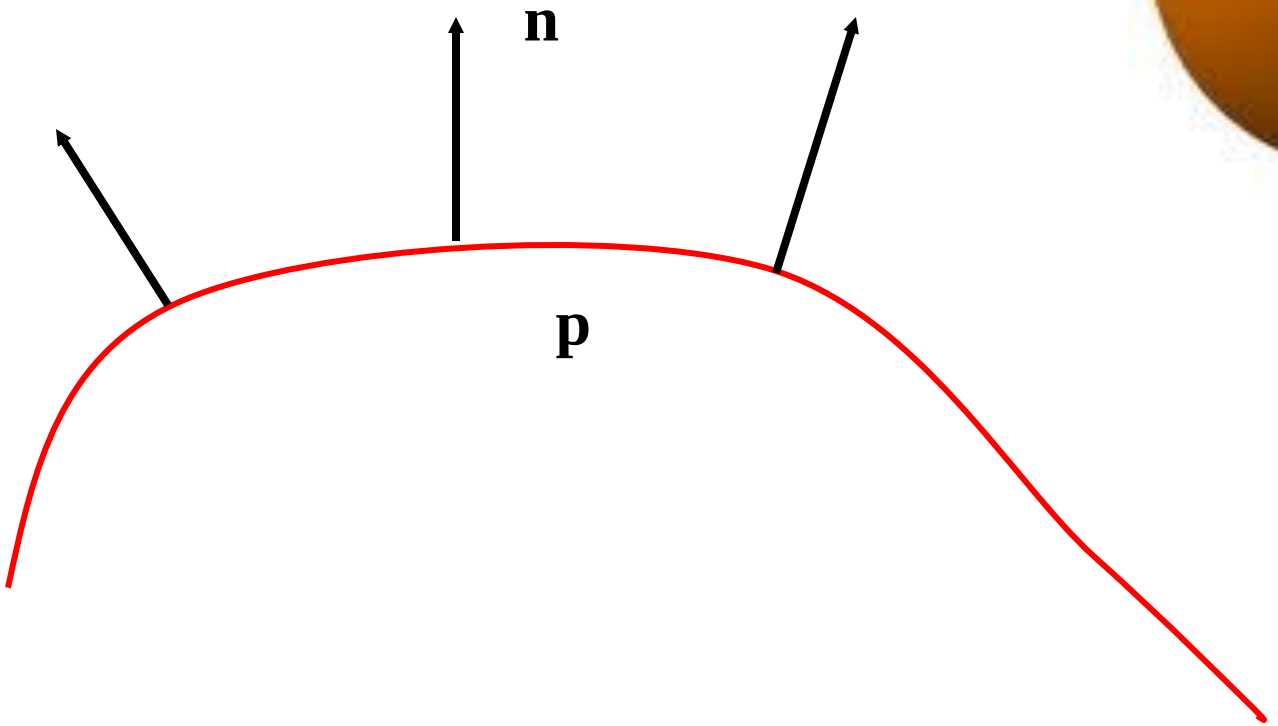
Klaus Mueller

Computer Science Department

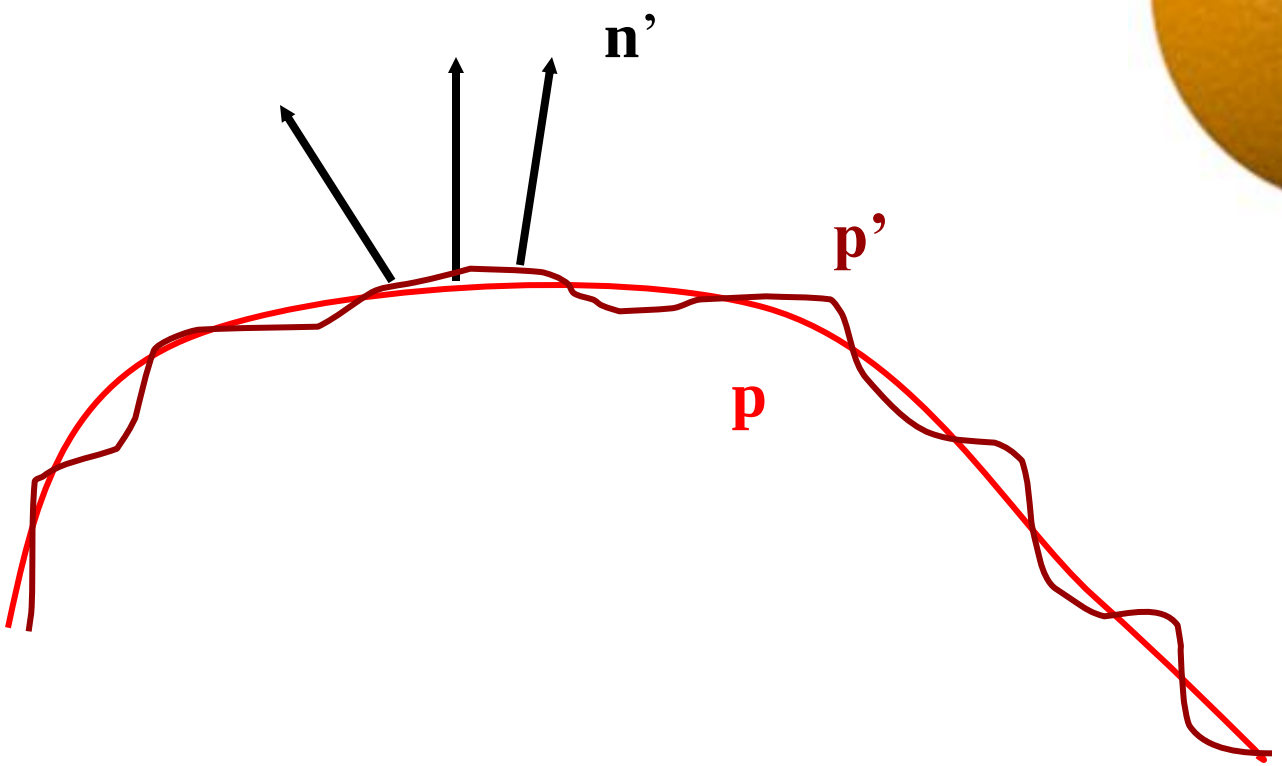
Stony Brook University

Some material from E.Angel, UNM, H. Winnemoeller, NWU, and the paper by J. Blinn

Smooth Surface



Rougher Surface



Bump Mapping Concept

Developed by James Blinn

- “Simulation of Wrinkled Surfaces”, ACM Siggraph 1978

Enables a surface to appear as exhibiting a macroscopic relief-structure

- but without the need to model this relief explicitly in geometry

Instead the effect is achieved by perturbing the surface normal angularly

- done according to information stored in a 2D bump map
- can also be achieved procedurally

These variations of the normal vector trick the viewer into believing to see more detail than is actually existent

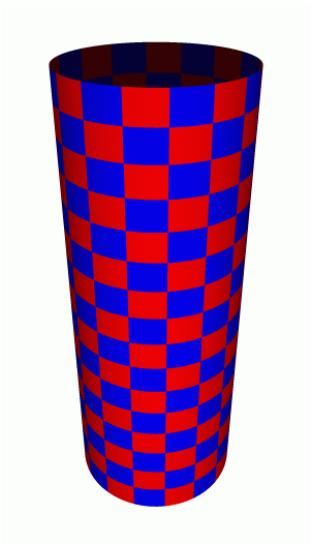
Bump Mapping Concept

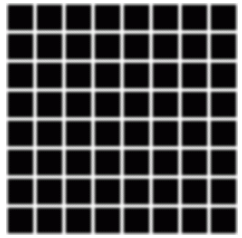
A bump map is a function F(u,v)

- each F(u,v) represents the change in the height of the surface relative to the original flat surface

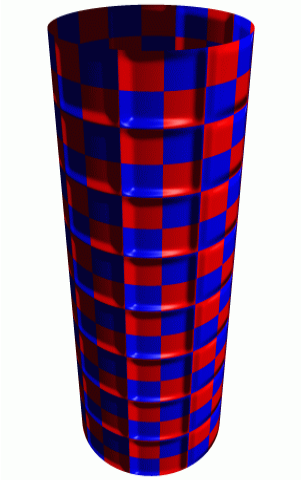
Function F(u,v) can be specified by:

- an analytic expression → procedural texture
- a 2D table (matrix) of values →bump (texture) map
- can either be a map of values or a map with normals (faster)





Bump Map b(u,v)



Cylinder w/Diffuse Texture Map

Cylinder w/Texture Map & Bump Map

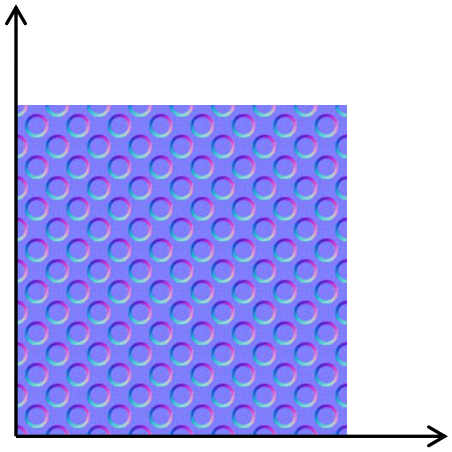
Derivation (1)

A bump map

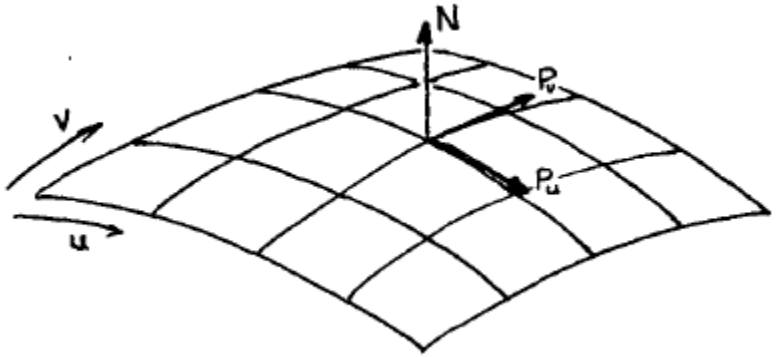
- is a function b(u,v) such that for each u,v, b(u,v) represents the change in the height of the surface relative to the original flat surface.

The function b(u,v)

- can be specified by an analytic expression, or by a 2D table (matrix) of values.



+

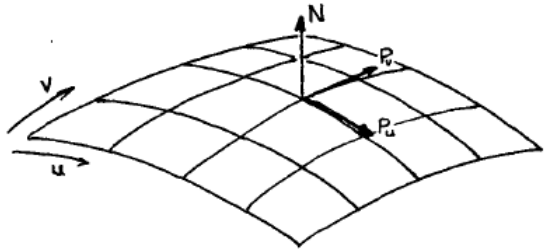


bump map b(u.v)

surface patch P(u,v)

Derivation (2)

For each point on the surface, we need to calculate the new normal, given the bump map



Let P(u,v) be a point on the surface

Then, **N** = P_u x P_v is the normal at P(u,v)

Let **n** be the unit vector of **N**

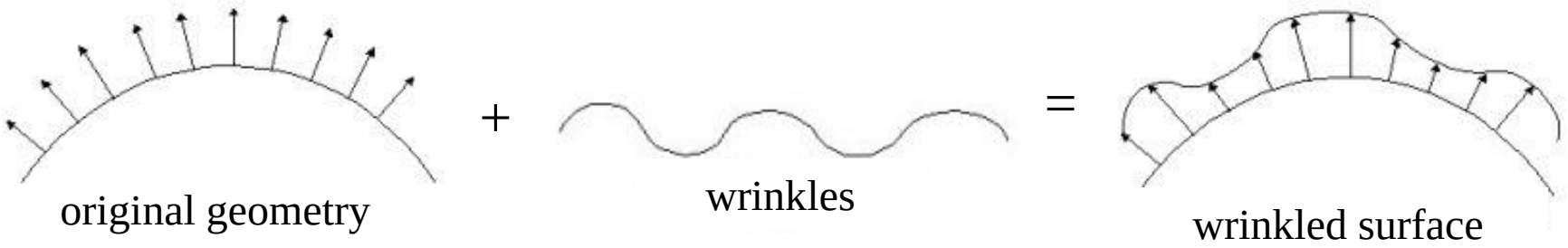
- where P_u and P_v represent the partial derivatives of P in the u and v directions

Using the bump map, updated surface position is P'(u,v) = P(u,v) + b(u,v)**n**

Approximate perturbed normal is **N**' = **N** - b_v(**n**xP_u) + b_u(**n**xP_v)

- where b_u and b_v represent the partial derivative of b in the u and v directions

Therefore, given the bump map, we find **N**' at each u,v location on the surface, and use **n**' for lighting calculations.

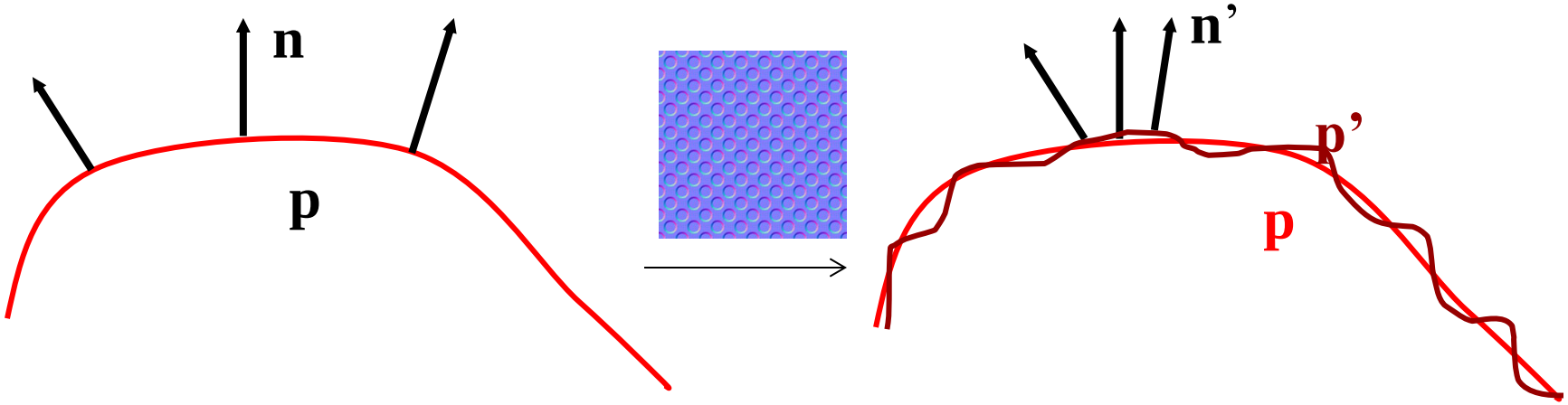


Lighting with Bump Maps

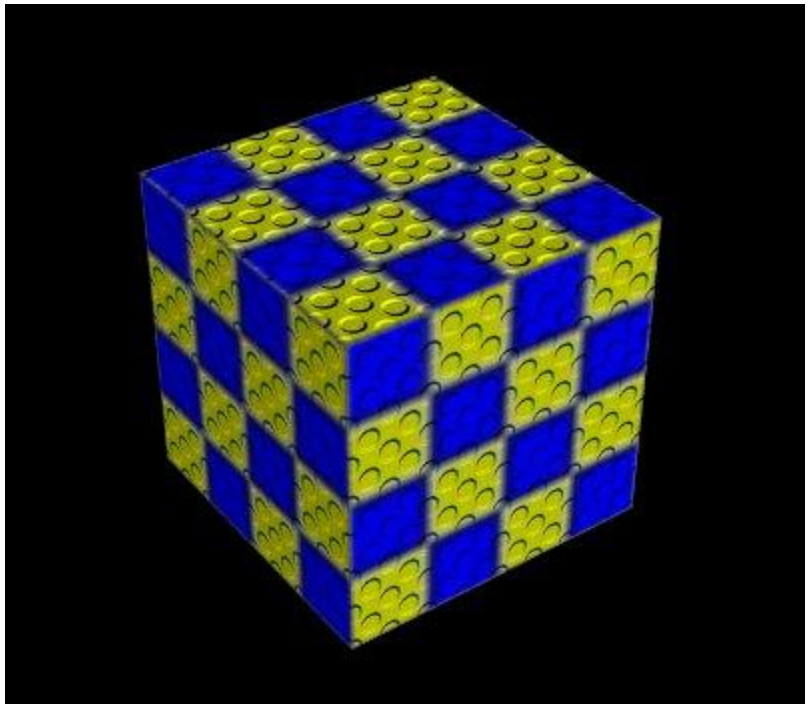
Calculate the interaction of the new "bumpy" surface with lights in the scene using, for example, Phong reflection

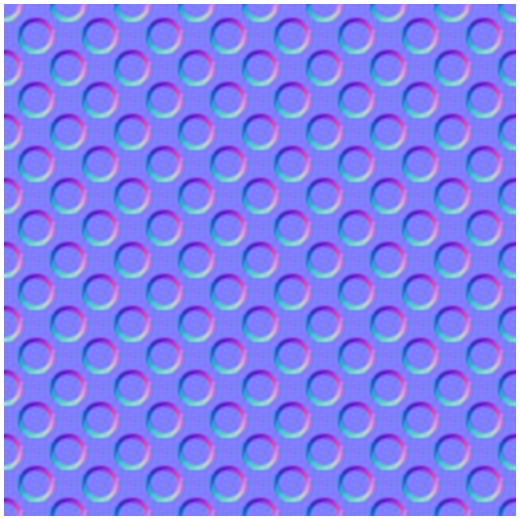
Alternatively

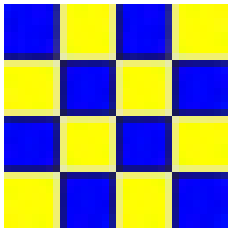
- store the normal map instead of calculating F_u and F_v
- store in RGB
- no perturbation then maps to (0.5. 0.5, 1.0) since the bump vector is aligned with z
- this is why bump map normal maps look blue when texture is displayed

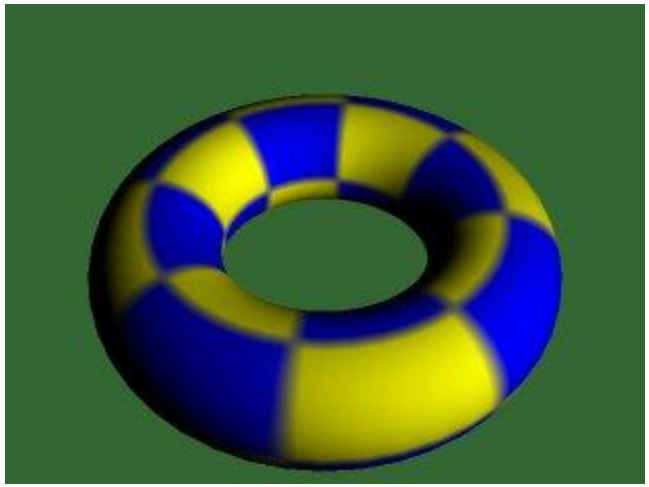


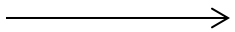
Bump Mapping Examples

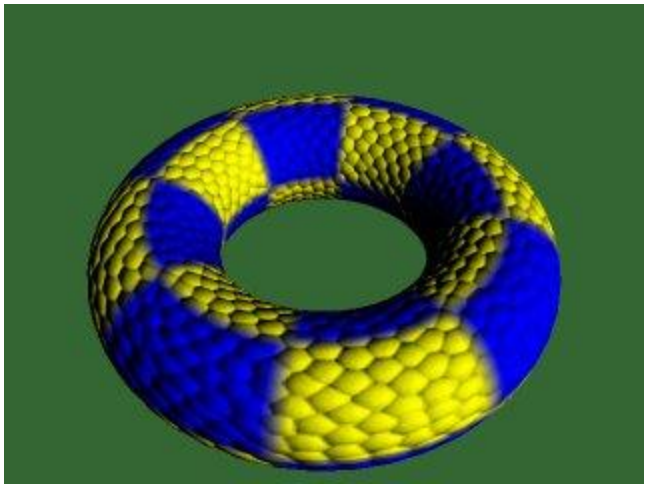






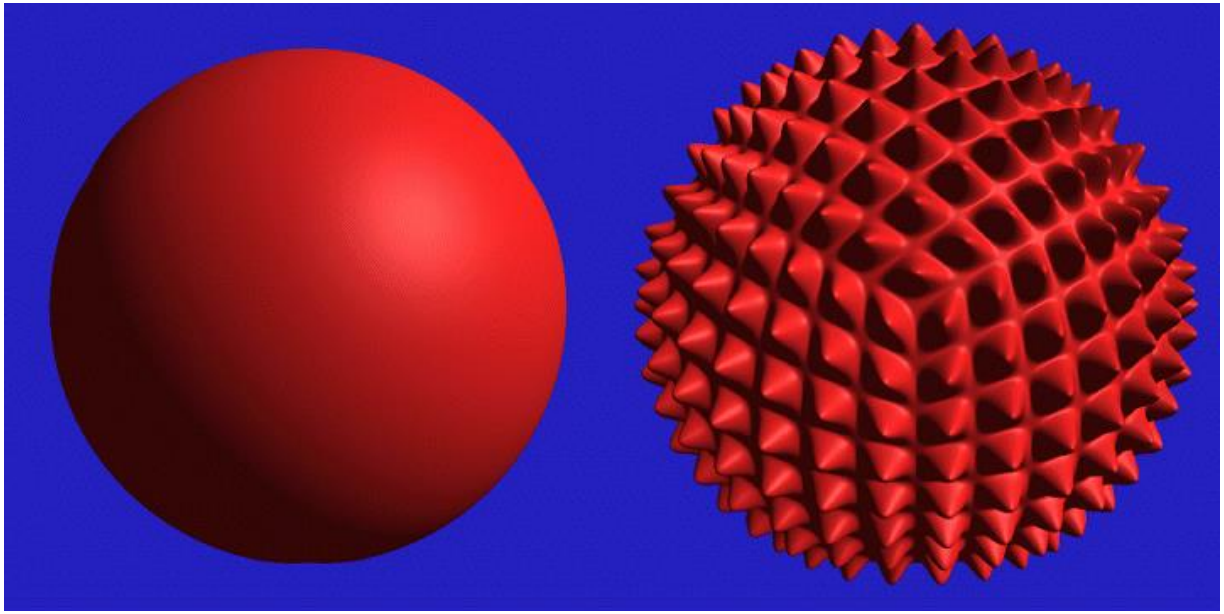






Displacement Mapping

We use the texture map to actually move the surface point. This is called *displacement mapping*. How is this fundamentally different than bump mapping?



The geometry must be displaced before visibility is determined.