CS451 Texturing 2

Bumpmap + Basic Lighting

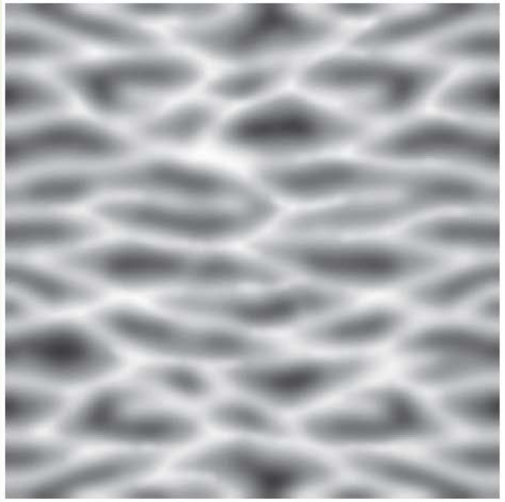
Jyh-Ming Lien

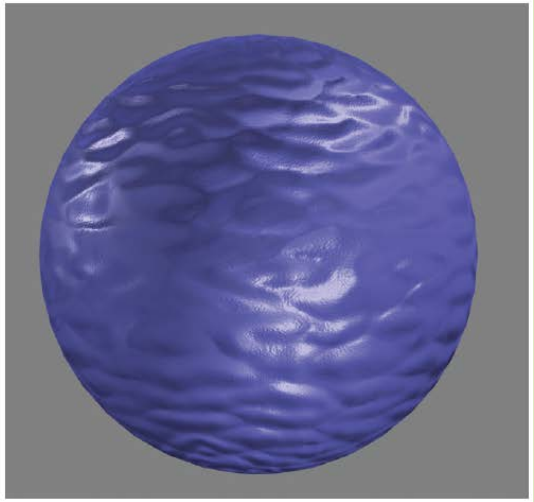
Department of Computer SCience

George Mason University

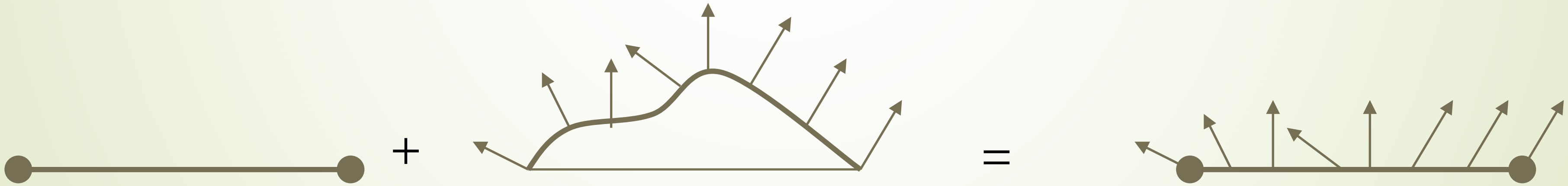
Based on Tomas Akenine-Möller's lecture note

Bump mapping





- by Blinn in 1978
- Inexpensive way of simulating wrinkles and bumps on geometry
  - Too expensive to model these geometrically
- Instead let a texture modify the normal at each pixel, and then use this normal to compute lighting

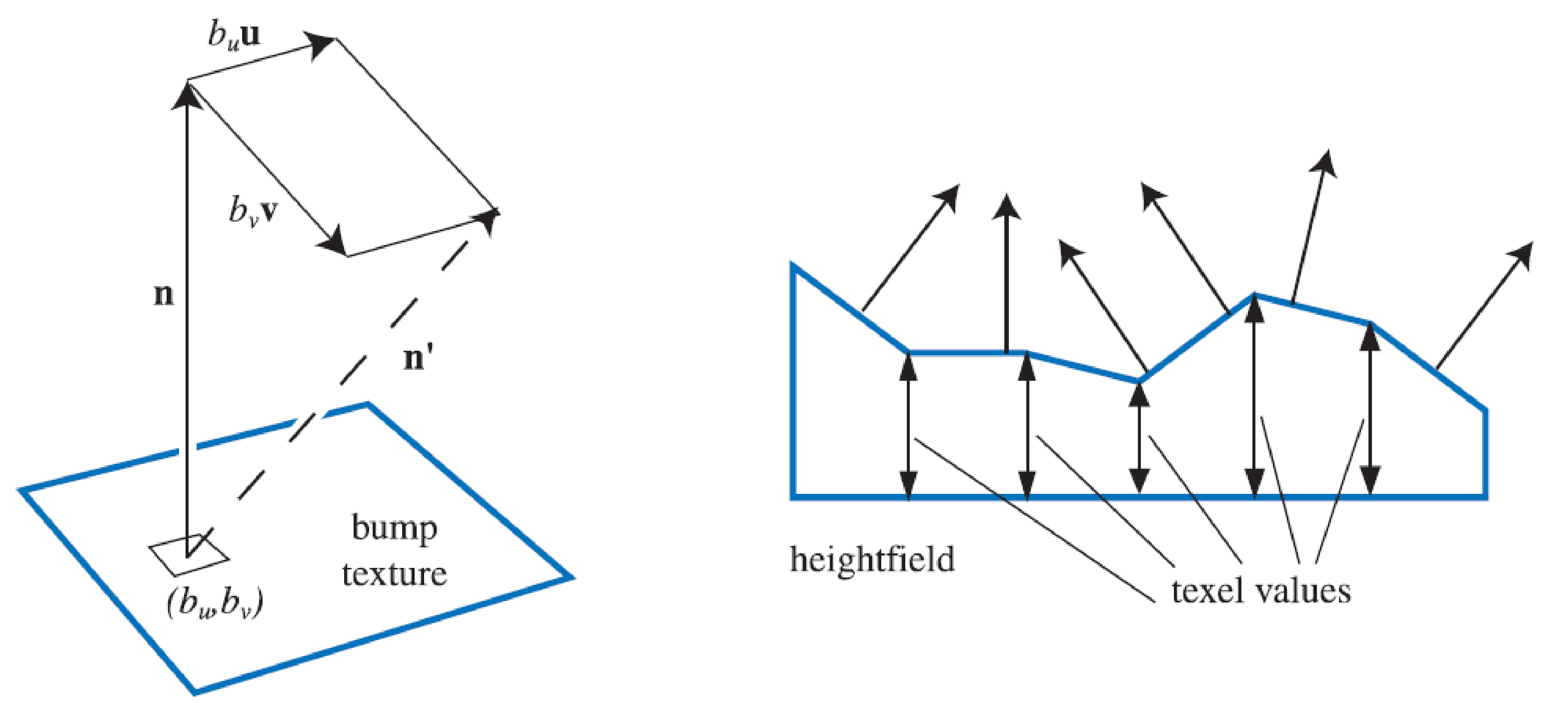


geometry

Bump map
Stores heights:
can derive normals

Bump mapped geometry

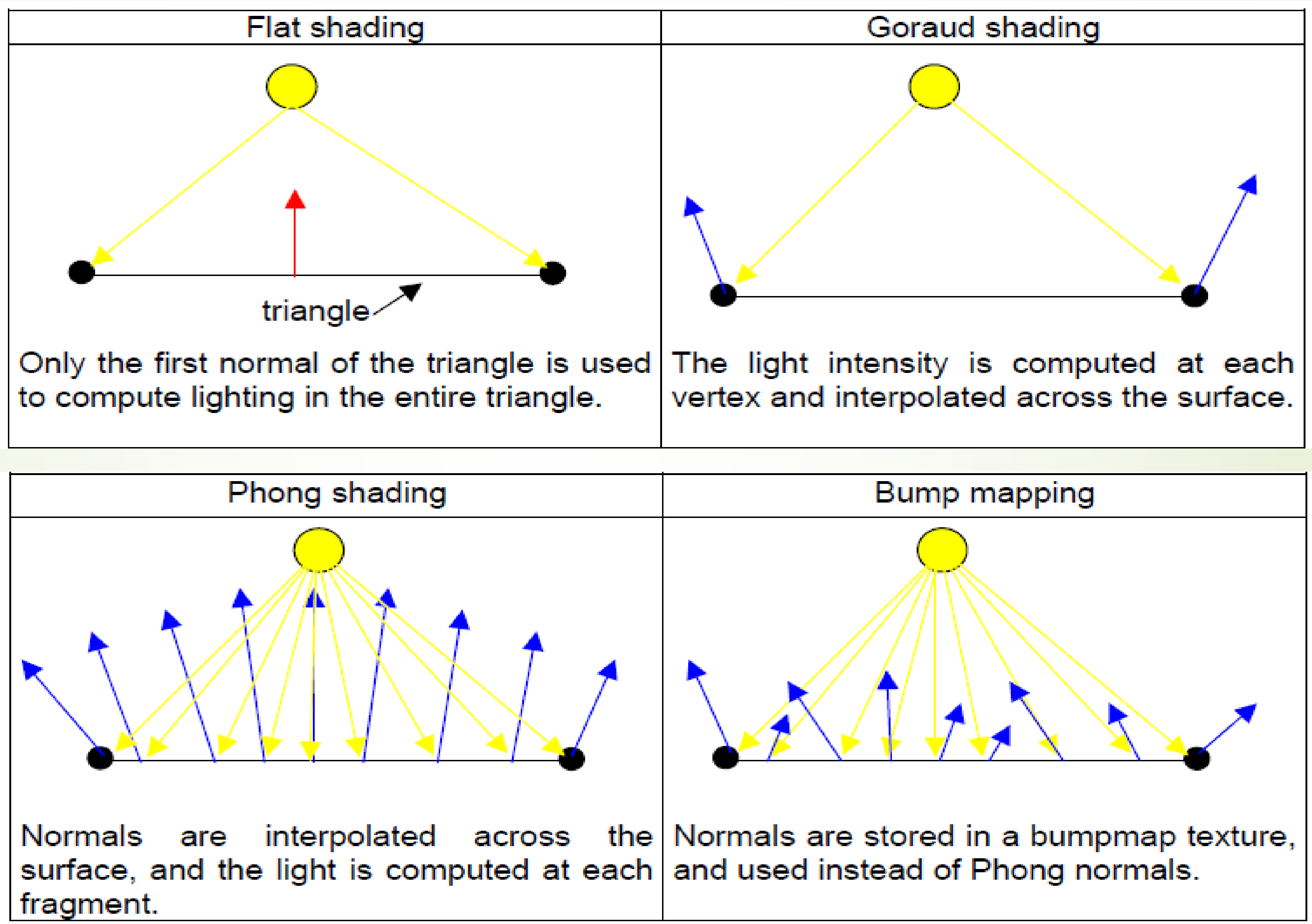
Normal directions in Bump Mapping



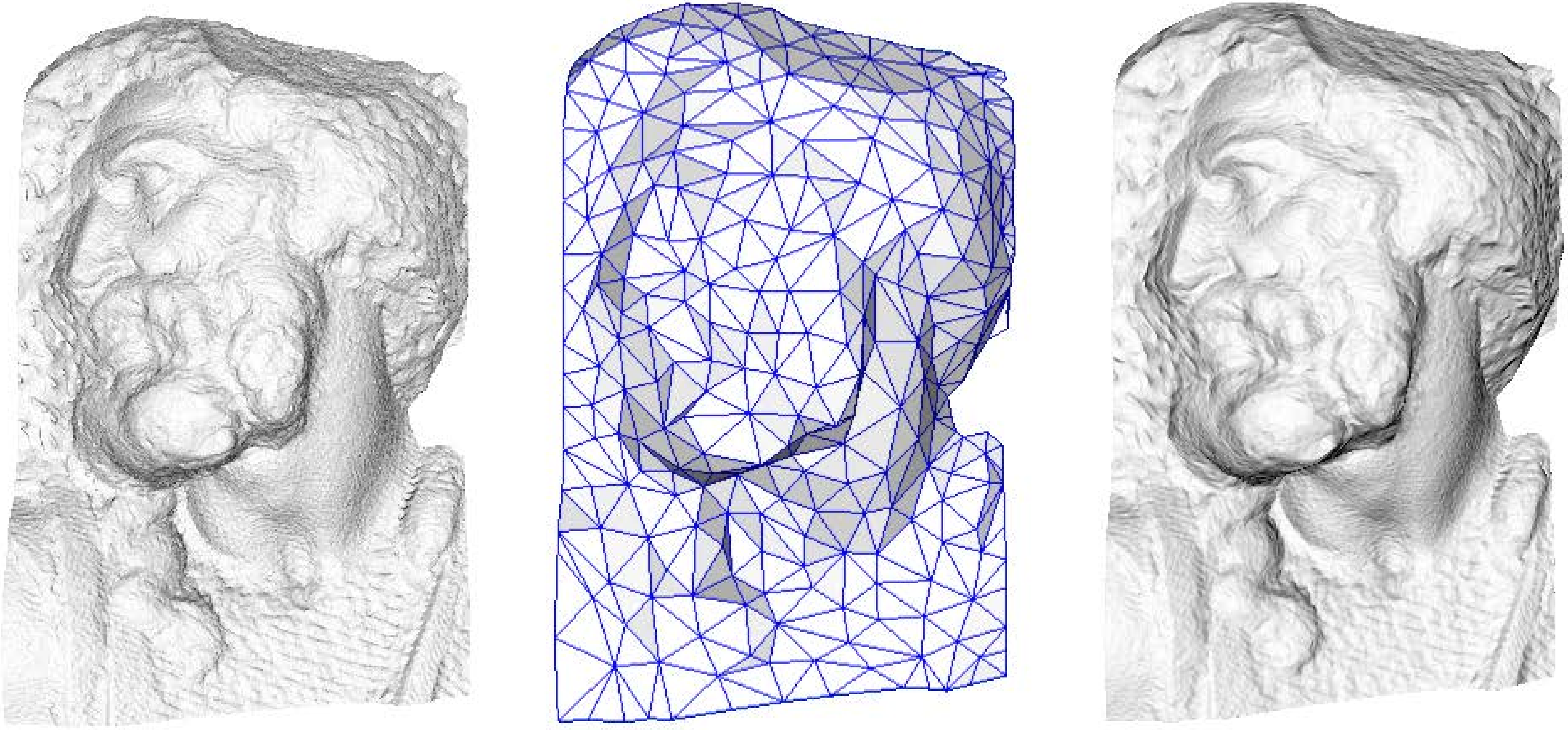
Approach #1
Offset vector map
Each pixel stores (b_w,b_v)

Approach #2
Height field, each pixel stores how high the center point is

Comparing to other Shading Methods



Flat shading	Goraud shading
Only the first normal of the triangle is used to compute lighting in the entire triangle.	The light intensity is computed at each vertex and interpolated across the surface.
Phong shading	Bump mapping
Normals are interpolated across the surface, and the light is computed at each fragment.	Normals are stored in a bumpmap texture, and used instead of Phong normals.

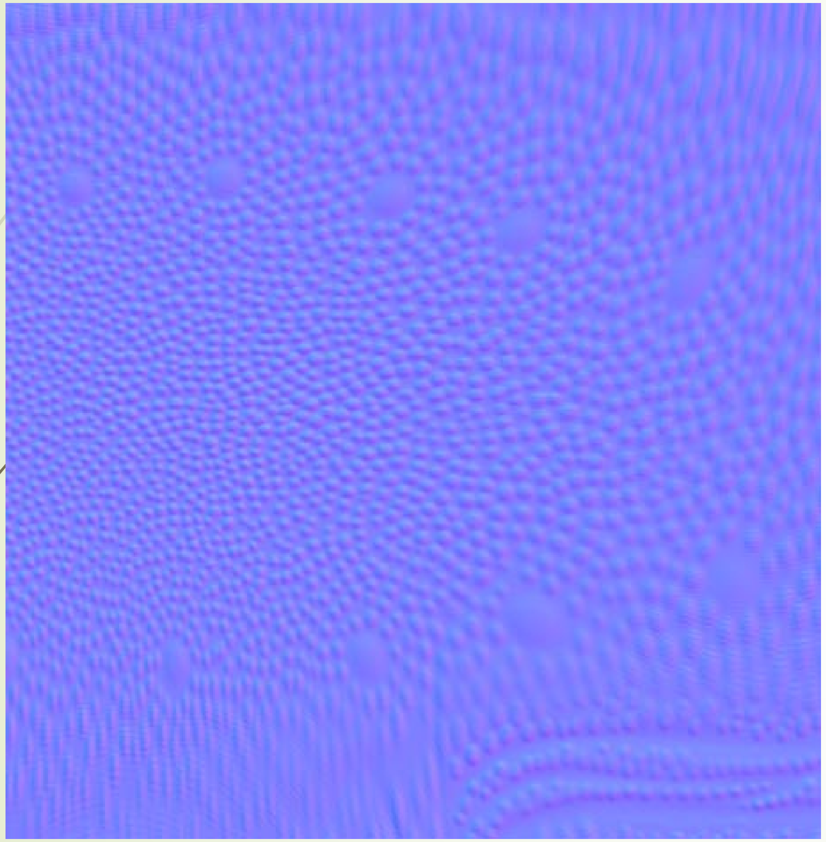


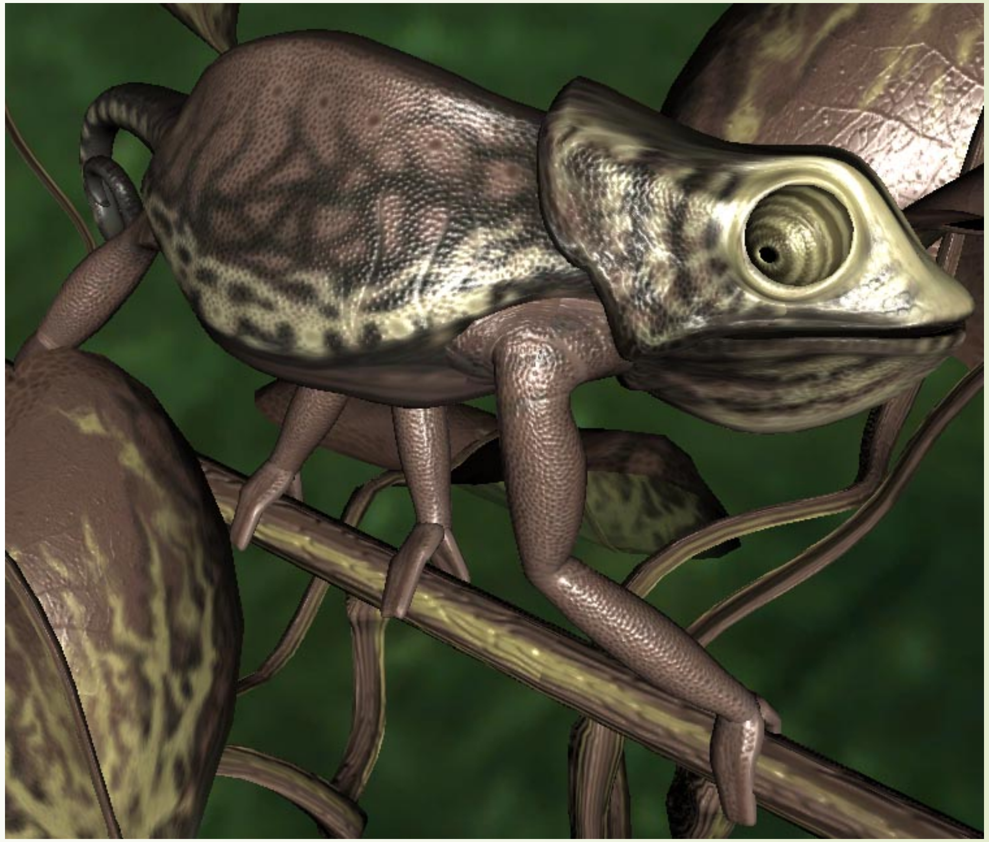
original mesh
4M triangles

simplified mesh
500 triangles

simplified mesh
and normal mapping
500 triangles

Normal mapping

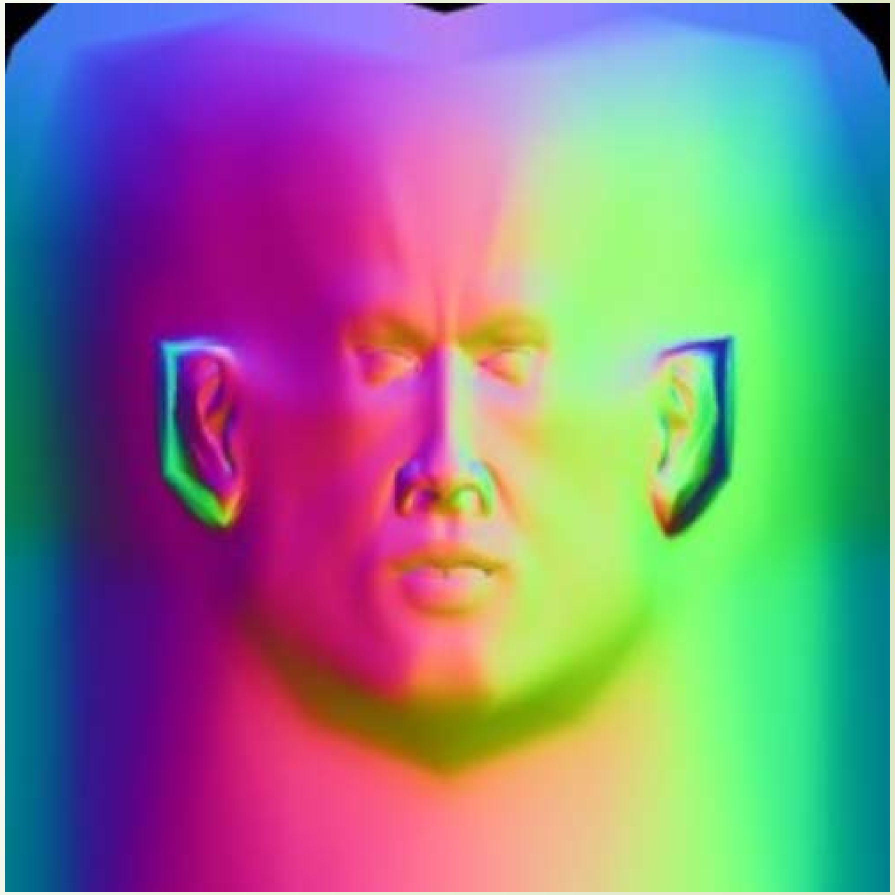




Normal map
Each pixel stores perturbed direction (x,y,z)

Normal mapping in RGB

- Store normals in texture
  - $n = (n_x, n_y, n_z)$ are in [-1,1]
  - $n = \left(\frac{n_x+1}{2}, \frac{n_y+1}{2}, \frac{n_z+1}{2}\right)$ in [0,1]
  - Mult by 255 (8 bit per color component)

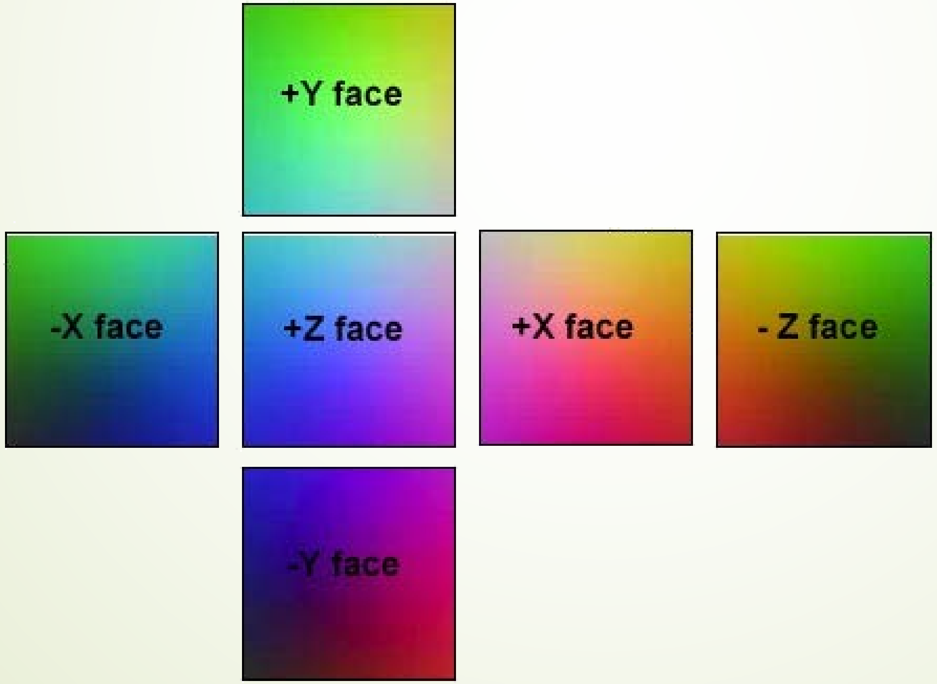


Spaces where your Normal map lives

- **World space** normal map
  - Each normal direction stored in texel is a world space vector
  - Usually applied to object using cube mapping
  - Rarely used for things that move
- **Object space** normal map
  - Each normal direction stored in texel is a vector in the space of the model
  - Each vertex must have unique (u,v) coordinates
- **Tangent space** normal map
  - Allow reuse of a normal map texture across multiple parts of the model

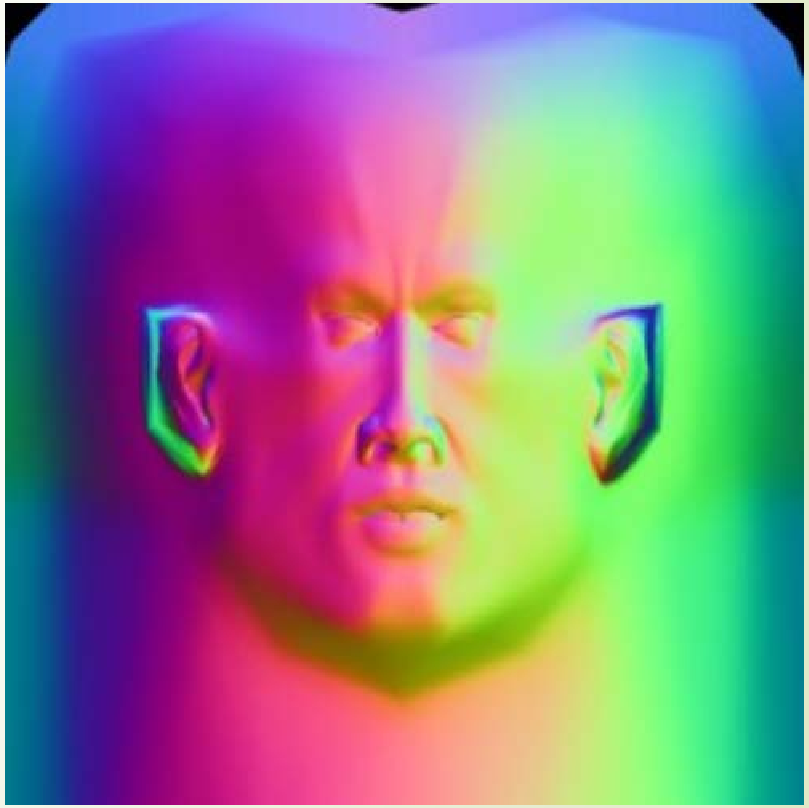
Normal mapping in World Space

- Each normal direction stored in texel is a world space vector
- Usually combine with cubemap texture



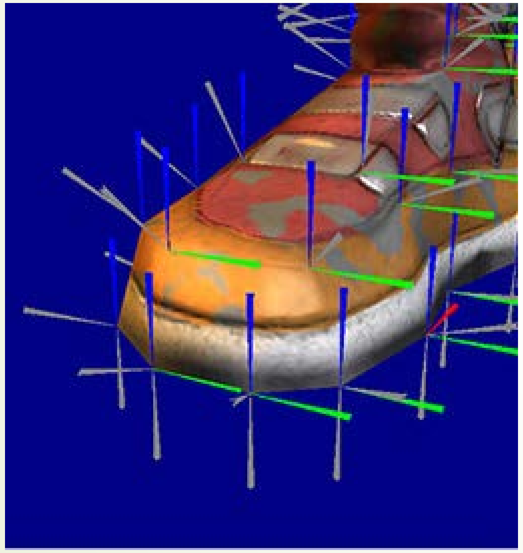
Object-Space Normal Mapping

- 3d vector encoded
- Simpler to implement
- Reuse limited to
  - translation
  - Scaling
  - Rotation

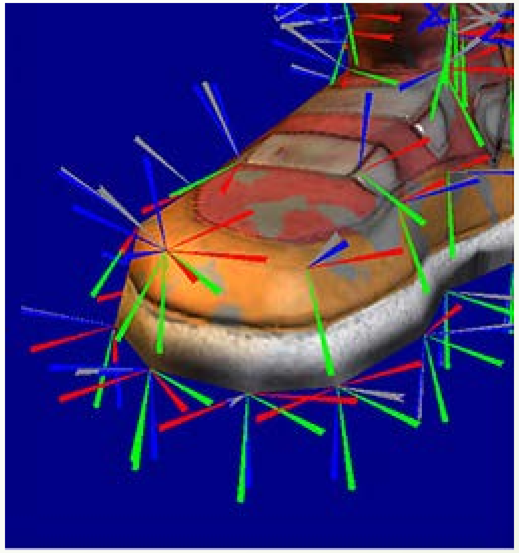


Comparison

- The spaces defined by each vertex

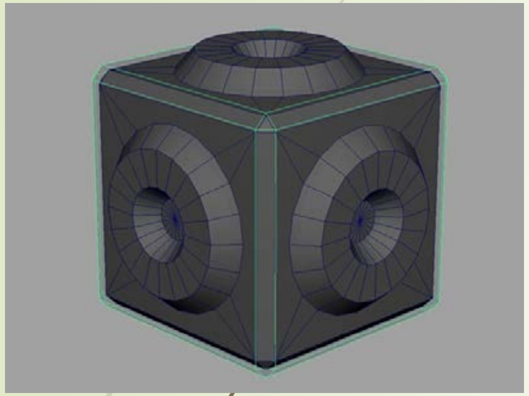


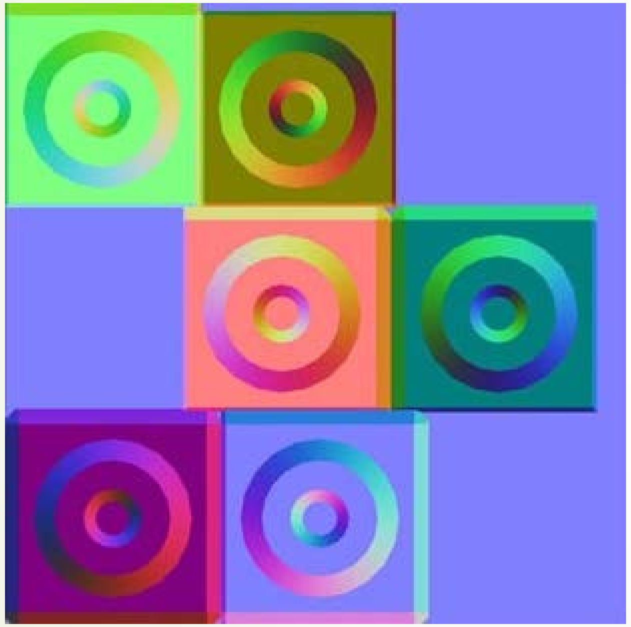
object space



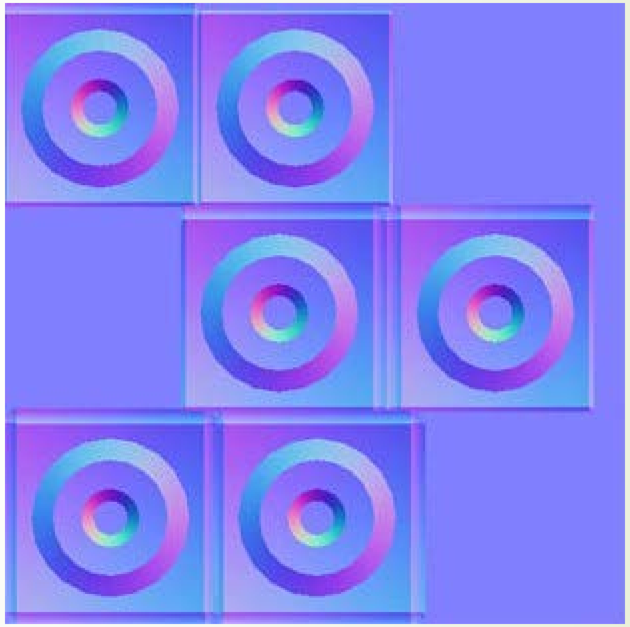
Tangent space

Comparison





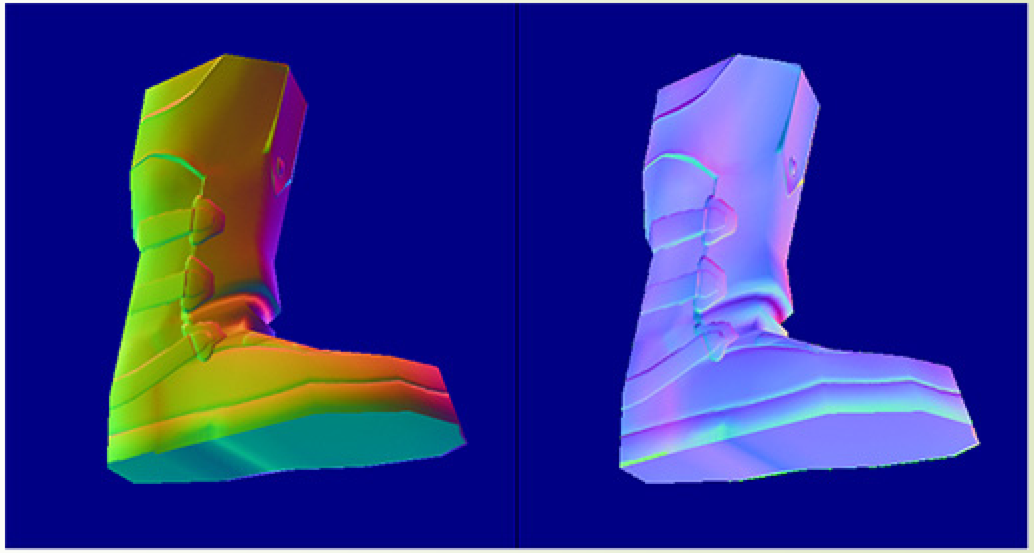
Object space



Tangent space

Advantages of Tangent Space

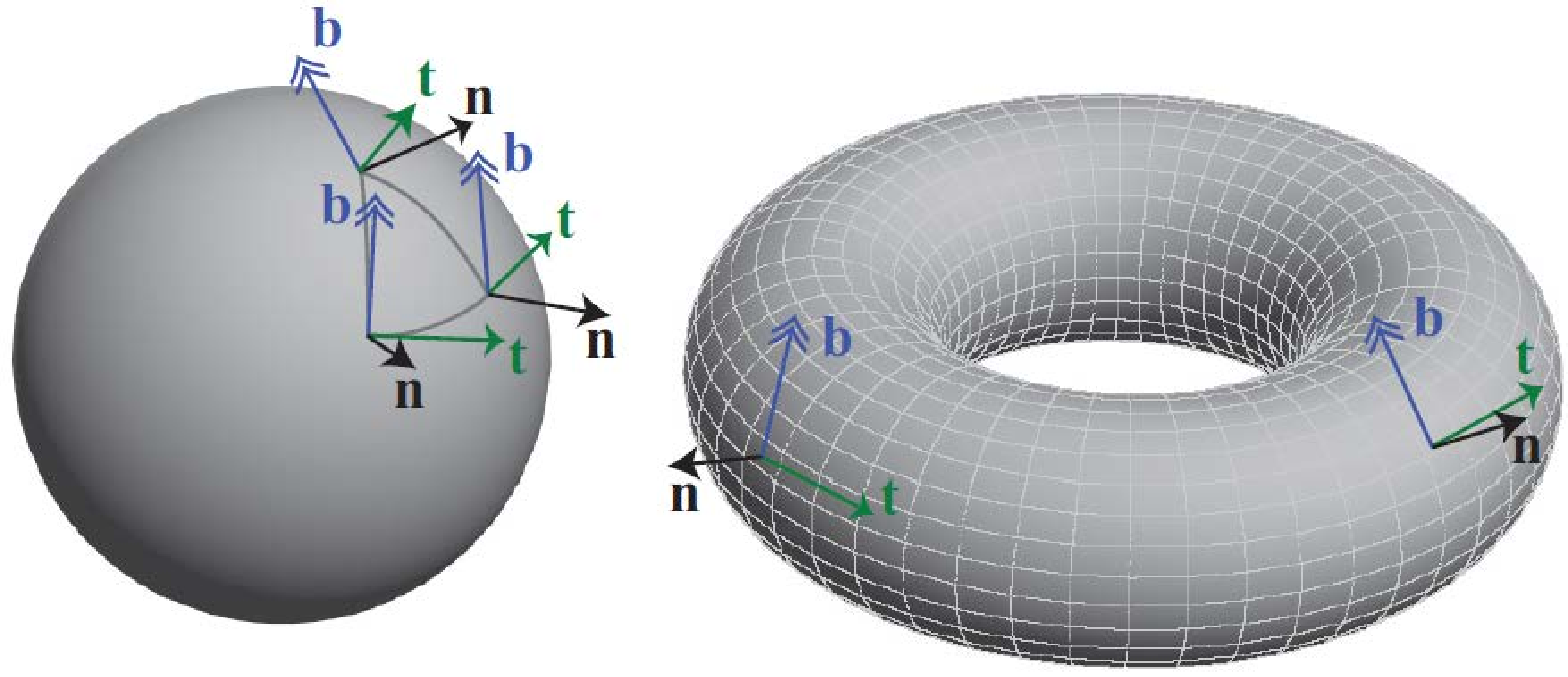
- Efficient
- Support for mirroring
- Tiling textures
- Higher image compression rate



object space Tangent space

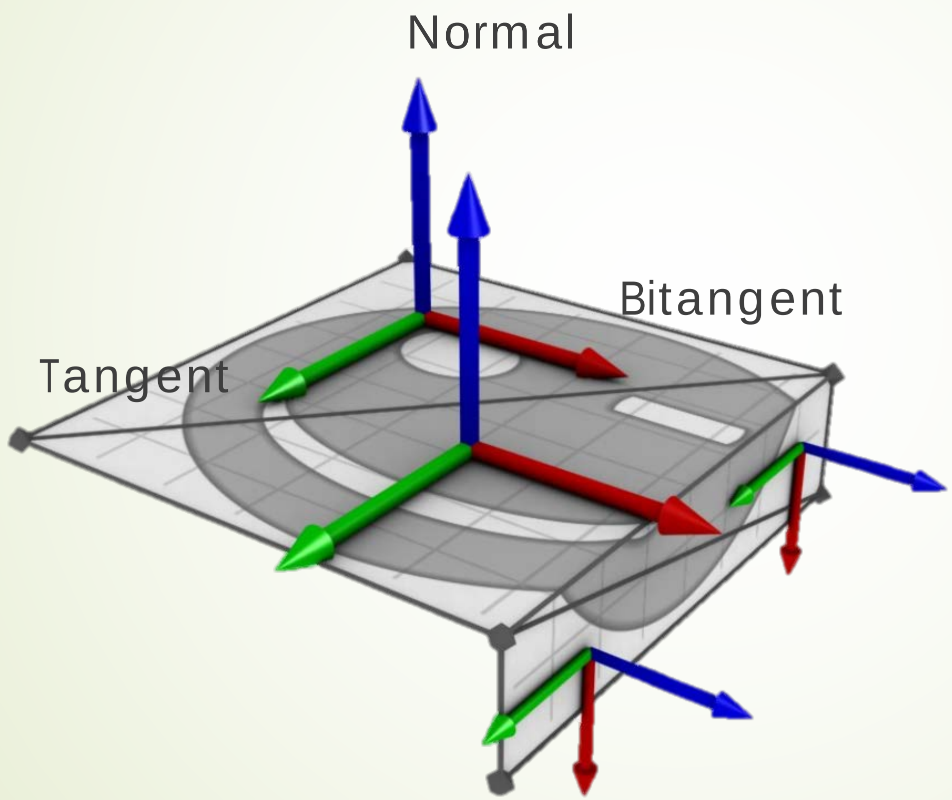
Tangent Space

- Normal, tangent, Bi-tangent



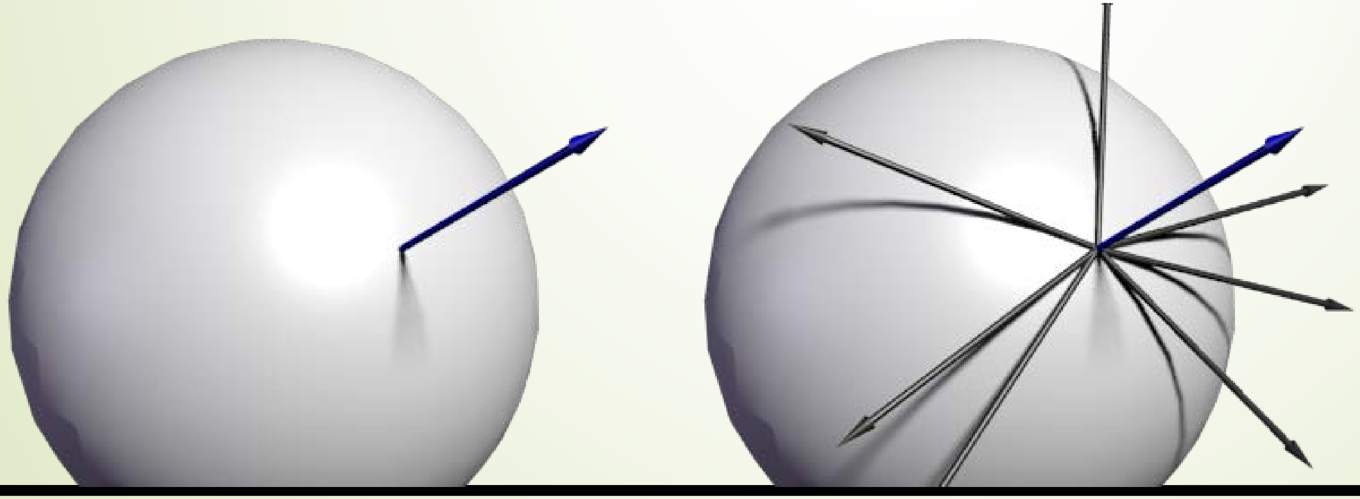
Tomas Akenine-Mől
ler © 2002

Tangent Space Per Triangle



How is Tangent Space Computed?

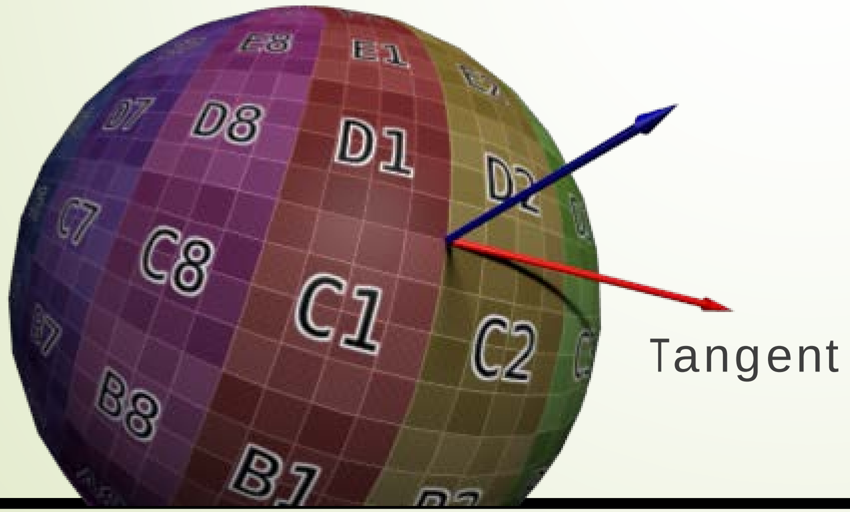
- Normal : perpendicular to the plane
- Tangent and bitangent are parallel to the plane
  - Tangent and bitangent are perpendicular to the normal
  - **There are many possible tangents and bitangents**

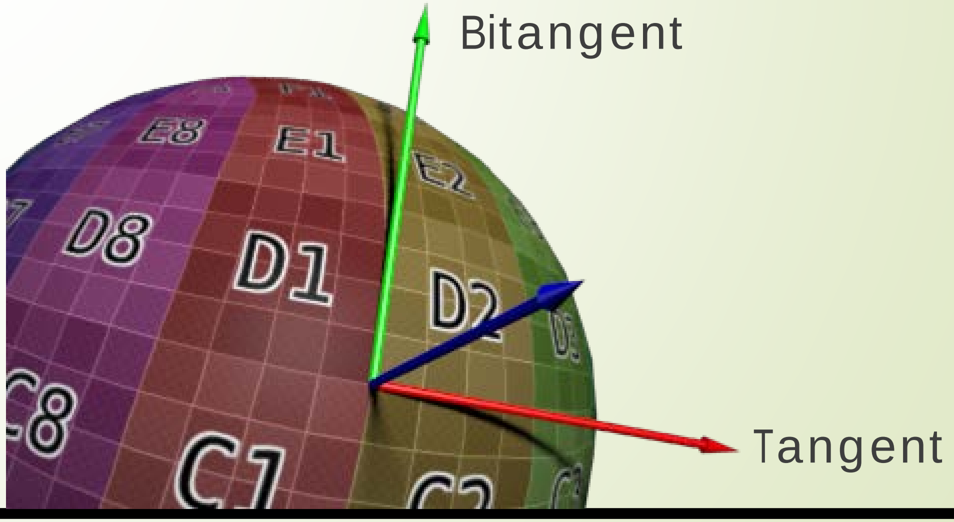


http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

How is Tangent Space Computed?

- Goal: determined a unique and smooth **BTN** by the **UV** coordinates
  - one points in the direction of U-axis in 3d space
  - the other in the direction of the V-axis
- Tangent space normal map stores the **length** of each vector

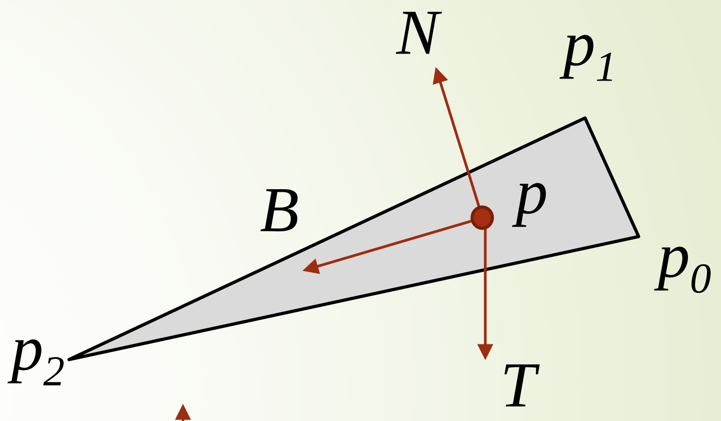


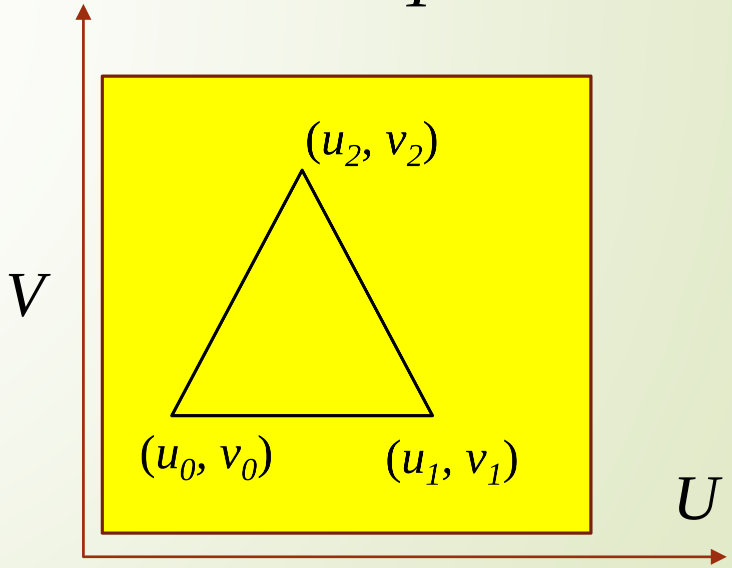


http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/

How is Tangent Space Computed?

- Goal: $P - P_0 = (u - u_0)T + (v - v_0)B$

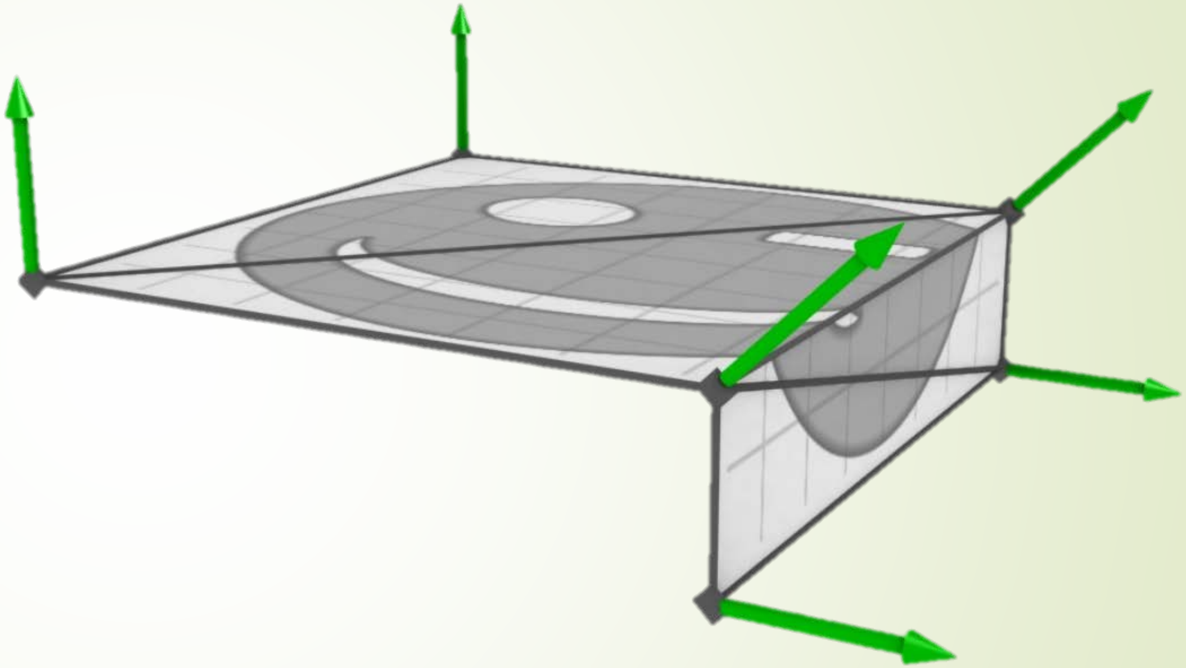




Note:

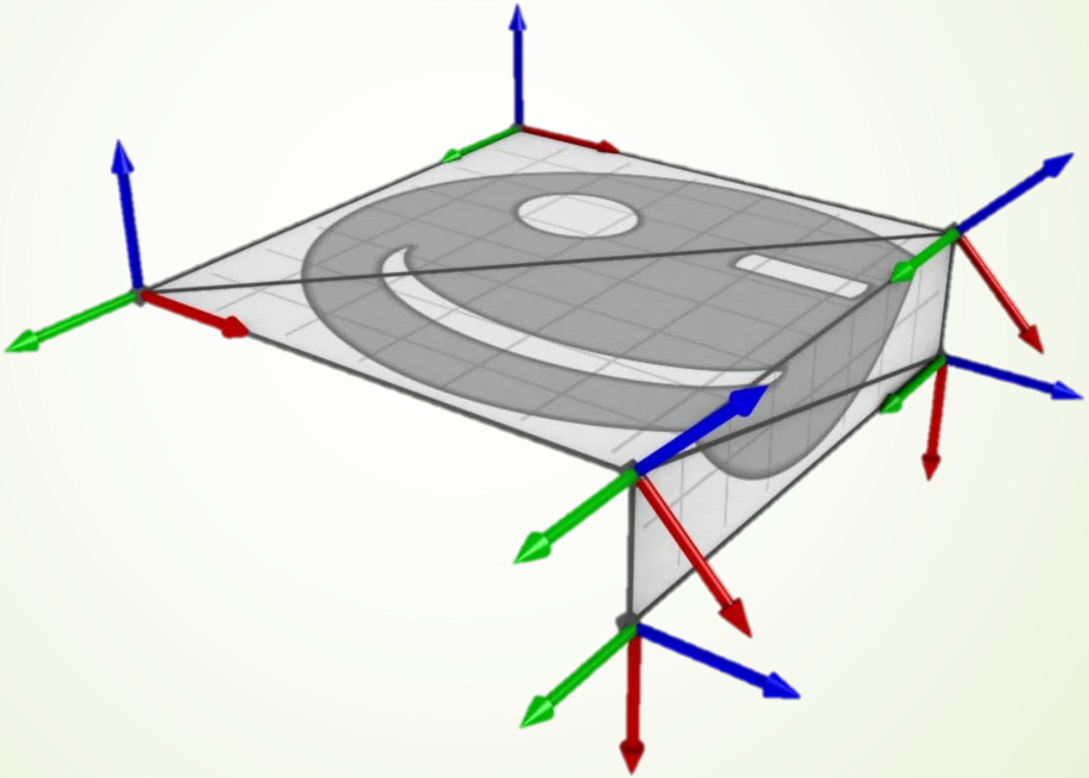
$$\begin{bmatrix} s_1 & t_1 \\ s_2 & t_2 \end{bmatrix}^{-1} = \frac{1}{s_1 t_2 - s_2 t_1} \begin{bmatrix} t_2 & -t_1 \\ -s_2 & s_1 \end{bmatrix}$$

Per Vertex Normal



- How do you compute per vertex normal?
- Is the normal affected by tessellation?

Per Vertex Tangent Space



tangent space

$$\begin{bmatrix} T_1 & B_1 & N_1 \\ T_2 & B_2 & N_2 \\ T_3 & B_3 & N_3 \end{bmatrix}$$

How do you convert a light source to tangent space?

What's Missing?

- There are no bumps on the silhouette of a bump-mapped object

- Bump maps don’t allow self-occlusion or self-shadowing