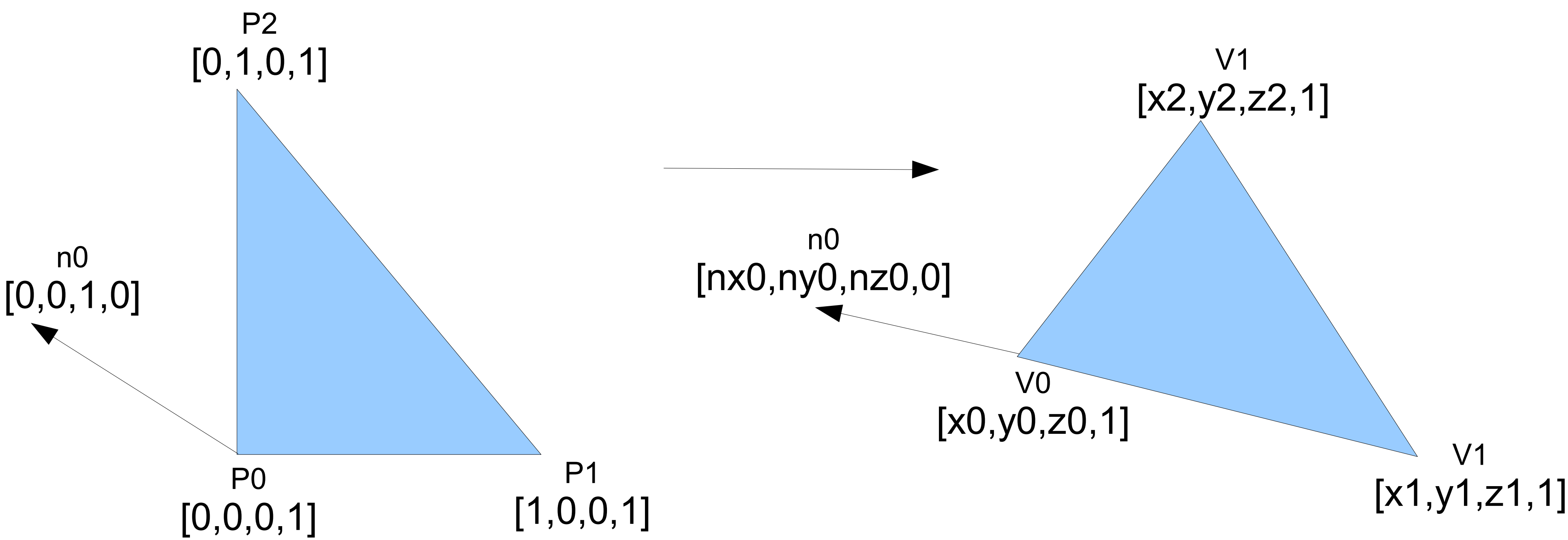
Space Transform



Space Transform Matrix

$$M=\begin{bmatrix} a00 & a01 & a02 & 0 \\ a10 & a11 & a12 & 0 \\ a20 & a21 & a22 & 0 \\ a30 & a31 & a32 & 1 \end{bmatrix}$$

$$P0 \times M = V0$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a00 & a01 & a02 & 0 \\ a10 & a11 & a12 & 0 \\ a20 & a21 & a22 & 0 \\ a30 & a31 & a32 & 1 \end{bmatrix} = \begin{bmatrix} x0 & y0 & z0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a30 & a31 & a32 & 1 \end{bmatrix} = \begin{bmatrix} x0 & y0 & z0 & 1 \end{bmatrix}$$

$$M=\begin{bmatrix} a00 & a01 & a02 & 0 \\ a10 & a11 & a12 & 0 \\ a20 & a21 & a22 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

$$P1 \times M = V1$$

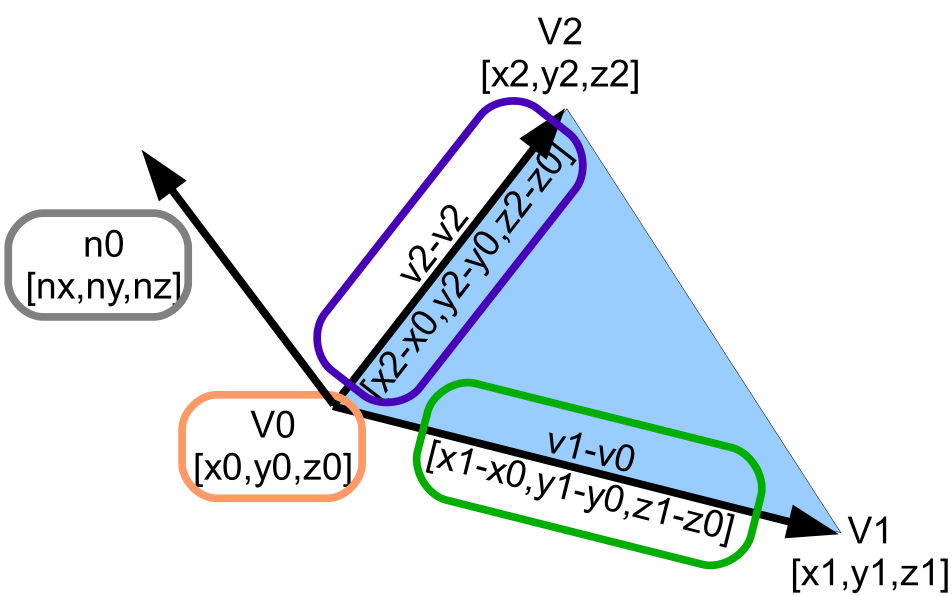
$$\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a00 & a01 & a02 & 0 \\ a10 & a11 & a12 & 0 \\ a20 & a21 & a22 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix} = \begin{bmatrix} x1 & y1 & z1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a00+x0 & a01+y0 & a02+z0 & 1 \end{bmatrix} = \begin{bmatrix} x1 & y1 & z1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a00+x0=x1 \\ a01+y0=y1 \\ a02+z0=z1 \end{bmatrix} = \begin{bmatrix} a00=x1-x0 \\ a01=y1-y0 \\ a02=z1-z0 \end{bmatrix} \quad M=\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ a10 & a11 & a12 & 0 \\ a20 & a21 & a22 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

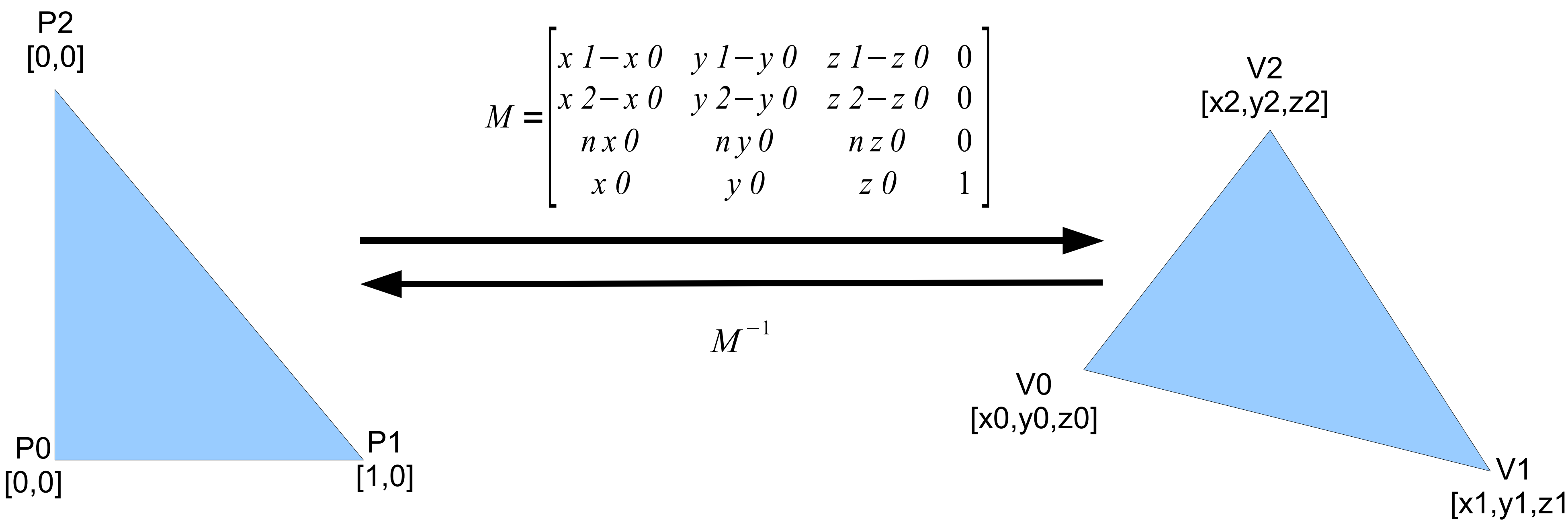
$$M=\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

Space Transform Matrix



$$M = \begin{bmatrix} x1 - x0 & y1 - y0 & z1 - z0 & 0 \\ x2 - x0 & y2 - y0 & z2 - z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

Space Transform Matrix



$$M = \begin{bmatrix} x1 - x0 & y1 - y0 & z1 - z0 & 0 \\ x2 - x0 & y2 - y0 & z2 - z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

$$M^{-1}$$

Illumination models

- Local
  - Blinn-Phong
  - Cook-Torrance
- Global
  - Ray-Tracing
  - Radiosity
  - Photon Mapping
- They all need a normal vector at illumination point !

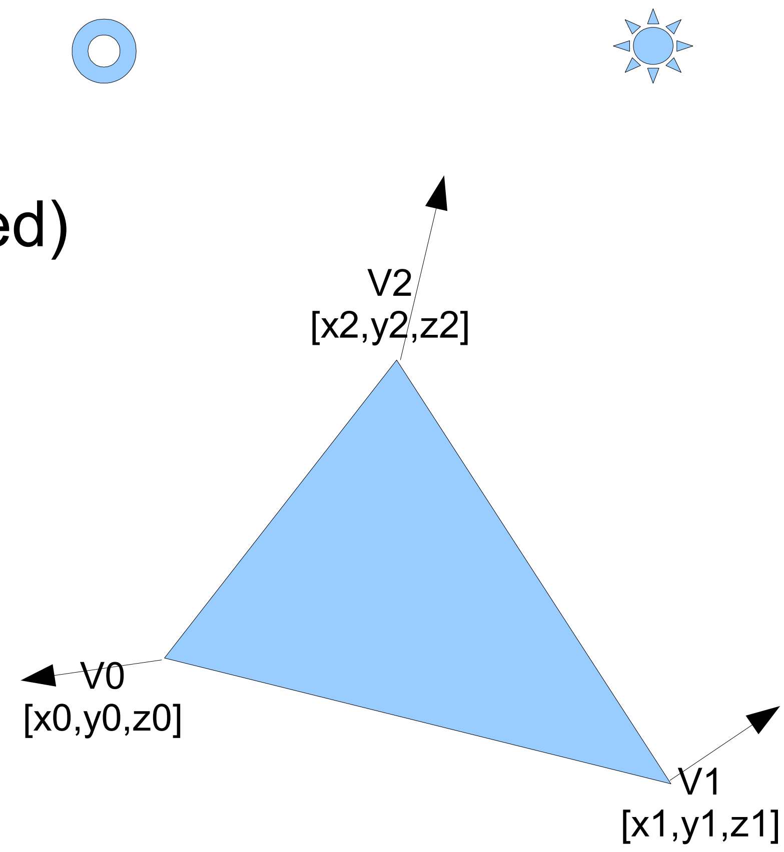
- Normal vector at a point on surface
  - Perpendicular to surface at that point
- Simulation of Curved surface
  - Normal vectors at polygon vertices
- Interpolation across surface
  - Simulation of surface curvature

Space Transforms

- World Space
  - Lights, Cameras
- Model (Object) Space
  - Vertices and normals of a 3D model
  - per-Model Matrix (Model-to-World transform)
- View (Camera) Space
  - Viewer position is in [0,0,0,1]
  - World-to-View transform matrix

Simple model
per-vertex lighting

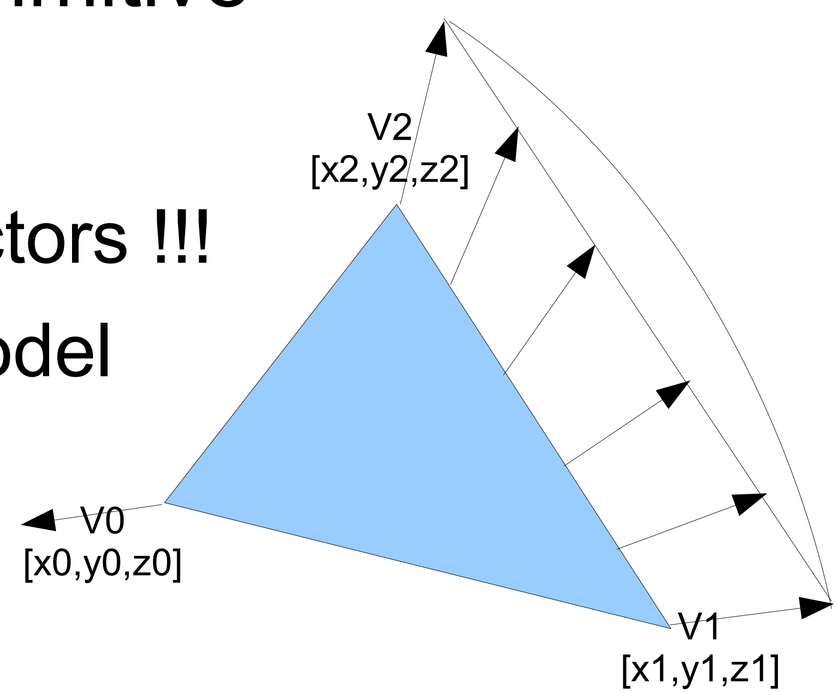
- Per-object
  - Transform from WS to MS
    - Light & View Position
- Per-vertex
  - Vectors in Model Space (normalized)
    - Normal vector at Vertex
    - Vertex - Light Position
    - Vertex - View Position
  - calculate local illumination model
    - Store as color
    - Interpolate across primitive
- Fast but Inaccurate
  - Needs finer tessellation



Simple model

per-fragment (per-pixel) lighting

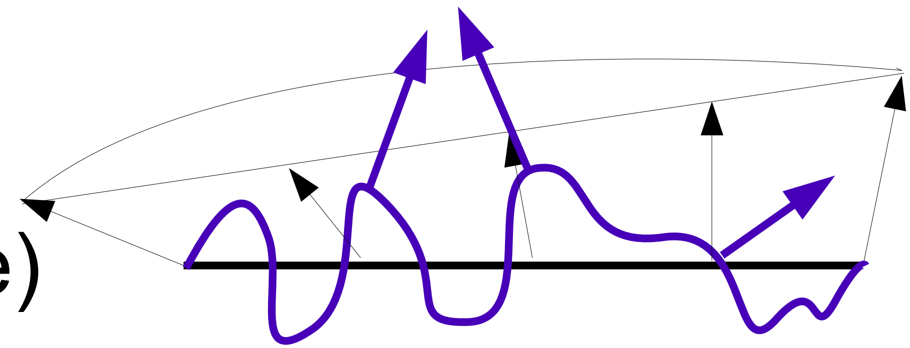
- Per-vertex vectors in model space
  - Normal vector at Vertex (normalized)
  - Vertex - Light Position (do NOT normalize !)
  - Vertex - View Position (do NOT normalize !)
- Interpolate vectors across primitive
- Per-fragment
  - Normalize all interpolated vectors !!!
  - calculate local illumination model
- More intensive



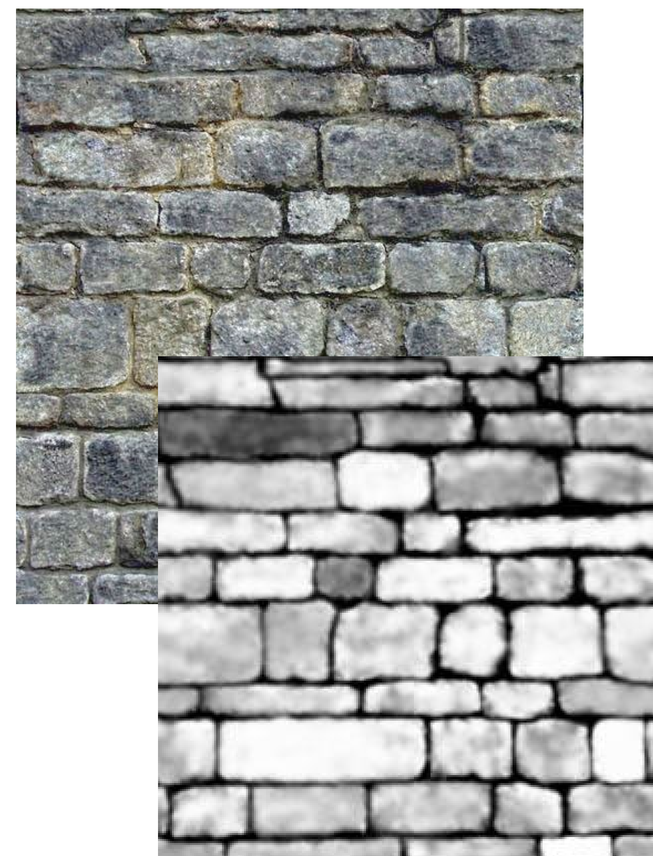
Shaders

- Smooth illumination
- Finer detail
  - Detailed color texture
  - Finer illumination ?
  - Bump mapping
    - Blinn, James F. "Simulation of Wrinkled Surfaces" 1978

Bump Mapping

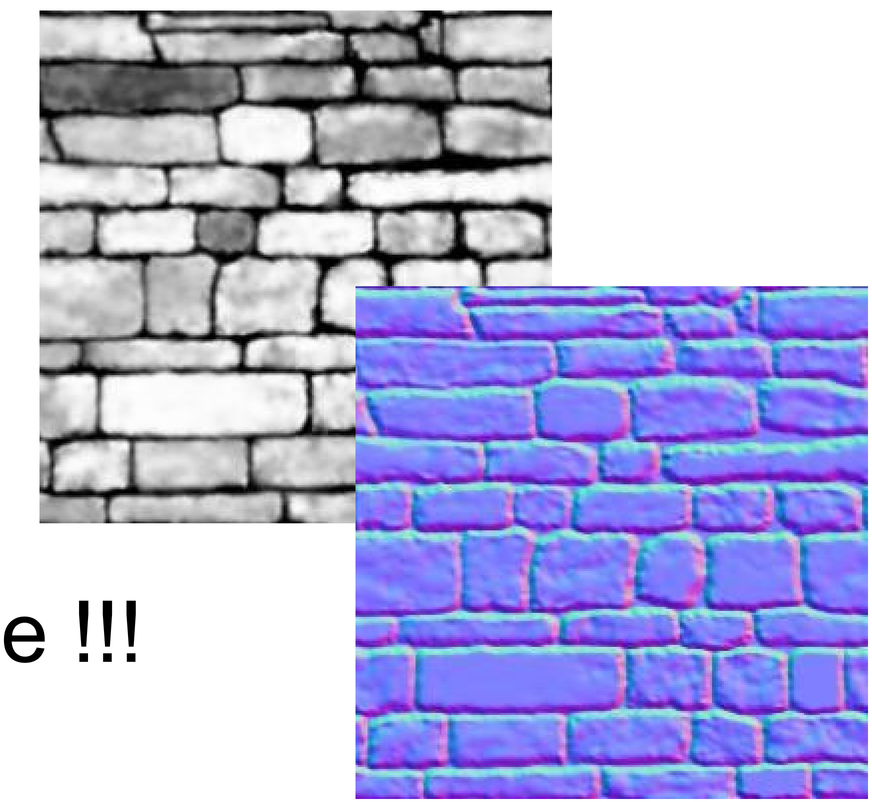


- Height Map
- Per-pixel (in Model Space)
  - Interpolated vectors, normalize
  - Look into height-map at given surface position and neighbors
  - Estimate bump normal
  - Perturb interpolated normal with bump normal, normalize
  - calculate local illumination model



Normal Mapping

- NormalMap
  - For every texel one 3D normal
- Convert HeightMap to NormalMap
  - For every texel
    - compute gradient
      - Central differences
    - Normalize gradient
    - Range compress into 8 bpp
      - (N+1.0)*255.0
- Normals are in UVW space !!!



Normal Mapping
in object space

- Per-Vertex
  - Object space
    - Vertex - Light Position (L) - do NOT normalize !
    - Vertex - View Position (V) - do NOT normalize !
  - Calculate UVW to Object Space matrix
    - T,B,N vectors
- Interpolate vectors L,V, T,B,N across primitive

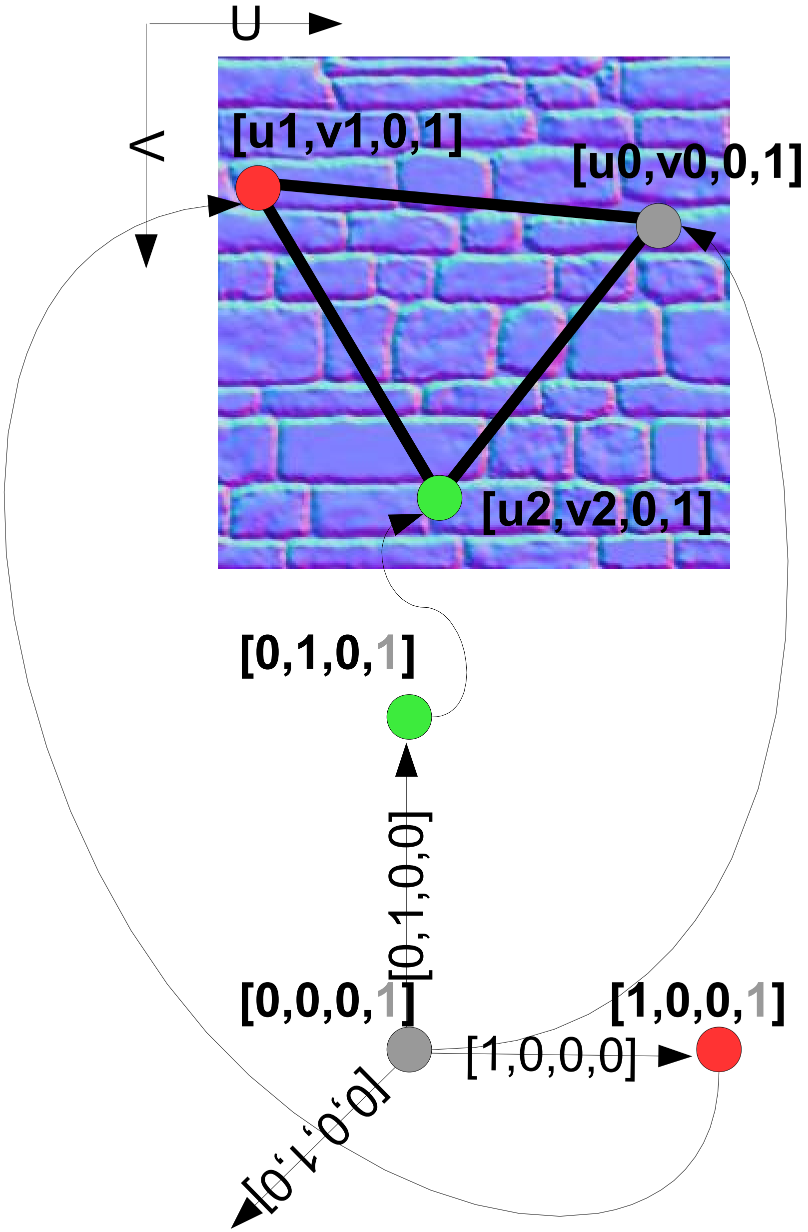
Normal Mapping
in object space

- **Per-Vertex**
  - **Object space**
    - **Vertex - Light Position (do NOT normalize !)**
    - **Vertex - View Position (do NOT normalize !)**
  - **Calculate UVW to Object Space matrix**
    - **T,B,N vectors**
- **Interpolate all vectors across primitive**
- Per-fragment
  - Sample compressed normal from normalmap
  - Uncompress n=n*2.0 – 1.0
  - Transform n to Model Space (T,B,N matrix)
    - Matrix multiply per-fragment !!!
  - Normalize vectors !
  - Calculate local illumination model

UVW to Object Space Matrix

- "to UVW" matrix

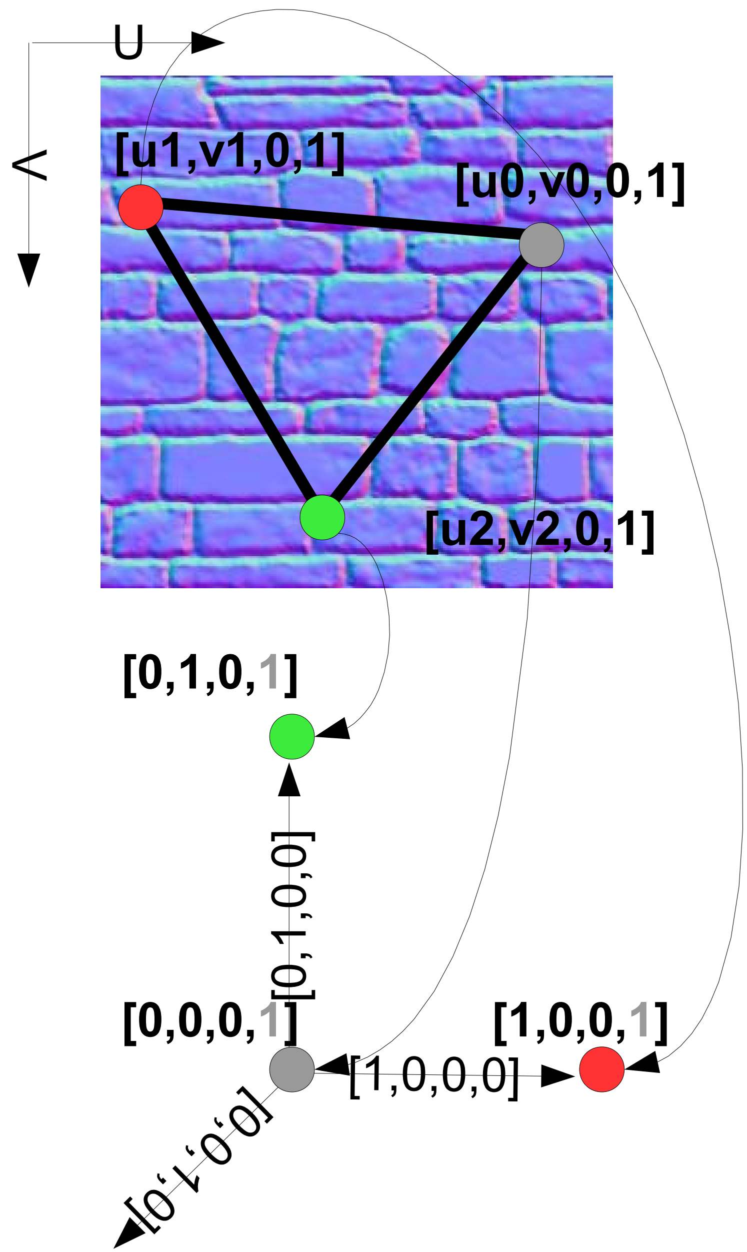
$$UVW = \begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u0 & v0 & 0 & 1 \end{bmatrix}$$



UVW to Object Space Matrix

- “from UVW” matrix

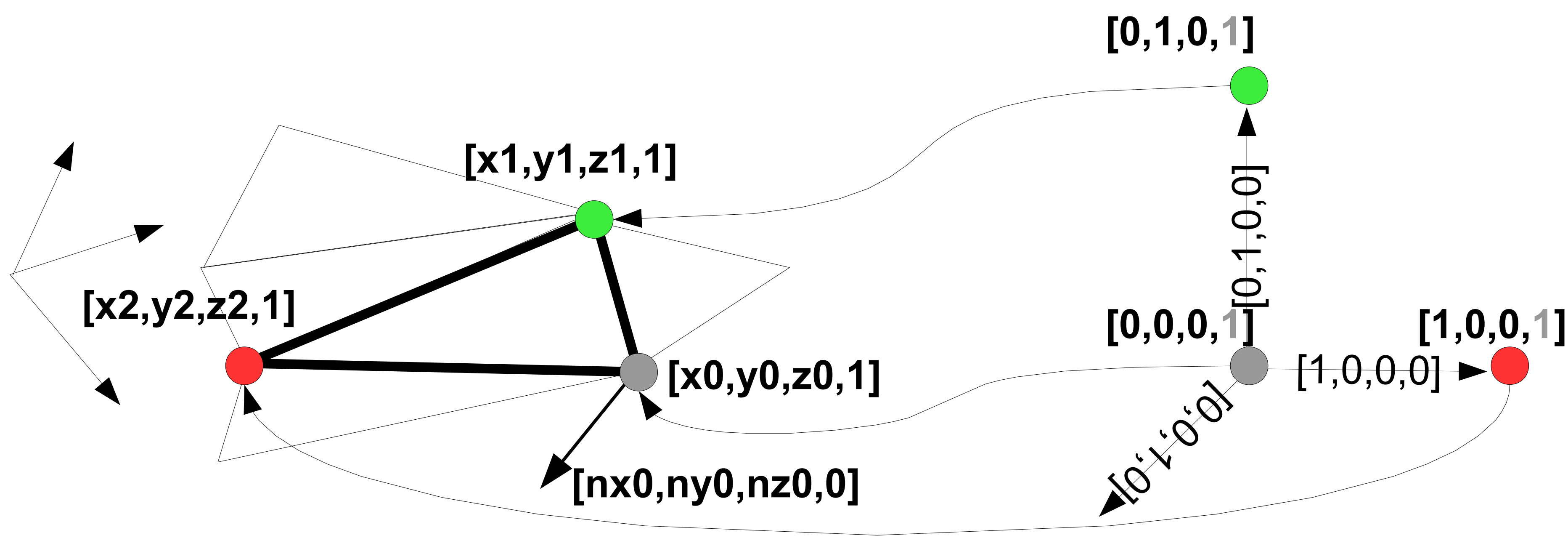
$$UVW^{-1} = \begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u0 & v0 & 0 & 1 \end{bmatrix}^{-1}$$



UVW to Object Space Matrix

- “to Triangle in object space” matrix

$$TriM=\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

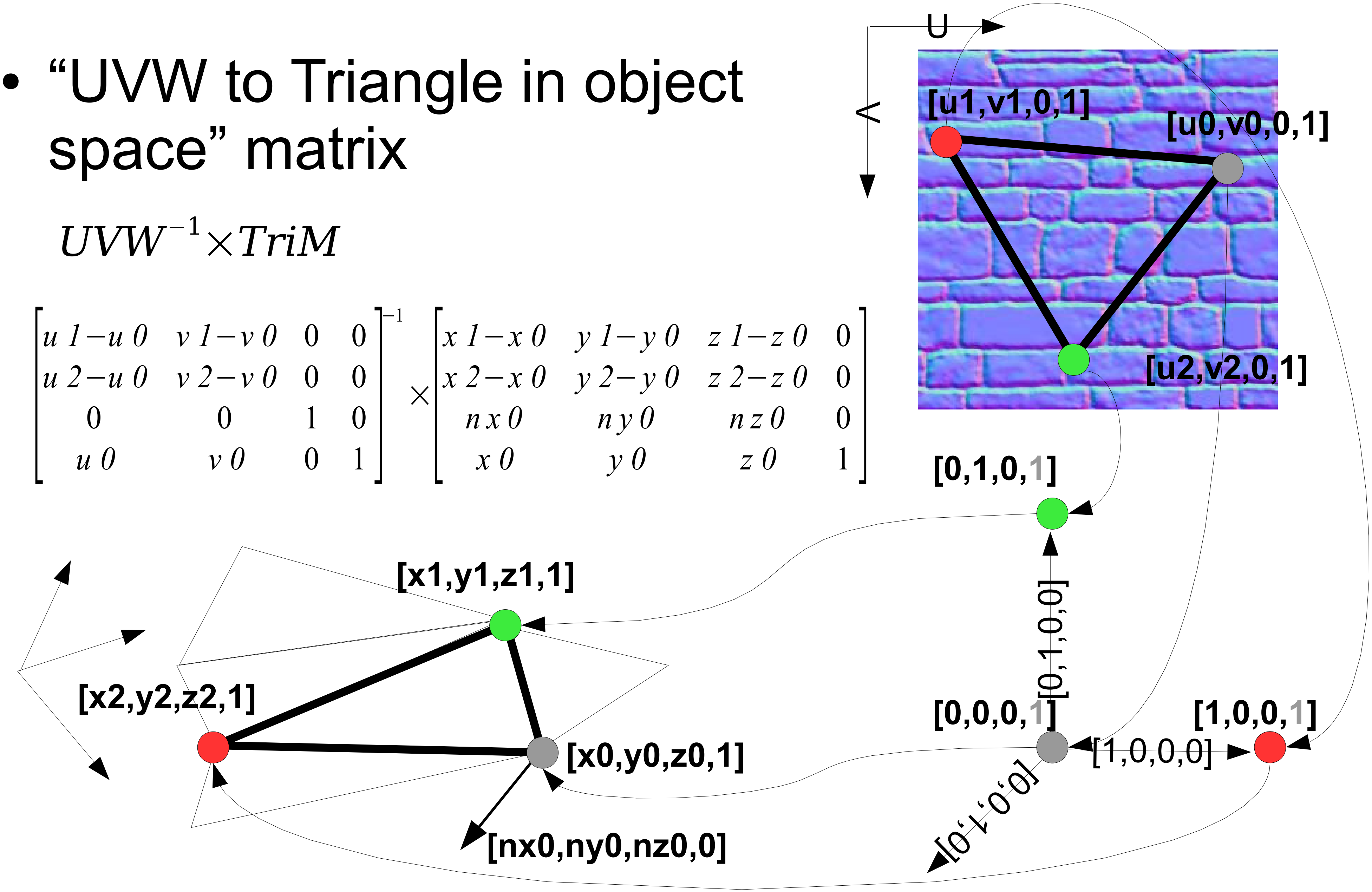


UVW to Object Space Matrix

- "UVW to Triangle in object space" matrix

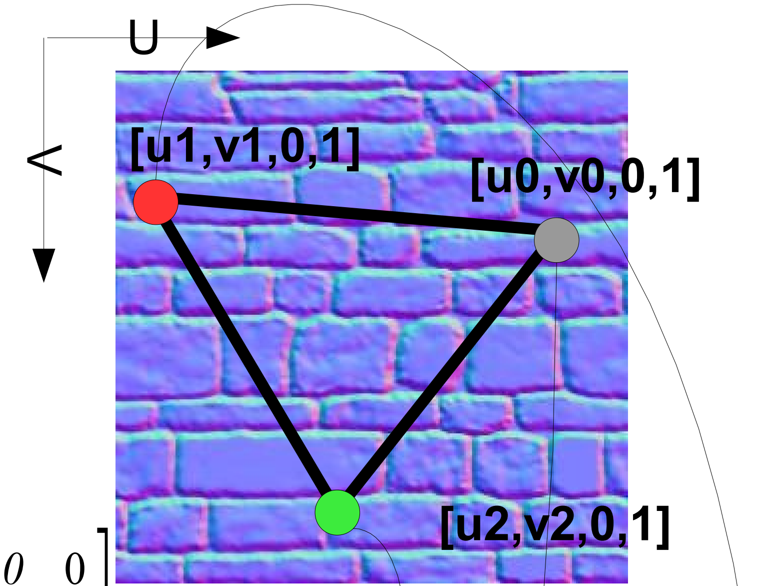
$$UVW^{-1} \times TriM$$

$$\begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u0 & v0 & 0 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$



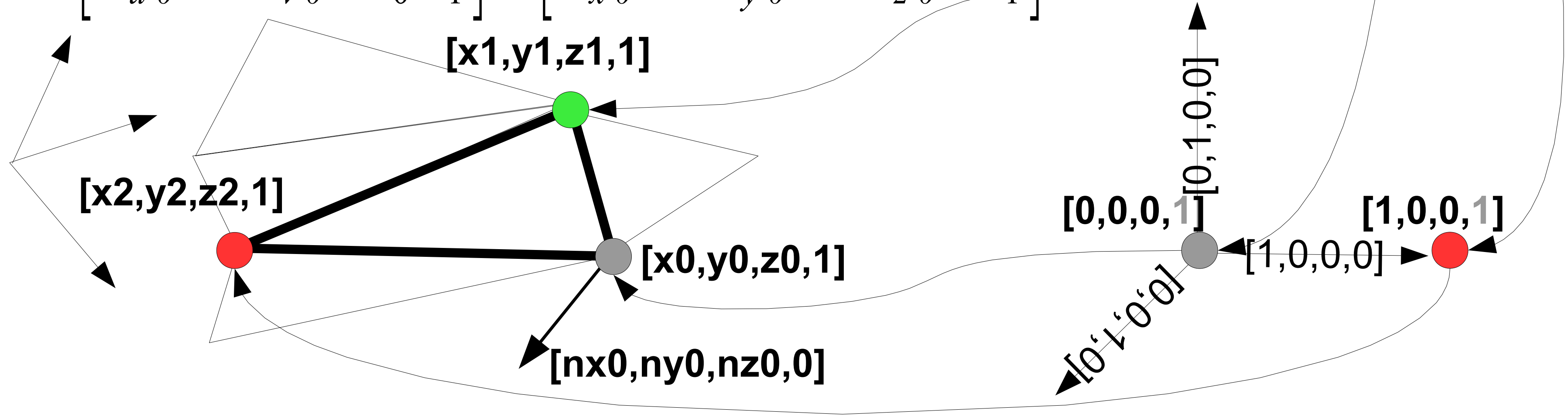
UVW to Object Space Matrix

- Sampled uncompressed normal n=[nx,ny,nz,0]
- Transform to object space



$$n \times UVW^{-1} \times TriM$$

$$n \times \begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ u0 & v0 & 0 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$



UVW to Object Space Matrix

- Transforming only vectors
  - n=[nx,ny,nz,0]
- Translation part set to zero

$$n\times\begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}\times\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \\ u2-u0 & v2-v0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1}\times\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \\ x2-x0 & y2-y0 & z2-z0 & 0 \\ nx0 & ny0 & nz0 & 0 \\ x0 & y0 & z0 & 1 \end{bmatrix}=\begin{bmatrix} T \\ B \\ N \end{bmatrix}$$

$$\begin{bmatrix} u & v & 0 & 1 \end{bmatrix}\times\begin{bmatrix} T \\ B \\ N \\ P \end{bmatrix}=\begin{bmatrix} x & y & z & 1 \end{bmatrix} \qquad \begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \end{bmatrix}\times\begin{bmatrix} T \\ B \\ N \\ P \end{bmatrix}=\begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \end{bmatrix}$$

UVW to Object Space Matrix

$$\begin{bmatrix} u1-u0 & v1-v0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} T \\ B \\ N \\ P \end{bmatrix} = \begin{bmatrix} x1-x0 & y1-y0 & z1-z0 & 0 \end{bmatrix}$$

$$(u1-u0)*Tx+(v1-v0)*Bx=x1-x0$$
$$(u1-u0)*Ty+(v1-v0)*By=y1-y0$$
$$(u1-u0)*Tz+(v1-v0)*Bz=z1-z0$$

$$\begin{bmatrix} u2-u0 & v2-v0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} T \\ B \\ N \\ P \end{bmatrix} = \begin{bmatrix} x2-x0 & y2-y0 & z2-z0 & 0 \end{bmatrix}$$

$$(u2-u0)*Tx+(v2-v0)*Bx=x2-x0$$
$$(u2-u0)*Ty+(v2-v0)*By=y2-y0$$
$$(u2-u0)*Tz+(v2-v0)*Bz=z2-z0$$

Normal Mapping
in tangent space

- Per-Vertex
  - Object space
    - Vertex - Light Position (do NOT normalize !)
    - Vertex - View Position (do NOT normalize !)
  - Calculate Object Space to UVW Space matrix
    - T,B,N vectors → inverse matrix
    - Transform vectors into UVW (tangent) space
      - Vertex - Light Position (L)
      - Vertex - View Position (V)
- Interpolate vectors L and V across primitive

Normal Mapping
in tangent space

- Per-Fragment
  - Sample compressed normal from normalmap
  - Uncompress n=n*2.0 – 1.0
  - Normalize vectors !
  - Calculate local illumination model
  - No matrix multiply per-fragment !
    - L,V and N are in tangent space