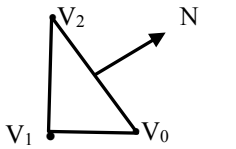
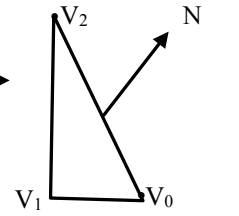
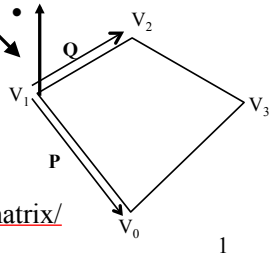
Transforming Normals

- Map vertex normals from obj coords to eye coords
  - w=0 in homogeneous coordinates is useful, but it has limits.
  - Consider a non-uniform scale
  - Could recalculate normals using the plane equation in the eye coordinate space
  - There is an alternative

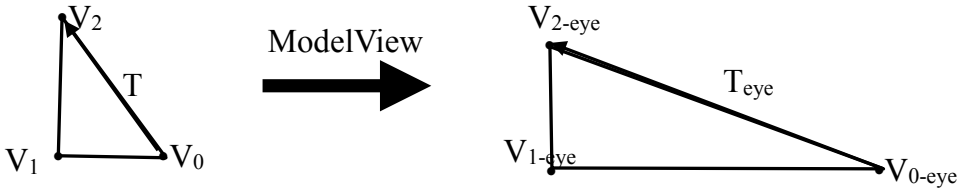






From http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/the-normal-matrix/

Transforming a Vector as a Tangent



A vector is the difference between 2 points (get w=0)

$$T = V_2 - V_0$$

Multiply both sides by the ModelView matrix, M

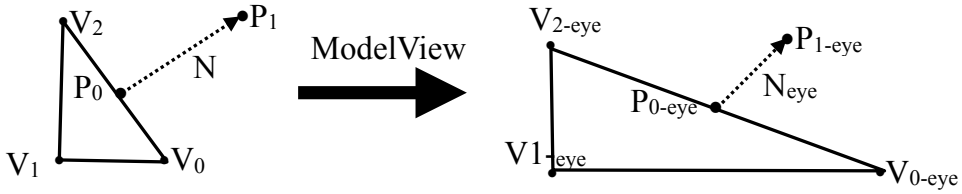
$$M*T = M* V_2 - M* V_0$$

T_{eye}=M*T is tangent transformed to eye coordinate system

$$T_{eye} = V_{2\text{-}eye} - V_{0\text{-}eye}$$

This IS correct.

Transforming a Normal Vector



Could find 2 points, P_0 and P_1, whose difference = N

any two such points will work, but pick easy ones

Multiply both sides by the ModelView matrix, M

$$M*N = M* P_1 - M* P_0$$

N_{eye}=M*N is Normal transformed to eye coordinate system

$$N_{eye} = P_{1\text{-}eye} - P_{0\text{-}eye}$$

This is **not correct**!

The OpenGL Normal Transform

Assume we want a matrix, G, that will do it right.

Let G be just 3x3; only using vectors; no translation needed.

Let $T = V_2 - V_0$. Let T'=GT and N'=GN.

We know that T•N = 0 and T'•N' = N'•T'= 0.

Let M be the 3x3 upper left component of the ModelView.

If G is correct:

N'•T' = (GN)•(MT) =0 where GN and MT are vectors

So $(GN)^T * (MT) = 0$. [$(GN)^T$ is the GN as a row vector]

We know that $(GN)^T = N^T G^T$ so $N^T G^T MT = 0$

If $G^T M = I$, then N'•T' = N•T = 0

Hence, $G = (M^{-1})^T$ G is called the Normal Transform