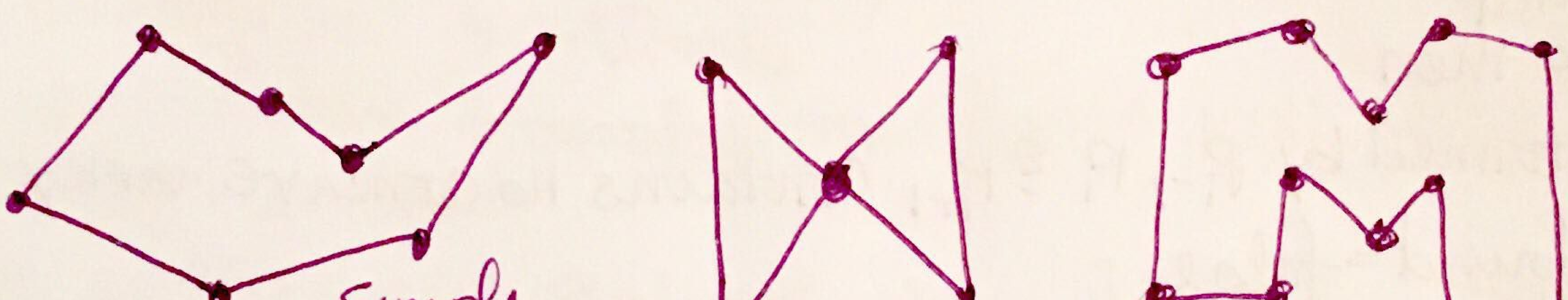
GRAPHICS
2/27/2018

OUTLINE: TRIANGULATING SIMPLE POLYGONS w/ EAR CLIPPINGS
TRANSFORMING NORMALS

TRIANGULATING SIMPLE POLYS w/ EAR CLIPPINGS

PROBLEM: Decompose a simple poly into a collection of Δs whose vertices are only those of the simple poly

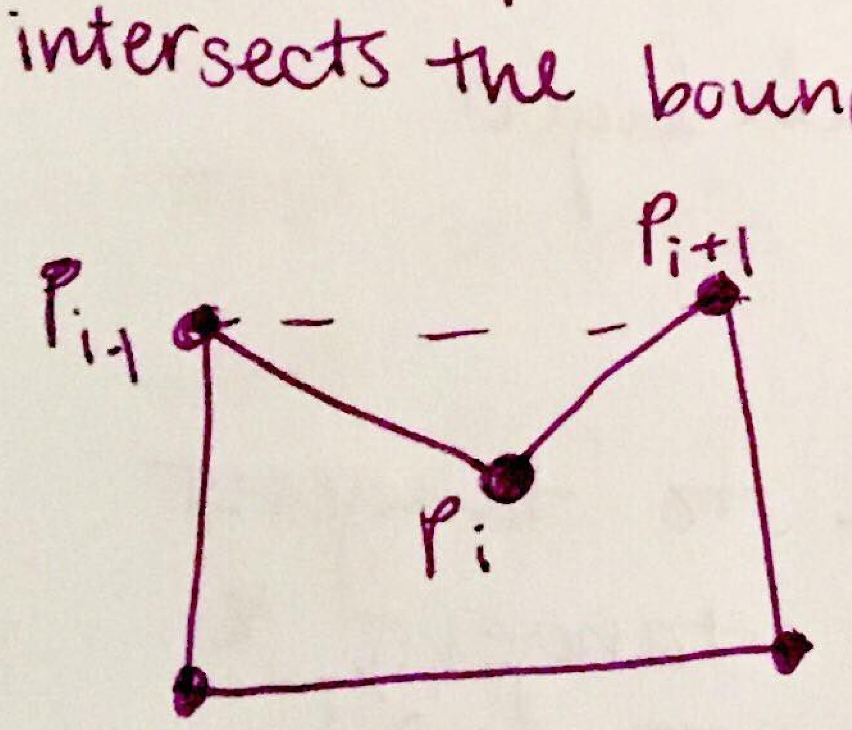
def: A simple poly is a poly, P, with no two non-consecutive edges intersecting.
(there is a well-defined bounded interior & unbounded exterior)



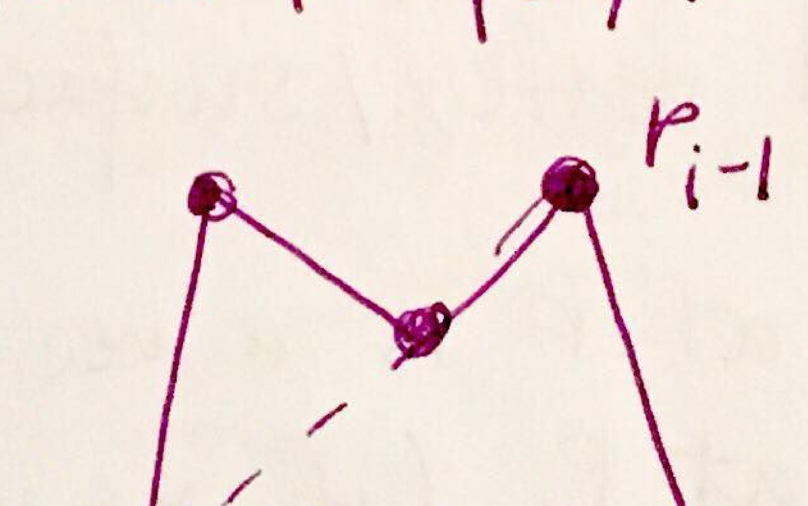
Simple — non-simple — Simple

def: The decomposition of a simple poly into Δs is a triangulation of the poly.

def: A vertex p_i is called a principal vertex if the diagonal (p_{i-1}, p_{i+1}) intersects the boundary of a simple poly, P only at p_{i-1} & p_{i+1}



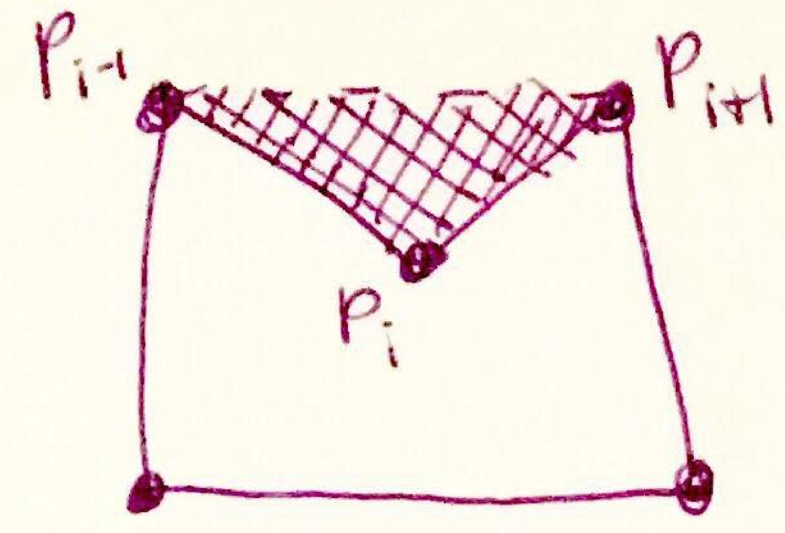
p_i is a principal vertex



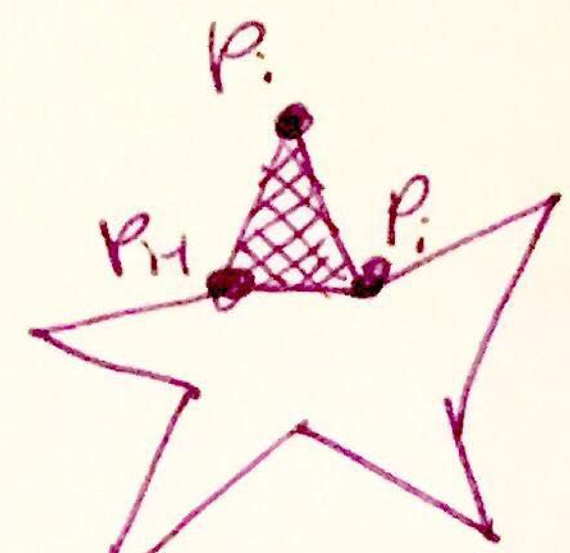
p_i is NOT a principal vertex

NOTE: need to exagerate this drawing better.

def: A principal vertex of a simple poly is called an ear if the diagonal (p_{i-1}, p_{i+1}) that bridges p_i lies entirely in P.



P_i is NOT AN EAR



P_i is an EAR

<u>TWO EARS THEOREM</u>: Except for Δs, every simple poly has @ least 2 non-overlapping ears.

<u>An $O(kn)$ Algorithm for finding an ear</u>:

* Algorithm can be as bad as $O(n^2)$
why? (since $k-1$ is # of concave vertices)

```
Function FindEar(P)
  i=0
  ear not found = true
  while ear not found
    if P_i is convex then
      if triangle formed by P_{i-1}, P_i & P_{i+1} contains no concave vertex then
        ear not found = false
    if ear not found then
      i = i+1
  return P_i
End FindEar
```

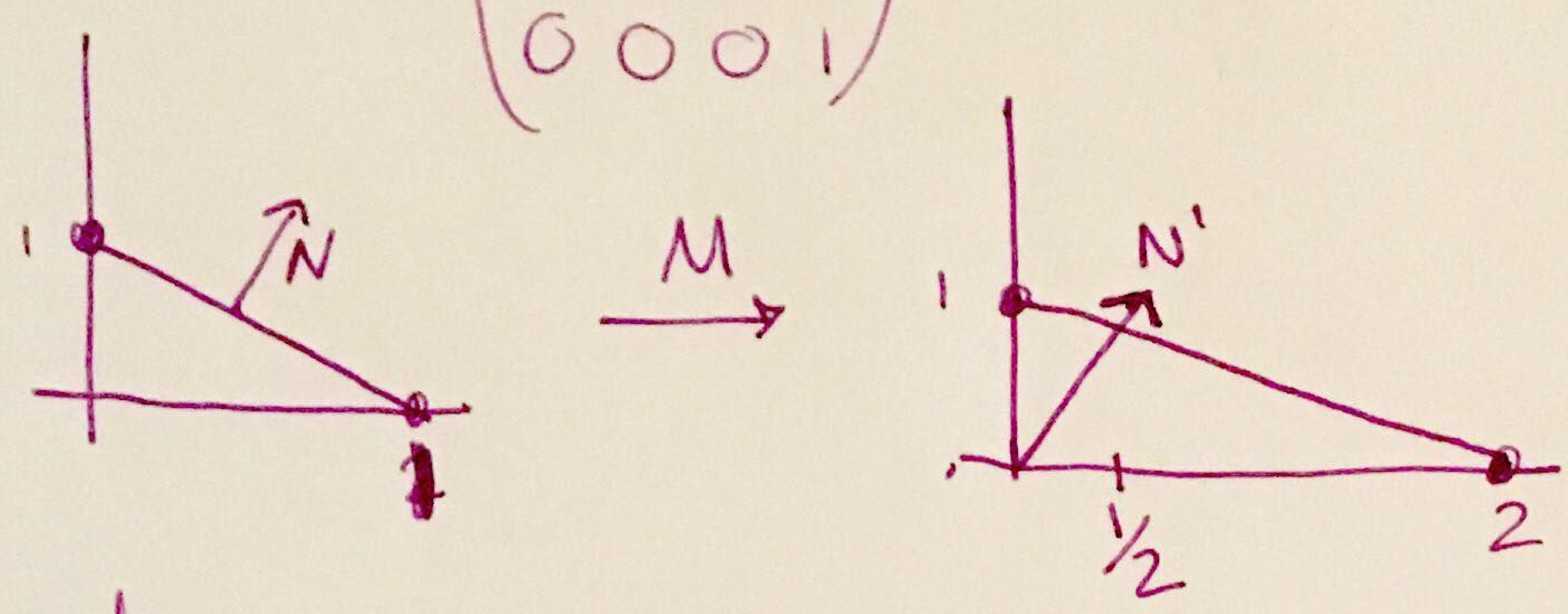
<u>TRANSFORMING NORMAL VECTORS</u>

* NORMAL VECTORS Are used to determine how much light is received @ a specified vertex/surface

Recall, a <u>normal vector</u> at P is a vector $\perp$ to tangent plane to that surface at P. (if you know tangent, T & bitangent, B of the surface @ P then $N = T \times B$)

- Prev. have seen we use matrix mult. to transform points & vectors
  - This will <u>not</u> Always work (infact rarely works) for normals only when scaling uniformly.

Non-Ex: $M = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ * scale ~~matrix~~ x-coord. by 2



Instead multiply N by the transpose of the inverse of that matrix

i.e. $N' = N \cdot M^{-1T}$

Why?

- N is not affected by translation so can ignore 4th row & column (left w/ rotat. & scaling part of matrix)
- transpose of orthogonal is its inverse & rotation matrices are orthogonal & $Q = Q^{-1T}$
  - so inverse ~~transfore~~ transpose will keep what we want intact.
- diag entries are scalings so $2 \mapsto 1/2$, $4 \mapsto 1/4$ etc. which is what we want.

Derivation: Let N be the normal @ P & V lie in the plane tangent to P. Assume V' & N' to be the images of the normal & tangent.

we have: $V \cdot N = (V_x \ V_y \ V_z) \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = V * N^T = 0$

$\Rightarrow V * N^T = V * M * M^{-1} * N^T = V * I * N^T$

$\Rightarrow V * N^T = \underbrace{(V * M)}_{\substack{V' \\ \text{(image of tangent)}}} * \underbrace{(n * M^{-1T})^T}_{N'^T}$

$\Rightarrow V * N^T = V' * N'^T$ & $N' = N * M^{-1T}$

WORK ON PROGRAMMING ASSIGNMENT/PROJECT.