

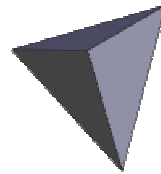
# Meshes, data structures, simplification

CS 175

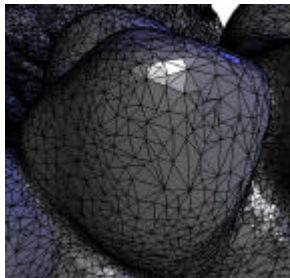
## Meshes



82K vertices  
164K triangles



4 vertices  
4 triangles



Simple file format?

Data structures?

# Wavefront file format

## ■ Tetrahedron

# OBJ file format with ext .obj

v 1.0 0.0 0.0

v 0.0 1.0 0.0

v 0.0 0.0 1.0

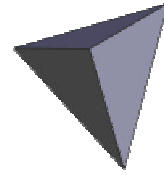
v 0.0 0.0 0.0

f 2 4 3

f 4 2 1

f 3 1 2

f 1 3 4



v x y z	vertex
f v <sub>1</sub> v <sub>2</sub> v <sub>3</sub>	face
# comment	

# Objects

## ■ Faces: polygons, triangles

- normal, area

## ■ Vertices

- coordinates, normal

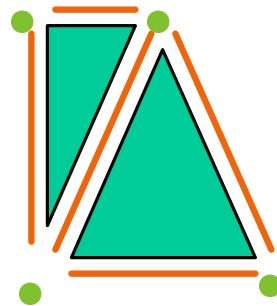
## ■ Edges

- creases, dihedral angles

## ■ Boundaries

## ■ Euler formula:

- $\text{genus} = 1 - (V - E + F + B) / 2$
- $4 - 5 + 2 + 1 = 2; 1 - 2/2 = 0$



## A fact

- usually for big meshes:

- $F \approx 2 V$

- $E \approx 3 V$

- why?

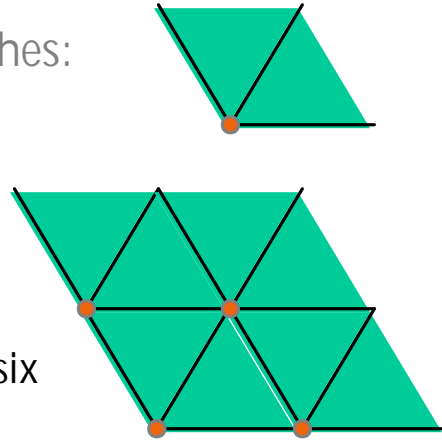
- use

- average valence is six

- go through verts

- count adjacent faces

- $6 V \approx 3 F$



## Data structures

- Operations

- find normal

- flat

- three verts of a face

- smooth

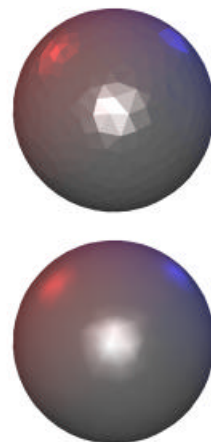
- average normals: one-ring

- » or

- use formula from subdivision

- given a face find vertices

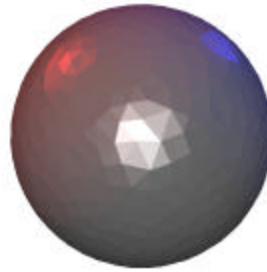
- given a vertex find faces



# OpenGL

## ■ Flat shaded triangles

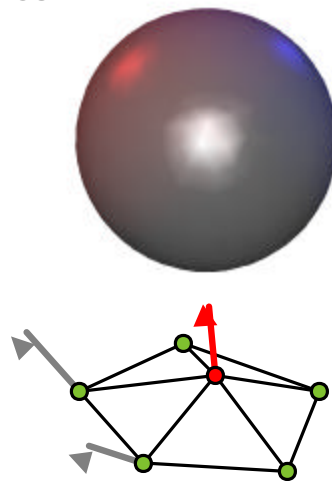
```
glBegin(GL_TRIANGLES);  
for(...) {  
    ...  
    glNormal3fv(n);  
    glVertex3fv(v1);  
    glVertex3fv(v2);  
    glVertex3fv(v3);  
}  
glEnd();
```



# OpenGL

## ■ Smooth shaded triangles

```
...  
glNormal3fv(n1);  
glVertex3fv(v1);  
glNormal3fv(n2);  
glVertex3fv(v2);  
glNormal3fv(n3);  
glVertex3fv(v3);  
...
```



# Operations

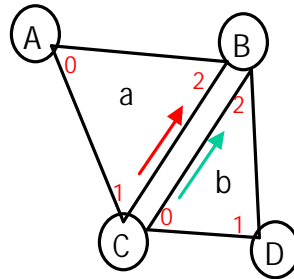
- naive procedure for smooth shaded:
  - iterate through verts
    - for every vertex compute normal
      - need one ring of neighbors
  - iterate through faces
    - for every face render a triangle with normals
      - face -> vertices

# Face based mesh

- Vertex container
  - vertex: 3 floats, 1 pointer
    - coordinates
    - pointer to adjacent face
- Face container
  - face: 6 pointers + ...
    - pointers to neighbors, vertices
    - permutations
- no edges explicitly
  - (v1, v2) or (f, side)

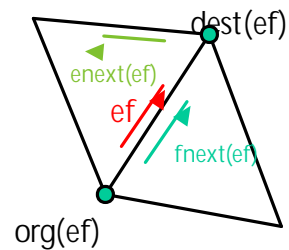
## “permutations”

- Useful to store orientation of the neighboring face
- $a = [ A, C, B ]$
- $b = [ C, D, B ]$
- a's neighbors:
  - $(a, (0,1)) \rightarrow ..$
  - $(a, (1,2)) \rightarrow (b, (0,2))$
  - $(a, (2,0)) \rightarrow ..$



## Navigation: efs

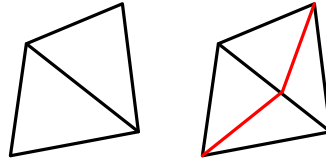
- Edge-face (ef)
- Operations
  - $dest, org, face$
  - $sym, enext, fnext$ 
    - $fnext(fnext(ef)) == ef: fnext^2 == id$
    - $enext^3 == id, sym^2 == id$
    - $org(fnext(ef)) == org(ef)$
    - $sym(org(ef)) == dest(ef)$



## Additional operations

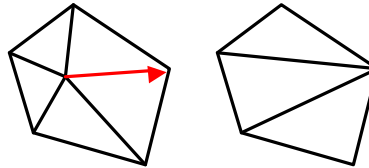
- Bisection

- adds a vertex
  - $\text{bisect}(ef, \alpha)$



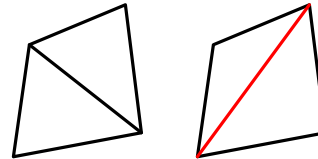
- Edge-collapse

- removes a vertex



- Edge-flip

- beware of valence three
- these are useful. why?



## Alternative: winged edge

- Edge knows

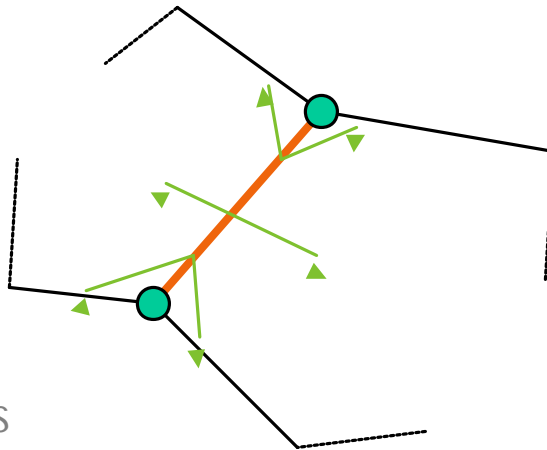
- four edges
- two verts
- two faces

- Face knows

- one edge

- Vertex knows

- one edge



# Manifold condition

- Two kinds of vertices allowed

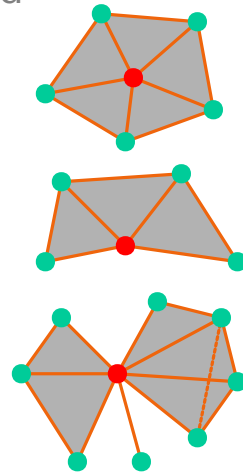
- inside vertex

- full umbrella neighborhood

- boundary vertex

- half umbrella neighborhood

- Prohibited configurations



# Normals

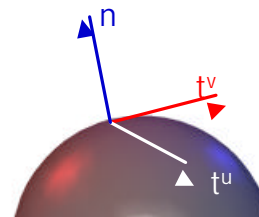
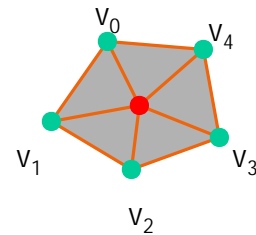
- At a vertex

- Tangent vectors

$$t^u = \sum_{i=0}^N p_i \cos(2\pi i/N)$$

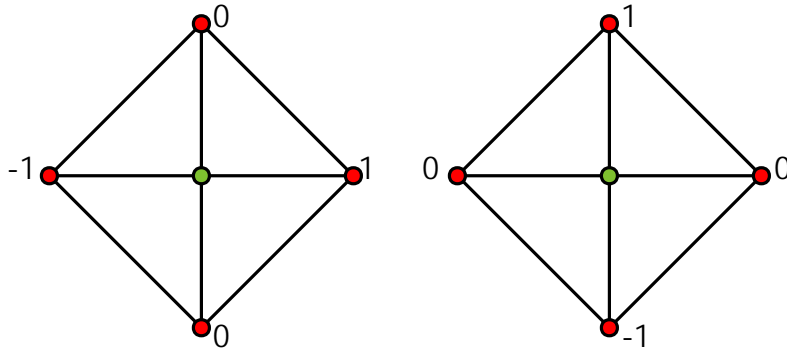
$$t^v = \sum_{i=0}^N p_i \sin(2\pi i/N)$$

$$n = t^u \times t^v$$



## Normals II

- Indeed for valence four



## Mean curvature

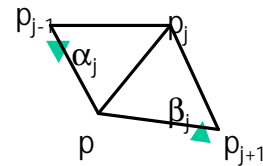
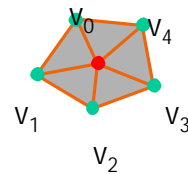
- Surface locally is like  $z = \mathbf{k}_1 x^2 + \mathbf{k}_2 y^2$

- mean curvature is  $\bar{\mathbf{k}} = \frac{\mathbf{k}_1 + \mathbf{k}_2}{2}$

- another way:  $\bar{\mathbf{k}} \cdot \mathbf{n} = \nabla A / A$

- Desbrun et al. '99:

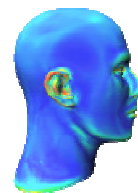
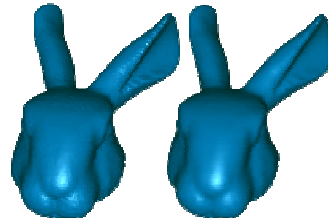
$$\bar{\mathbf{k}} = -\frac{1}{4A} \sum_{j=0}^{N-1} (\cot \mathbf{a}_j + \cot \mathbf{b}_j) (\mathbf{p}_j - \mathbf{p}) \cdot \mathbf{n}$$



- Why do we need curvature?

## Programming Assignment 1

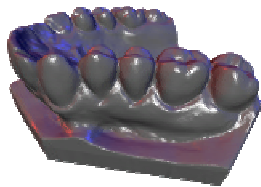
- Implement a mesh class
  - Read/write .obj file
  - Show it in the viewer
    - compute normals
      - flat and smooth shading
    - compute curvatures
- OpenGL code provided
- Keep in mind:
  - p.a. #2 is mesh simplification
  - p.a. #3 is subdivision



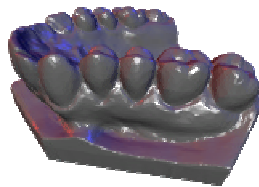
## Mesh simplification



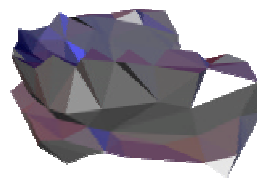
num of verts: 81457  
num of faces: 162910



60K verts



17K verts



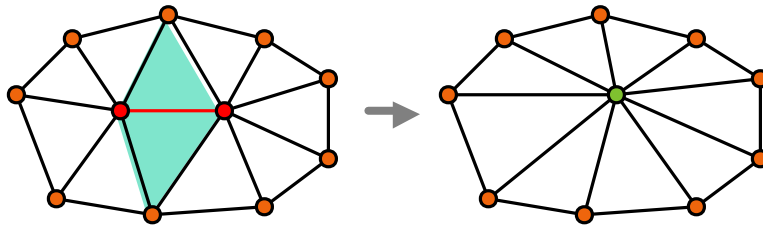
400 verts

## Clearly very important

- Laser scan is too fine
- Large planar areas
- Rendering performance
- Base for multires algorithms
  - parameterization
  - filtering
  - editing
  - progressive compression/transmission

## Edge contraction

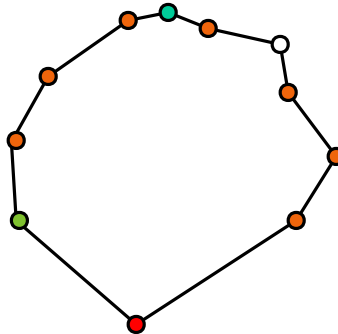
- Hoppe and others ...



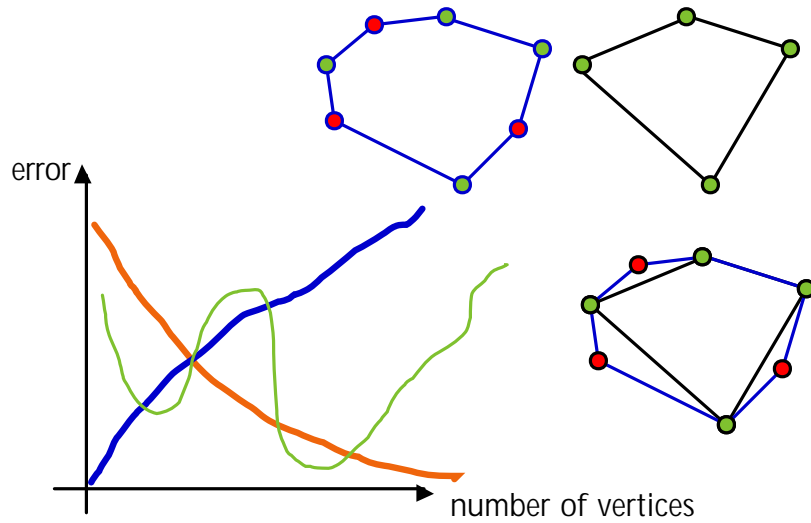
- need order in which to remove points
  - suggestions?

## Polyline simplification

- How to simplify?
- What is our goal?
- pause.

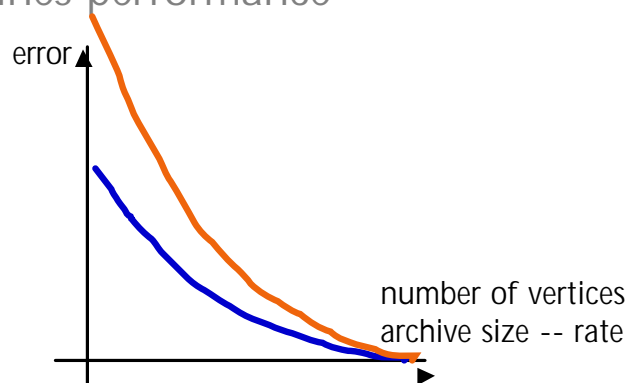


## Size vs error



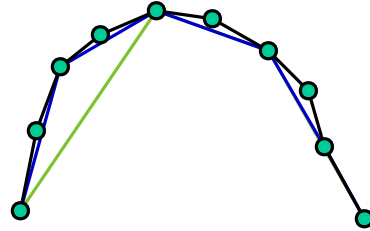
## Performance

- Rate distortion curve
  - compare different methods
- order defines performance



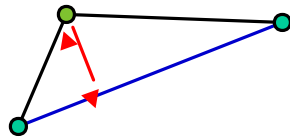
## Choices

- Error from original
  - usually too slow
- Local criteria
  - faster
  - many small errors add up
  - each face responsible for region of the original geometry
- Need to accumulate error
  - quadrics by Garland & Heckbert '97
    - programming assignment 2



## Error

- Distance from a plane



- $\langle \mathbf{n}, \mathbf{p} \rangle = d$

## Error II

- Distance from several planes
  - max
  - mean
    - advantages