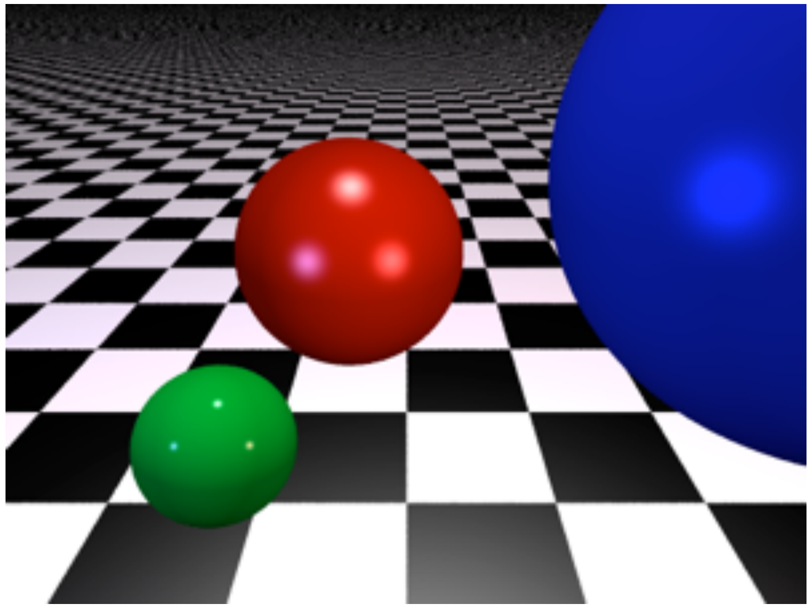
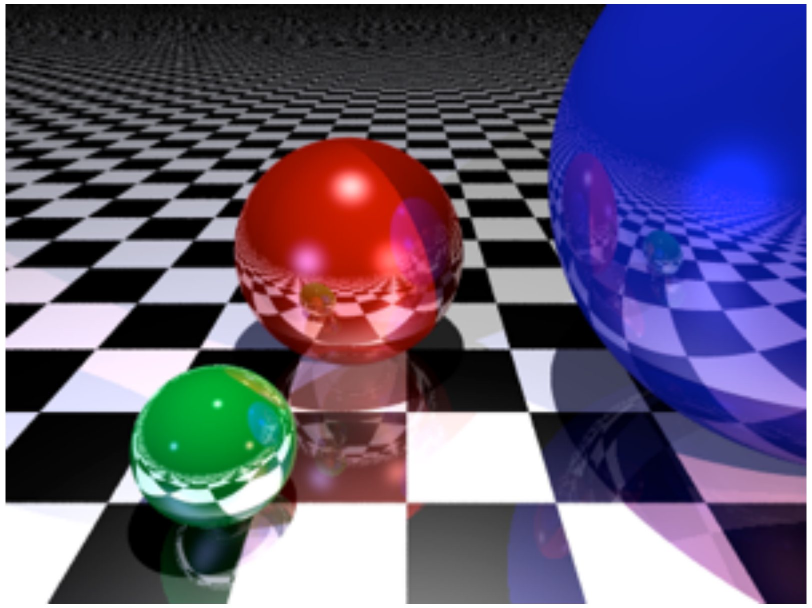
Introduction to Computer Graphics
Triangle Meshes





Prof. Dr. Mario Botsch
Computer Graphics & Geometry Processing

Universität Bielefeld

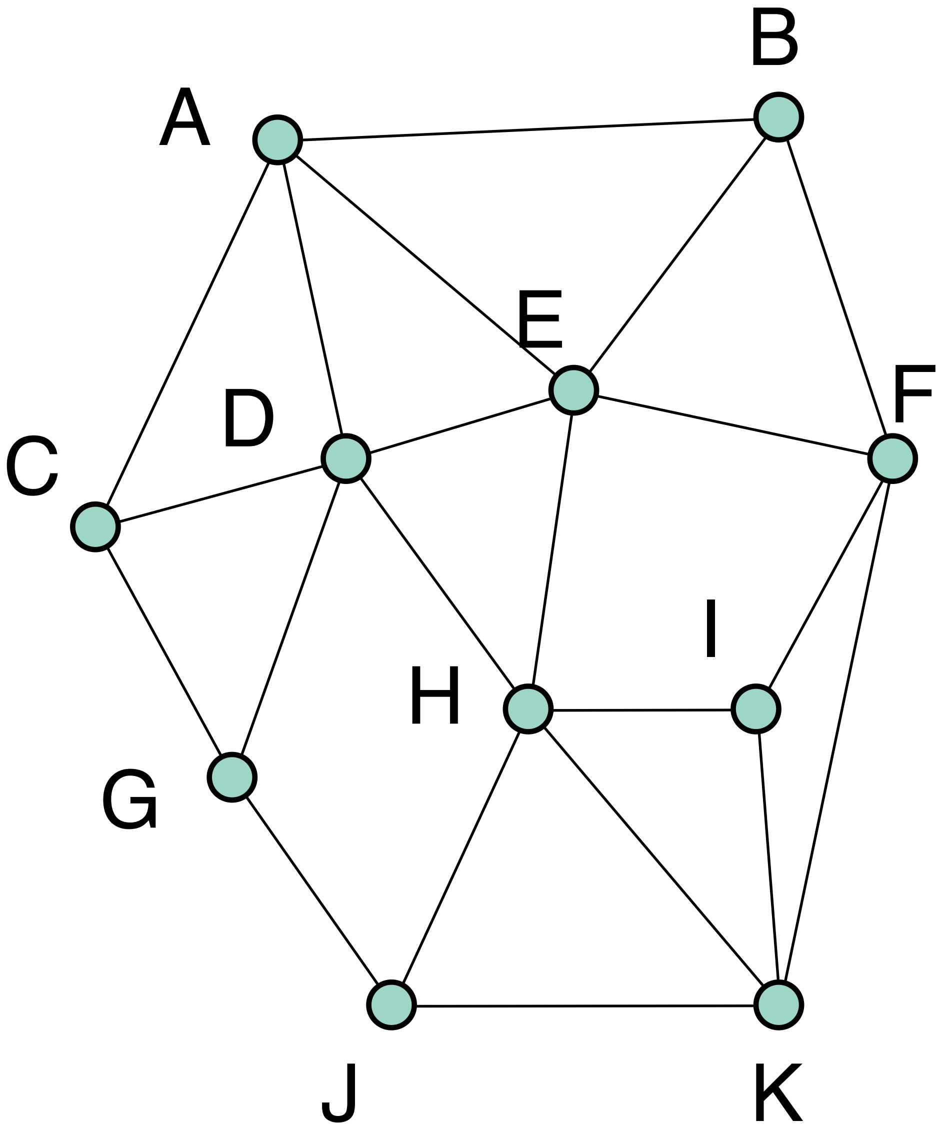
CITEC

Polygon Meshes

"I hate meshes. I cannot believe how hard this is. Geometry is hard."

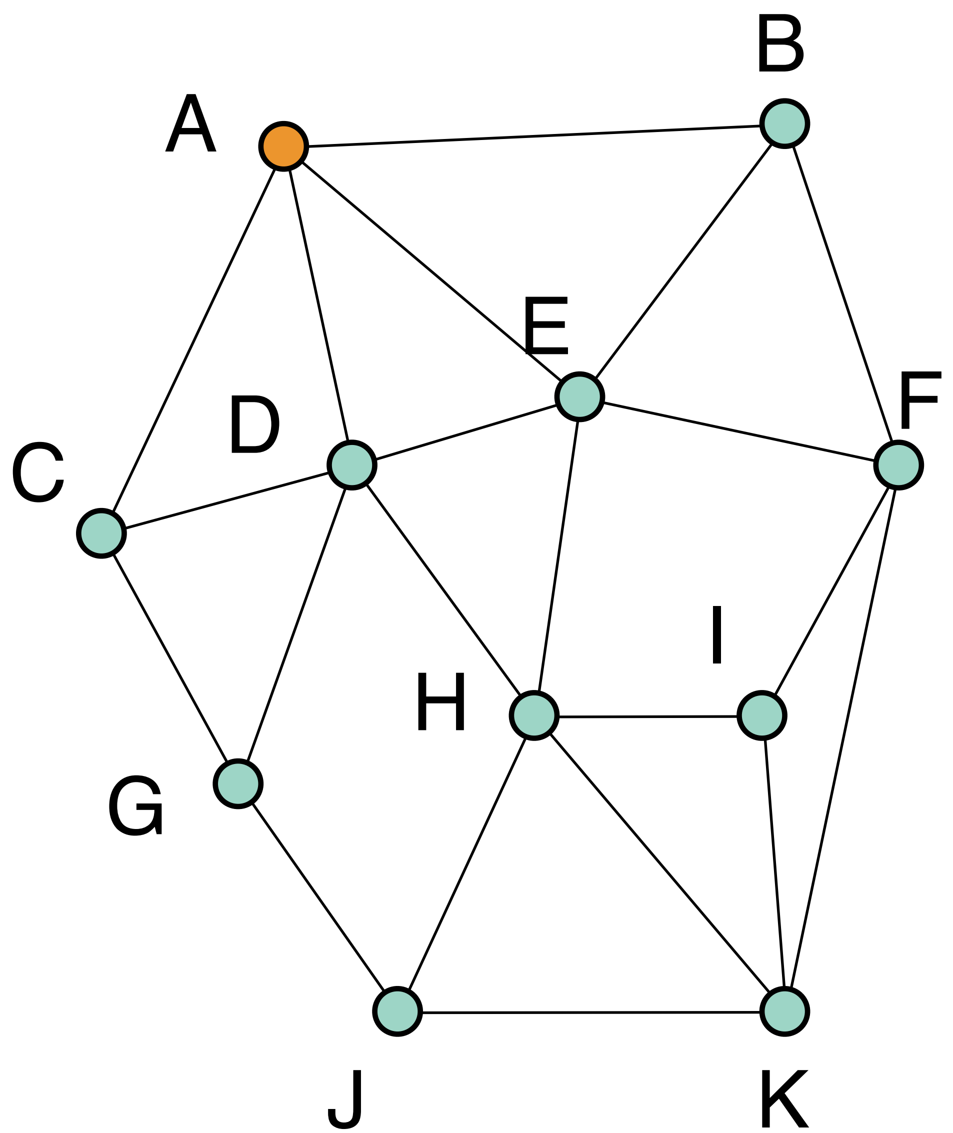
— David Baraff, Senior Research Scientist, Pixar Animation Studios

Graph Definitions



Graph $\{V, E\}$

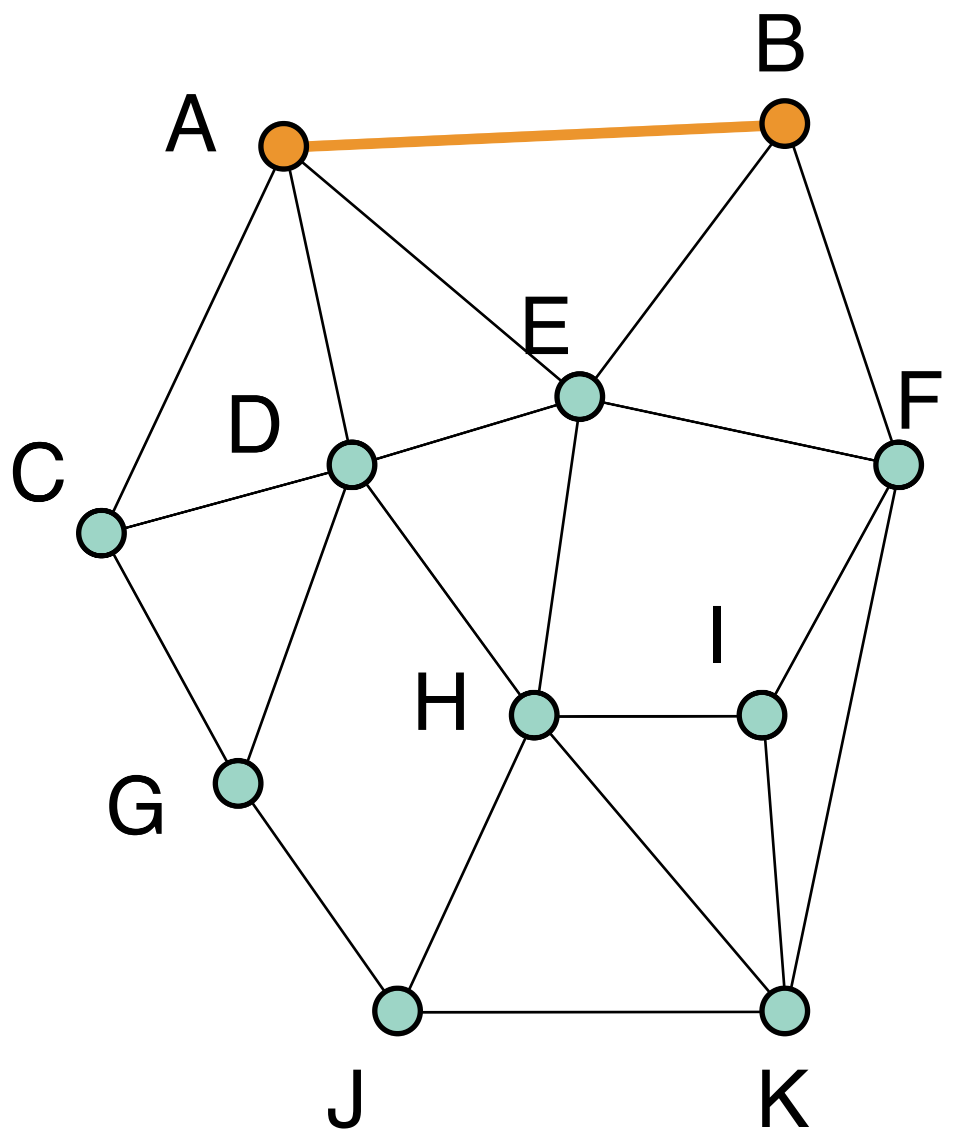
Graph Definitions



Graph $\{V, E\}$

Vertices $V = \{A, B, C, \ldots, K\}$

Graph Definitions

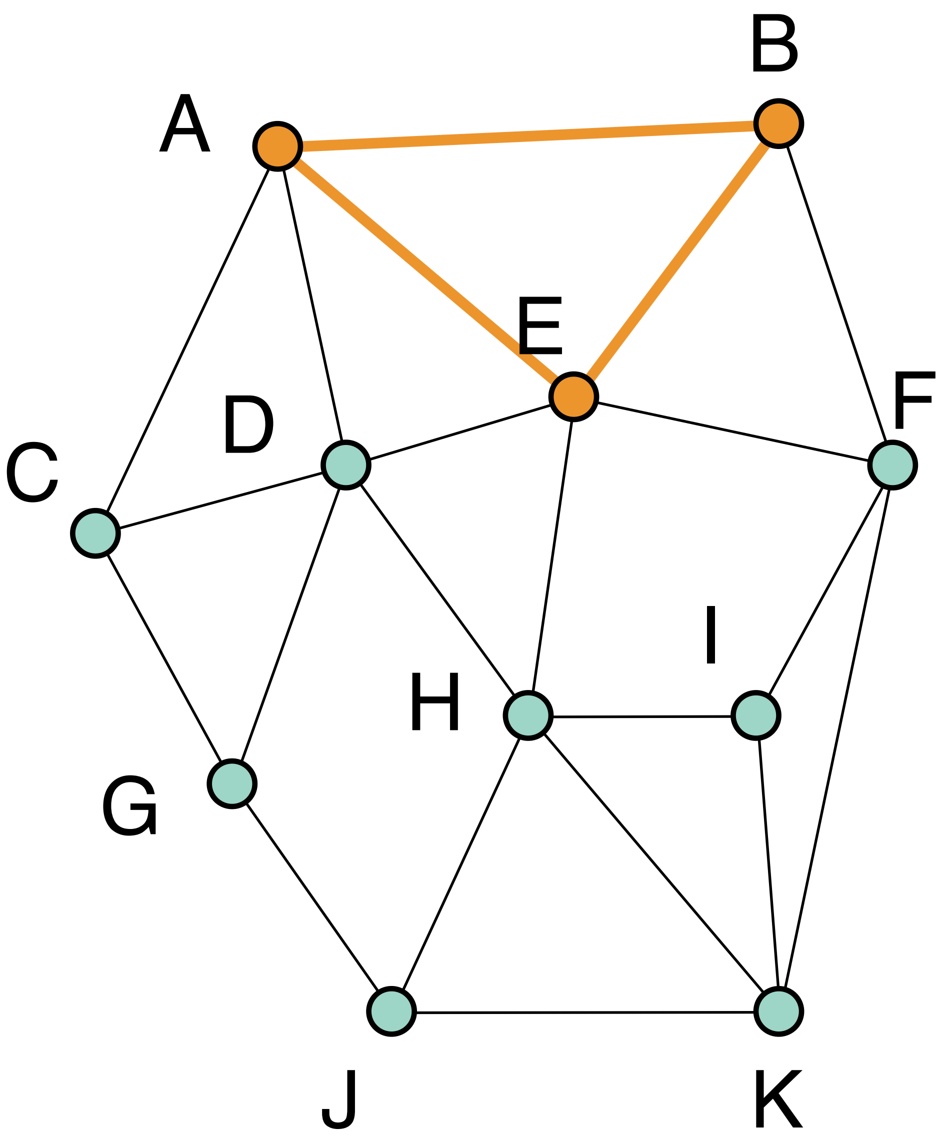


Graph $\{V, E\}$

Vertices $V = \{A, B, C, \ldots, K\}$

Edges $E = \{(AB), (AE), (CD), \ldots\}$

Graph Definitions



Graph $\{V, E\}$

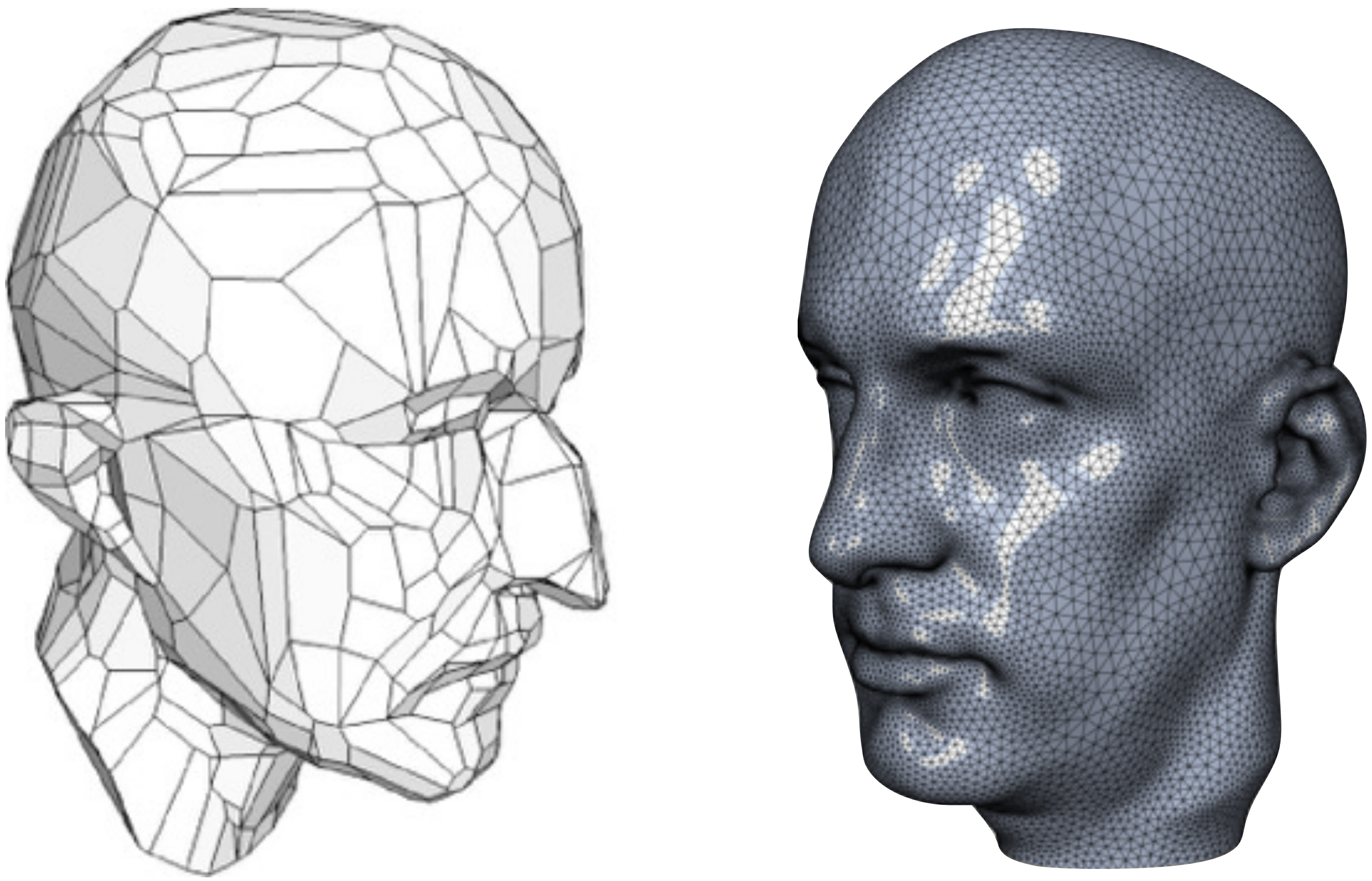
Vertices $V = \{A, B, C, \ldots, K\}$

Edges $E = \{(AB), (AE), (CD), \ldots\}$

Faces $F = \{(ABE), (EBF), (EFIH), \ldots\}$

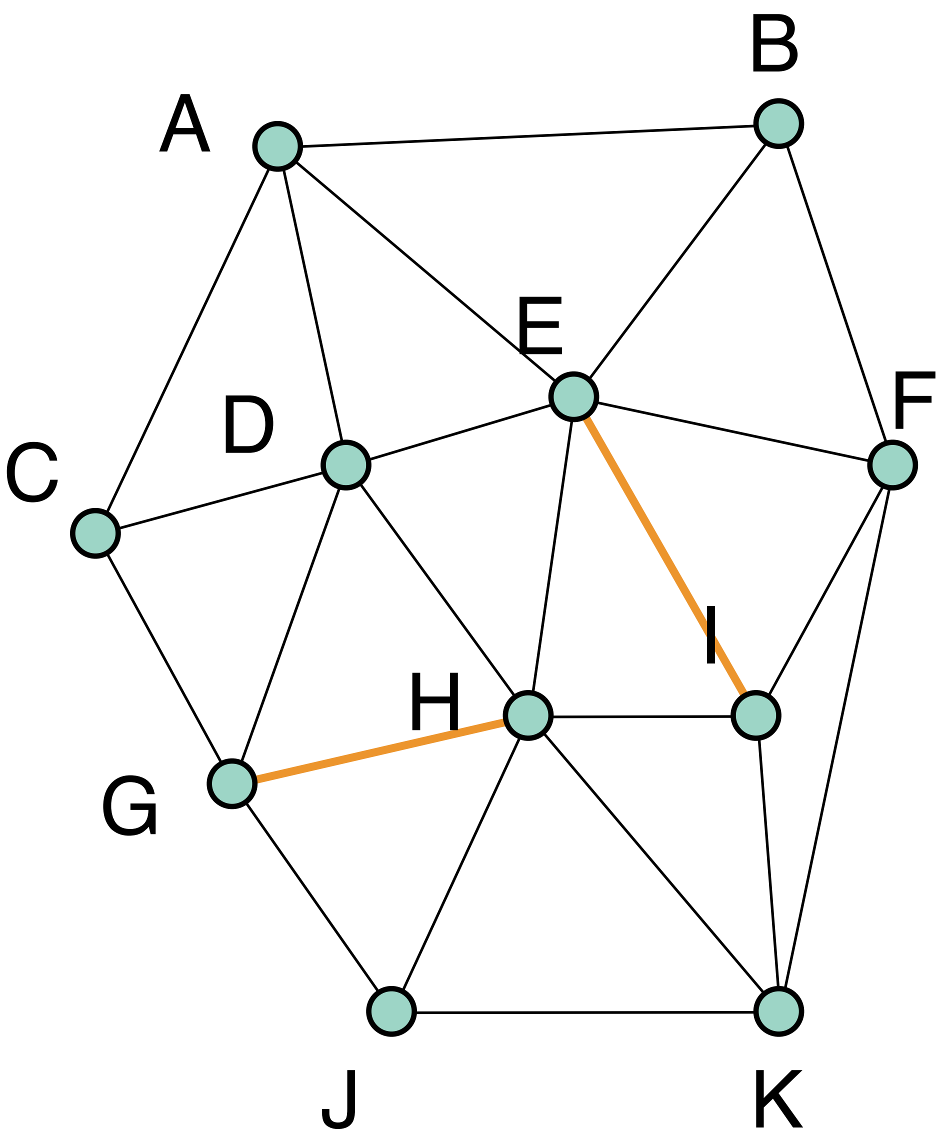
Graph Embedding

Embedding: Graph is embedded in $\mathbf{R}^d$, if each vertex is assigned a position in $\mathbf{R}^d$.



Embedded in $\mathbf{R}^3$

Triangulation



Triangulation: Graph where every face is a triangle.

Why...?

- ➡ simplifies data structures
- ➡ simplifies rendering
- ➡ simplifies algorithms
- ➡ by definition, triangle is planar
- ➡ any polygon can be triangulated

Triangle Meshes

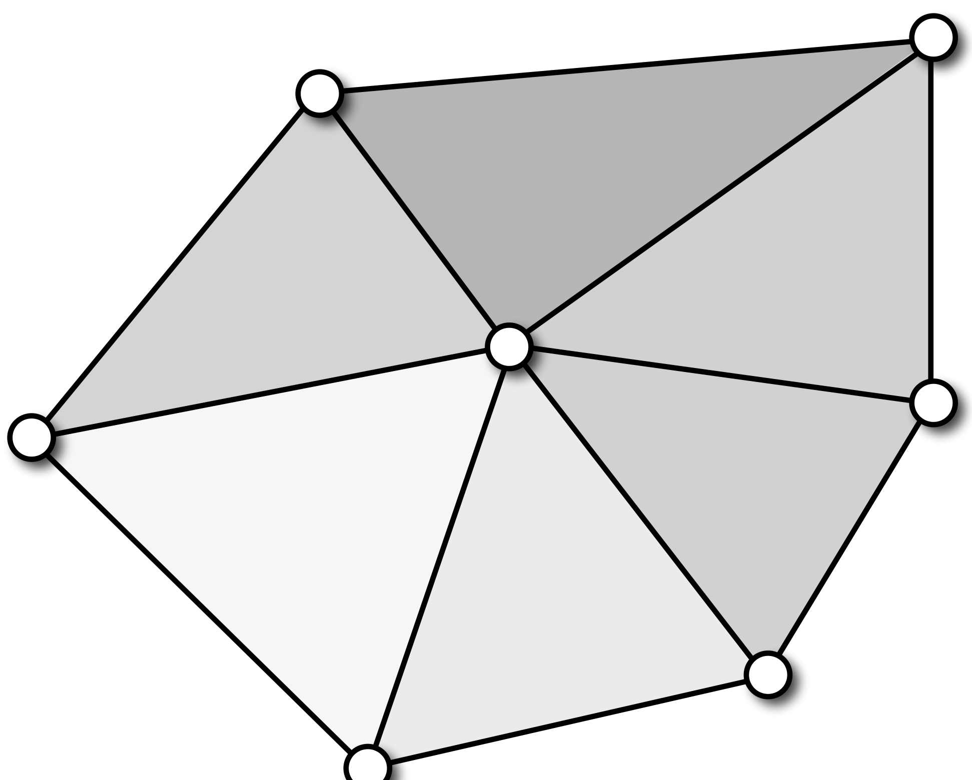
- Connectivity: vertices/nodes and triangles

$$\mathcal{V} = \{v_1, \ldots, v_n\}$$

$$\mathcal{F} = \{f_1, \ldots, f_m\} \ , \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$$

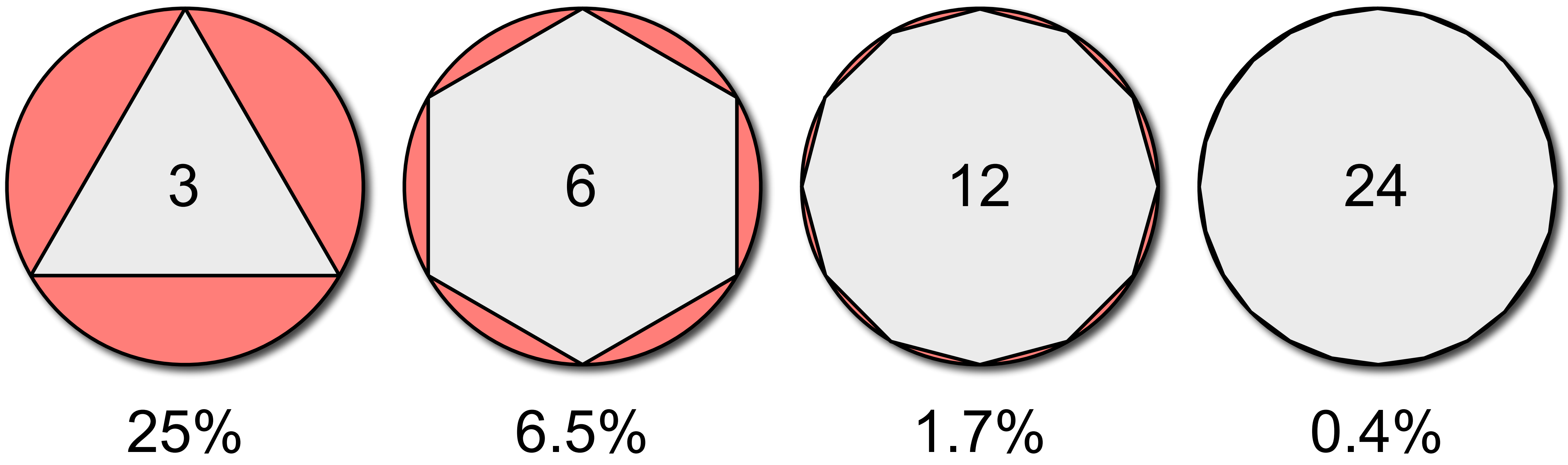
- Geometry: vertex positions

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \ , \quad \mathbf{p}_i \in \mathbb{R}^3$$



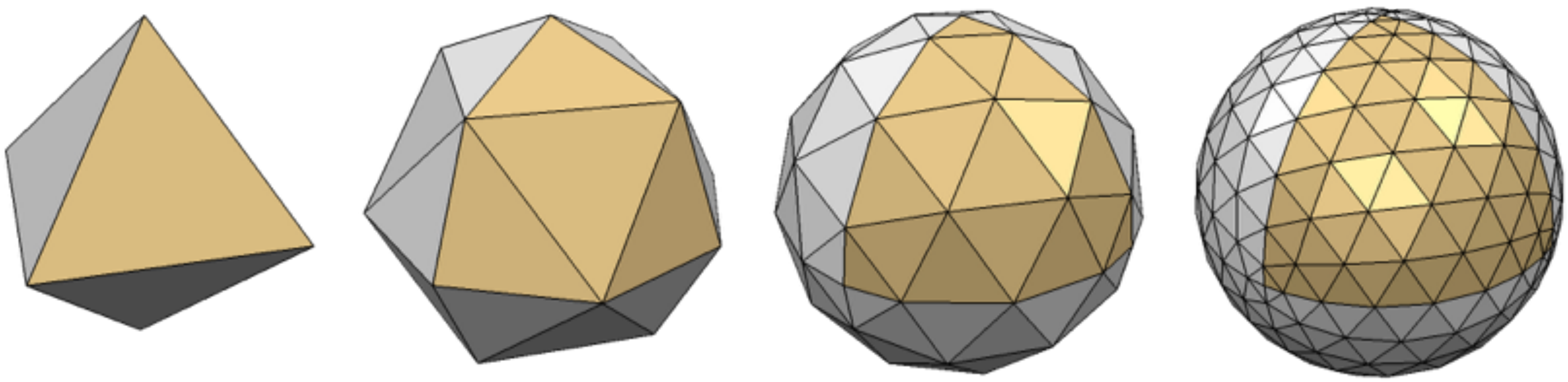
Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$



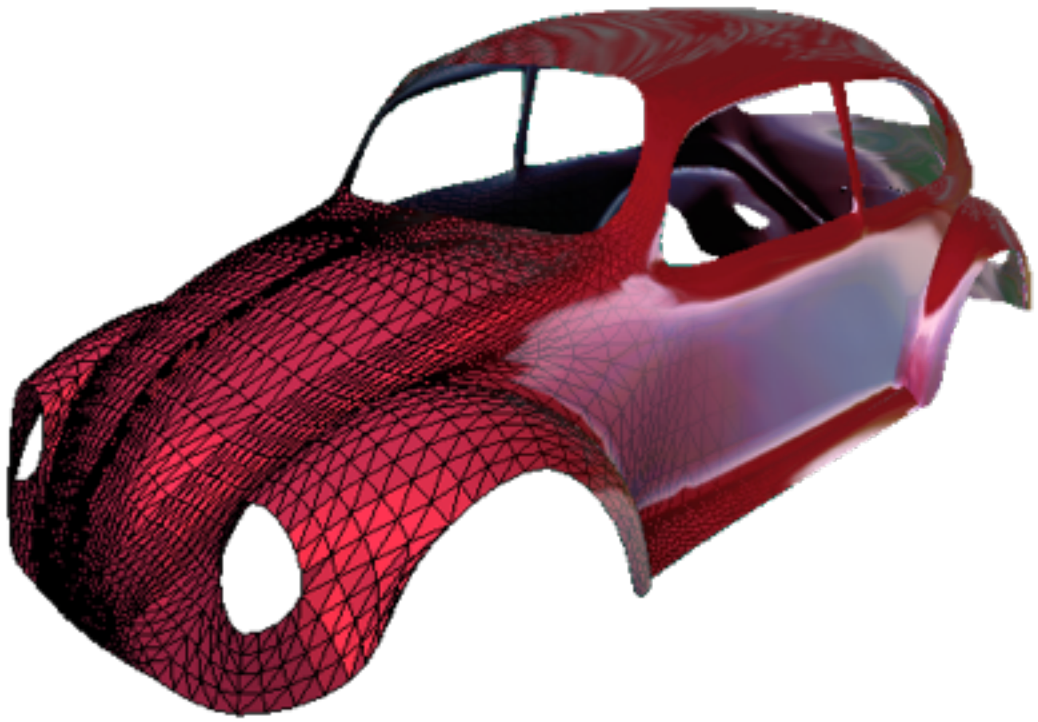
Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inversely proportional to #faces



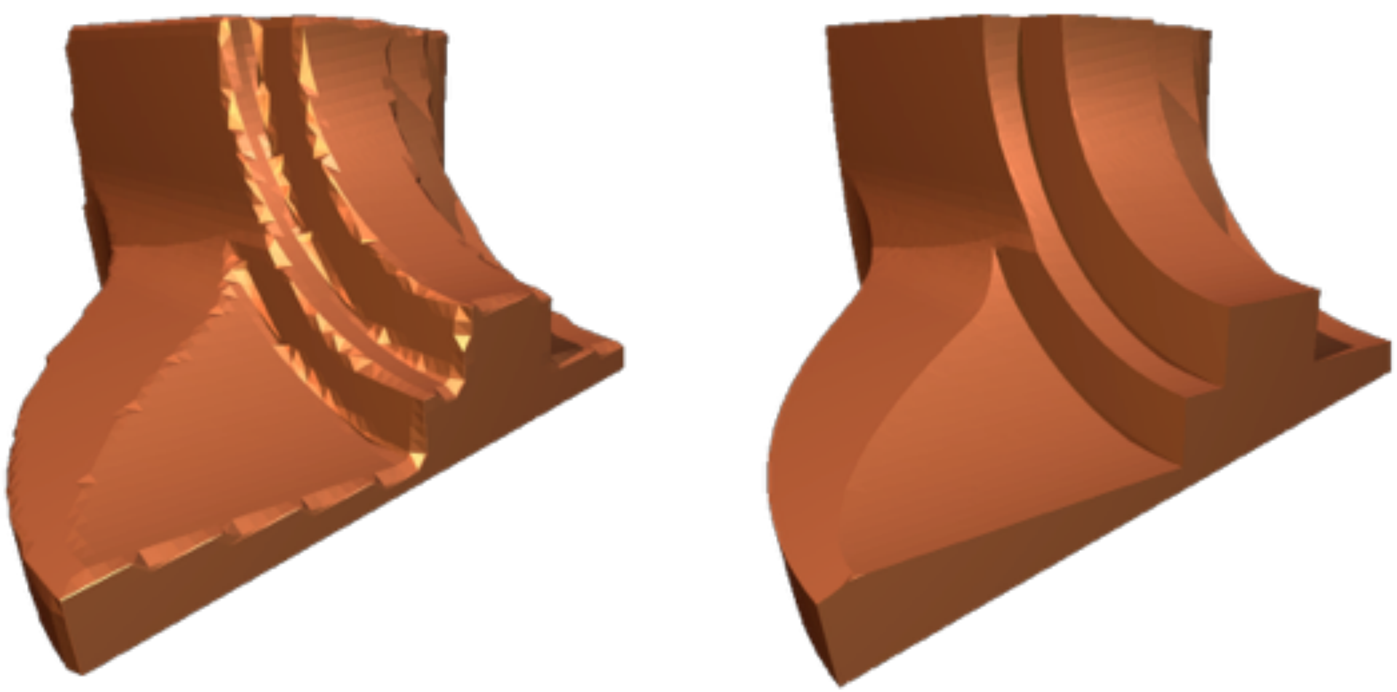
Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inversely proportional to #faces
  - Arbitrary topology surfaces



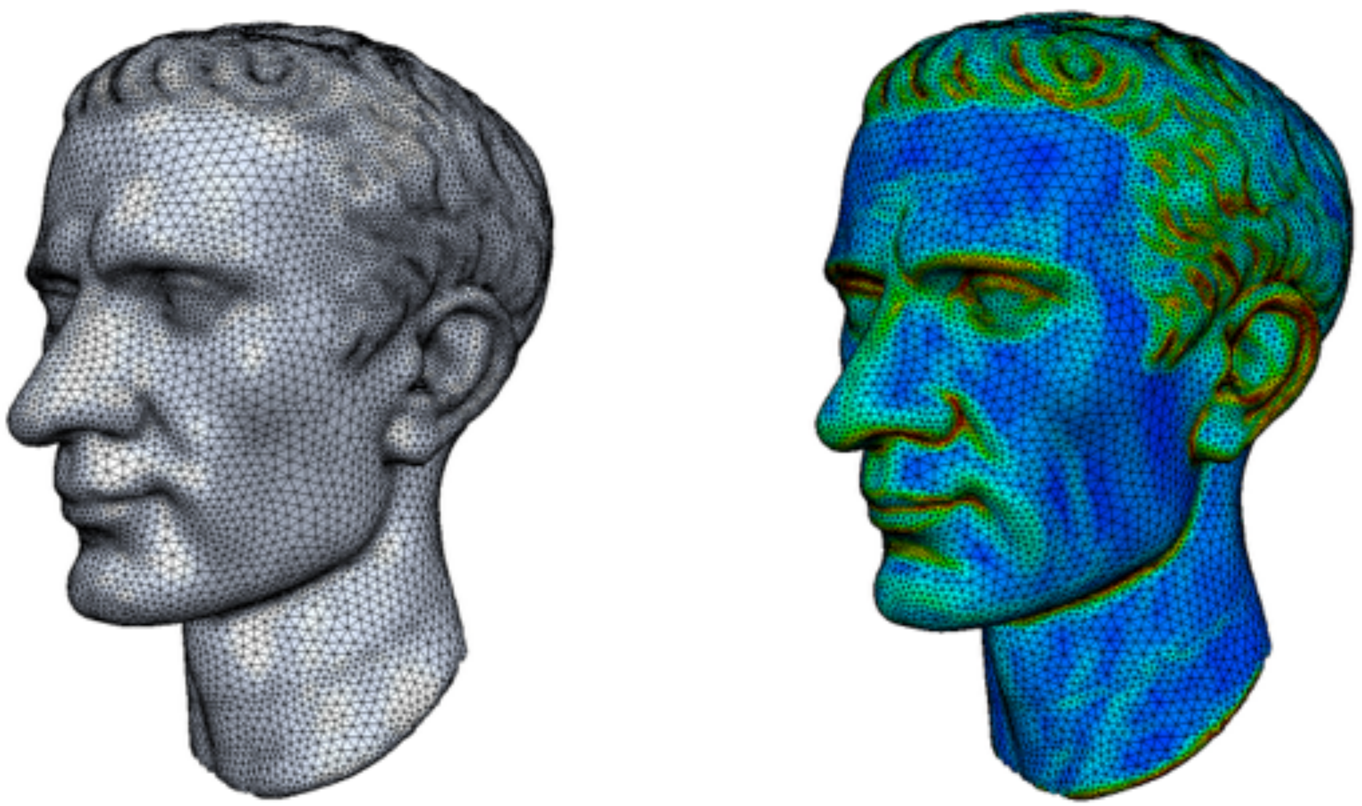
Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inversely proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces



Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inversely proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces
  - Curvature adaptive sampling

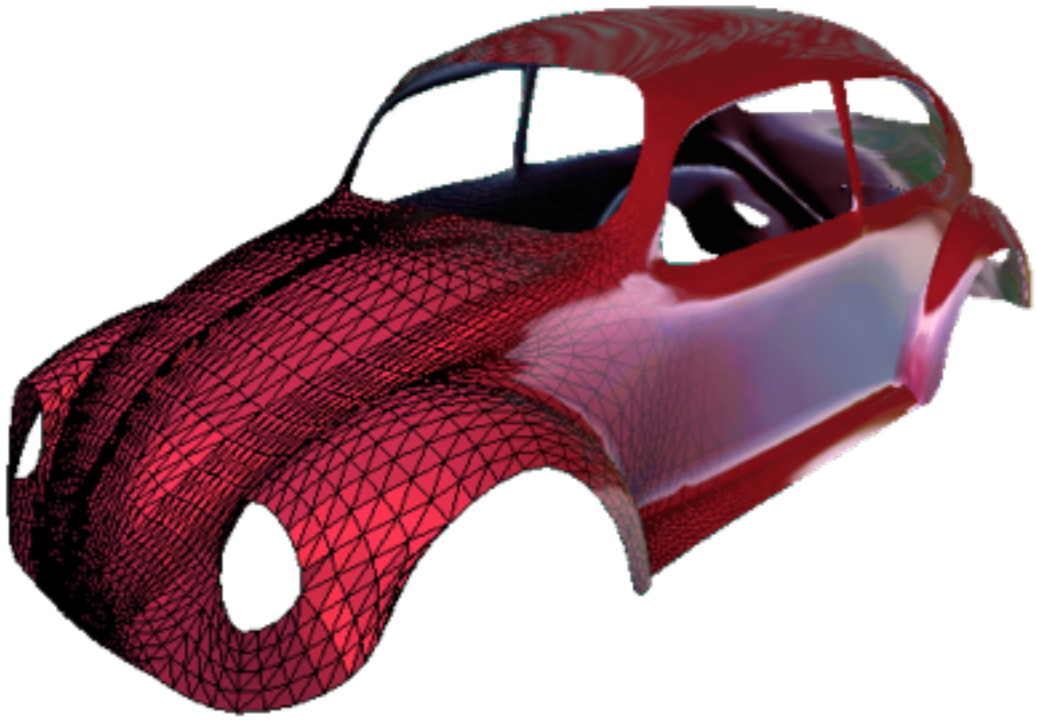


Triangle Meshes

- Advantages of triangle meshes
  - Piecewise linear approximation → error is $O(h^2)$
  - Error inversely proportional to #faces
  - Arbitrary topology surfaces
  - Piecewise smooth surfaces
  - Curvature adaptive sampling
  - Efficient GPU-based rendering

Triangle Meshes

- **Data structures**
- Ray Intersection
- Lighting



Euler Formula

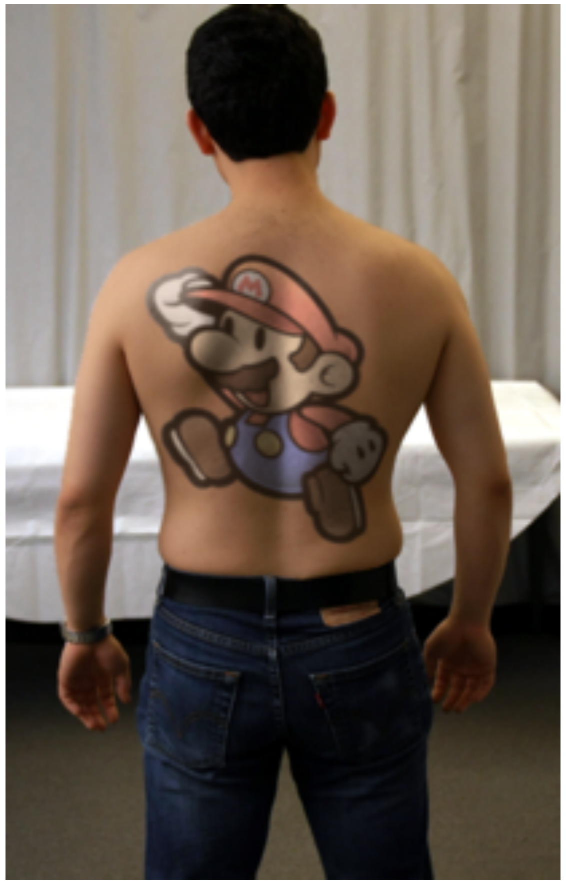
- For a closed polygonal mesh of genus *g*, the relation of the number V of vertices, E of edges, and F of faces is given by *Euler's formula*

$$V - E + F = 2(1-g)$$

- The term $2(1-g)$ is called Euler characteristic χ
- χ only depends on the geometric shape, <u>not</u> on its triangulation (cool!)

Euler Formula

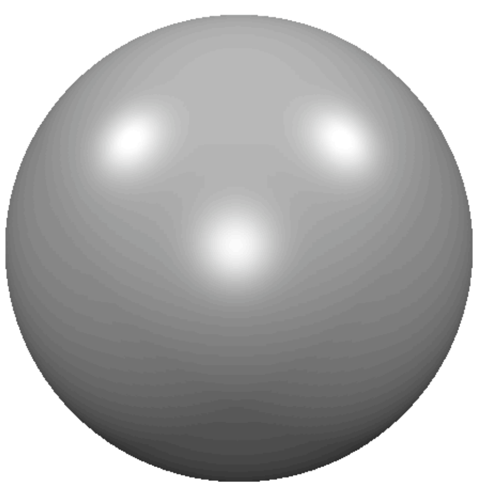
- The Euler formula is important / cool, because it is related to beer mugs, soccer, and tattoos.
  (and because it helps us design good data structures)



Topology: Genus

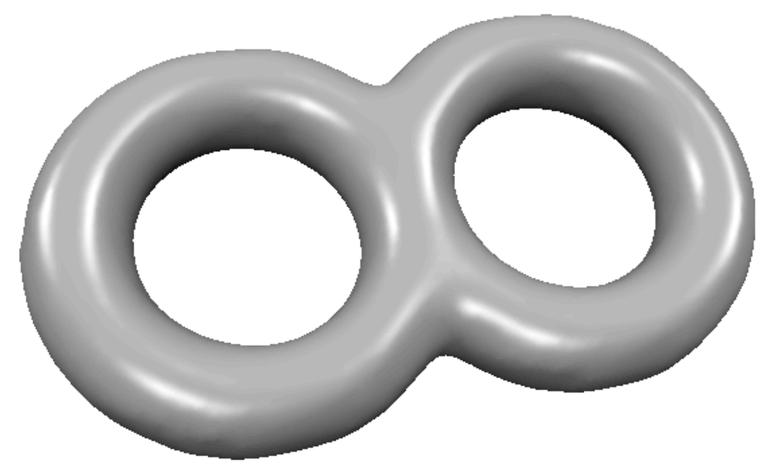
Genus: Maximal number of closed simple cutting curves that do not disconnect the graph into multiple components.

(Informally, the number of holes or handles.)



Genus 0

Genus 1



Genus 2

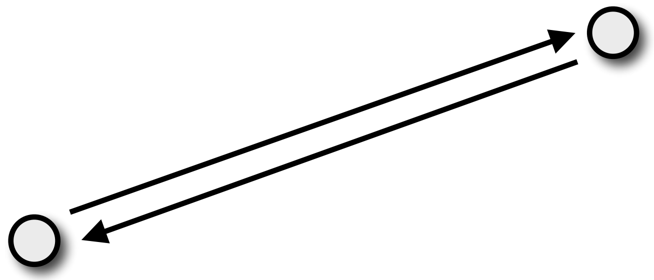
Euler Formula

- Euler formula

$$V - E + F \;=\; 2\,(1-g) \;\approx\; 0$$

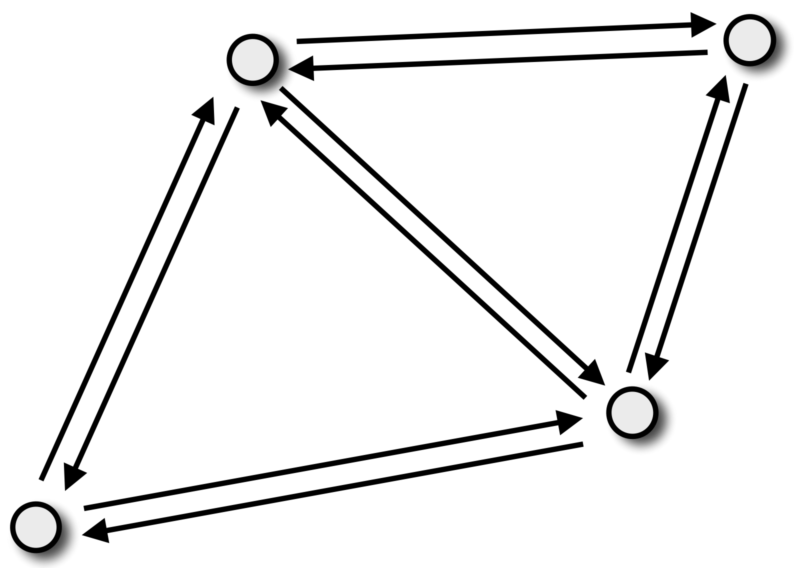
- Split edges into halfedges

$$H \;=\; 2E$$



- Focus on triangle meshes

$$H \;=\; 3F$$



Euler Formula

- Express E in terms of F

$$V - \frac{3}{2}F + F \;\approx\; 0 \quad\Leftrightarrow\quad V \;\approx\; \frac{1}{2}F$$

- Express F in terms of E

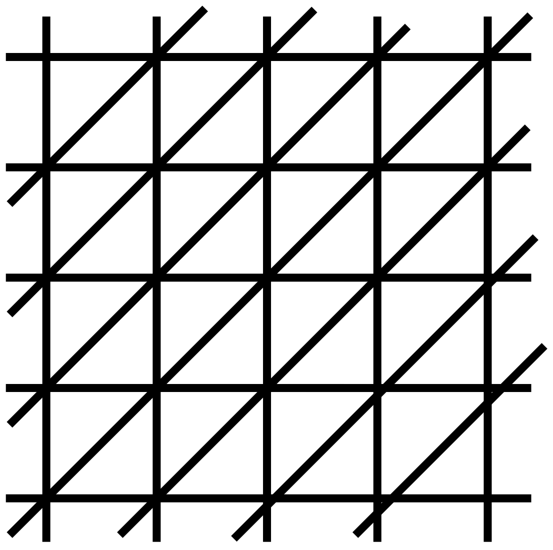
$$V - E + \frac{2}{3}E \;\approx\; 0 \quad\Leftrightarrow\quad V \;\approx\; \frac{1}{3}E$$

- Valence: How many halfedges per vertex?

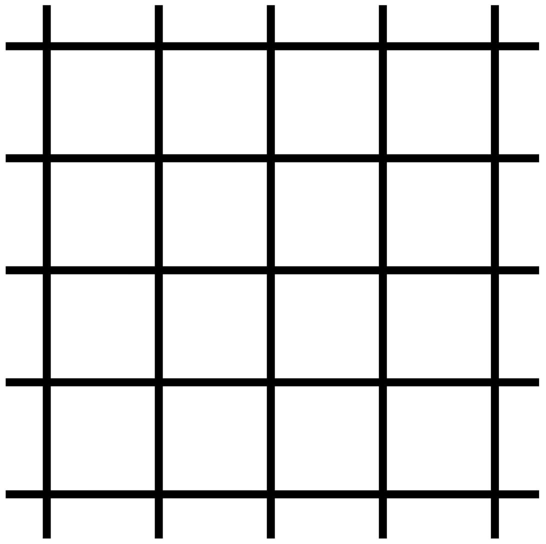
$$V \;\approx\; \frac{1}{3}E \;=\; \frac{1}{6}H$$

Mesh Statistics

- **Triangle meshes**
  - F ≈ 2V
  - E ≈ 3V
  - Average valence = 6



- **Quad meshes**
  - F ≈ V
  - E ≈ 2V
  - Average valence = 4



Soccer Ball

- How many pentagon/hexagons do we need to make a soccer ball?

Face Set Data Structure

- Store for each face:
  - 3 positions
- Used in STL file format
  - not efficient!

Triangles		
x_{11} y_{11} z_{11}	x_{12} y_{12} z_{12}	x_{13} y_{13} z_{13}
x_{21} y_{21} z_{21}	x_{22} y_{22} z_{22}	x_{23} y_{23} z_{23}
...	...	...
x_{F1} y_{F1} z_{F1}	x_{F2} y_{F2} z_{F2}	x_{F3} y_{F3} z_{F3}

36 B/f = 72 B/v

Indexed Face Set Data Structure

- Store for each vertex
  - its position
- Store for each face
  - indices corresponding to its thee vertices
- Used in many file formats
  - OFF, OBJ, VRML

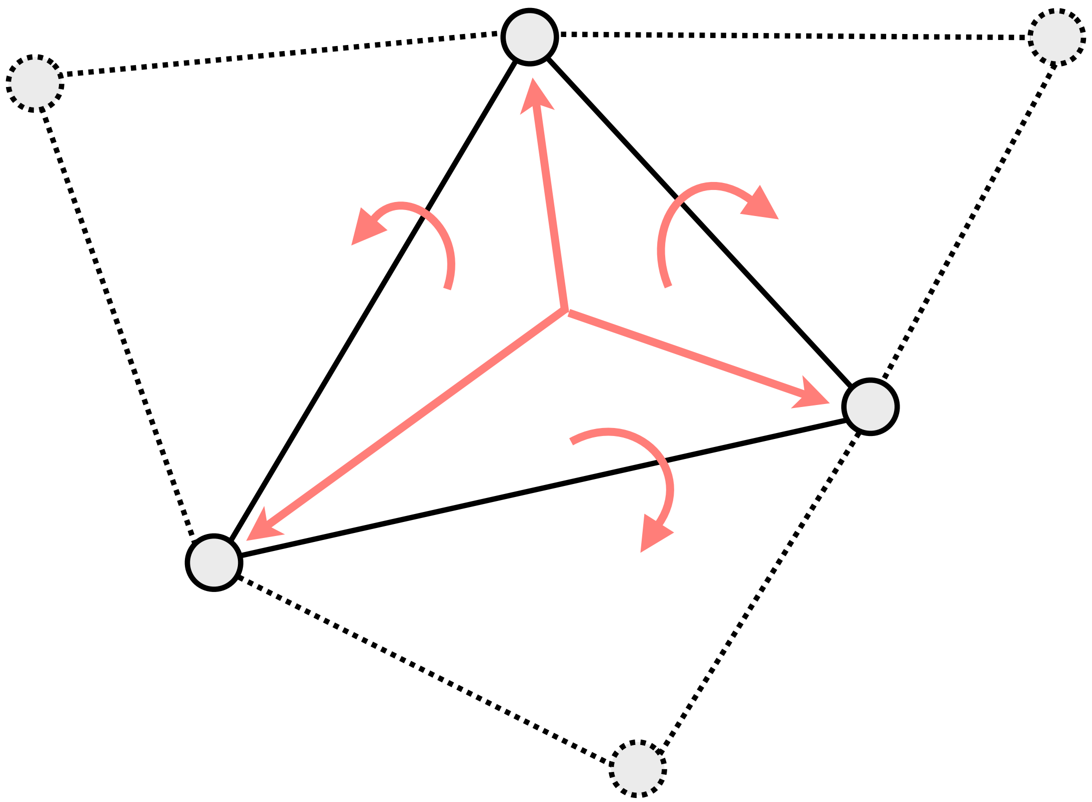
Vertices
x_1 y_1 z_1
. . .
x_V y_V z_V

Triangles
i_{11} i_{12} i_{13}
. . .
. . .
. . .
. . .
i_{F1} i_{F2} i_{F3}

12 B/v + 12 B/f = 36 B/v

Face-Based Connectivity

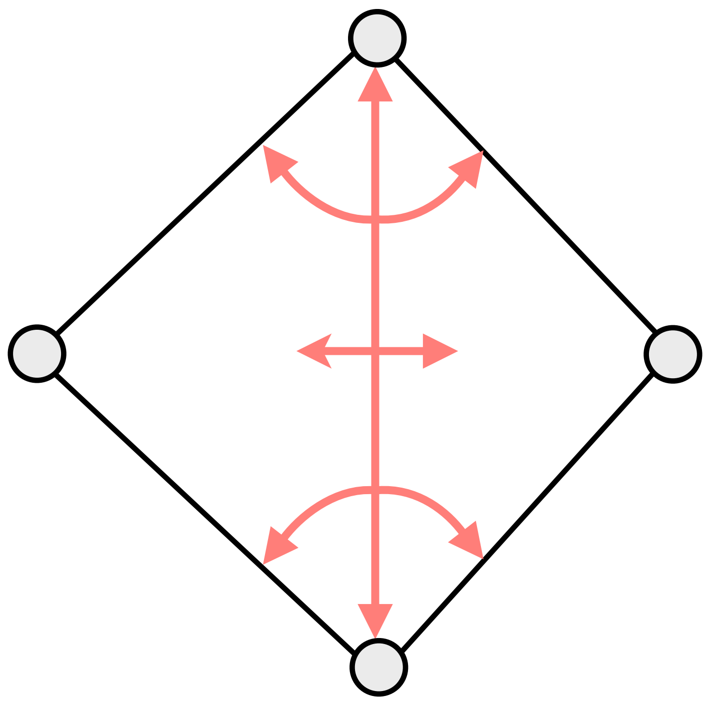
- Vertex:
  - position
  - 1 face
- Face:
  - 3 vertices
  - 3 face neighbors



64 B/v
no edges!

Edge-Based Connectivity

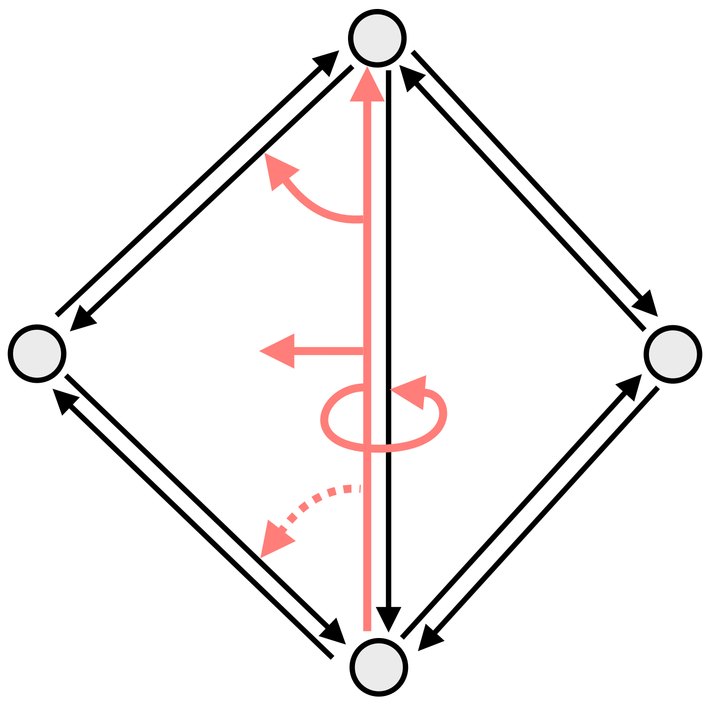
- Vertex
  - position
  - 1 edge
- Edge
  - 2 vertices
  - 2 faces
  - 4 edges
- Face
  - 1 edge



120 B/v
edge orientation?

Halfedge-Based Connectivity

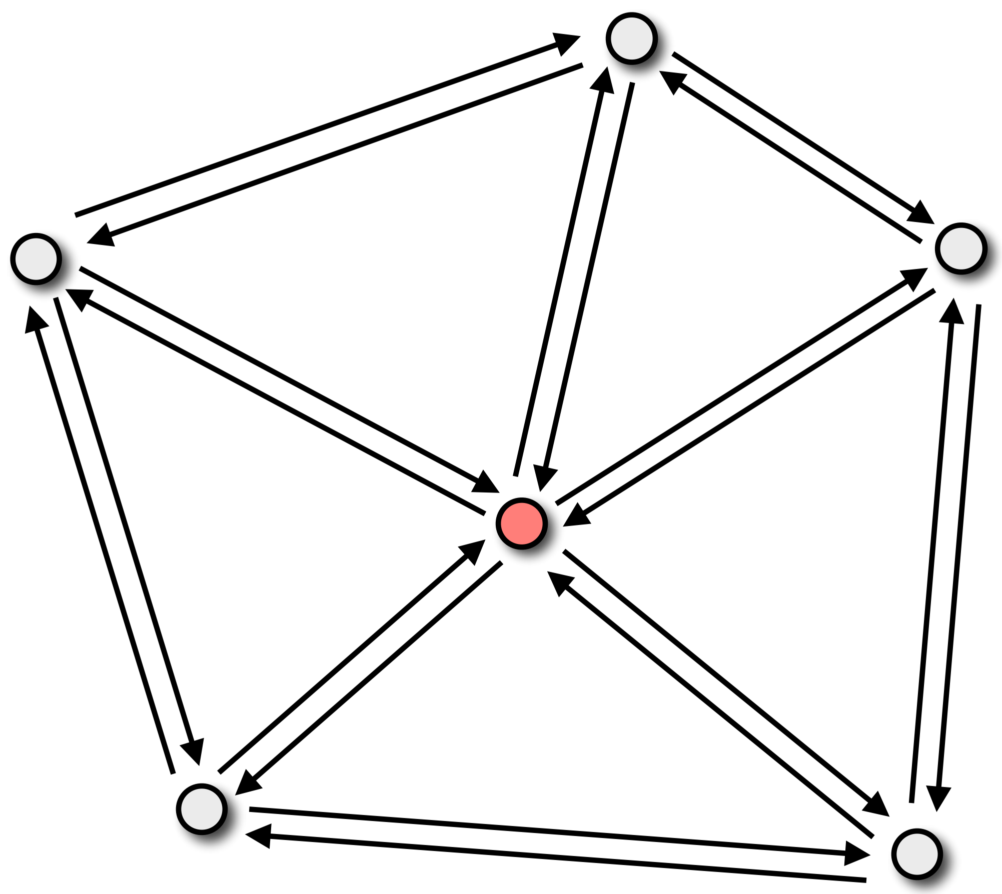
- Vertex
  - position
  - 1 halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 1, 2, or 3 halfedges
- Face
  - 1 halfedge



96 to 144 B/v
no case distinctions
during traversal

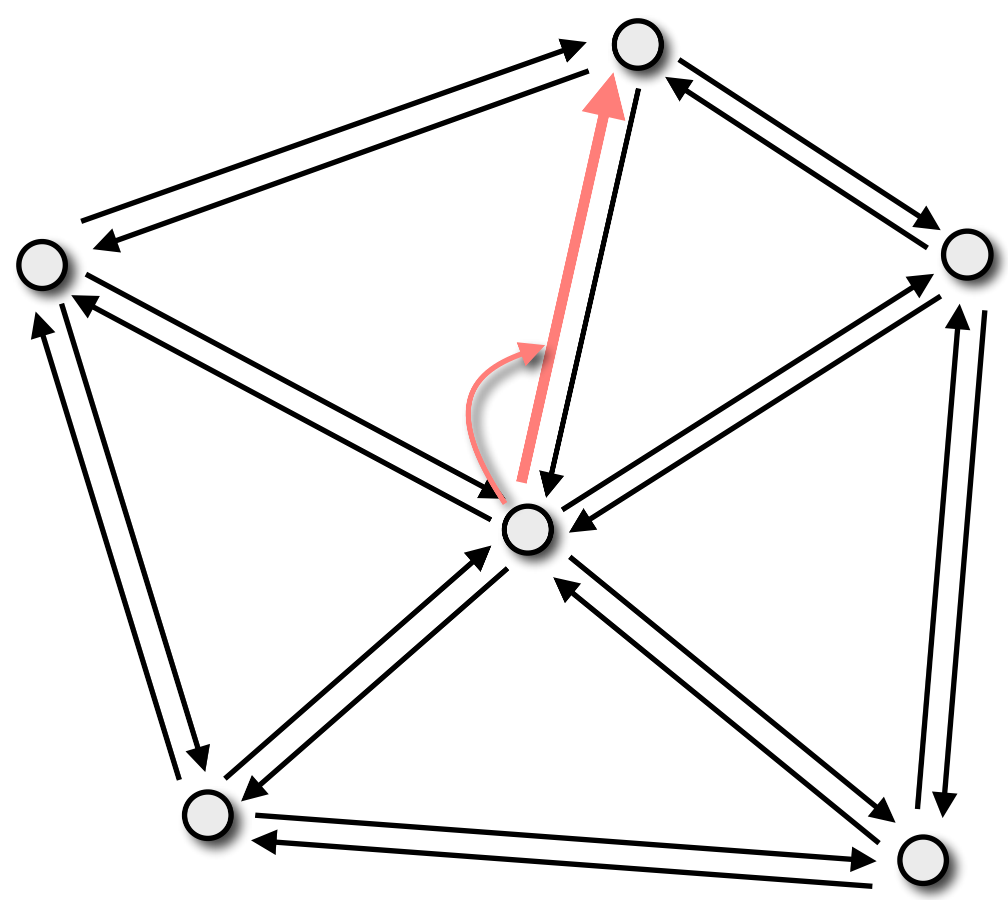
One-Ring Traversal

1. Start at vertex



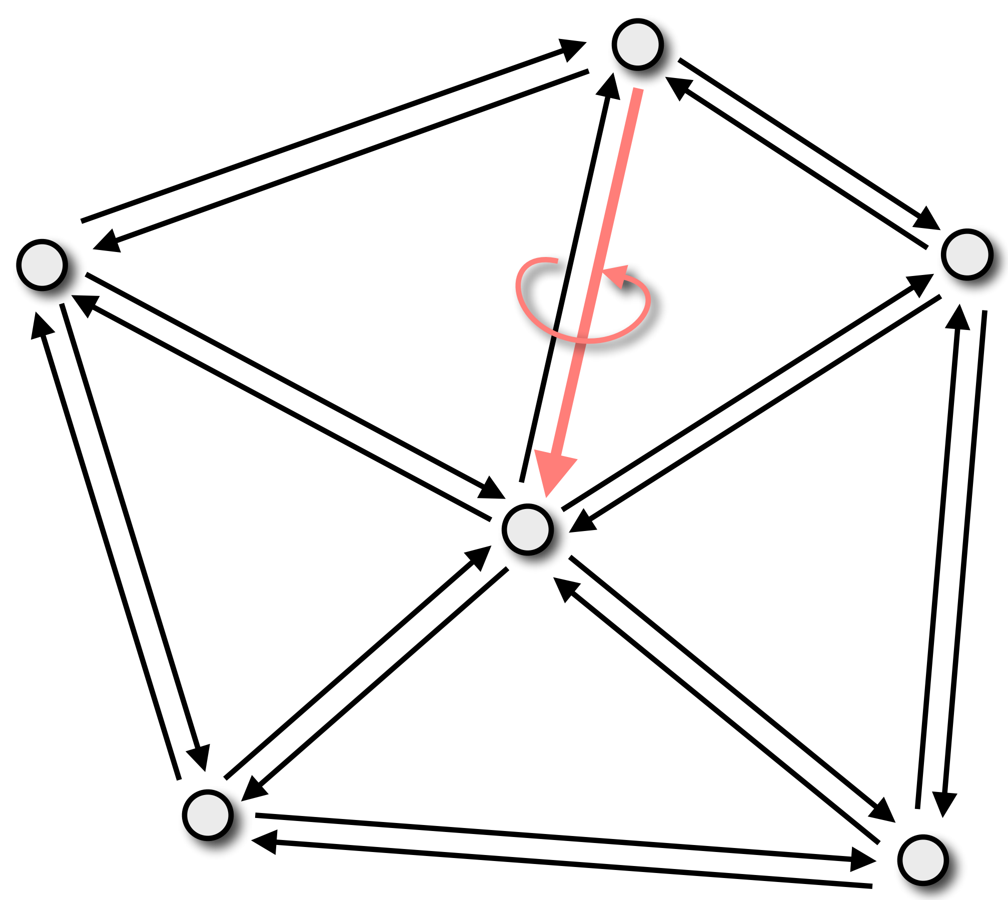
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



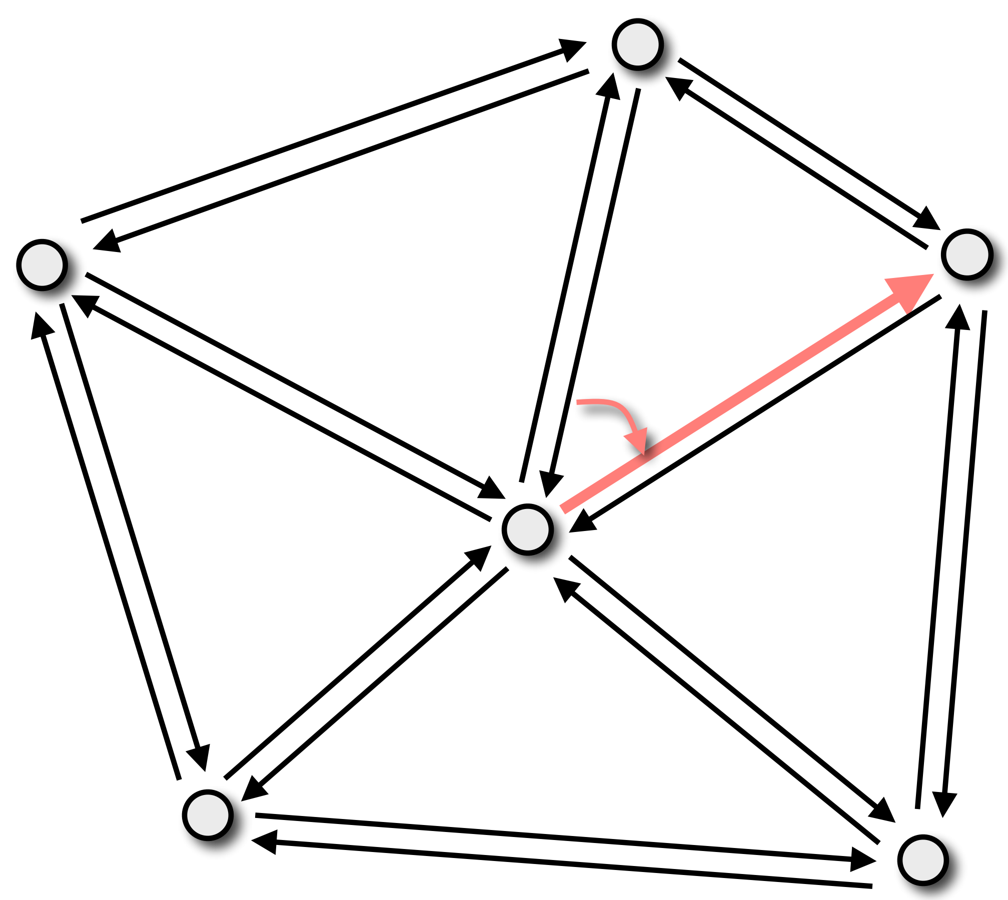
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge



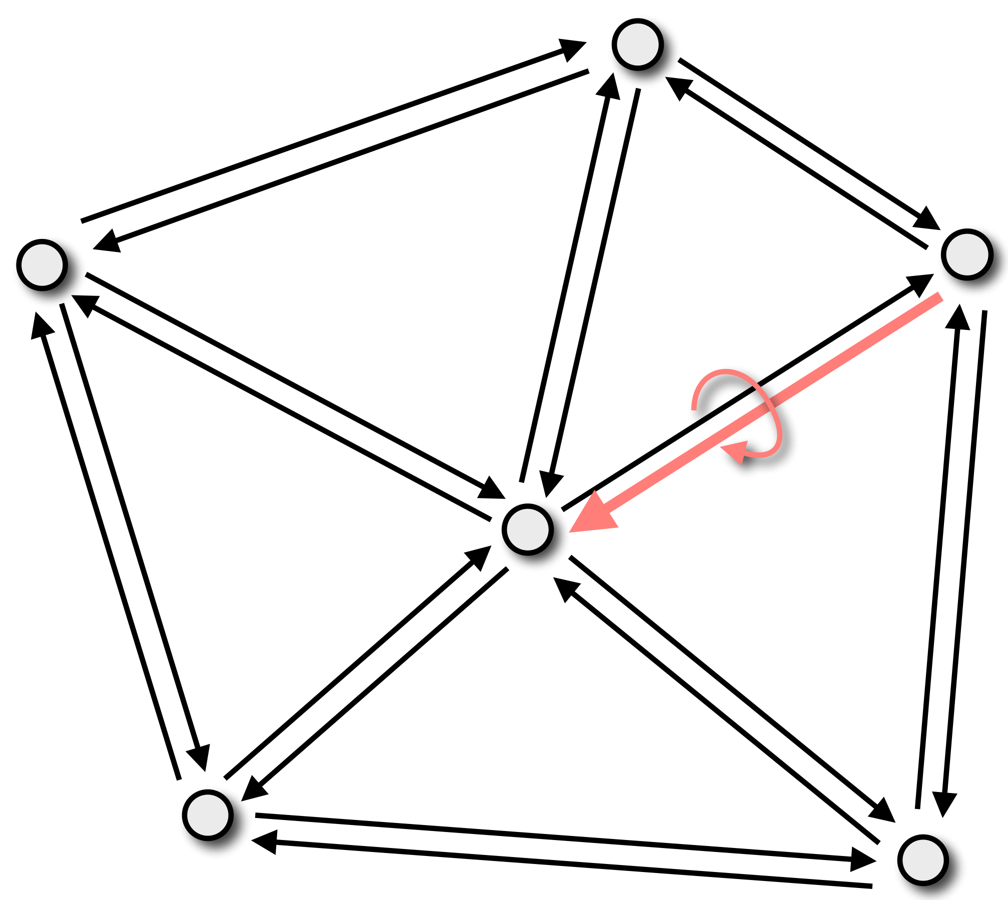
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge



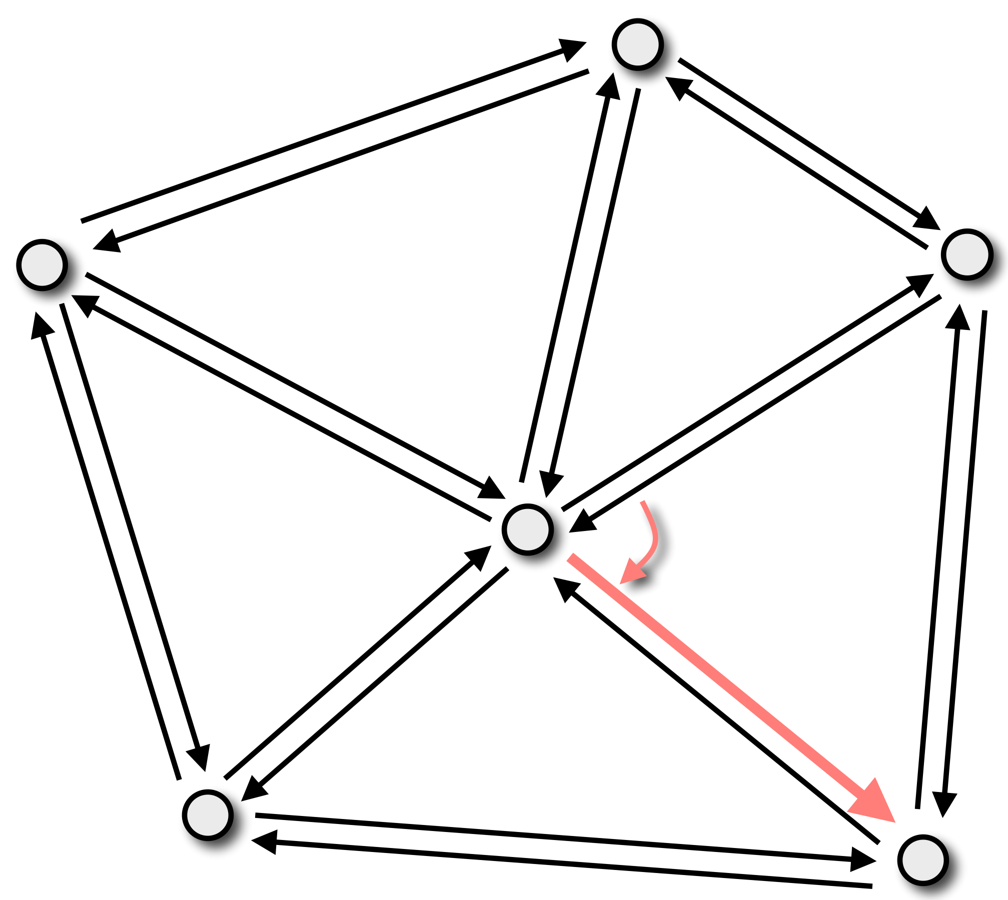
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite



One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge
3. Opposite halfedge
4. Next halfedge
5. Opposite
6. Next
7. ...

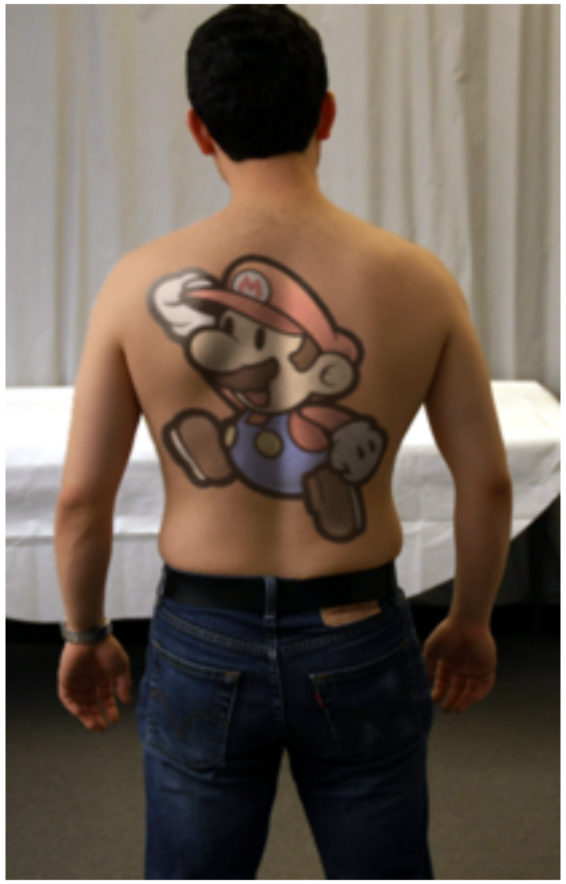


Halfedge-Based Libraries

- CGAL
  - `www.cgal.org`
  - Computational geometry
- OpenMesh
  - `www.openmesh.org`
  - Mesh processing
- `Surface_mesh` of our exercises
  - Like OpenMesh, but easier to use
  - Cool C++11 features ;-)

Euler Formula

- The Euler formula is important / cool, because it is related to beer mugs, soccer, and tattoos.
  (and because it helps us design good data structures)

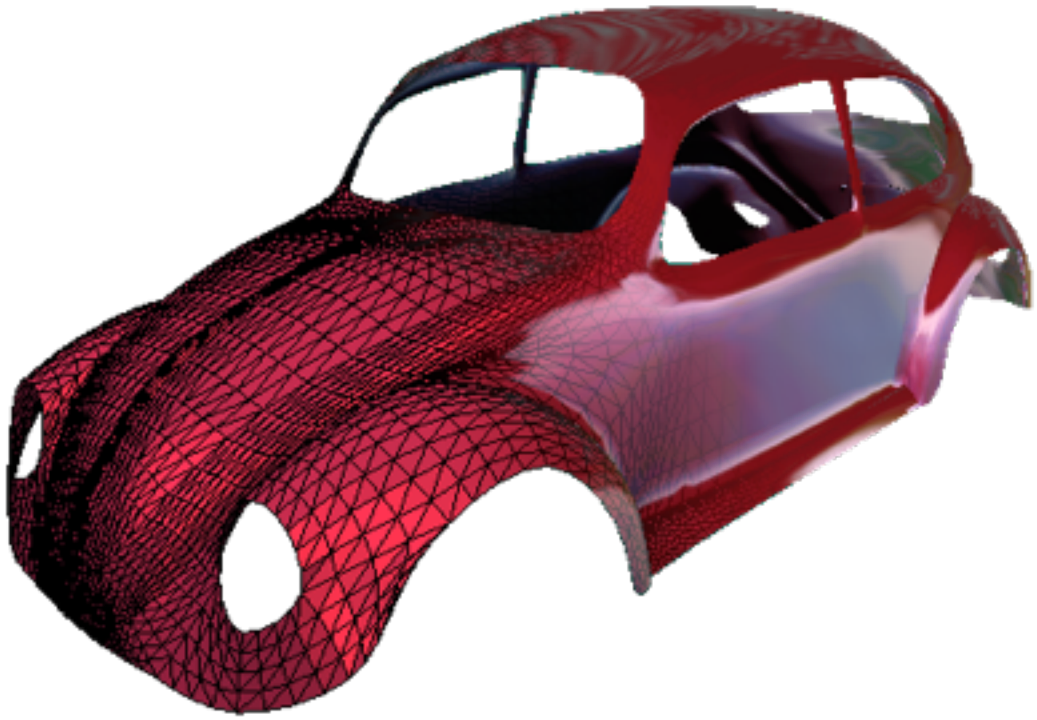


Euler Formula

- The Euler formula gives a cool tattoo ;-)

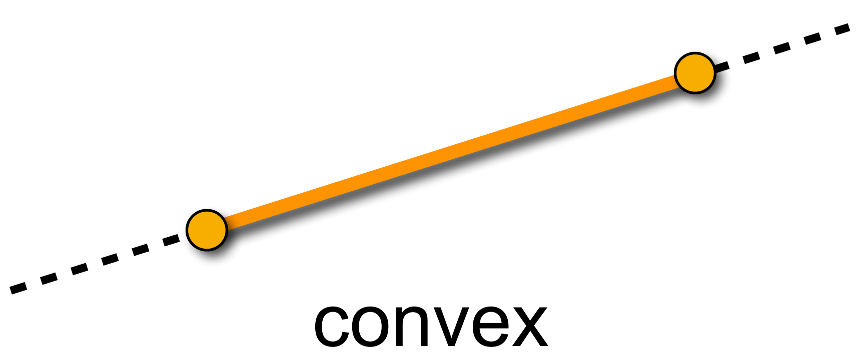
Triangle Meshes

- Data structures
- **Ray Intersection**
- Lighting



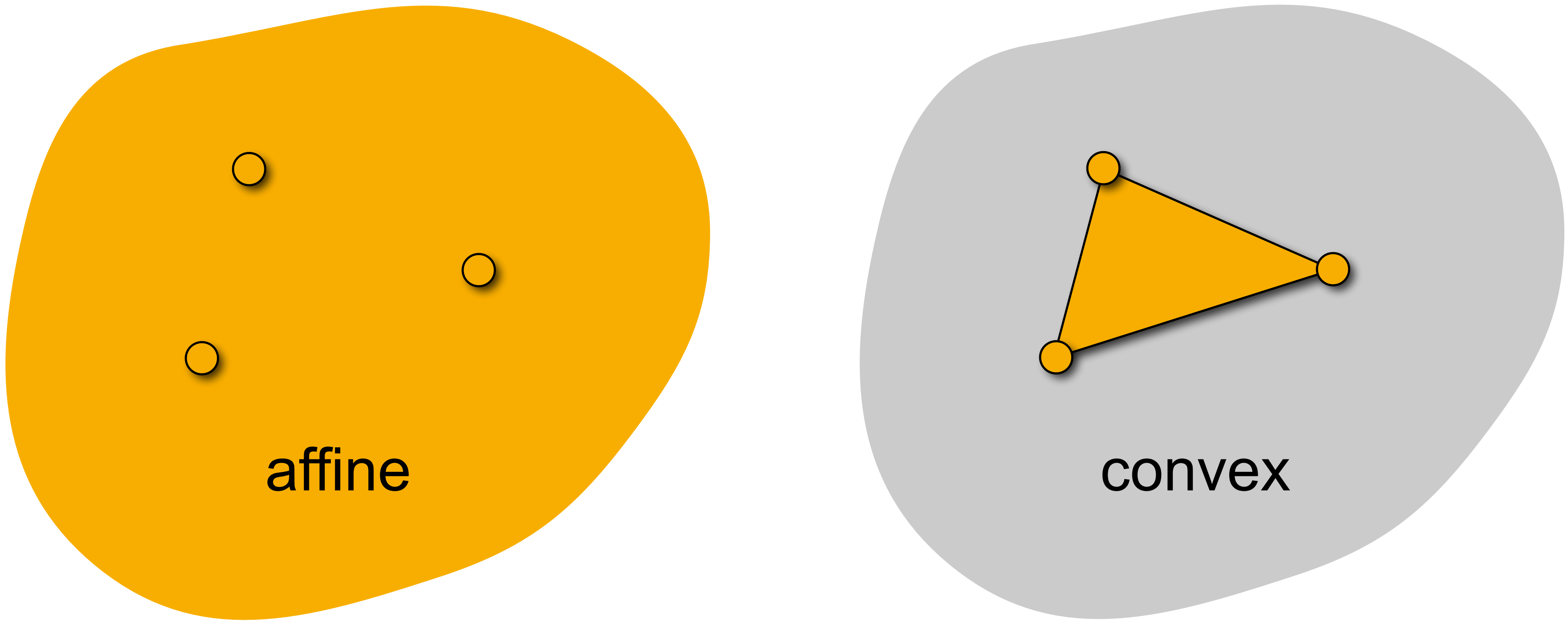
Barycentric Coordinates

- Meaningful linear combinations of *points*
  - *Affine* combination $\sum_i \alpha_i \mathbf{x}_i \quad \text{with} \quad \sum_i \alpha_i = 1$
  - *Convex* combination $\sum_i \alpha_i \mathbf{x}_i \quad \text{with} \quad \sum_i \alpha_i = 1 \ \wedge \ \alpha_i \geq 0$



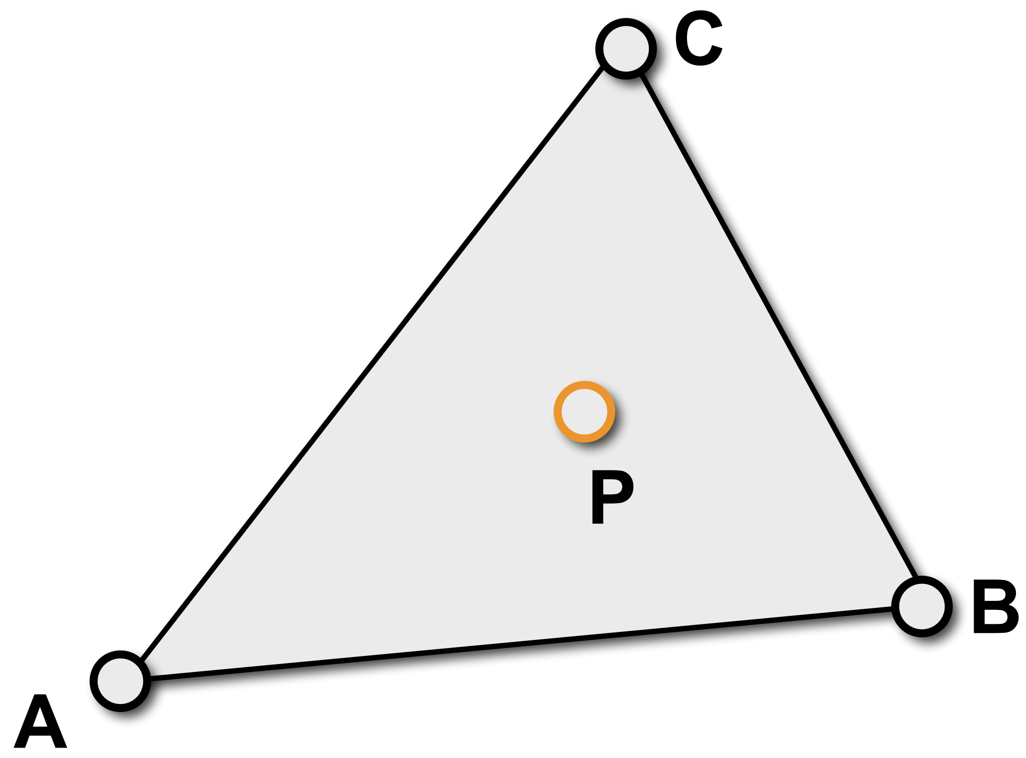
Barycentric Coordinates

- Meaningful linear combinations of *points*
  - *Affine* combination $\sum_i \alpha_i \mathbf{x}_i \quad \text{with} \quad \sum_i \alpha_i = 1$
  - *Convex* combination $\sum_i \alpha_i \mathbf{x}_i \quad \text{with} \quad \sum_i \alpha_i = 1 \ \wedge \ \alpha_i \geq 0$



Barycentric Coordinates

- Represent point as affine combination of $\mathbf{A},\mathbf{B},\mathbf{C}$
  - $\mathbf{P} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$



Barycentric Coordinates

- Represent point as affine combination of $\mathbf{A},\mathbf{B},\mathbf{C}$
  - $\mathbf{P} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-colinear $\mathbf{A},\mathbf{B},\mathbf{C}$

$$\begin{bmatrix} \mathbf{A}_x & \mathbf{B}_x & \mathbf{C}_x \\ \mathbf{A}_y & \mathbf{B}_y & \mathbf{C}_y \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} \mathbf{P}_x \\ \mathbf{P}_y \\ 1 \end{bmatrix}$$

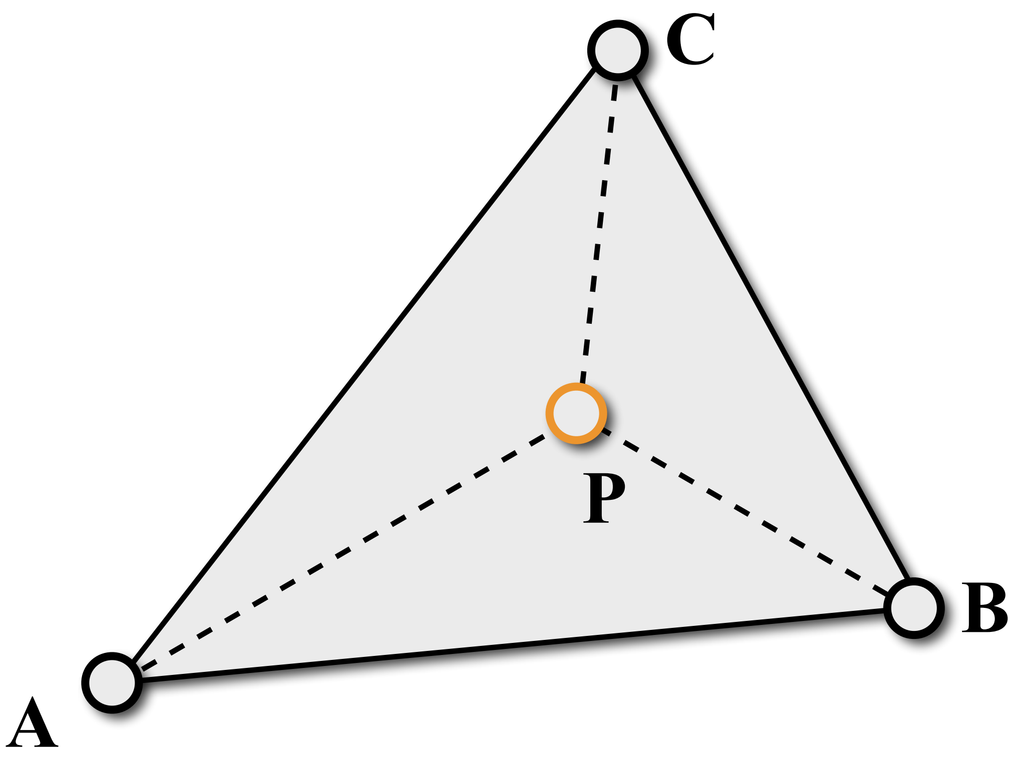
Barycentric Coordinates

- Represent point as affine combination of **A**,**B**,**C**
  - $\mathbf{P} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-colinear **A**,**B**,**C**
  - Ratio of <u>signed</u> triangle areas

$$\alpha(\mathbf{P}) = \frac{\mathrm{area}(\mathbf{P},\mathbf{B},\mathbf{C})}{\mathrm{area}(\mathbf{A},\mathbf{B},\mathbf{C})}$$

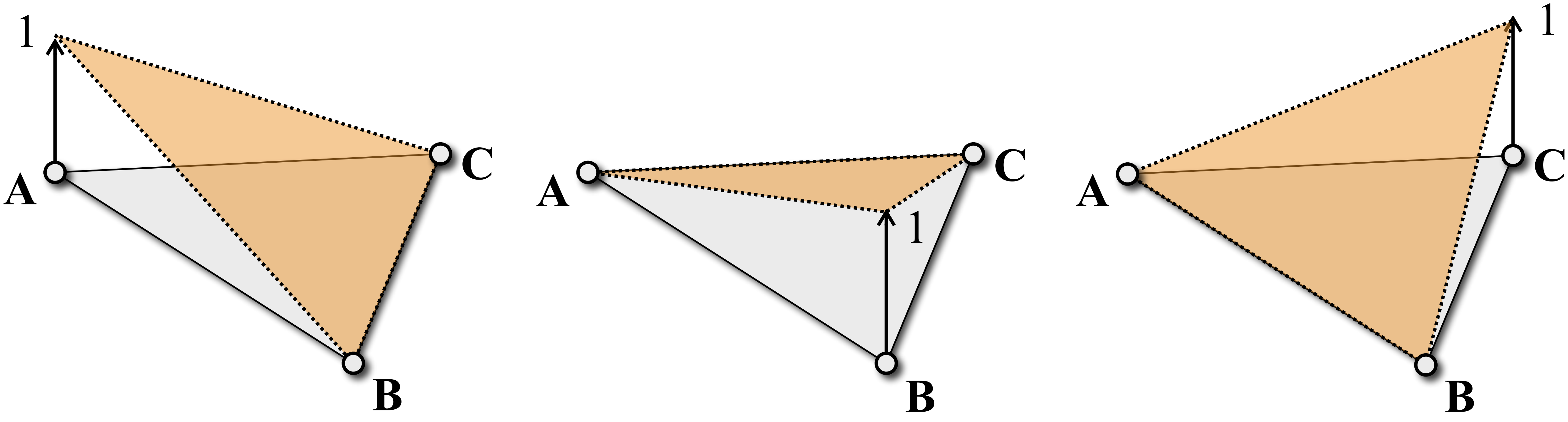
$$\beta(\mathbf{P}) = \frac{\mathrm{area}(\mathbf{P},\mathbf{C},\mathbf{A})}{\mathrm{area}(\mathbf{A},\mathbf{B},\mathbf{C})}$$

$$\gamma(\mathbf{P}) = \frac{\mathrm{area}(\mathbf{P},\mathbf{A},\mathbf{B})}{\mathrm{area}(\mathbf{A},\mathbf{B},\mathbf{C})}$$



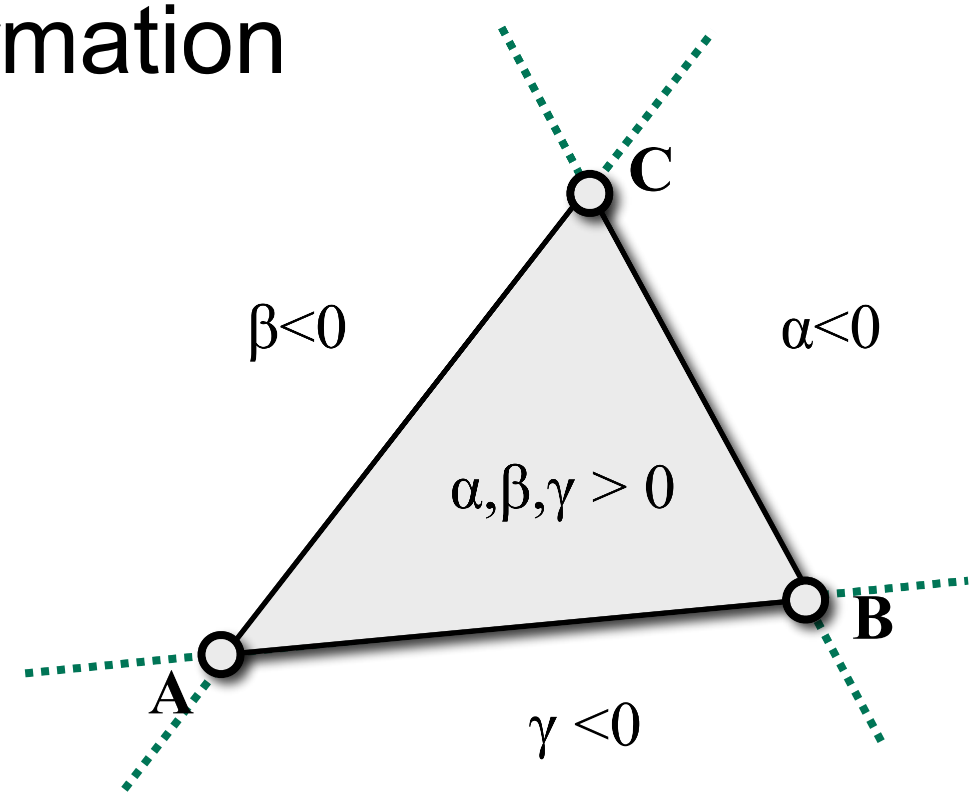
Barycentric Coordinates

- Represent point as affine combination of **A**,**B**,**C**
  - $\mathbf{P} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-colinear **A**,**B**,**C**
  - Ratio of <u>signed</u> triangle areas
  - $\alpha(\mathbf{P})$, $\beta(\mathbf{P})$, $\gamma(\mathbf{P})$ are linear functions



Barycentric Coordinates

- Represent point as affine combination of $\mathbf{A},\mathbf{B},\mathbf{C}$
  - $\mathbf{P} = \alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}$ with $\alpha + \beta + \gamma = 1$
  - Unique for non-colinear $\mathbf{A},\mathbf{B},\mathbf{C}$
  - Ratio of <u>signed</u> triangle areas
  - $\alpha(\mathbf{P})$, $\beta(\mathbf{P})$, $\gamma(\mathbf{P})$ are linear functions
  - Gives inside/outside information



Ray-Triangle Intersection

- Point on plane by barycentric coordinates

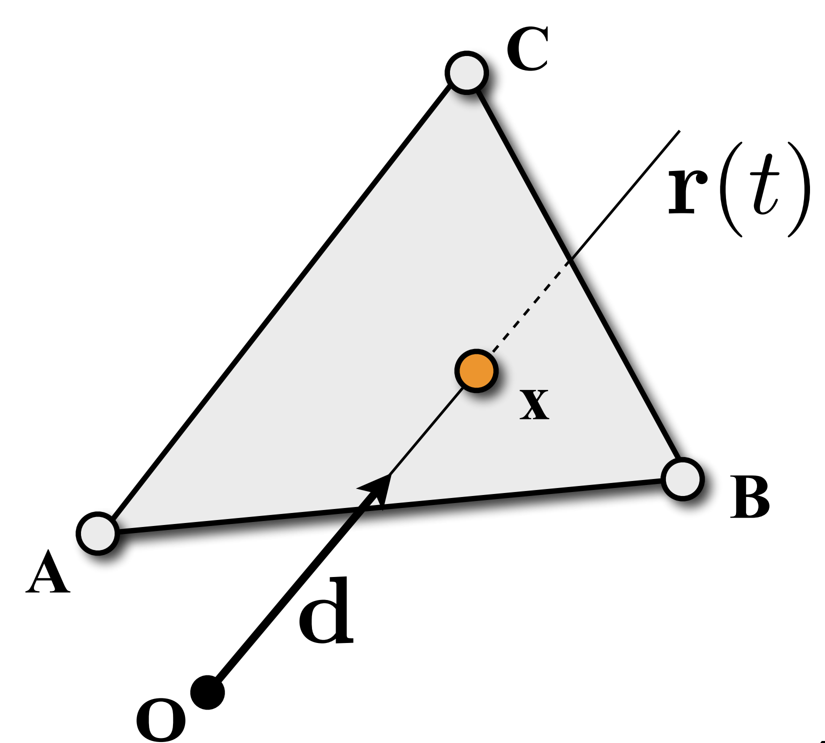
$$\mathbf{x} = \alpha\mathbf{A} + \beta\mathbf{B} + \underbrace{(1 - \alpha - \beta)}_{=\gamma}\mathbf{C}$$

- Solve 3×3 linear system for t, α, β

$$\mathbf{o} + t\mathbf{d} = \alpha\mathbf{A} + \beta\mathbf{B} + (1 - \alpha - \beta)\mathbf{C}$$

- Check inside condition

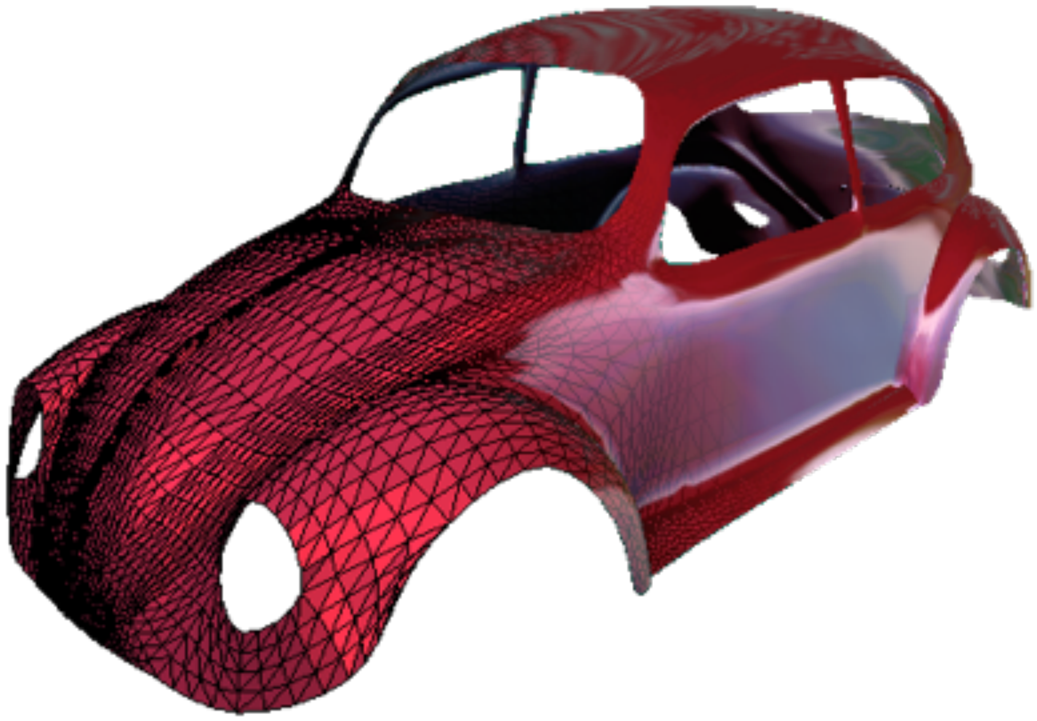
$$0 \leq \alpha, \beta, \gamma \leq 1$$



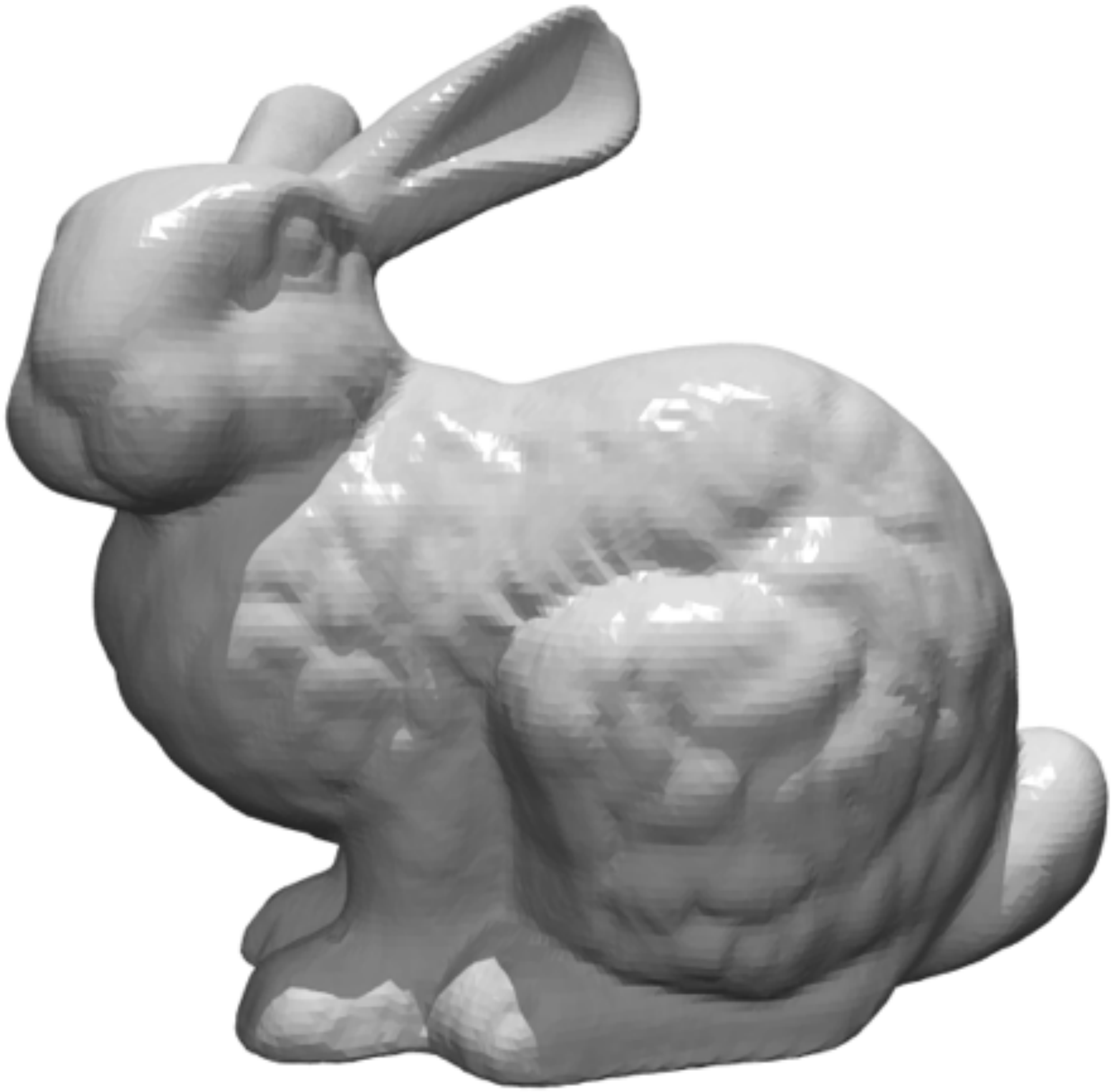
See Pharr 3.6.2 for details

Triangle Meshes

- Data structures
- Ray Intersection
- **Lighting**



Ray Tracing of Tri-Meshes



We get this...

...but want this

Flat Shading

- Use constant face normal for lighting
  - Facetted appearence
  - Mach band effect

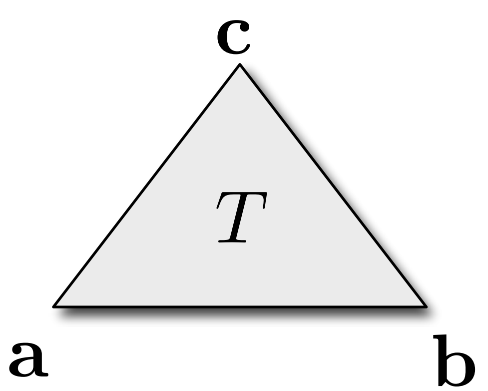
Phong Shading

- Use smooth normal field for lighting
  - Compute normal vectors per vertex
  - Barycentric interpolation of normal vectors
  - Use interpolated normals for lighting

Per-Vertex Normals

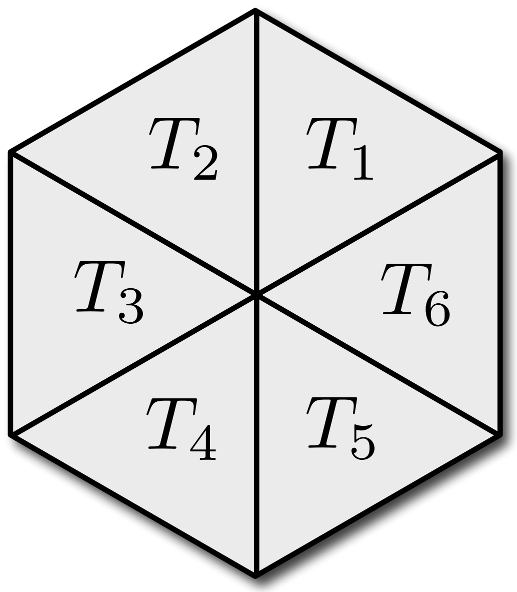
- Per-triangle normal vector

$$\mathbf{n}_T = \frac{(\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a})}{\|(\mathbf{b}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a})\|}$$

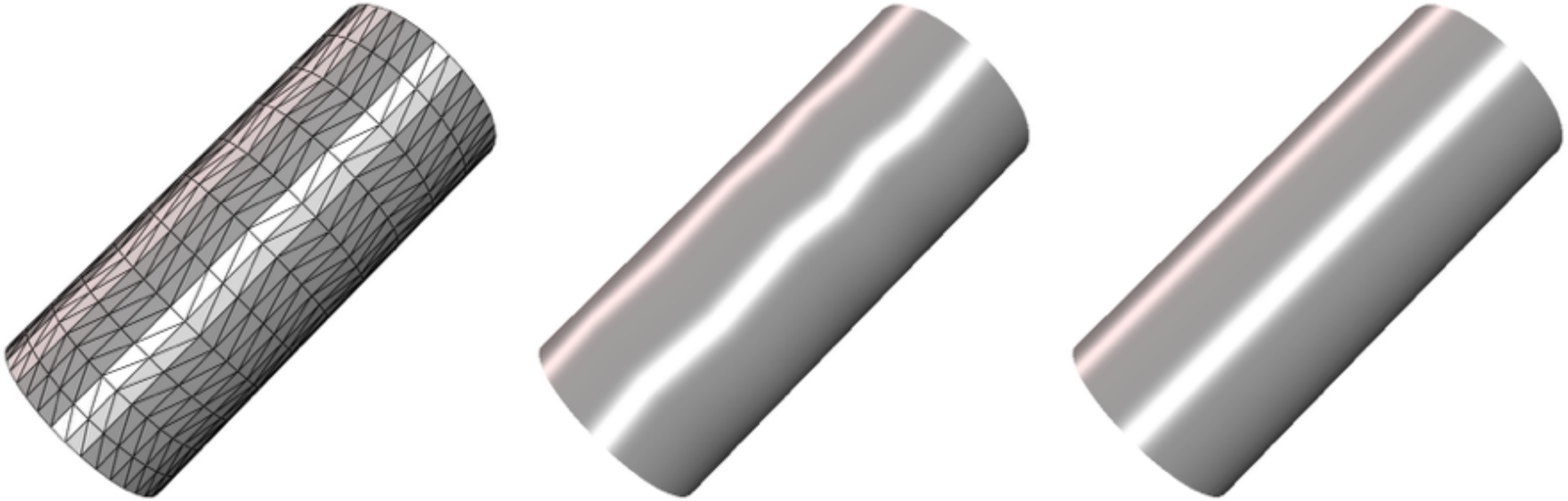


- Per-vertex normal vector
  - Average of incident triangles' normals
  - Weighted by area or opening angle

$$\mathbf{n}_V = \frac{\sum_{T_i \ni V} w_{T_i} \cdot \mathbf{n}_{T_i}}{\left\|\sum_{T_i \ni V} w_{T_i} \cdot \mathbf{n}_{T_i}\right\|}$$



Per-Vertex Normals



triangulation　　no weighting　　angle-weighted

Interpolate Vertex Normals

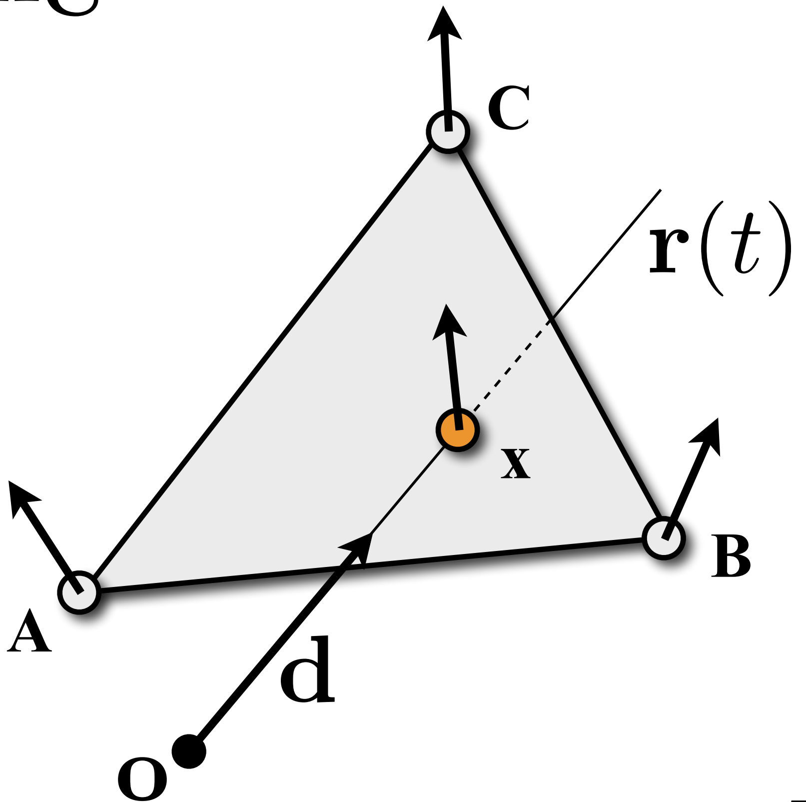
- Intersection point with barycentric coordinates

$$\mathbf{x} = \alpha \mathbf{A} + \beta \mathbf{B} + \gamma \mathbf{C}$$

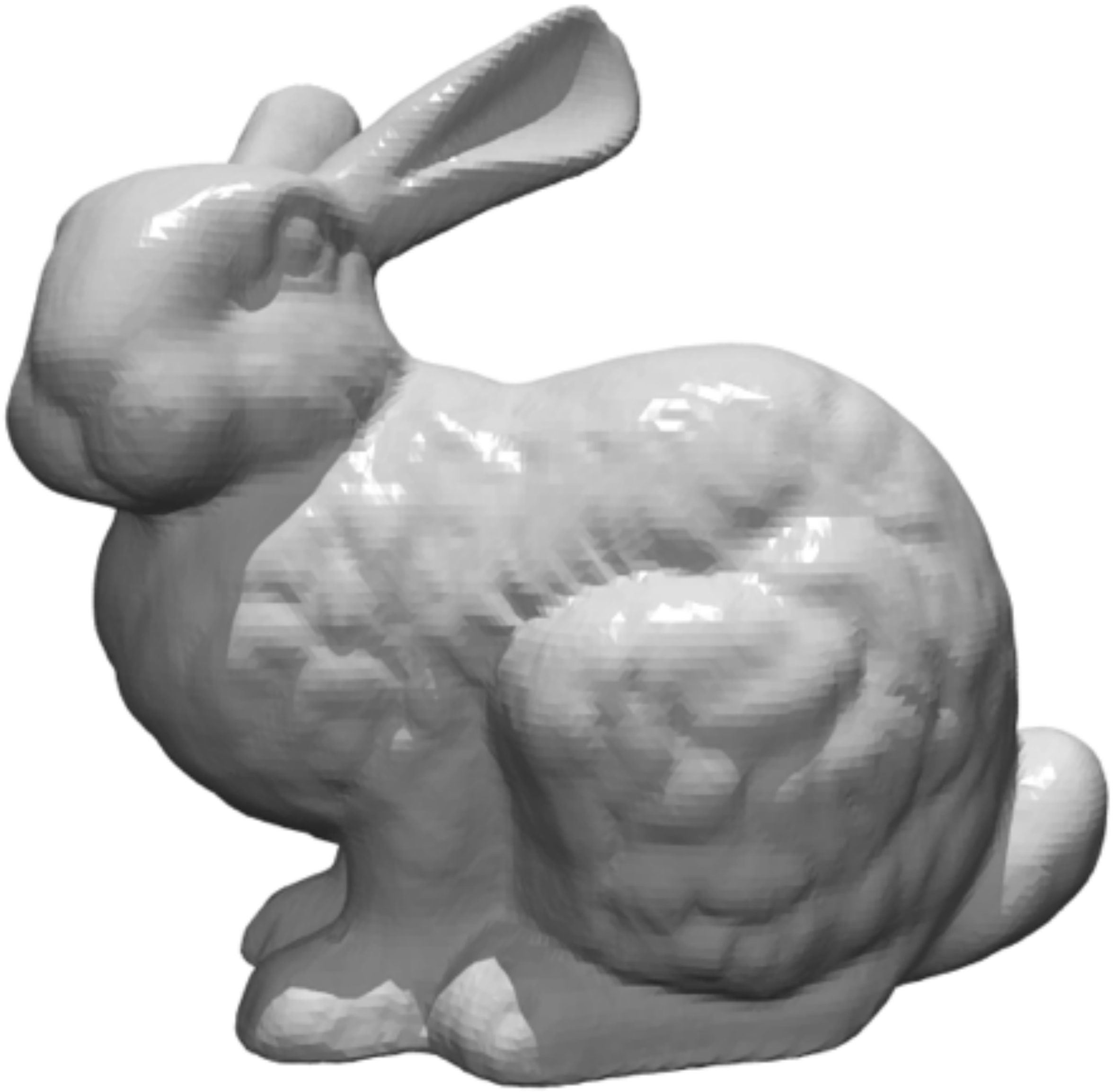
- Linearly interpolate vertex normals

$$\mathbf{n}(\mathbf{x}) \; = \; \alpha \, \mathbf{n}_{\mathbf{A}} + \beta \, \mathbf{n}_{\mathbf{B}} + \gamma \, \mathbf{n}_{\mathbf{C}}$$

- Use $\mathbf{n}(\mathbf{x})$ for lighting point $\mathbf{x}$



Shading

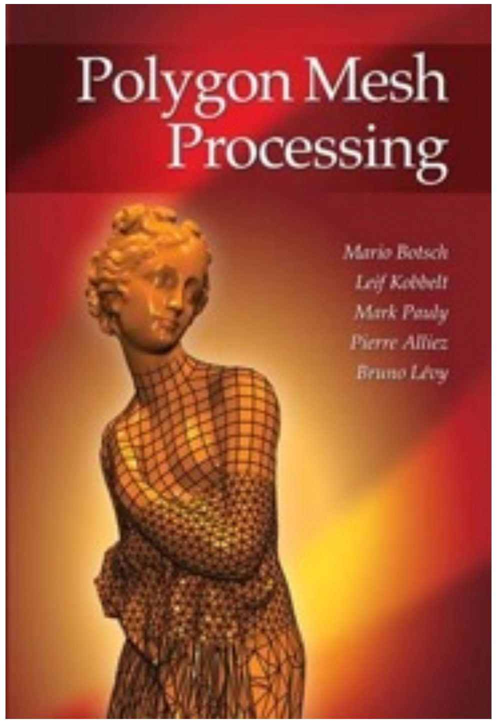


Flat Shading

Phong Shading

Literature

- Botsch, Kobbelt, Pauly, Alliez, Levy: ***Polygon Mesh Processing***, AK Peters, 2010
  - Chapters 1.3 & 2



- Pharr, Humphreys: ***Physically Based Rendering***, Morgan Kaufmann, 2004.
  - Chapter 3