Computer Animation and Games I

CM50244

3D Surface Representation Recap

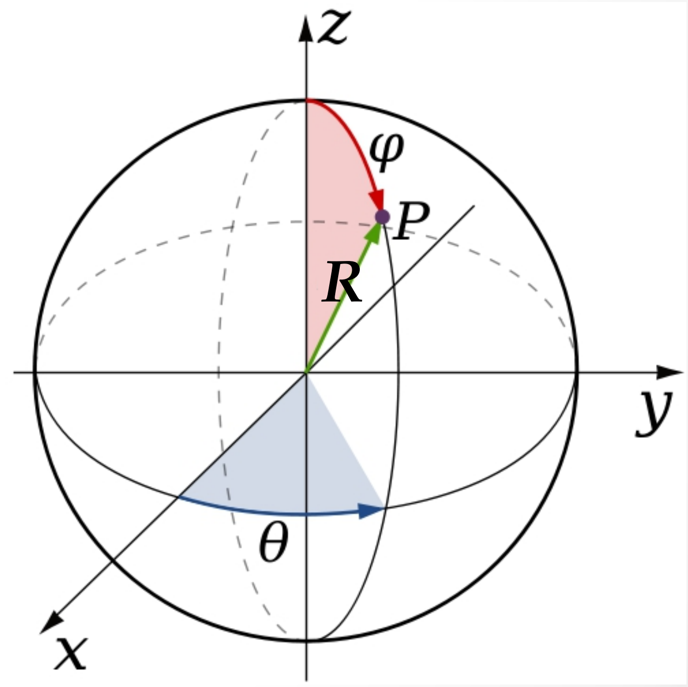
- **Parametric** representation
- **Implicit** representation
- **Explicit** representation

Surface Representation

- **Parametric** representation $x = x(u, v), y = y(u, v), z = z(u, v)$
  - the x, y, z coordinates of a surface point are functions of two parameters *u and v*

$$\mathbf{f} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3, \quad \mathcal{S}_\Omega = \mathbf{f}(\Omega)$$

$$x(\theta, \varphi) = R sin\varphi cos\theta$$
$$y(\theta, \varphi) = R sin\varphi sin\theta$$
$$z(\theta, \varphi) = R cos\varphi$$

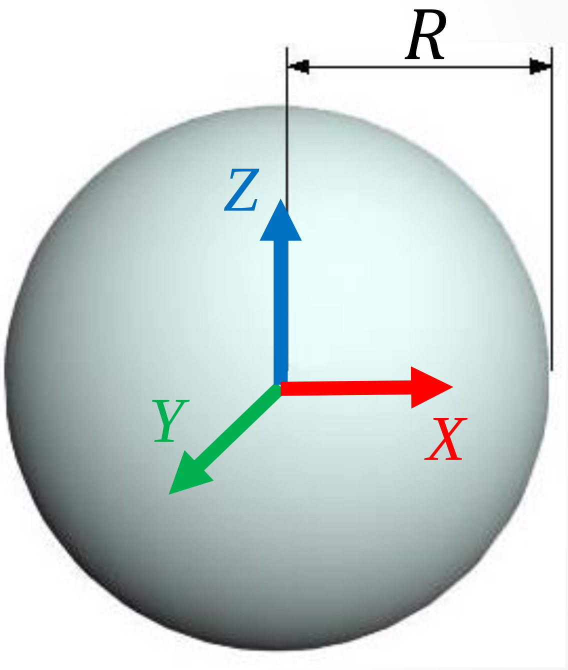


parameter (u, v) provide an easy way to iterate points on the surface

Surface Representation

- **Implicit** representation $f(x, y, z) = 0$
  - For any surface point, x, y, z coordinates should satisfy a single equation

$$x^2 + y^2 + z^2 - R^2 = 0$$



easily check if a point is on/inside/outside the surface by evaluating the function value

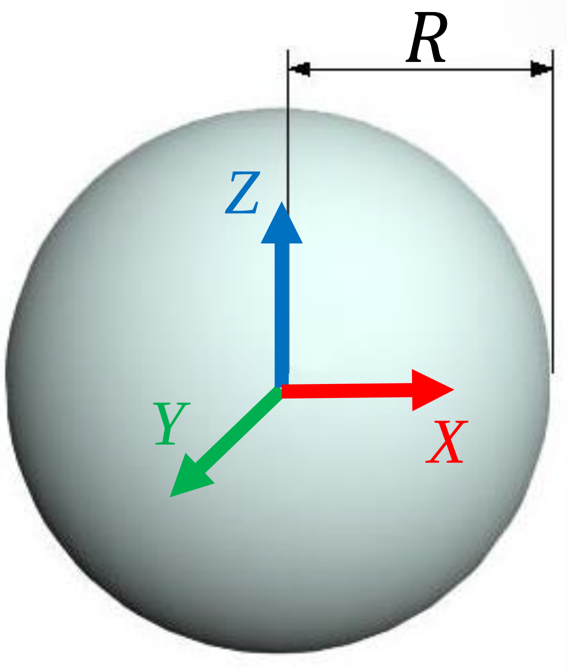
Surface Representation

- **Explicit** representation $z = f(x, y)$
  - z coordinate is explicitly represented as a function of x and y coordinates

$$\boxed{x^2 + y^2 + z^2 - R^2 = 0}$$

$$z = \sqrt{R^2 - x^2 - y^2}$$

$$z = -\sqrt{R^2 - x^2 - y^2}$$



simple since z coord. is explicitly represented by x and y coord.
but not convenient for multi-valued surface function

Mesh Representation

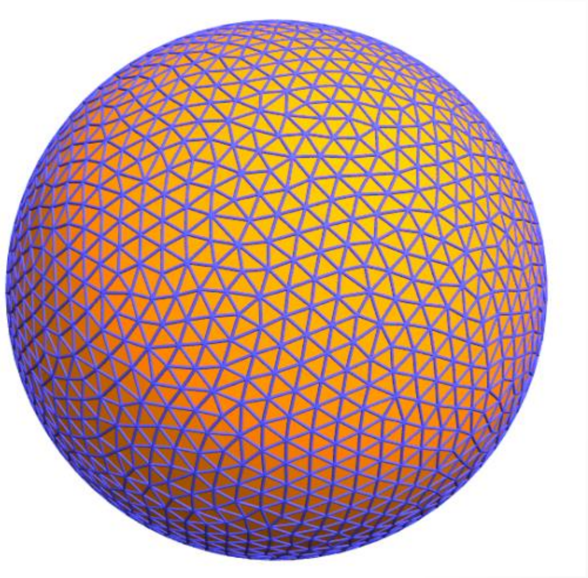
- A **discrete** 3D surface representation

$$M = (V, E, F)$$

$V: mesh\ vertex\ set$

$E: mesh\ edge\ set$

$F: mesh\ face\ set$



Today's Lectures

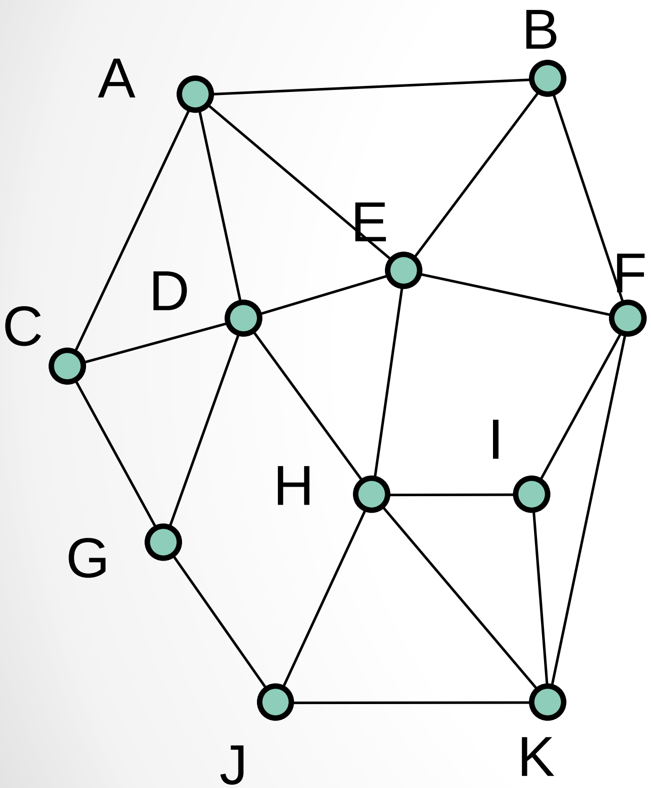
- 3D Mesh Representation and Data Structures
- 2D and 3D Transformations

Mesh Representation and Mesh Data Structures

Overview

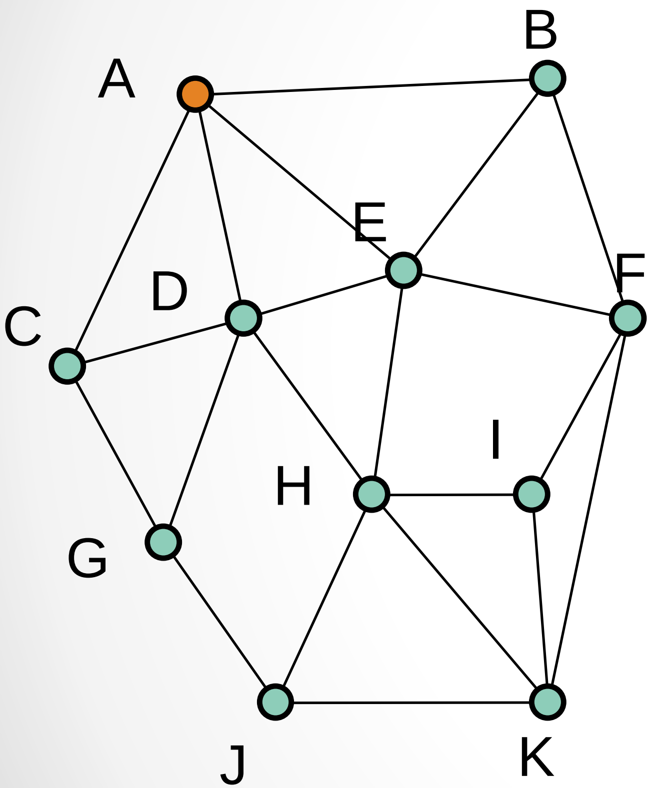
- **3D Mesh Representation**
- Mesh Data Structures

Graph Definitions



Graph $\{V, E\}$

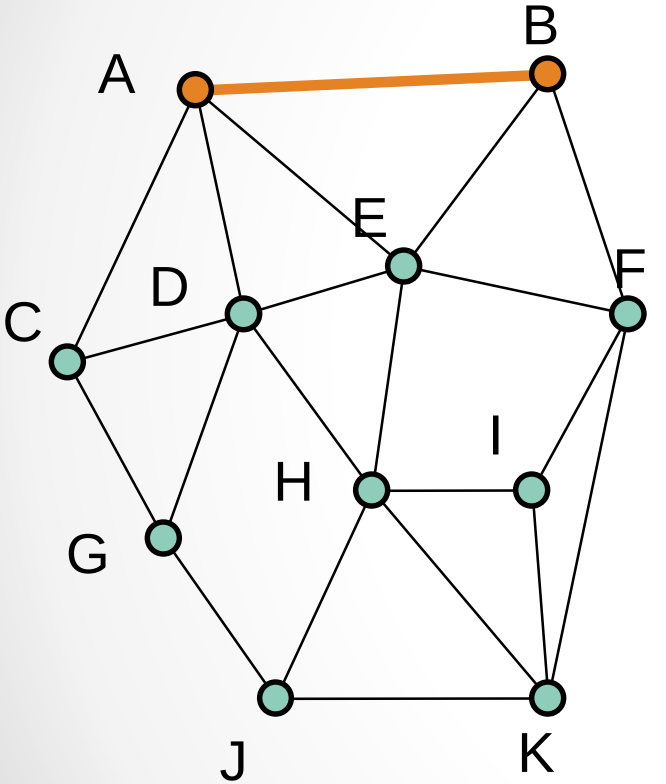
Graph Definitions



Graph $\{V, E\}$

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

Graph Definitions

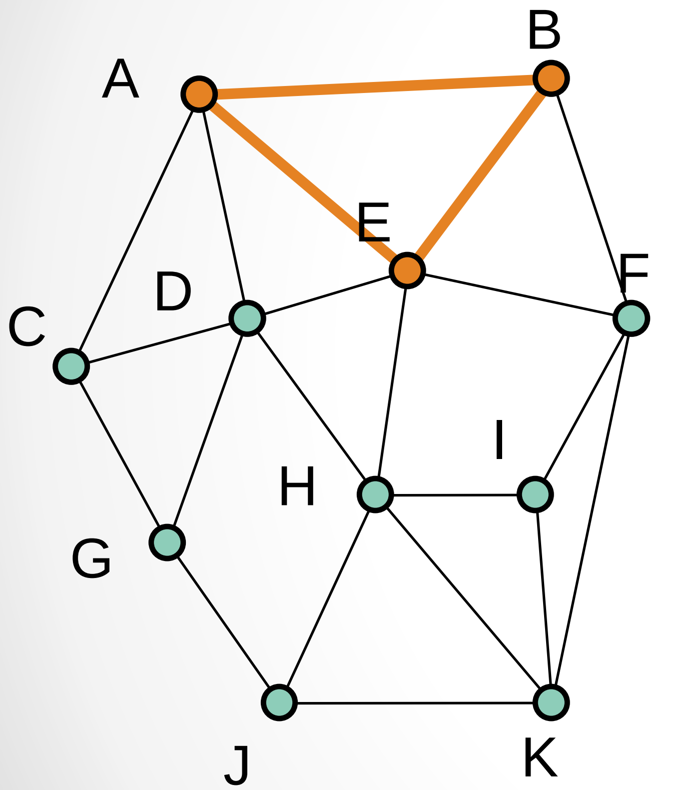


Graph {V, E }

Vertices V = {A, B, C, …, K}

Edges E = {(AB), (AE), (CD), …}

Graph Definitions



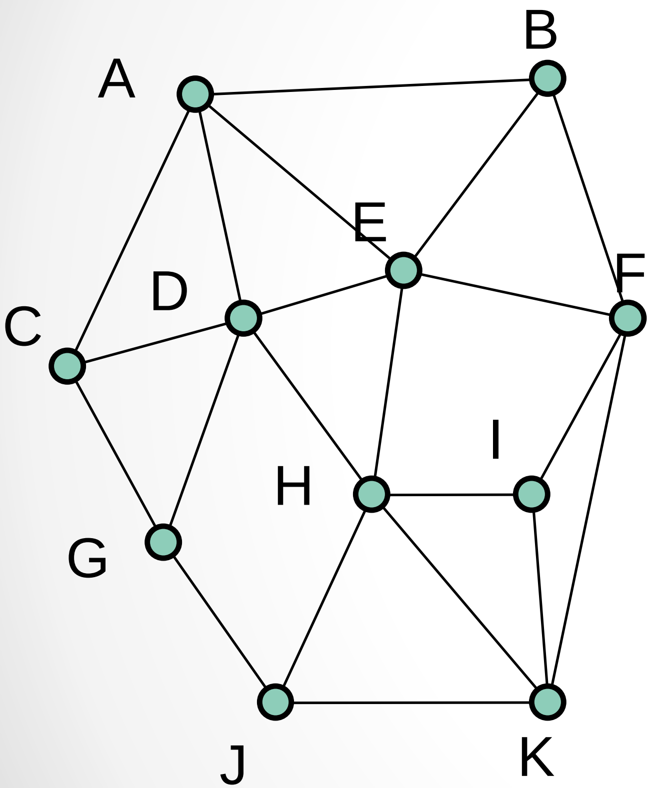
Graph $\{V, E\}$

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

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

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

Graph Definitions

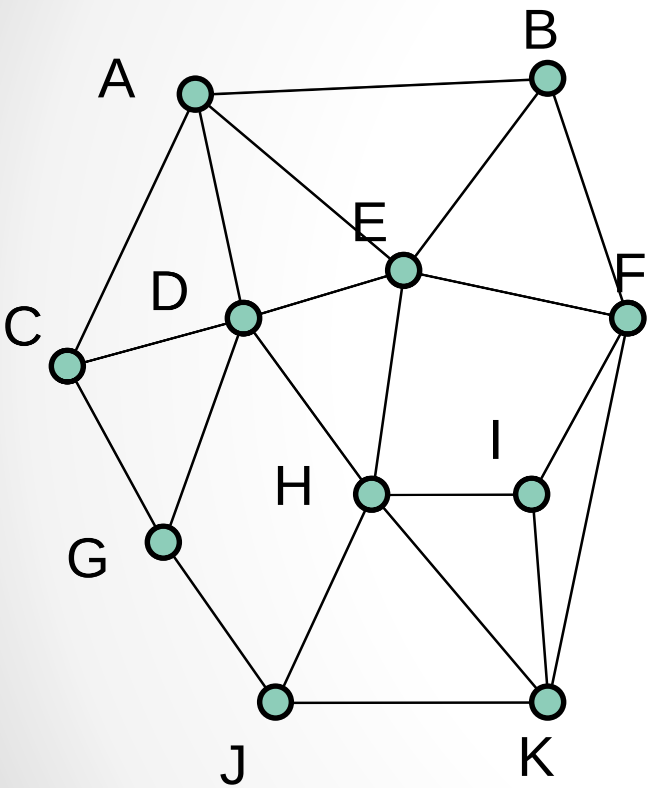


Vertex degree or **valence**:
number of incident edges.

deg(A) = 4

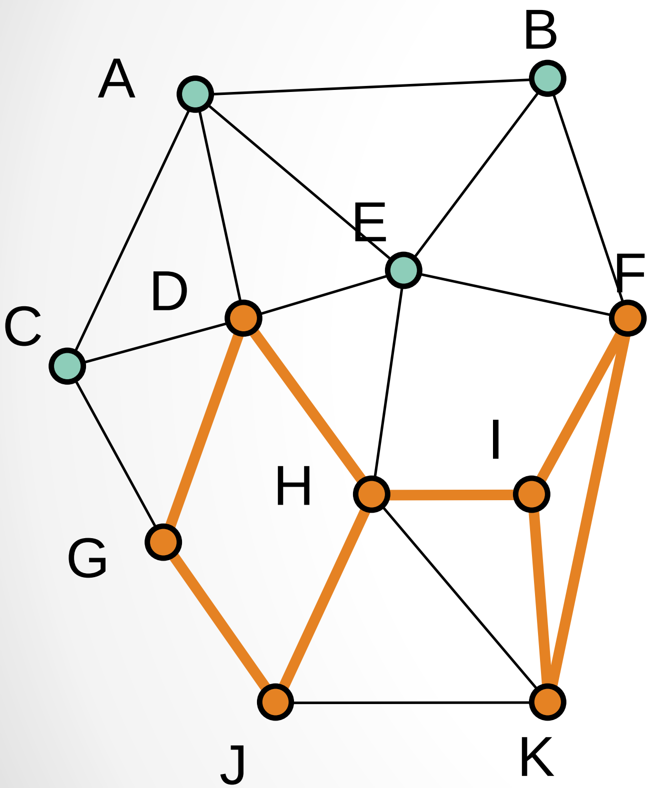
deg(E) = 5

Connectivity



Connected: Path of edges connecting every two vertices.

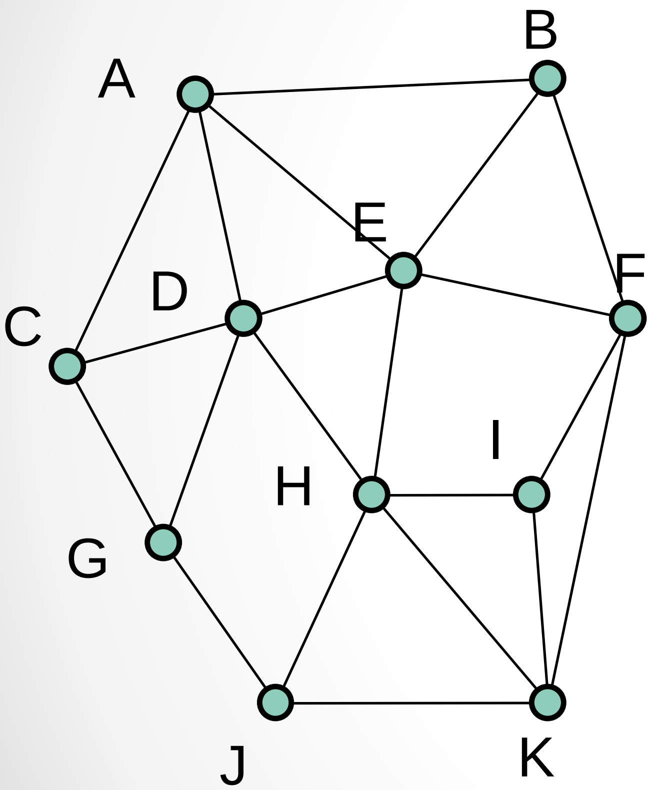
Connectivity



Connected: Path of edges connecting every two vertices.

Subgraph: Graph {*V'*, *E'*} is a subgraph of graph {*V*, *E* } if *V'* is a subset of *V* and *E'* is a subset of *E* incident on *V'*.

Connectivity

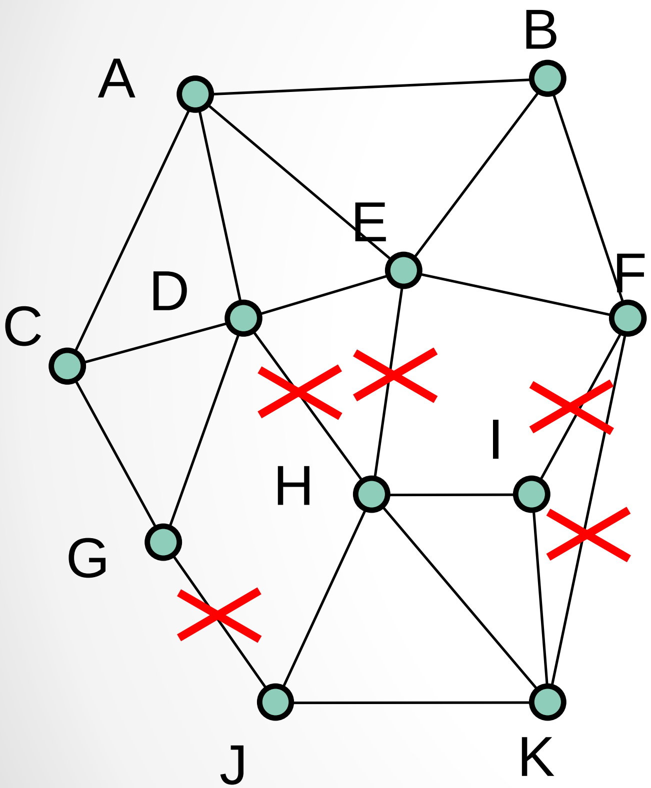


Connected: Path of edges connecting every two vertices.

Subgraph: Graph $\{V', E'\}$ is a subgraph of graph $\{V, E\}$ if V' is a subset of V and E' is a subset of E incident on V'.

Connected component: Maximally connected subgraph.

Connectivity

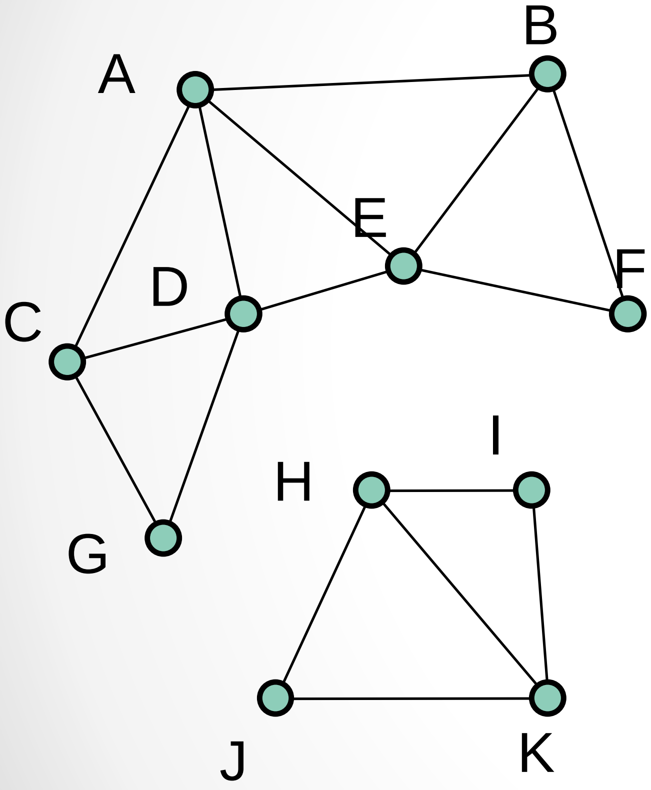


Connected: Path of edges connecting every two vertices.

Subgraph: Graph {*V'*, *E'*} is a subgraph of graph {*V*, *E* } if *V'* is a subset of *V* and *E'* is a subset of *E* incident on *V'*.

Connected component: Maximally connected subgraph.

Connectivity



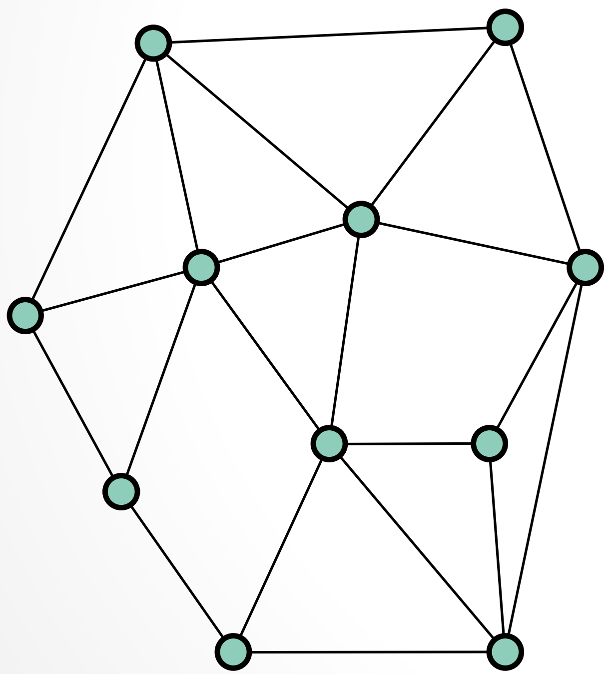
Connected: Path of edges connecting every two vertices.

Subgraph: Graph {*V'*, *E'*} is a subgraph of graph {*V*, *E* } if *V'* is a subset of *V* and *E'* is a subset of *E* incident on *V'*.

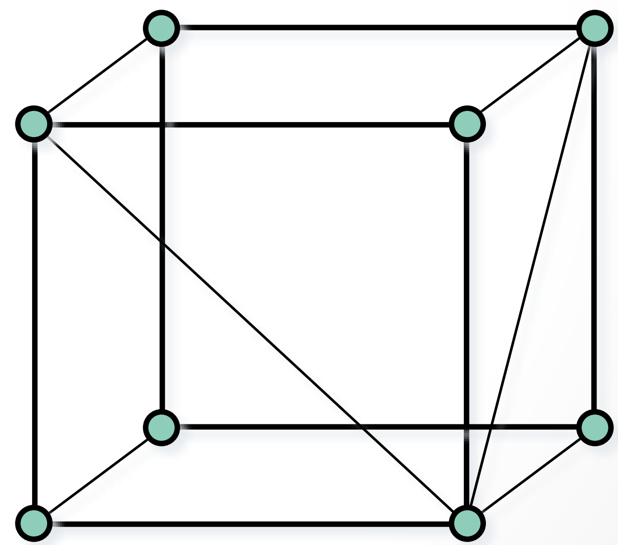
Connected component: Maximally connected subgraph.

Graph Embedding

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



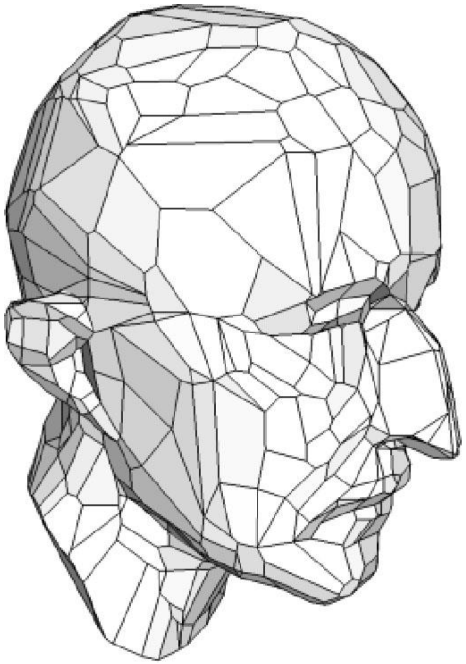
Embedded in $\mathbf{R}^2$



Embedded in $\mathbf{R}^3$

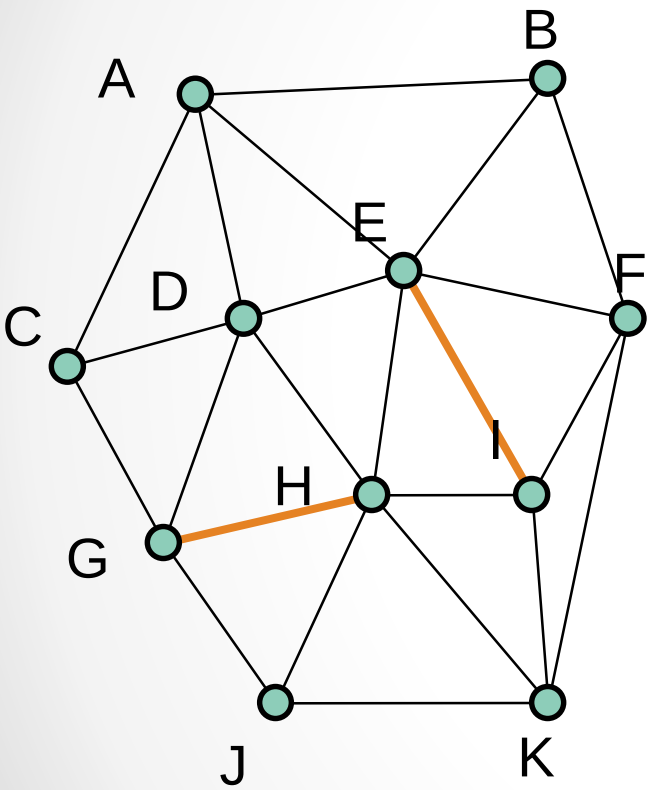
Graph Embedding

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 and convex
- any polygon can be triangulated

Mesh Representation

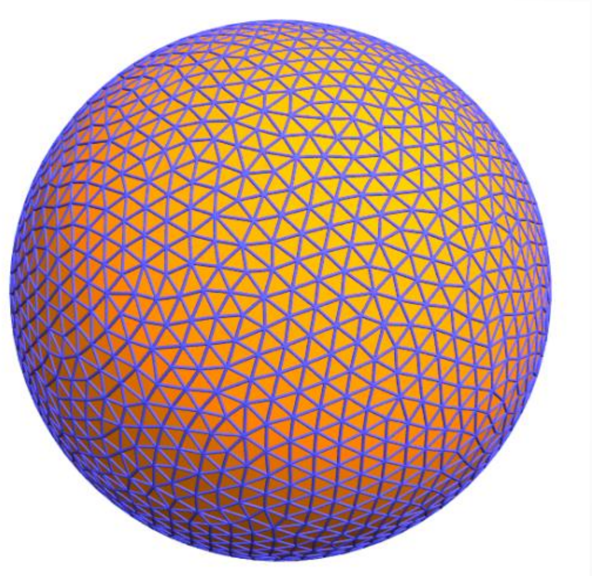
- A **discrete** 3D surface representation

$$M = (V, E, F)$$

$V: mesh\ vertex\ set$

$E: mesh\ edge\ set$

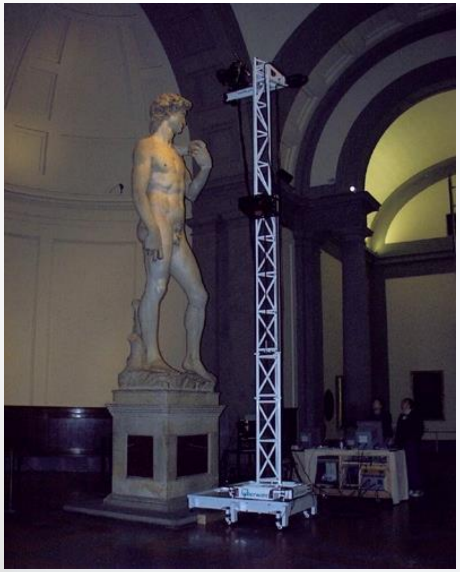
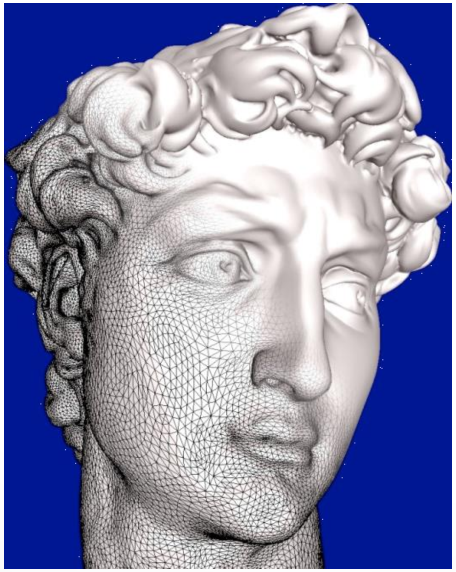
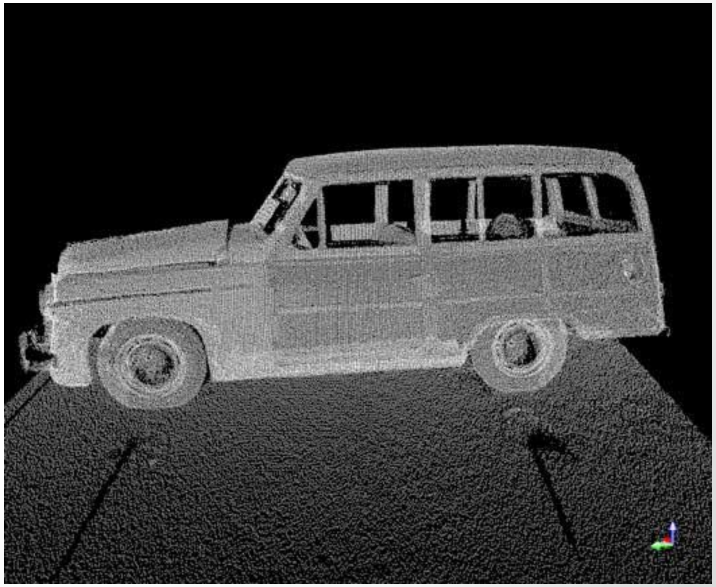
$F: mesh\ face\ set$



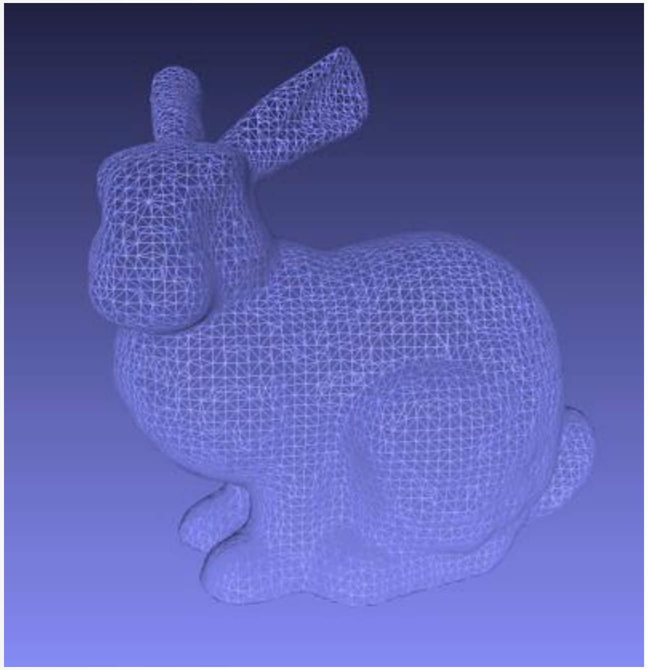
Can be treated as a special **graph** embedded in 3D

Why Mesh?

- Simplicity and generality (a set of vertices & a set of faces)
- Efficiently rendered by graphics hardware
- Output of most acquisition tools (Laser Scanner, Kinect…)
- Input to most simulation/analysis tools (FE solvers)

Real Meshes

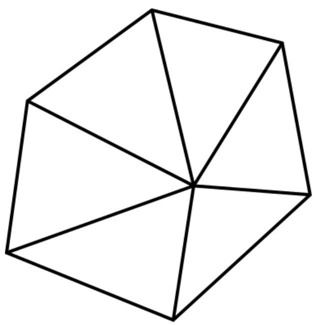


Stanford Bunny
8171 vertices,
16301 triangles
http://graphics.stanford.edu/data/3Dscanrep/

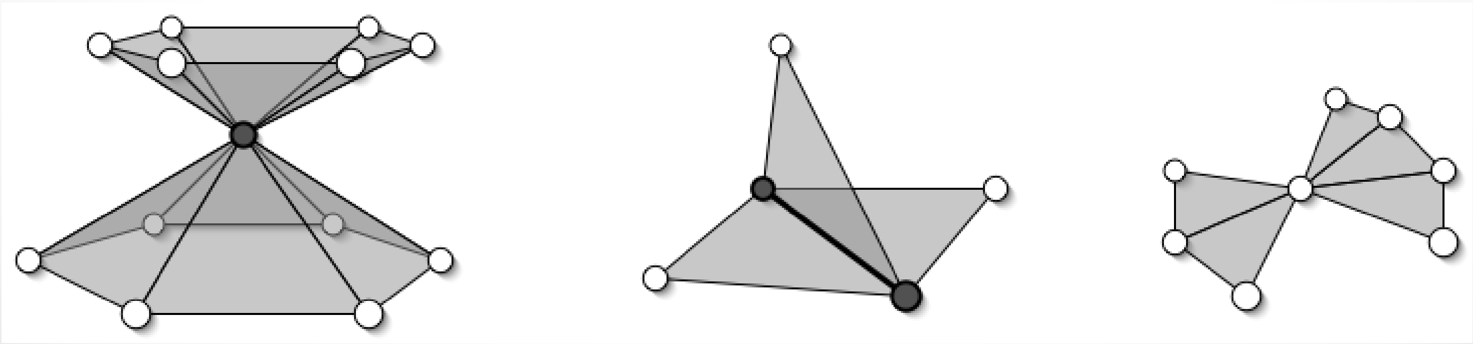
Digital Michelangelo Project
28,184,526 vertices,
56,230,343 triangles
http://graphics.stanford.edu/projects/mich/

Manifold Mesh

- Local neighborhoods are disk-shaped

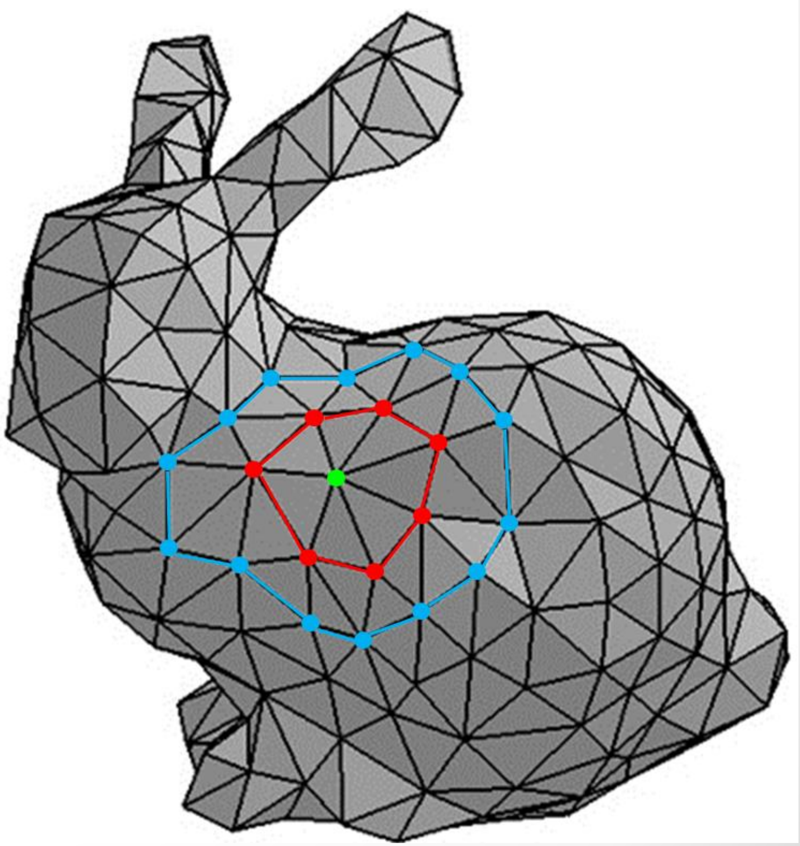


- Required by lots of mesh processing algorithms
- Non-manifold examples:



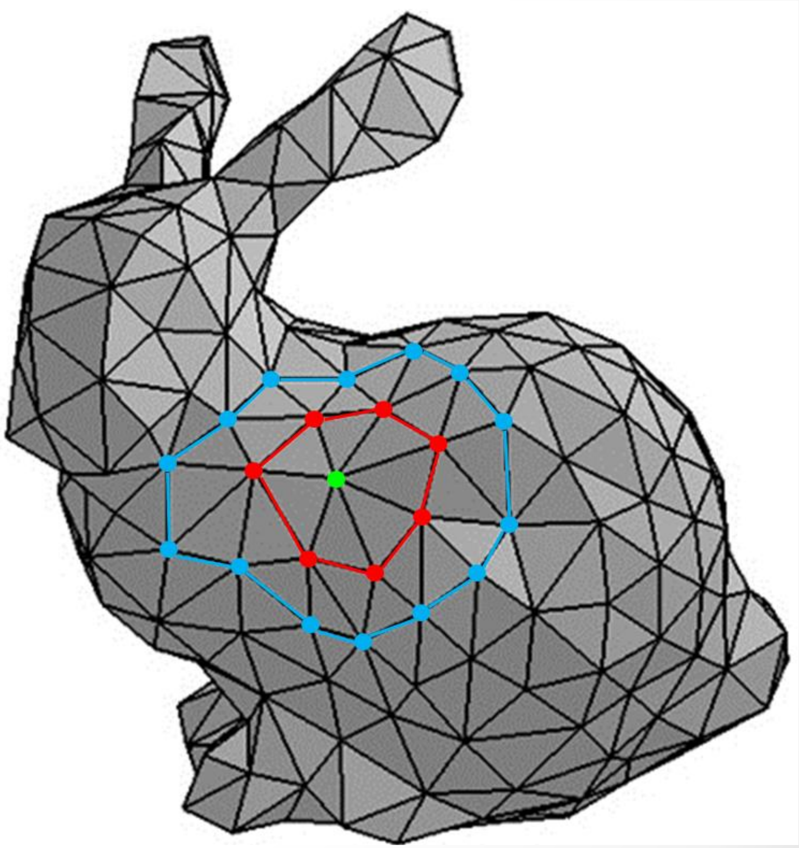
n-ring Neighborhood

- *n*-ring neighborhood of a vertex (recursive definition on manifold mesh)
  - 0-ring neighborhood only contains the center vertex, no face
  - *n*-ring neighborhood contains (*n-1*)-ring neighborhood and its incident faces



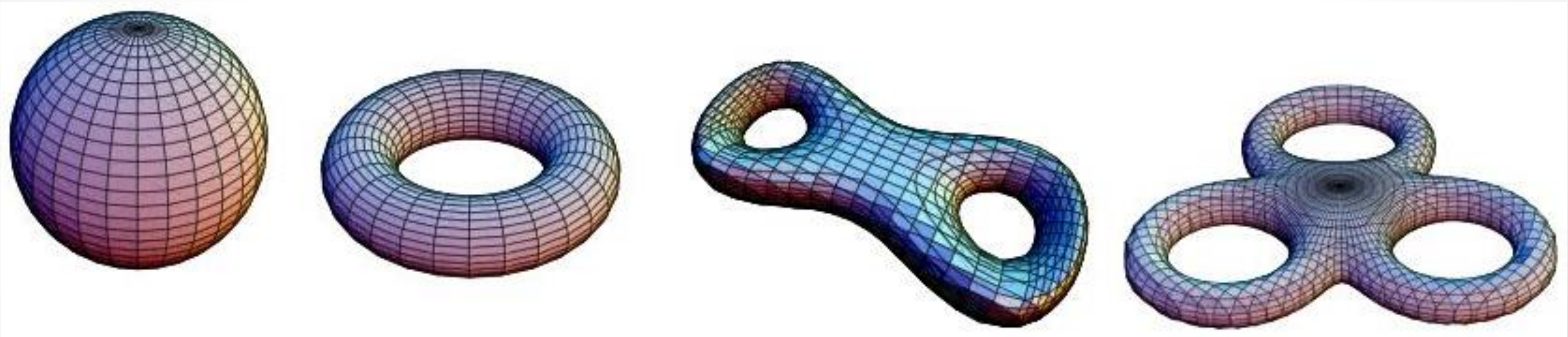
n-ring Neighborhood

- Q: How to compute the normal of a mesh vertex?



Global Topology: Genus

Genus: (informally) the number of holes or handles.



Genus 0 Genus 1 Genus 2 Genus 3

Euler Formula

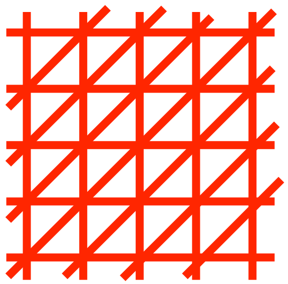
- For a closed polygonal mesh of genus g, the relation of the number V of vertices, E of edges, and F of faces is given by Euler's formula

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

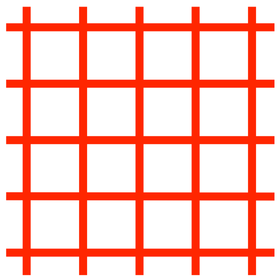
- The term 2(1-g) is called the Euler characteristic

Euler Consequences

- Triangle meshes
  - $F \approx 2V$
  - $E \approx 3V$
  - Average valence = 6



- Quad meshes
  - $F \approx V$
  - $E \approx 2V$
  - Average valence = 4



Overview

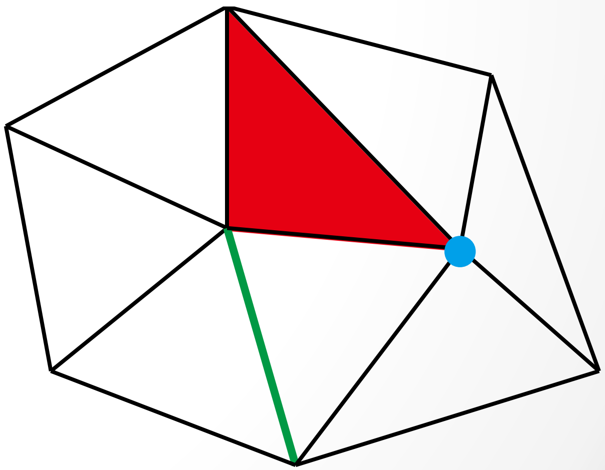
- 3D Mesh Representation
- **Mesh Data Structures**

Mesh Data Structures

- What should be stored?
  - Geometry: 3D coordinates
  - Attributes
    - e.g., normal, color, texture coordinate
  - Connectivity
    - What is adjacent to what

Mesh Data Structures

- What should it support?
  - Rendering
  - Queries (constant time access to neighbours)
    - given a vertex, which faces/edges share it
    - given an edge, which two triangles share it
    - given a triangle, what are the three adjacent triangles
  - Modifications
    - Remove/add a vertex/face
    - Vertex split, edge collapse



Mesh Data Structures

- How good is a data structure?
  - Time to construct (preprocessing)
  - Time to answer a query
  - Time to perform an operation
  - Space complexity
  - Redundancy

Face Set (STL)

- Face:
  - 3 positions

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

no connectivity!

redundancy

Shared Vertex (OBJ, OFF)

- Indexed Face List
  - Vertex: position
  - Face: vertex indices

Vertices
x_1 y_1 z_1
...
x_V y_V z_V

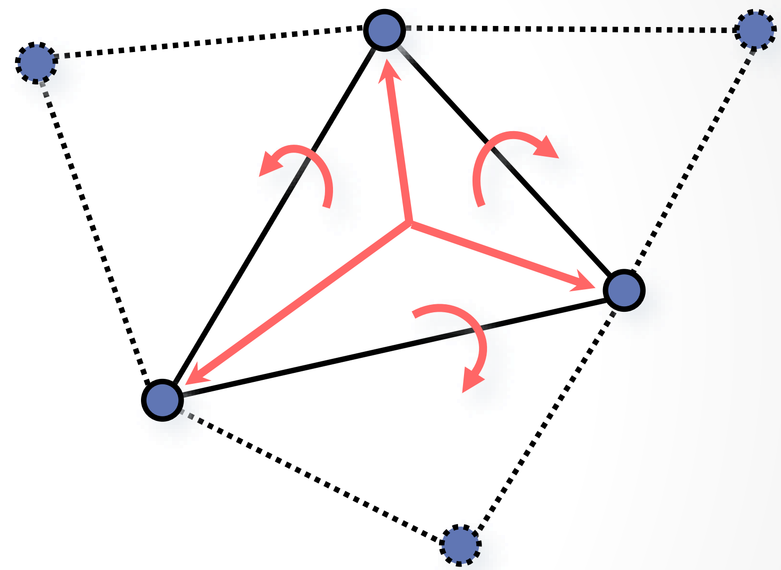
Triangles
i_{11} i_{12} i_{13}
...
...
...
...
i_{F1} i_{F2} i_{F3}

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

no neighborhood info

Face-Based Connectivity

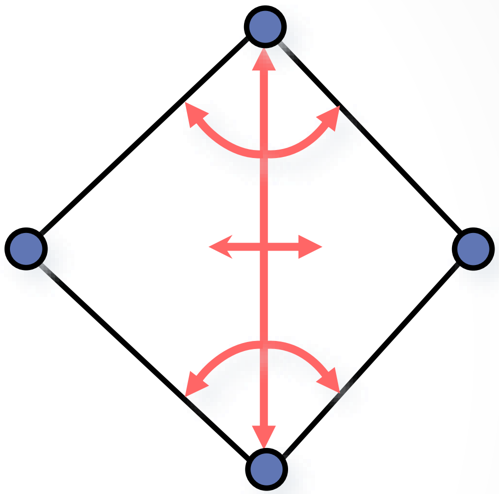
- Vertex:
  - position
  - 1 face
- Face:
  - 3 vertices
  - 3 face neighbors



64 B/v
no edges!

Edge-Based Connectivity

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

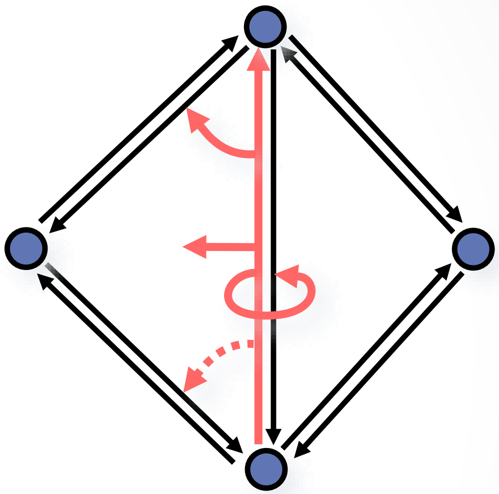


120 B/v

edge orientation?

Halfedge-Based Connectivity

- Vertex
  - position
  - 1 (outgoing) halfedge
- Halfedge
  - 1 vertex
  - 1 face
  - 3 halfedges (next/previous/opposite)
- Face
  - 1 halfedge

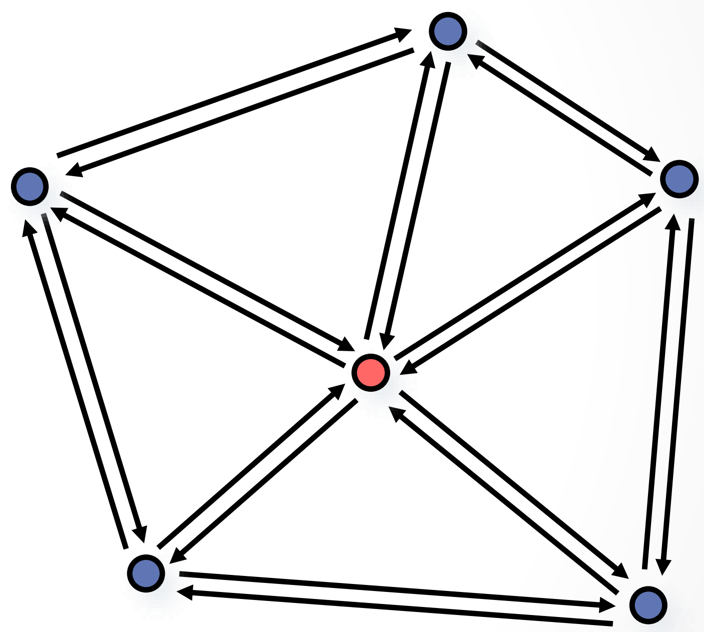


144 B/v

no case distinctions during traversal

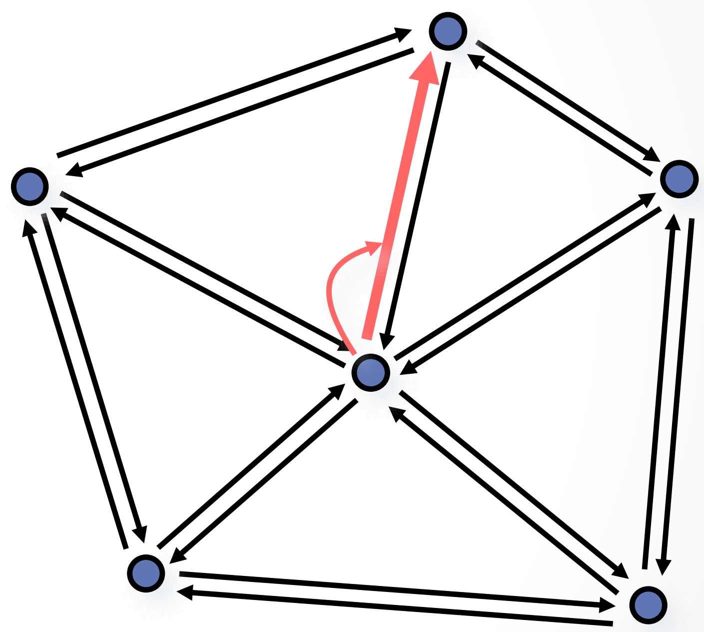
One-Ring Traversal

1. Start at vertex



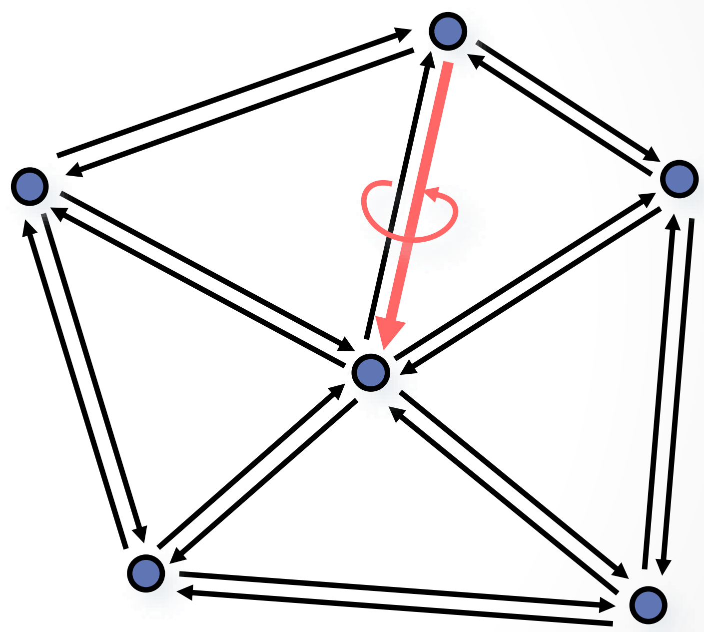
One-Ring Traversal

1. Start at vertex
2. Outgoing halfedge



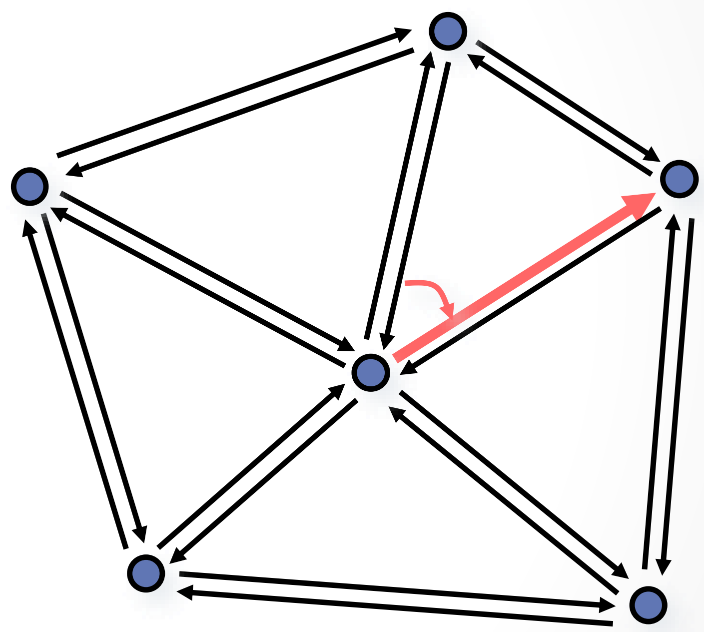
One-Ring Traversal

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



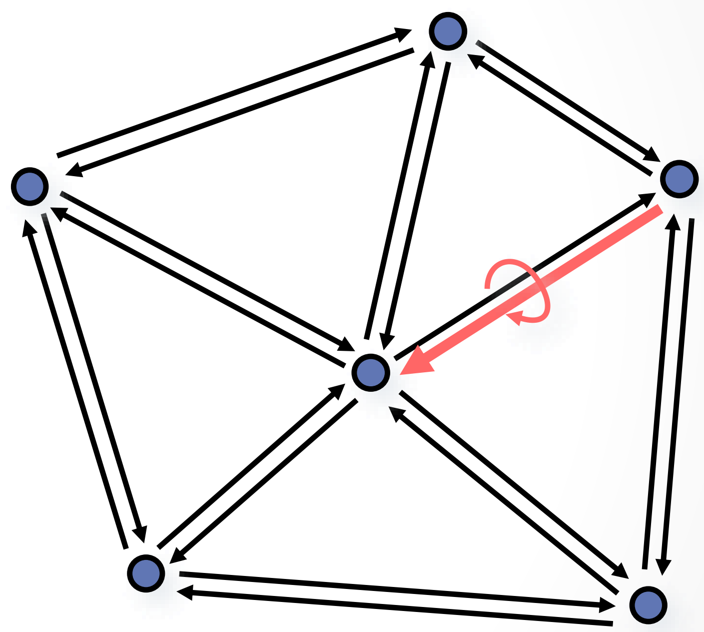
One-Ring Traversal

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



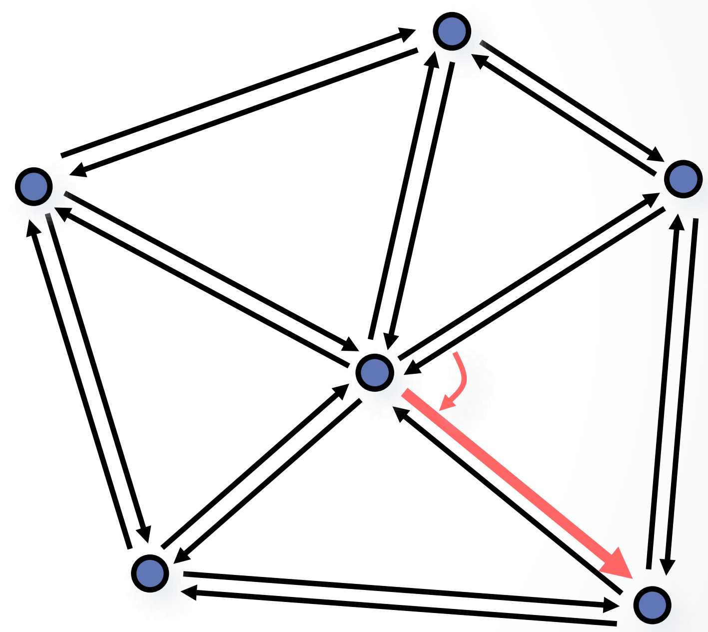
One-Ring Traversal

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
  - Free for non-commercial use

- OpenMesh
  - www.openmesh.org
  - Mesh processing
  - Free, LGPL licence