FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Computeranimation

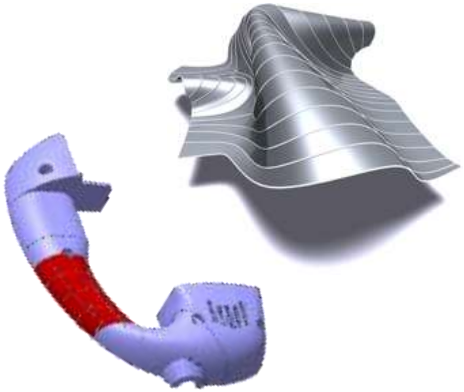
Appendix A –Meshes

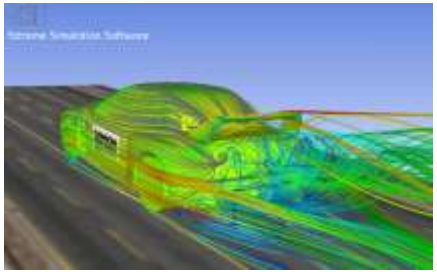
LGDV
COMPUTER GRAPHICS • ERLANGEN

Surface Representations

Why do we need surfaces at all

- Design (CAD)
- Rendering
- Simulation
- Documentation Preservation
- Quality Control
- Custom Fitting



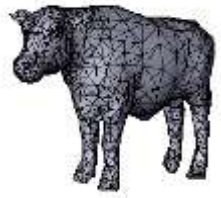


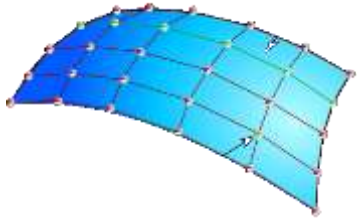
LGDV
COMPUTER GRAPHICS • ERLANGEN

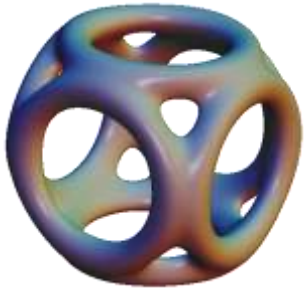
Surface Representations

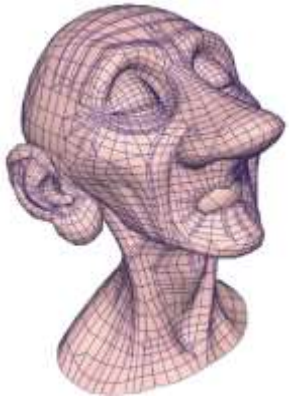
Description of Objects‘ Surfaces

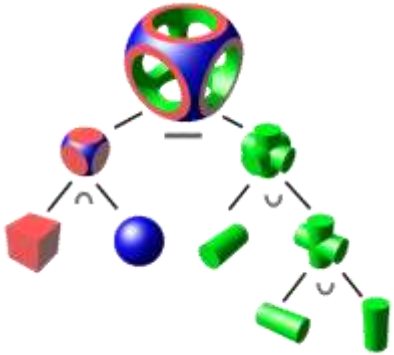
- Polygonal Meshes
- Parametric Surfaces
- Implicit Surfaces
- Constructive Solid Geometry
- Subdivision Surfaces









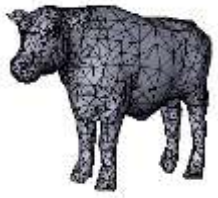


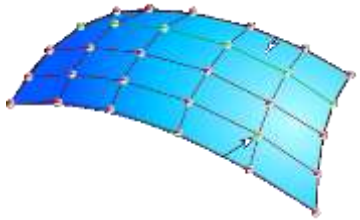
LGDV
COMPUTER GRAPHICS • ERLANGEN

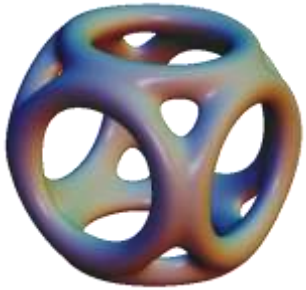
Surface Representations

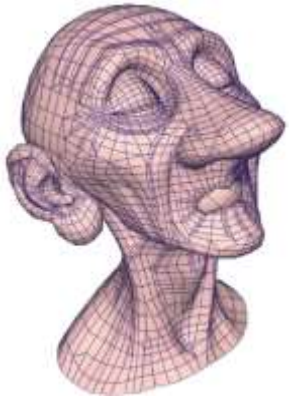
Description of Objects‘ Surfaces

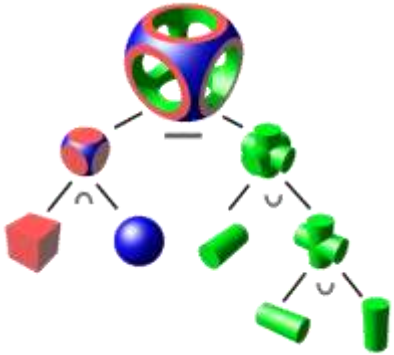
- Polygonal Meshes ***← in this course***
- Parametric Surfaces
- Implicit Surfaces
- Constructive Solid Geometry
- Subdivision Surfaces









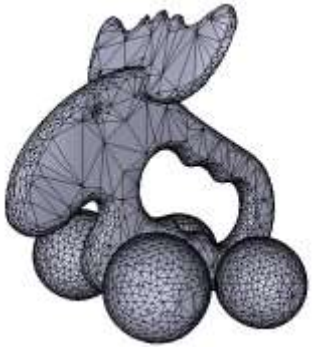


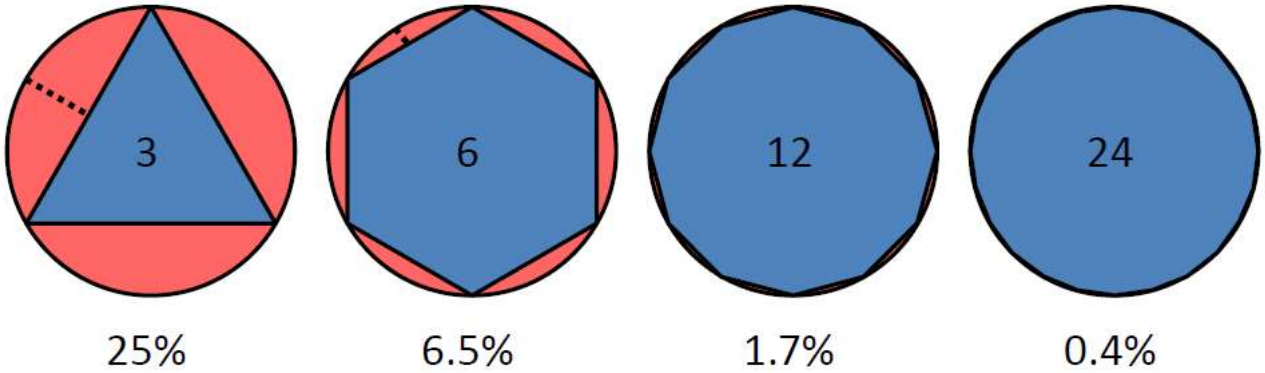
LGDV
COMPUTER GRAPHICS • ERLANGEN

Polygonal Meshes

Piecewise Linear Approximation

- Error $O(h^2)$



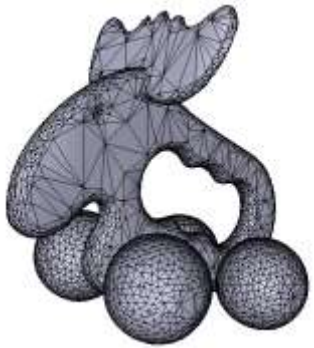


LGDV
COMPUTER GRAPHICS • ERLANGEN

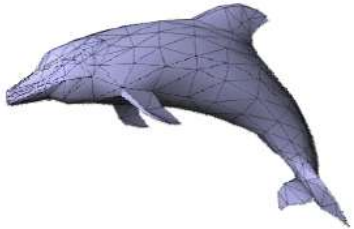
Polygonal Meshes

Polygonal Meshes are Good!

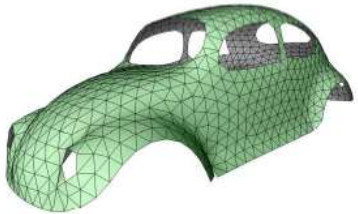
- Approximation $O(h^2)$
- Arbitrary Topology
- Piecewise smooth Surfaces
- Adaptive Refinement
- Efficient Rendering



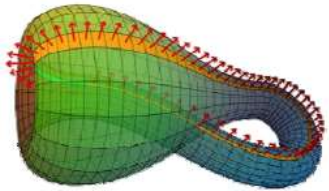
Mesh Zoo



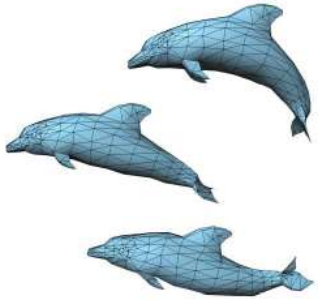
Single component closed, triangular, orientable manifold



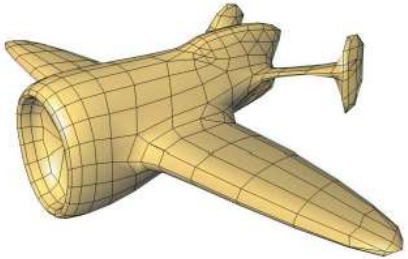
With boundaries



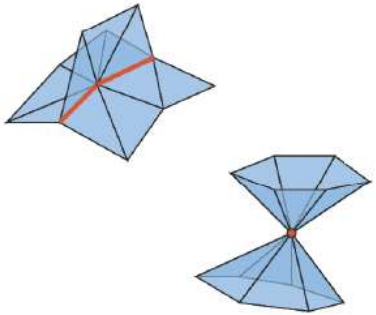
Non-orientable



Many components



Triangles and Quads



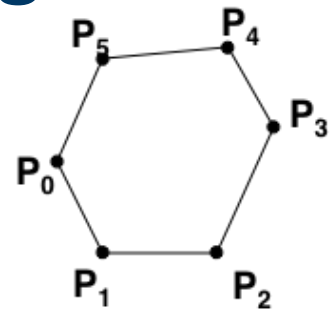
Non-manifold

LGDV COMPUTER GRAPHICS • ERLANGEN

Polygonal Meshes

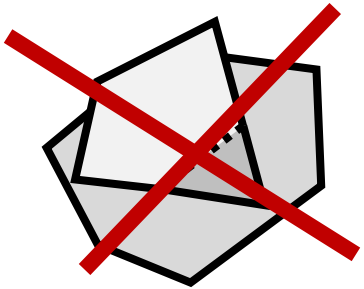
Polygon

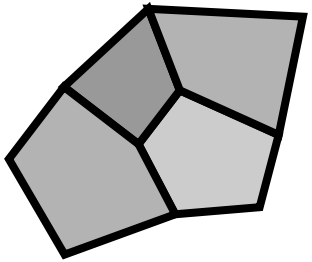
- Ordered set of vertices



Polygon mesh

- Collection of Polygons satisfying certain restrictions





Polygonal Meshes

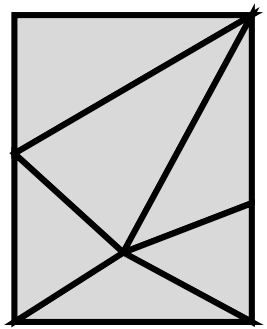
Most Often Used Polygonal Meshes in Computer Graphics

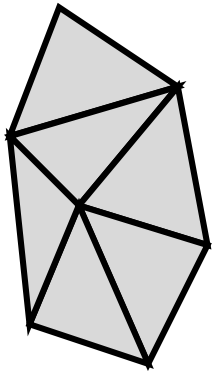
- Triangle meshes
- Quad meshes

Topology vs. Geometry

- **Geometry**: The shape of an object: Basically the vertex position
- **Topology**: connectivity / neighbourhood relation

Polygonal Meshes





Topologically Equivalent

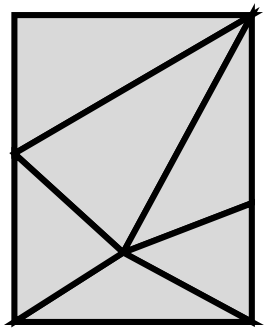
- Mesh only differs in vertex Positions

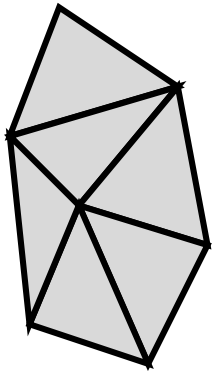
Geometrically (almost) equivalent

- Same shape, different connectivity of mesh-elements

LGDV
COMPUTER GRAPHICS • ERLANGEN

Polygonal Meshes



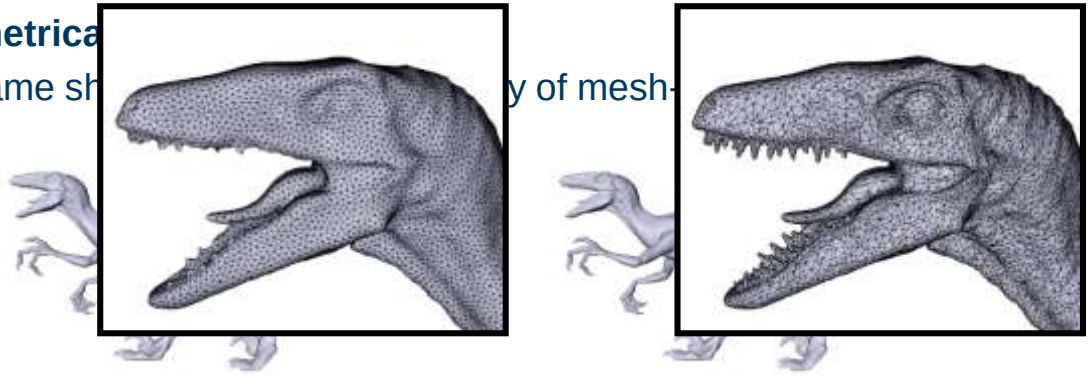


Topologically Equivalent

- Mesh only differs in vertex Positions

Geometrica

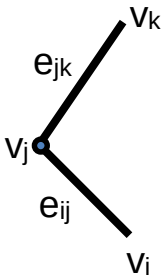
- Same sh y of mesh-

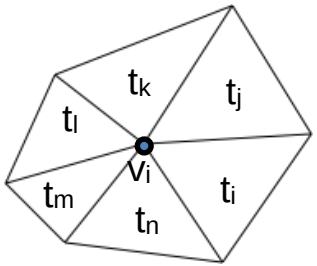


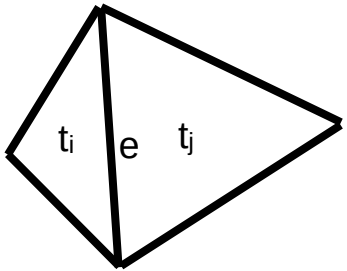
Triangle Meshes

Definition: A triangulation of a set of points consists of:

- Vertices $v_i, \quad \{v_i, i = 1, \ldots, N\} \subseteq \Re^3$
- Edges $\subseteq \{\overline{v_i v_j}, 1 \le i, j \le N\}$
- Triangles $\subseteq \{\Delta(v_i v_j v_k), 1 \le i, j, k \le N\}$
- The triangulation can also have attributes (normals, texture-coordinates, etc…)
- Edges sharing a vertex called adjacent
- Triangles sharing a vertex or an edge are called adjacent



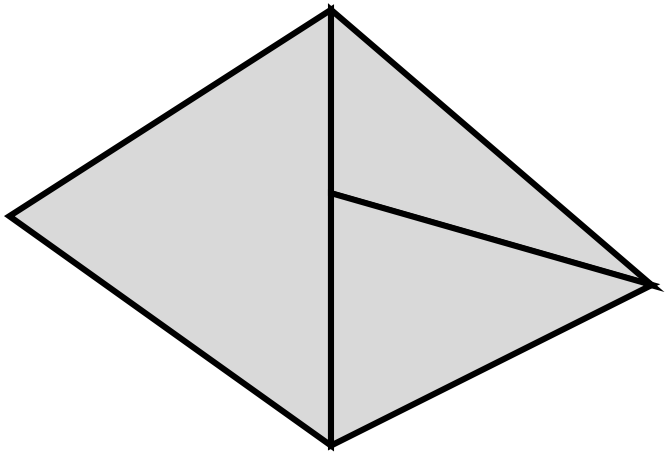




LGDV
COMPUTER GRAPHICS • ERLANGEN

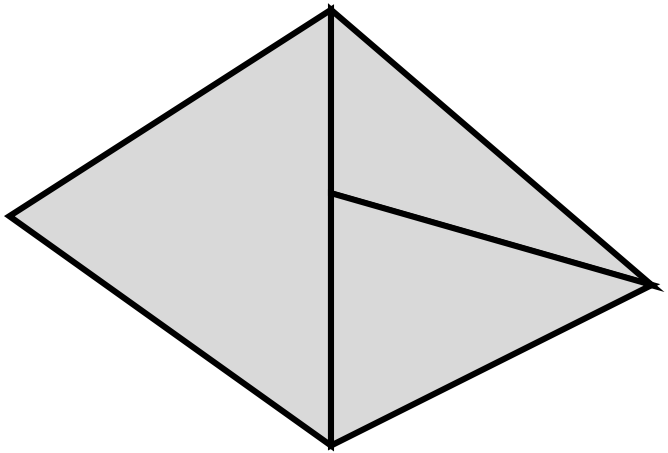
Triangle Meshes

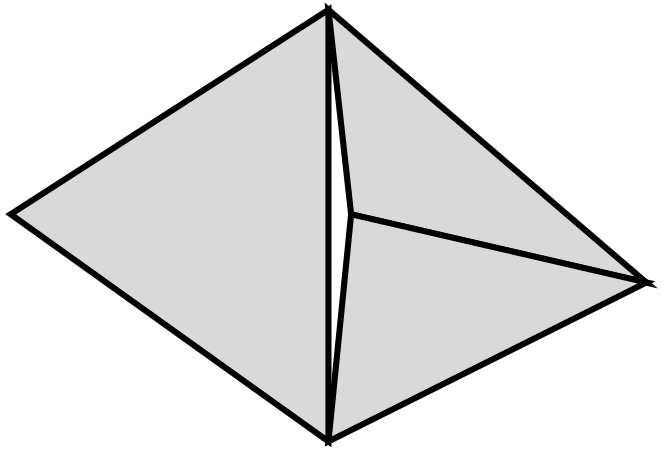
Is this a valid Triangle Mesh?



Triangle Meshes

Is this a valid Triangle Mesh?





Mesh Data Structures

How to represent Triangulations

- Considerations: Rendering, Memory usage, complexity of queries
- There is no perfect data structure just a trade off
- Triangle list
- Shared Vertex Structure
- There are many more…

Mesh Data Structures

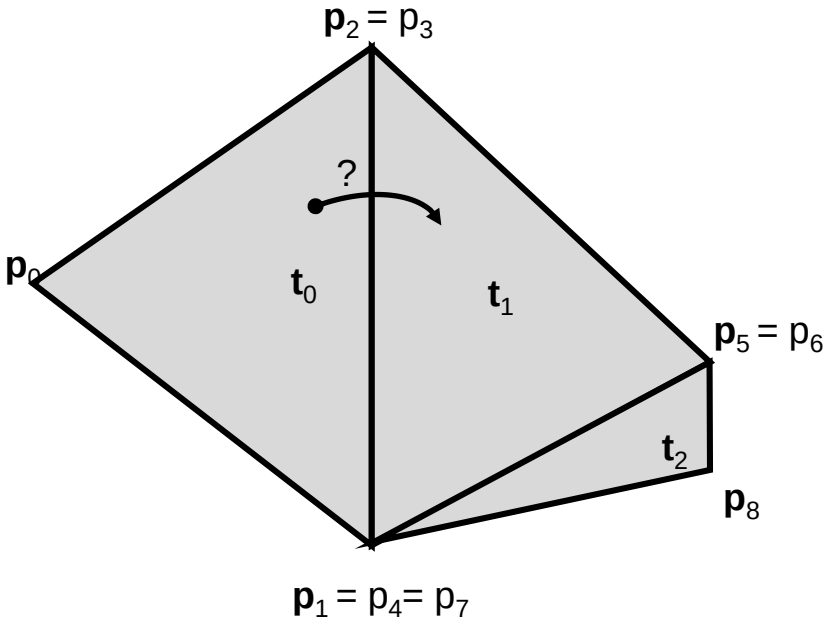
Triangle List

Triangle list

p_0: x_0, y_0, z_0;
p_1: x_1, y_1, z_1; } triangle 0
p_2: x_2, y_2, z_2;

p_3: x_3, y_3, z_3;
p_4: x_4, y_4, z_4; } triangle 1
p_5: x_5, y_5, z_5;

p_6: x_6, y_6, z_6;
p_7: x_7, y_7, z_7;
…



Mesh Data Structures

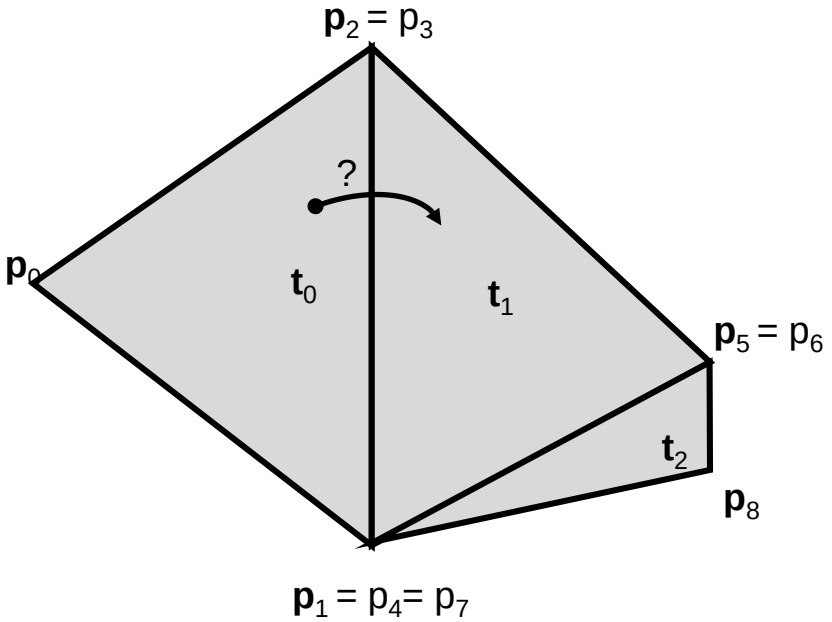
Triangle List

Pros:

- Trivial
- Easy Triangle Removal
- Easy to render

Cons:

- No neighbourhood info (hard to traverse)
- Duplicate Points (waste of storage)



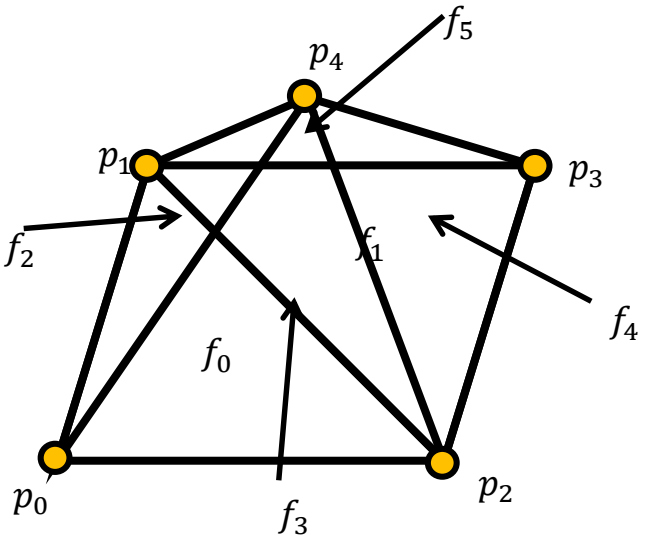
Mesh Data Structures

Shared Vertex Data Structure

- Also called „indexed face set“
- Consists of two lists

vertex list
p_0: x_0, y_0, z_0;
p_1: x_1, y_1, z_1;
p_2: x_2, y_2, z_2;
p_3: x_3, y_3, z_3;
p_4: x_4, y_4, z_4;

face list
f_0: 0, 1, 2;
f_1: 2, 1, 3;
f_2: 1, 0, 4;
f_3: 4, 0, 2;
f_4: 4, 2, 3;
f_5: 4, 3, 1;

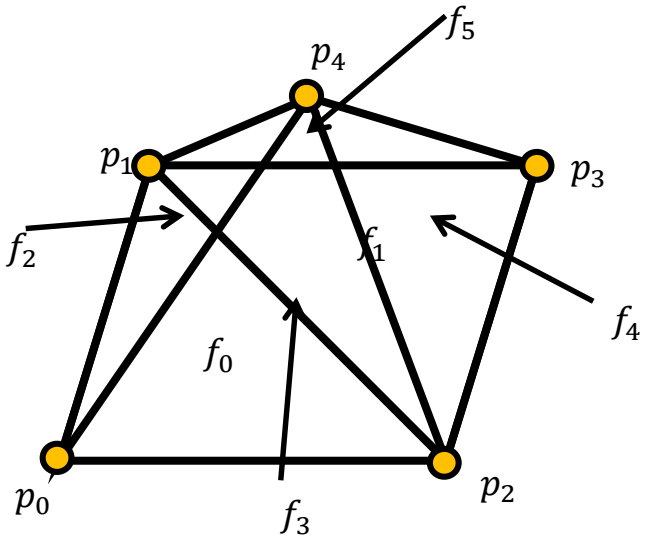


LGDV COMPUTER GRAPHICS • ERLANGEN

Mesh Data Structures

Shared Vertex Data Structure

- Remark: geometric queries are difficult
- Extend face list by reference to neighbour faces



vertex list
p_0: x_0, y_0, z_0;
p_1: x_1, y_1, z_1;
p_2: x_2, y_2, z_2;
p_3: x_3, y_3, z_3;
p_4: x_4, y_4, z_4;

Extended face list
f_0: 0, 1, 2; 1, 3, 2
f_1: 2, 1, 3; 5, 4, 3
f_2: 1, 0, 4; …
f_3: 4, 0, 2; …
f_4: 4, 2, 3; …
f_5: 4, 3, 1; …

LGDV COMPUTER GRAPHICS • ERLANGEN

Mesh Data Structures

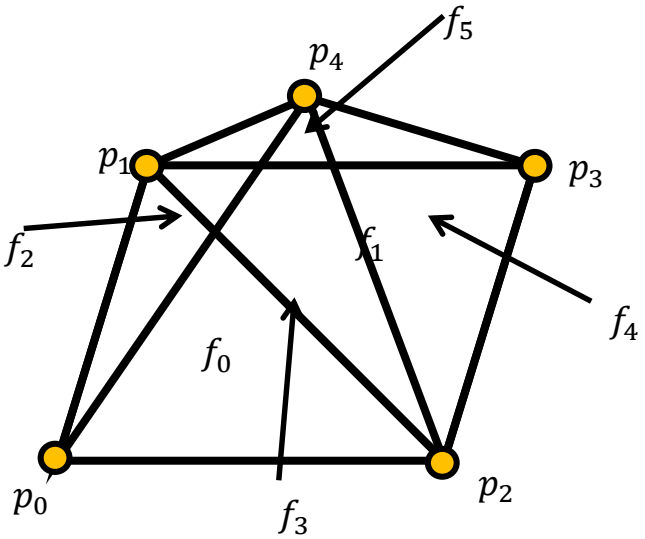
Shared Vertex Data Structure

Pros:

- Easy
- Gold standard for mesh file formats (obj, off)
- Neighbourhood queries cheap (extended version)
- Low storage cost (each vertex stored once)
- Works for Tri/Quad/Poly/Mixed meshes

Cons:

- Neighbourhood queries expensive (non-extended version)
- High Storage cost (extended version)



LGDV
COMPUTER GRAPHICS • ERLANGEN