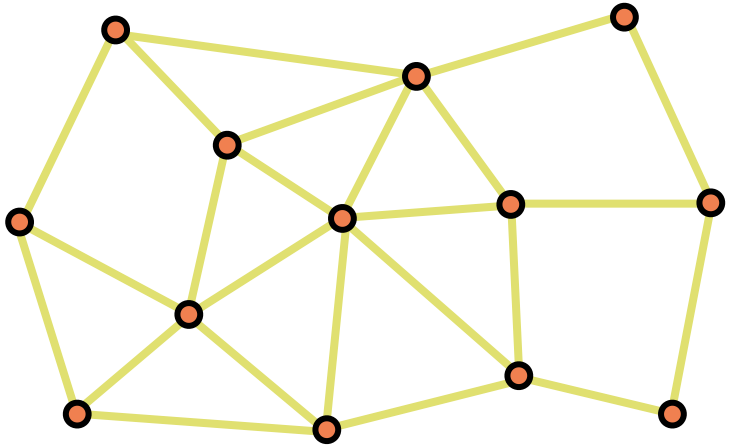
Vorlesung
Computergrafik

Media Informatics
Graphics & Geometric Computing
Prof. Dr. M. Campen

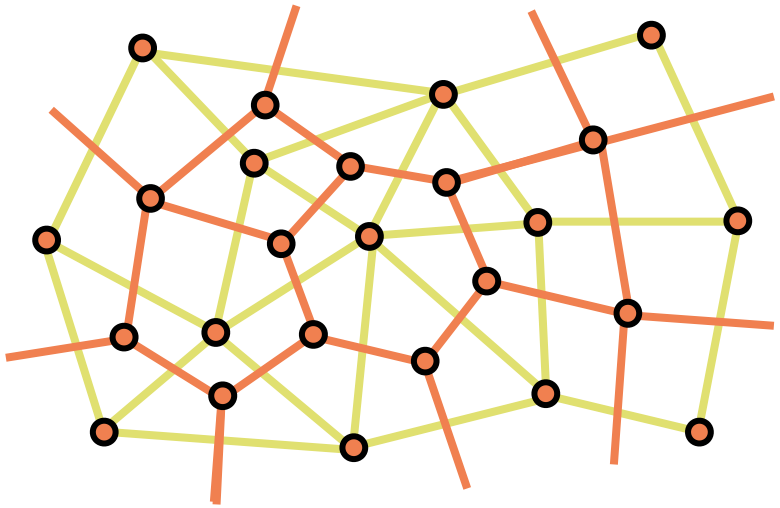
Polygon-Netze

- Bestehend aus
  - "Geometrie": Vertices mit ihren Punktkoordinaten
  - "Topologie" oder "Konnektivität": Edges (Kanten) und Faces (Polygone, Dreiecke) zwischen den Vertices



Polygon-Netze

- *Duales Netz:*
  - für jedes *primale* Face einen Vertex,
  - für jede primale Edge eine "gedrehte" Edge,
  - für jeden *primalen* Vertex ein Face.



  - Geometrie (Vertexpositionen) des dualen Netzes nicht allgemein definiert; fallspezifisch. Zur exemplarische Darstellung z.B. einfach im Face-Zentrum platzieren.

Polygon-Netze

- *1-Ring* eines Vertex v:
    - Zykel von Vertices, die durch eine Edge mit v verbunden (zu v "benachbart") sind.
- *Geschlossen ("closed")*
    - Keine Edge hat nur ein (oder gar kein) angrenzendes ("inzidentes") Face
    - d.h. keine Randkanten ("boundary edges")
- *Mannigfaltig ("manifold"):*
    - Jede Edge hat genau zwei inzidente Faces
    - Jeder Vertex hat genau einen 1-Ring
- *Valenz:*
    - eines Vertex: Anzahl inzidenter Edges
    - eines Faces: Anzahl inzidenter Edges (bei Dreiecken also 3)

Polygon-Netze

- *Genus:*
    - Anzahl der Schnitte (entlang geschlossener Kurven) die man maximal durchführen kann, bevor das Netz in mehrere unverbundene Teile zerfällt.

Polygon-Netze

- *Genus:*
  - Anzahl der Schnitte (entlang geschlossener Kurven) die man maximal durchführen kann, bevor das Netz in mehrere unverbundene Teile zerfällt.
  - z.B.: Kugel: 0; Torus: 1; Brezel: 3; …
  - unabhängig von Geometrie

Polygon-Netze

- *Genus:*
  - Anzahl der Schnitte (entlang geschlossener Kurven) die man maximal durchführen kann, bevor das Netz in mehrere unverbundene Teile zerfällt.
  - z.B.: Kugel: 0; Torus: 1; Brezel: 3; …
  - unabhängig von Geometrie

Polygon-Netze

- *Genus:*
  - Anzahl der Schnitte (entlang geschlossener Kurven) die man maximal durchführen kann, bevor das Netz in mehrere unverbundene Teile zerfällt.
  - z.B.: Kugel: 0; Torus: 1; Brezel: 3; …
  - unabhängig von Geometrie

- *Euler-Formel:*
  - V-E+F = 2(1-g)
    - V, E, F = Anzahl Vertices, Edges, Faces
    - g = Genus
    - Gilt für jedes mannigfaltige Polygonnetz!

Dreiecks-Netze

- *Modifikationsoperatoren:*
    - Face Split
    - Edge Split
    - Vertex Split
    - Edge Flip
    - Edge Collapse

    - erhalten alle die Mannigfaltigkeit eines Netzes
    - respektieren alle (wie zu erwarten) die Euler-Formel

    - Operatoren, die hingegen z.B. einen neuen Tunnel einbauen verändern den Genus und damit die Euler-Charakteristik.

Dreiecks-Netze

- Eigenschaften von Dreiecksnetzen:
  - Sei H = 2E die Anzahl der "Halbkanten"
  - Dann: 3F = H, also 3F = 2E (nur weil alle Faces Dreiecke sind!)
  - Daher 4(1-g) = 2V-2E+2F = 2V-3F+2F = 2V-F
    - also: F = 2V - c (wobei c eine Konstante, abhängig vom Genus ist)
    - also: ein Dreiecksnetz enthält etwa doppelt so viele Faces wie Vertices
  - Und 6(1-g) = 3V-3E+3F = 3V-3E+2E = 3V-E
    - also E = 3V - c (wobei c eine Konstante, abhängig vom Genus ist)
    - also: ein Dreiecksnetz enthält etwa dreimal so viele Edges wie Vertices
    - da jede Edge zu zwei Vertices inzident ist, gibt es im Schnitt etwa 6 Edge-Inzidenzen pro Vertex.
      - die durschnittliche Vertex-Valenz in einem Dreiecksnetz ist ca. 6.

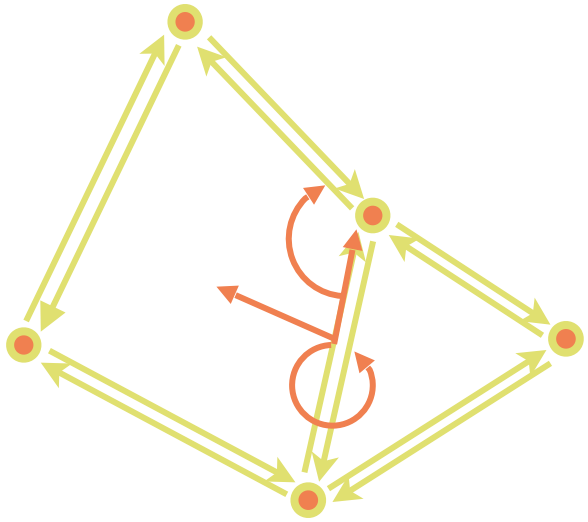
Datenstrukturen/Dateiformate für Polygonnetze

- Face List (.stl)
  - [(x,y,z), (x,y,z), (x,y,z)]
    [(x,y,z), (x,y,z), (x,y,z)]
    [(x,y,z), (x,y,z), (x,y,z), (x,y,z)]
    [(x,y,z), (x,y,z), (x,y,z)]
    …
- Indexed/Shared Vertex (.off, .obj)
  - 0: (x,y,z)
    1: (x,y,z)
    2: (x,y,z)
    …
    [0,2,1]
    [2,3,0]
    …

Vertices kommen (mit ihren Koordinaten) mehrfach vor, wenn Sie zu mehr als einem Face inzident sind.

Datenstrukturen/Dateiformate für Polygonnetze

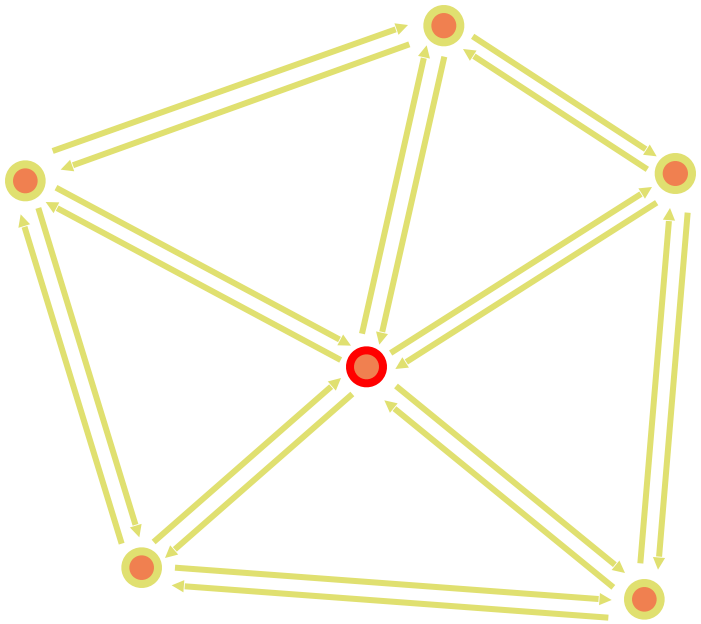
- Halfedge Mesh:
  - Jede Edge wird als zwei gegensätzlich gerichtete *Halfedges* angesehen.
    - innerhalb jedes Faces: gegen den Uhrzeigersinn!
  - Jede Halfedge kennt:
    - ihren Nachfolger im Face ("next")
    - ihre entgegengesetzte Halfedge ("opposite")
    - ihr inzidentes Face ("face")
    - ihren vorderen inzidenten Vertex ("to")
  - Jedes Face kennt eine inzidente Halfedge ("halfedge")
  - Jeder Vertex kennt eine inzidente (ausgehende) Halfedge ("out")
  - Shortcuts:
    - "from(h)" = "to(opposite(h))"
    - "prev(h)" = "next(next(h))" (im Dreiecksnetz)



Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

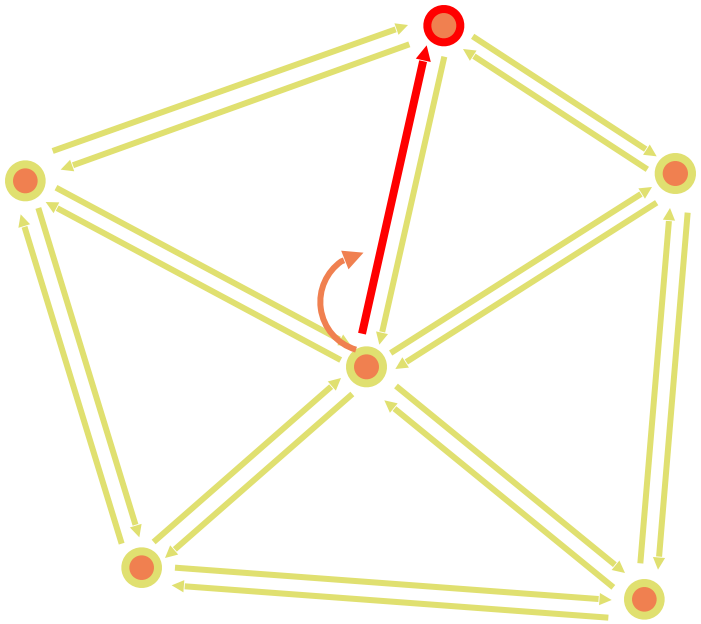
1. Starte bei v



Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

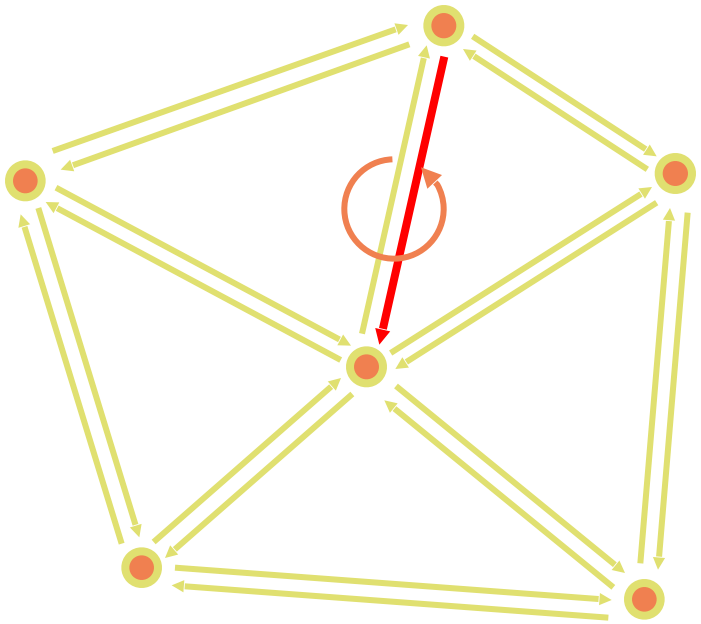
1. Starte bei v
2. out
3. to



Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

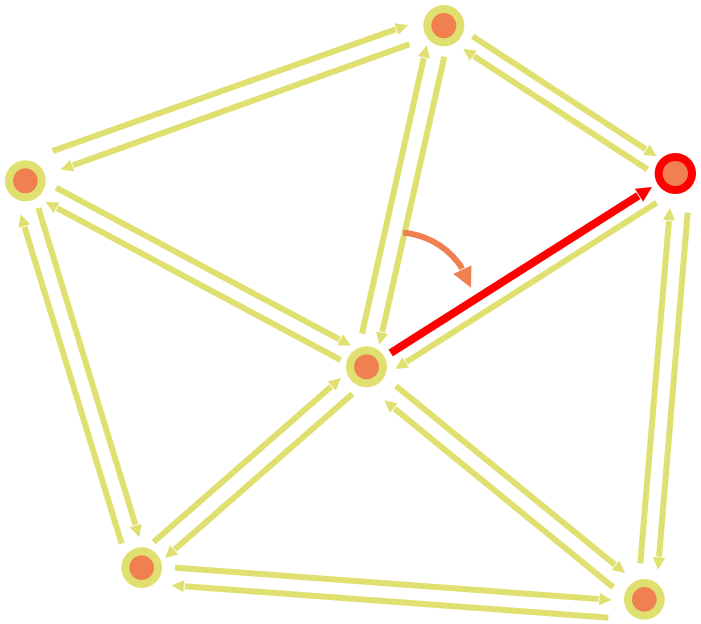
1. Starte bei v
2. out
3. to
4. opposite



Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

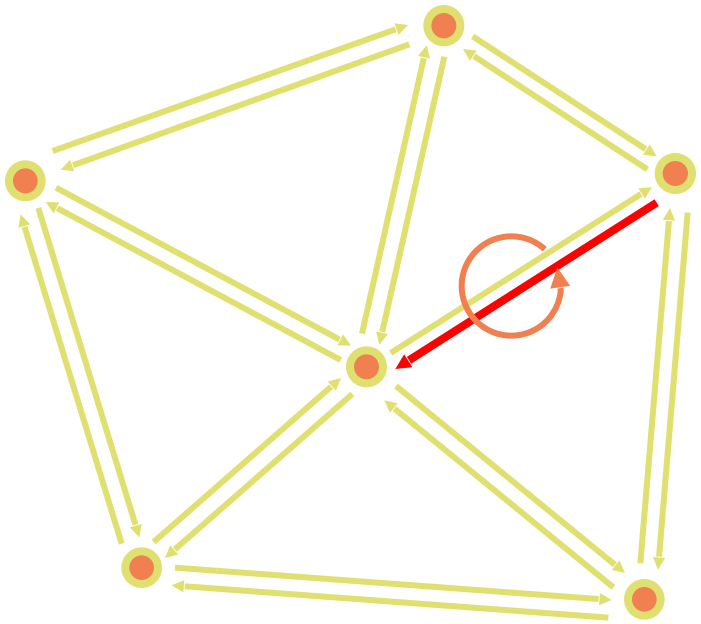
1. Starte bei v
2. out
3. to
4. opposite
5. next
6. to



Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

1. Starte bei v
2. out
3. to
4. opposite
5. next
6. to
7. opposite

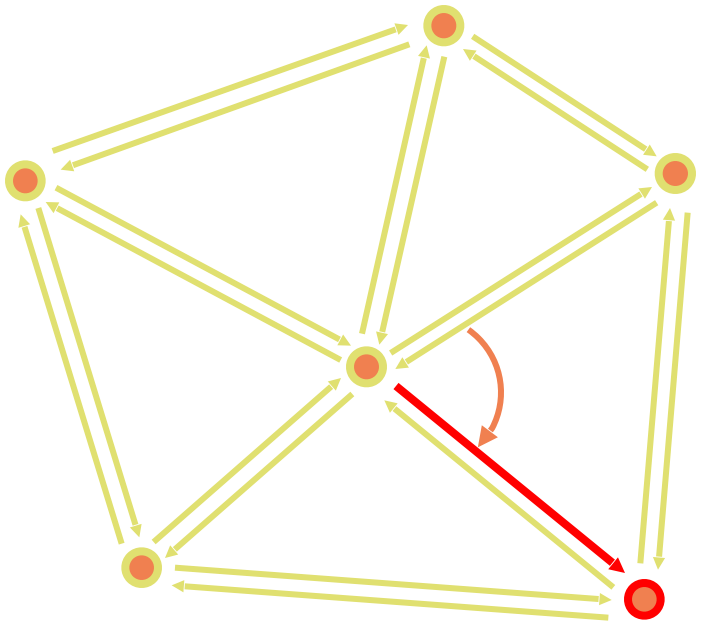


Datenstrukturen/Dateiformate für Polygonnetze

- Halfedge Mesh:
  - Ermöglicht effiziente Navigation auf dem Netz
    - Beispiel: 1-Ring ablaufen:

1. Starte bei v
2. out
3. to
4. opposite
5. next
6. to
7. opposite
8. next
9. to

   …



Halfedge Mesh

- Repräsentationsformen:
  - *objektorientiert:*
    - Vertices, Faces und Halfedges sind Objekte
    - die Operatoren (next, opposite, face, to, out, halfedge) sind als Zeiger/ Referenzen dieser Objekte realisiert, die auf die entsprechenden anderen Objekte zeigen.
      - z.B. ist `h.face` dann das Objekt des Faces, das an die Halfedge, welche durch Objekt h repräsentiert wird, angrenzt.
    - Koordinaten (u. a. Eigenschaften): als Objekt-Attribute (Member-Variablen)
  - *array-basiert* (in der Regel effizienter)
    - Vertices, Faces und Halfedges werden jeweils einfach durch Integer (0,1,2,…) identifiziert (genannt Index oder ID)
    - die Operatoren sind als Arrays realisiert.
      - z.B. ist `face[7]` dann der Integer-Index des Faces, das an die Halfedge mit dem Index 7 angrenzt.
    - Koordinaten (u. a. Eigenschaften): als zusätzliche Arrays (z.B. `coord[47]`)

Halfedge Mesh

- Ränder
  - In Halfedge-Datenstrukturen sind Randkanten daran ersichtlich, dass die außenliegende Halfedge kein angrenzendes Face hat:
    - in der objektorientierten Variante: `h.face = undefined/NULL`
    - in der array-basierten Variante: `face[h] = -1`
  - Alternativ gibt es die Möglichkeit, die außenliegenden Halfedges selbst wegzulassen. Dann ist z.B. `opposite[h] = -1`, wobei `h` die innenliegende Halfedge ist.
    - Diese Variante wird praktisch selten verwendet, da die Navigation über das Netz bei solch "fehlenden" Randhalfedges etwas umständlicher ist.

Bis zum nächsten Mal!

Media Informatics
Graphics & Geometric Computing
Prof. Dr. M. Campen