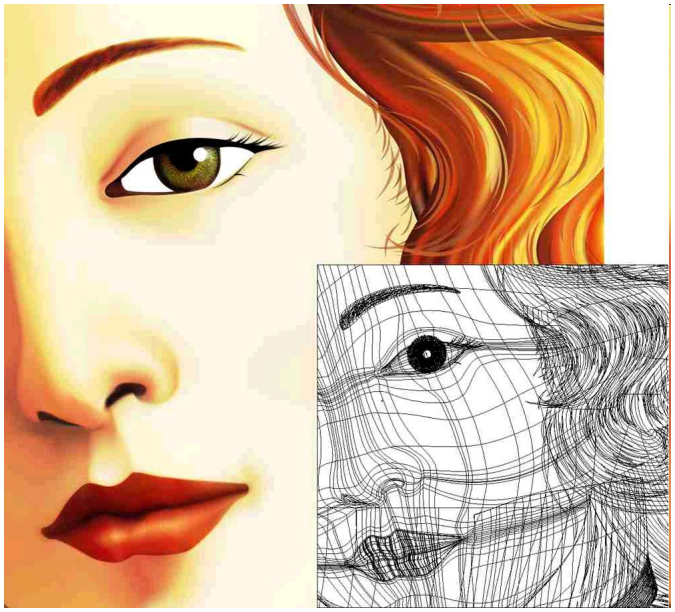
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN
LEHR- UND FORSCHUNGSEINHEIT MEDIENINFORMATIK

Prof. Dr. Andreas Butz

Dipl.-Medieninf. Hendrik Richter
Dipl.-Medieninf. Raphael Wimmer

Computergrafik 1 Übung 6

Meshes



http://codeidol.com/img/illustrator-cs/54084xfg1001_0.jpg

Primitive

2D-Objekte

- Beispiel:

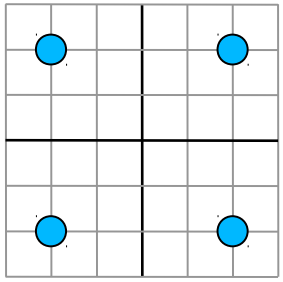
```
glBegin(GL_POLYGON);
    glColor3f(1.0f, 0.0f, 0.0f); // red
    glVertex3f(-1.0f, -1.0f, 0.0f);
    glVertex3f(1.0f, -1.0f, 0.0f);
    glColor3f(0.0f, 0.0f, 1.0f); // blue
    glVertex3f(1.0f, 1.0f, 0.0f);
    glVertex3f(-1.0f, 1.0f, 0.0f);
glEnd();
```

- Weitere Formen:

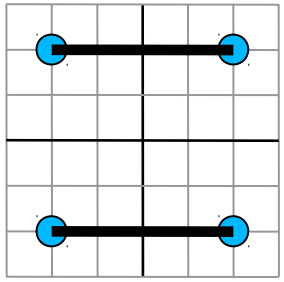
```
GL_POINTS, GL_LINES (je 2 Punkte verbunden)
GL_LINE_STRIP, GL_LINE_LOOP
GL_QUADS, GL_POLYGON, GL_TRIANGLES,
GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUAD_STRIP
```

- 3D-Objekte müssen aus 2D-Objekten zusammengesetzt werden

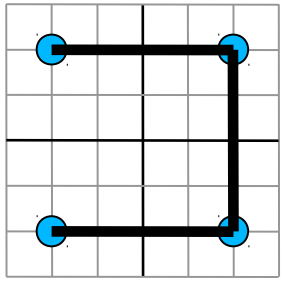
2D-Objekte



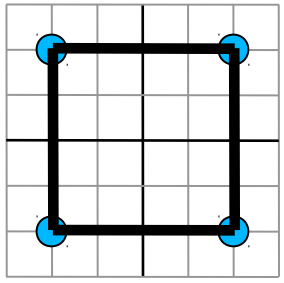
GL_POINTS



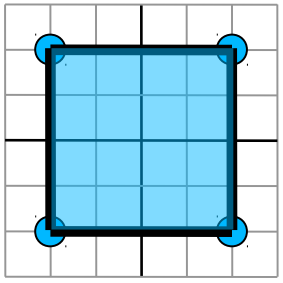
GL_LINES



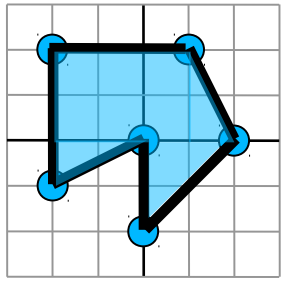
GL_LINE_STRIP



GL_LINE_LOOP



GL_QUADS

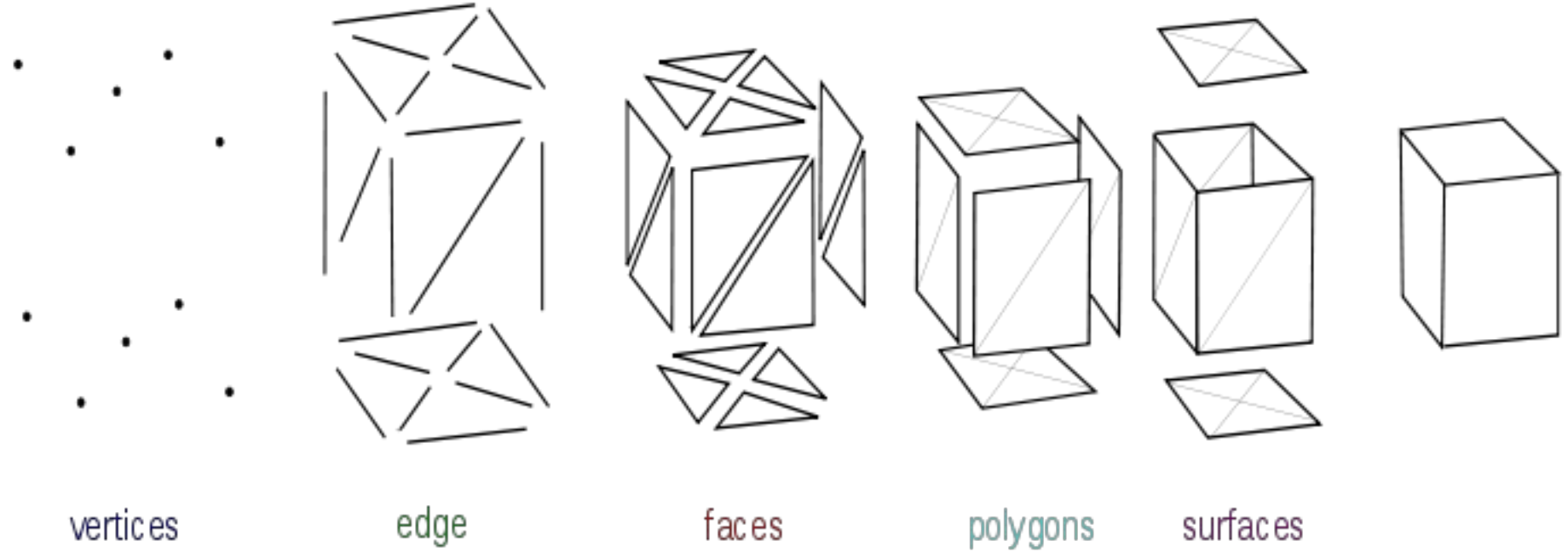


GL_POLYGON

Polygon Meshes

Meshes - Grundlagen

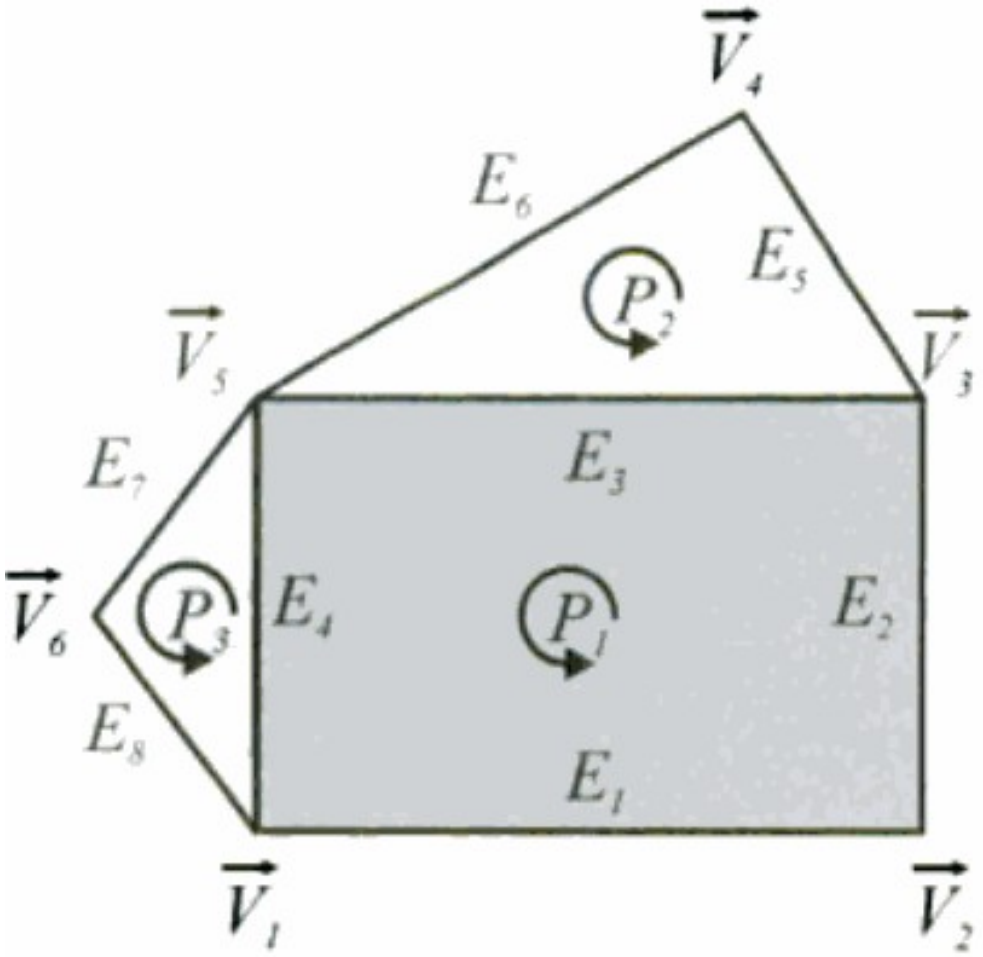
http://upload.wikimedia.org/wikipedia/commons/thumb/6/6d/Mesh_overview.svg/720px-Mes



- Untereinander mit Kanten verbundene Punkte = Polygonnetz
- jeder Knoten ist von jedem anderen im Netz aus erreichbar
- Polygonnetze sind ungerichtete Graphen ohne Mehrfachkanten
- Annäherung an gebogene Oberflächen durch höhere Polygonanzahl

Meshes - Beispiel

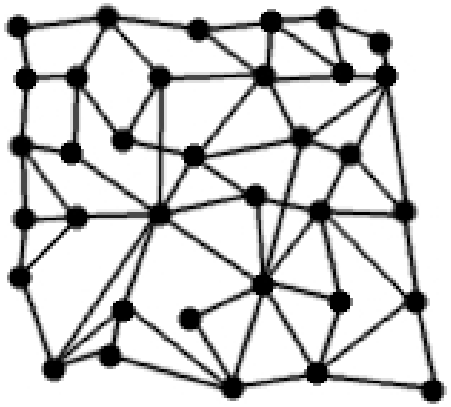
Computergrafik und OpenGL: Eine systematische Einführung
By Dieter Orlamünder, Wilfried Mascolus



$\vec{V}_i$: Knoten oder Vertices
E_i: Kanten oder Edges
P_i: Polygone
Erzeugungsrichtung

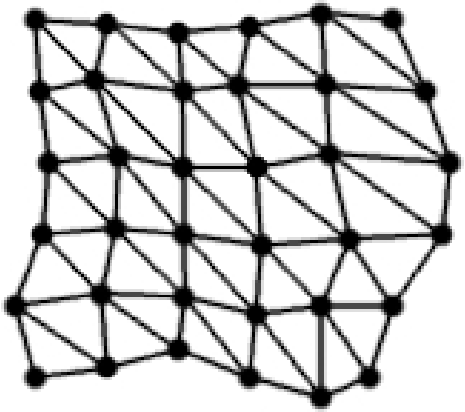
Meshes - Eigenschaften

http://upload.wikimedia.org/wikipedia/de/e/ec/Netzeigenschaften.jpg -



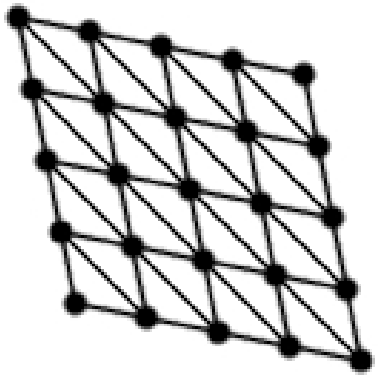
- strukturiert

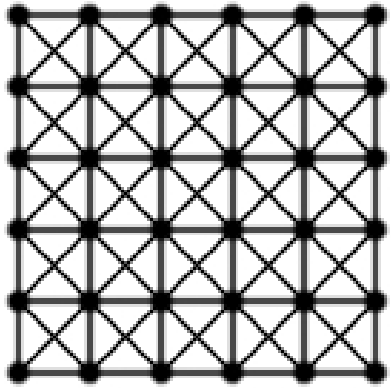
Jeder innere Punkt hat die gleiche Anzahl anliegender Kanten und Flächen



- strukturiert
- regulär

Die Kantenlänge ist in jede Richtung konstant.



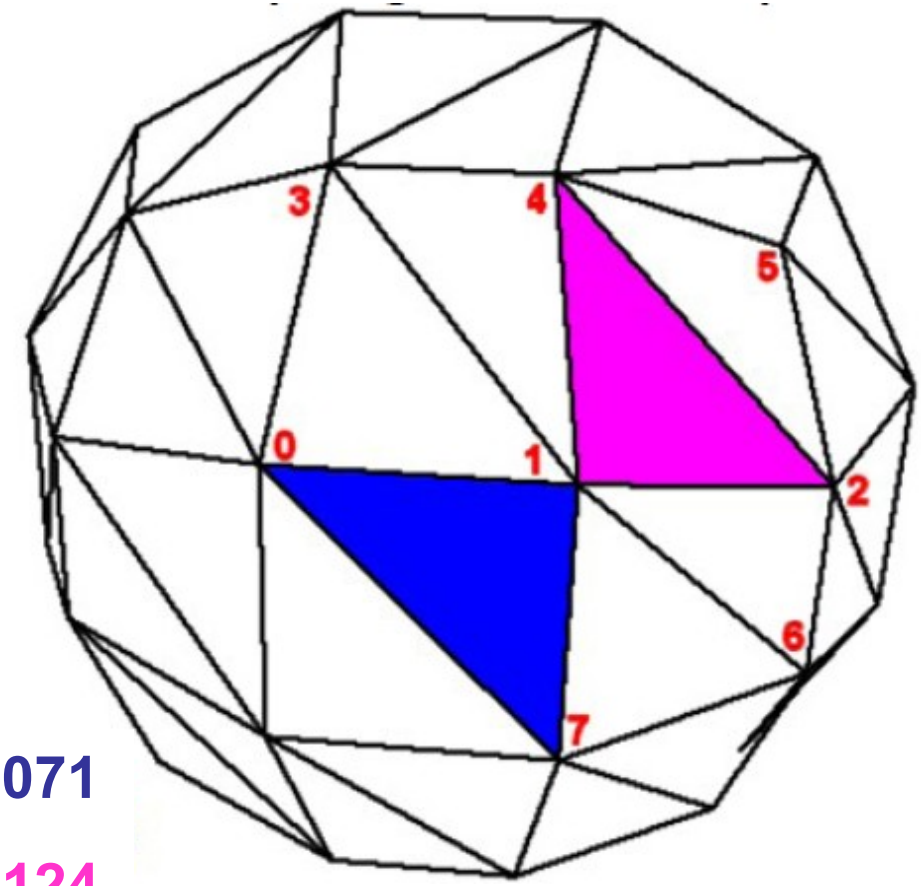


- strukturiert
- regulär
- orthogonal

Alle Netzkanten bilden rechte Winkel.

Meshes - Umlaufsinn

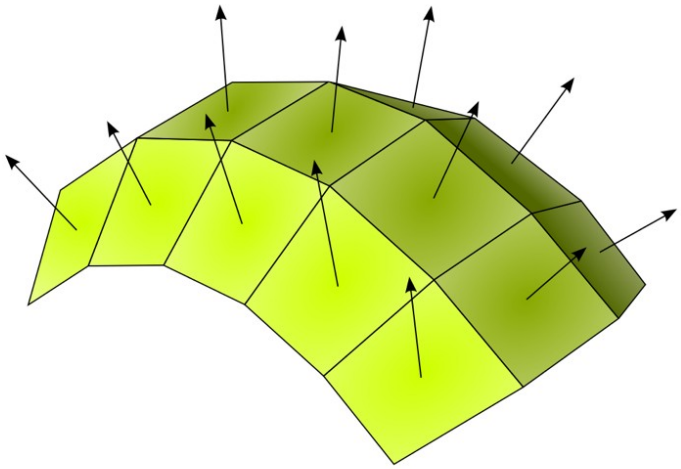
http://www.zusi.de/zusi3/Kapitel_5_1.pdf



Normalen

- Die Normale einer Ebene bzw. eines Vektorpaars ist derjenige Vektor der darauf senkrecht steht (und ungleich 0 ist)
- Bestimmung entweder über lineares Gleichungssystem oder Kreuzprodukt:

$$\vec{u} \times \vec{v} = \begin{pmatrix} u_2 \cdot v_3 - u_3 \cdot v_2 \\ u_3 \cdot v_1 - u_1 \cdot v_3 \\ u_1 \cdot v_2 - u_2 \cdot v_1 \end{pmatrix}$$



(Quelle: http://de.wikipedia.org/wiki/Normale)

Datenstrukturen

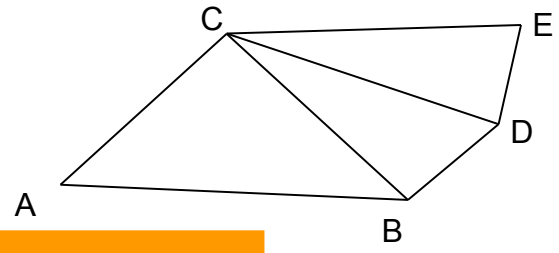
Meshes – Explizite Speicherung

```
glBegin(GL_TRIANGLES);
glColor3f(1.0f,0.0f,0.0f);          // Set The Color To Red
glVertex3f(-1.0f, -0.5f, -4.0f); // lower left vertex
glVertex3f( 1.0f, -0.5f, -4.0f); // lower right vertex glVertex3f( 0.0f,
0.5f, -4.0f); // upper vertex

glColor3f(0.0f,1.0f,0.0f); // Set The Color To Green
glVertex3f(-0.5f, -1.5f, -4.0f);
glVertex3f( 0.5f, -1.5f, -4.0f);
glVertex3f( 0.0f, -1.0f, -4.0f);
glEnd();
```

- Jedes Polygon wird durch eine Liste seiner Eckpunktkoordinaten repräsentiert
- Zwischen jedem Paar von Ecken ist eine Kante, auch zwischen letzter und erster Ecke
- Speicheraufwändige Darstellung
- Koordinaten von Ecken werden mehrfach aufgeführt
- Keine Speicherung gemeinsamer Ecken und Kanten

Meshes – Explizite Speicherung




```
glBegin(GL_TRIANGLE_STRIP);
        //Basisdreieck:
        glVertex3f(-1.0f, -0.5f, -4.0f); // A
        glVertex3f( 1.0f, -0.5f, -4.0f); // B
        glVertex3f( 0.0f, 0.5f, -4.0f); // C

        glVertex3f(1.5f, 0.0f, -4.0f); // D
        glVertex3f(2.0f, -1.5f, -4.0f); // E
glEnd();
```

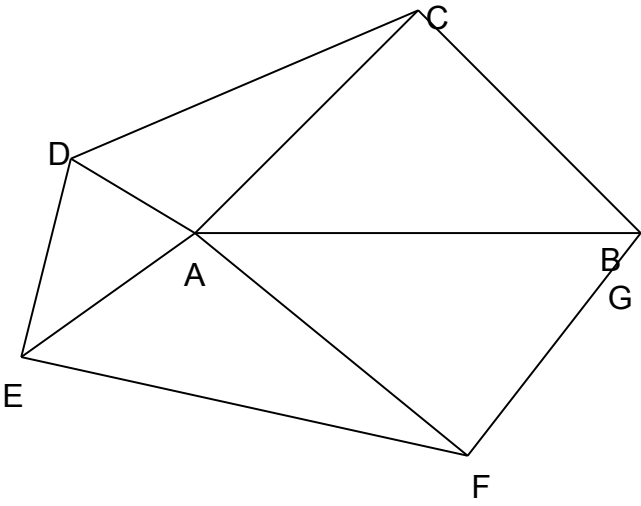
- GL_LINE_STRIP | GL_TRIANGLE_STRIP | GL_QUAD_STRIP
- Definition einer Grundform
- Hinzufügen von Eckpunkten durch einzelne Vertices
- Keine Mehrfachdefinition von Knoten mehr

Meshes – Explizite Speicherung

```
glBegin(GL_TRIANGLE_FAN);
        glVertex3f(-1.0f, -0.5f, -4.0f); // A
        glVertex3f( 1.0f, -0.5f, -4.0f); // B
        glVertex3f( 0.0f, 0.5f, -4.0f); // C

        glVertex3f(-1.5f, 0.0f, -4.0f); // D
        glVertex3f(-1.8f, -1.0f, -4.0f); // E
        glVertex3f( 0.2f, -1.5f, -4.0f); // F

        glVertex3f( 1.0f, -0.5f, -4.0f); // G
glEnd();
```



- Definition einer Mitte
- Hinzufügen von Eckpunkten durch einzelne Vertices
- Keine Mehrfachdefinition von Knoten

Meshes - Knotenliste

Verfahren in OpenGL:

- Knotenlisten anlegen
- Arrays von Knotenindizes zeichnen

```
GLfloat vertices[][3] =
{{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0},...};

glVertexPointer(3, GL_FLOAT, 0, vertices);

GLubyte cubeIndices[24] = {0,3,2,1,...};

glDrawElements(GL_QUADS, 24, GL_UNSIGNED_BYTE, cubeIndices);
```

http://homepages.uni-paderborn.de/fschopp/hauptstudium/docs/computergrafikI.pdf

Meshes – Speichermethoden

Polygon meshes may be represented in a variety of ways, using different methods to store the vertex, edge and face data. These include:

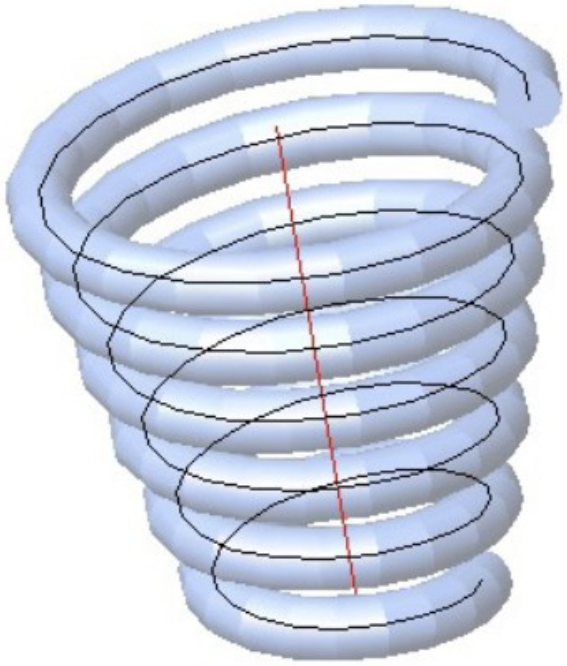
Face-Vertex Meshes – A simple list of vertices, and a set of polygons that point to the vertices it uses.
Winged-Edge Meshes – In which each edge points to two vertices, two faces, and the four (clockwise and counterclockwise) edges that touch it. Winged-Edge meshes allow constant time traversal of the surface, but with higher storage requirements.
Half-Edge Meshes – Similar to Winged-Edge meshes except that only half the edge traversal information is used.
Quad-Edge Meshes – A quad-edge mesh stores edges, half-edges, and vertices without any reference to polygons. The polygons are implicit in the representation, and may be found by traversing the structure. Memory requirements are similar to half-edge meshes.
Corner-Table – A corner-table stores vertices in a predefined table, such that traversing the table implicitly defines polygons. This is in essence the "triangle fan" used in hardware graphics rendering. The representation is more compact, and more efficient to retrieve polygons, but operations to change polygons are slow. Furthermore, Corner-Tables do not represent meshes completely. Multiple corner-tables (triangle fans) are needed to represent most meshes.
Vertex-Vertex Meshes – A *vv* mesh represents only vertices, which point to other vertices. Both the edge and face information is implicit in the representation. However, the simplicity of the representation allows for many efficient operations to be performed on meshes.

http://www.cs.princeton.edu/courses/archive/fall02/cs526/lectures/meshes.pdf
http://en.wikipedia.org/wiki/Polygon_mesh

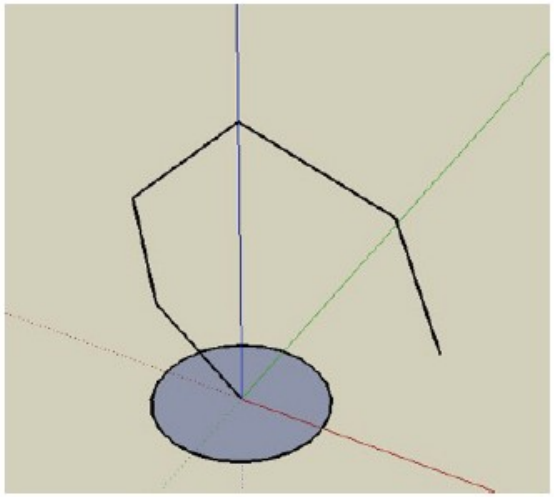
Extrusion & Lofts

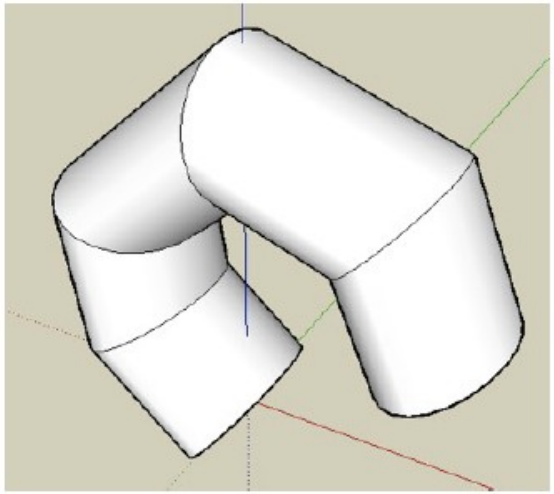
Extrusion (sweep object)

- Move a 2D shape along an arbitrary path
- possibly also scale in each step

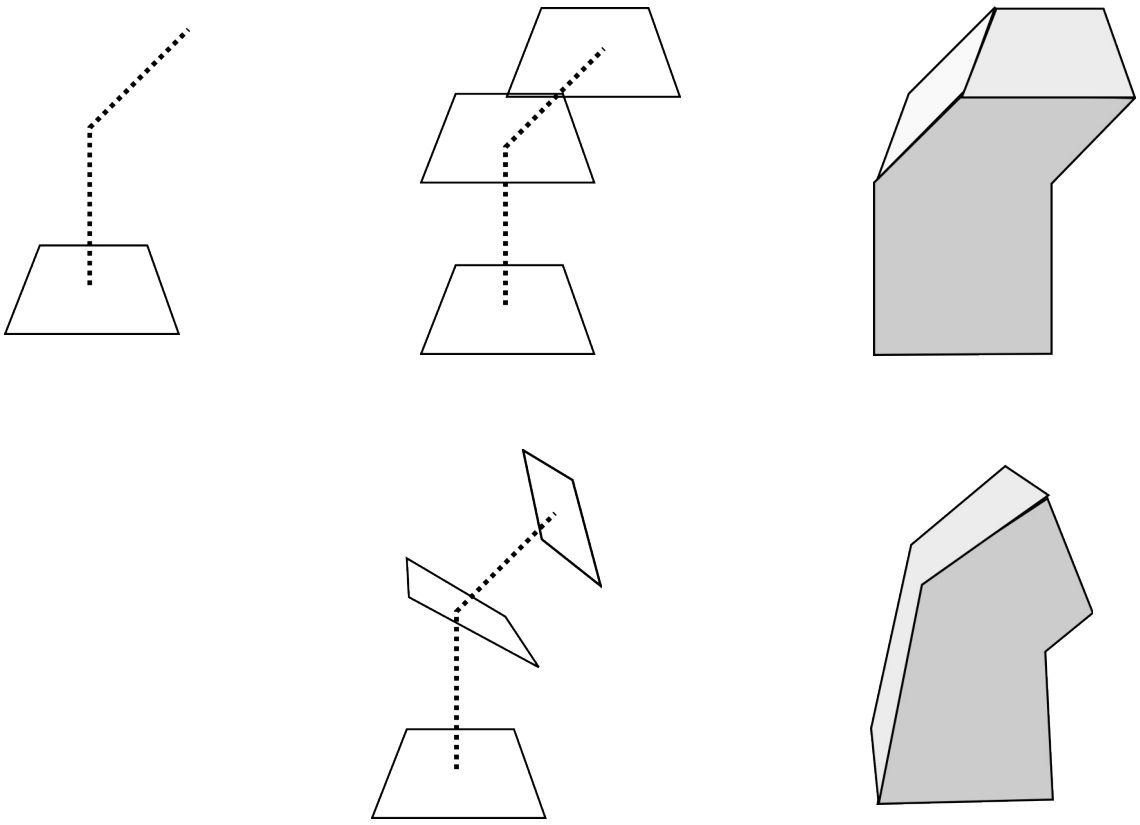


http://www.cadimage.net/cadtutor/lisp/helix-02.gif





Meshes - Extrusion



Meshes - Extrusion

array of original points O[x]
array of new points N[x]
direction to extrude: 'd'. → new points: N[x] = O[x] + d.

```
glBegin(GL_TRIANGLE_STRIP);

        for(int i=0; i<numPoints; i++)
        {
                glVertex3f(O[i].x, O[i].y, O[i].z);
                glVertex3f(N[i].x, N[i].y, N[i].z);
        }

glEnd();
```

Was fehlt?

Literatur

http://mi.informatik.uni-siegen.de/teaching/winter_0809/CG2_0809/script/cg2_04.pdf
http://www.sci.utah.edu/~bavoil/opengl/bavoil_trimeshes_2005.pdf
http://www.falloutsoftware.com/tutorials/gl/gl3.htm

http://www.oreillygmt.eu/images/wireframesubaru2.jpg

Vielen Dank!