# Chapter 4 - 3D Modeling

- **Polygon Meshes**
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

The 3D rendering pipeline (our version for this class)

```
┌─────────────────────────────────────────────┐
│  ┌──────────────────┐      ┌──────────────────┐│      ┌──────────────────┐      ┌──────────────────┐
│  │ 3D models in     │ ───▶ │ 3D models in world││ ───▶ │ 2D Polygons in   │ ───▶ │ Pixels in image  │
│  │ model coordinates│      │ coordinates      ││      │ camera coordinates│      │ coordinates      │
│  └──────────────────┘      └──────────────────┘│      └──────────────────┘      └──────────────────┘
└─────────────────────────────────────────────┘
         ▲                        ▲                        ▲                        ▲
  ┌─────────────┐          ┌─────────────┐          ┌─────────────┐
  │ Scene graph │          │   Camera    │          │Rasterization│
  └─────────────┘          └─────────────┘          └─────────────┘
         ▲        ▲                                        ▲
          ┌──────────────┐                          ┌─────────────┐
          │ Animation,   │                          │   Lights    │
          │ Interaction  │                          └─────────────┘
          └──────────────┘
```
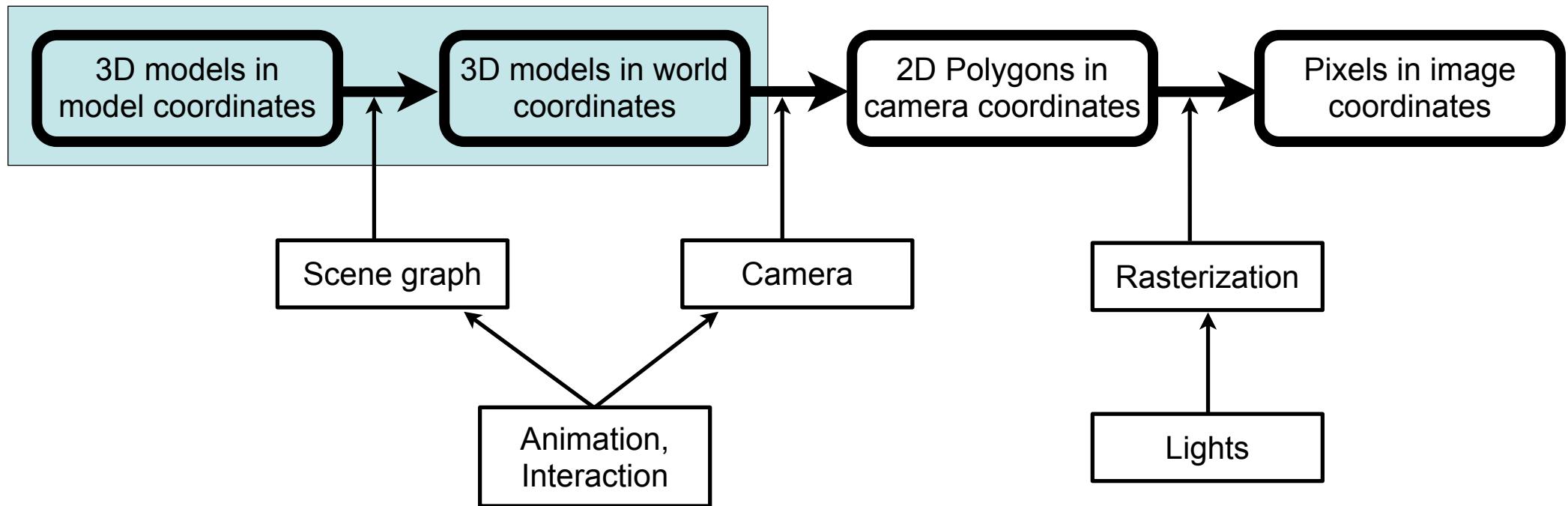
Representations of (Solid) 3D Objects

- Complex 3D objects need to be constructed from a set of primitives
  - Representation schema is a mapping 3D objects --> primitives
  - Primitives should be efficiently supported by graphics hardware
- Desirable properties of representation schemata:
  - Representative power: Can represent many (or all) possible 3D objects
  - Representation is a mapping: Unique representation for any 3D object
  - Representation mapping is injective: Represented 3D object is unique
  - Representation mapping is surjective: Each possible representation value is valid
  - Representation is precise, does not make use of approximations
  - Representation is compact in terms of storage space
  - Representation enables simple algorithms for manipulation and rendering
- Most popular on modern graphics hardware:
  - *Boundary representations* (B-Reps) using *vertices*, *edges* and *faces*.

Polygon Meshes

- Describe the surface of an object as a set of polygons
- Mostly use triangles, since they are trivially convex and flat
- Current graphics hardware is optimized for triangle meshes

vertices   edge   faces   polygons   surfaces
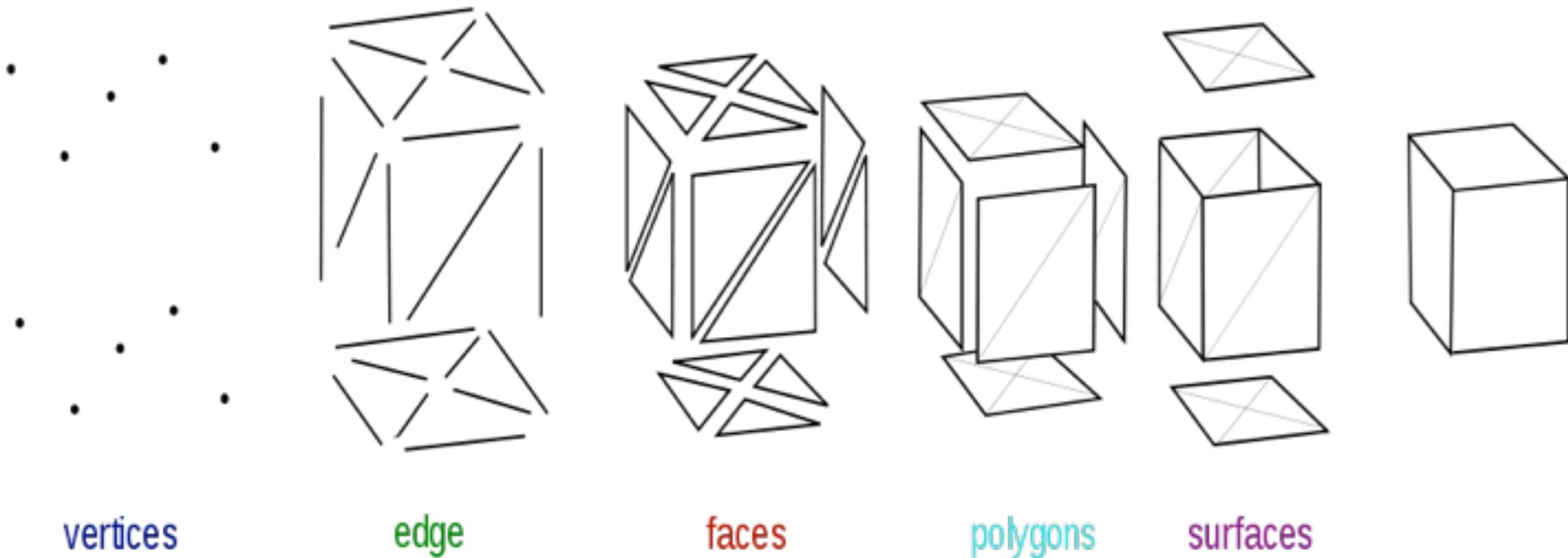
http://en.wikipedia.org/wiki/File:Mesh_overview.svg

3D Polygons and Planes

- A polygon in 3D space should be *flat*, i.e. all vertices in one 2D plane
  - Trivially fulfilled for triangles
- Mathematical descriptions of a 2D plane in 3D space (hyperplane)
  - Method 1: Point *p* and two non-parallel vectors *v* and *w*

    $$x = p + s\vec{v} + t\vec{w}$$

  - Method 2: Three non-collinear points
    (take one point and the difference vectors to the other two)
  - Method 3: Point *p* and normal vector *n* for the plane

    $$\vec{n} \cdot (x - \vec{p}) = 0 \quad \text{using the dot product}$$

  - Method 4: Single plane equation

    $$Ax + By + Cz + D = 0 \quad \text{A, B, C, D real numbers}$$

    (*A*, *B*, *C*) is normal vector of the plane

- All description methods easily convertible from one to the other

    (E.g. using cross product to compute normal vector)

Example: Triangle and Associated Plane

Three points (corners of the unit cube)
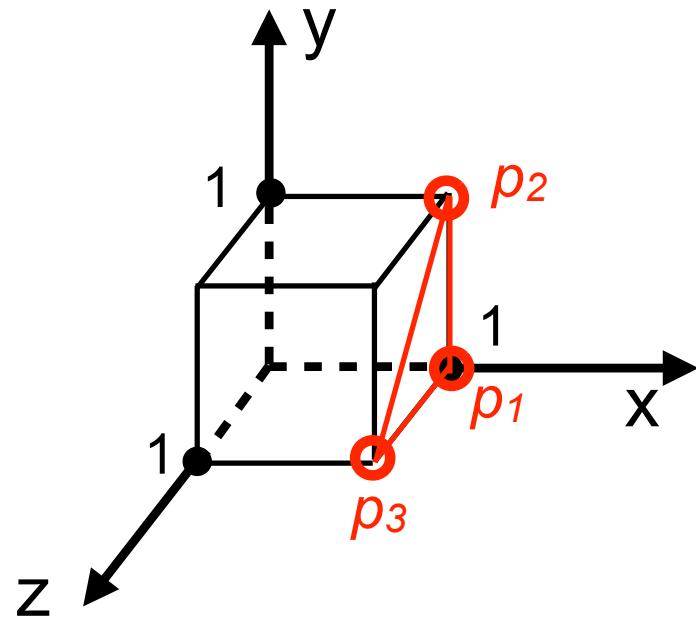
$$p_1 = (1,0,0)$$

$$p_2 = (1,1,0)$$

$$p_3 = (1,0,1)$$

Two in-plane vectors:

$$\vec{v} = p_2 - p_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \qquad \vec{w} = p_3 - p_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Normal vector:

$$\vec{v} \times \vec{w} = \begin{pmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{pmatrix} = \begin{pmatrix} 1 - 0 \\ 0 - 0 \\ 0 - 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Plane equation:

$$1x + 0y + 0z + D = 0$$

Inserting any point gives:

$$x - 1 = 0$$

Example (contd.): Triangle Front and Back Face

Three points (corners of the unit cube)

$$p_1 = (1,0,0)$$

$$p_2 = (1,0,1)$$

$$p_3 = (1,1,0)$$

Different order of vertices: mathematically negative (clockwise)
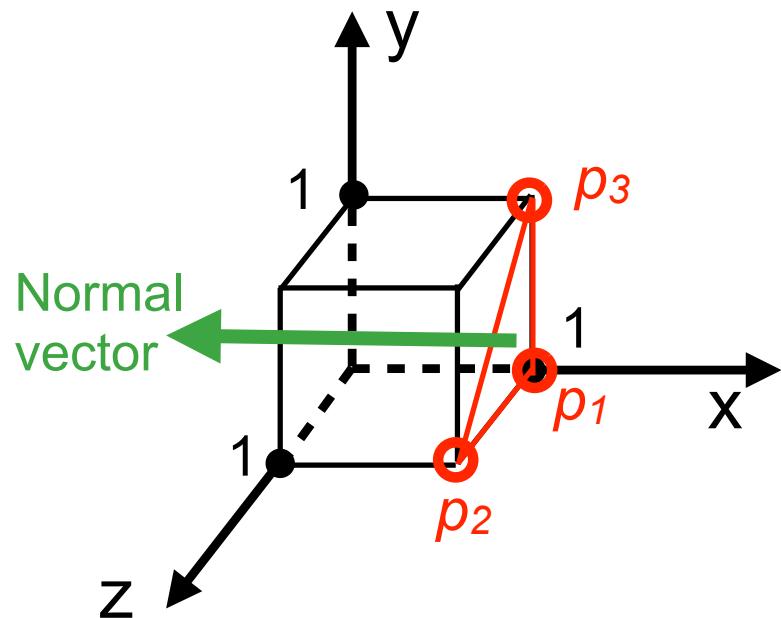
Normal vector:

$$\vec{v} \times \vec{w} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}$$

Plane equation:

$$(-1)x + 0y + 0z + D = 0$$

Inserting any point gives:

$$-x + 1 = 0$$

For an arbitrary point:
Left hand side of plane equation gives value > 0: Point is in front of plane
Left hand side of plane equation gives value < 0: Point is behind plane

Right Hand Rule for Polygons
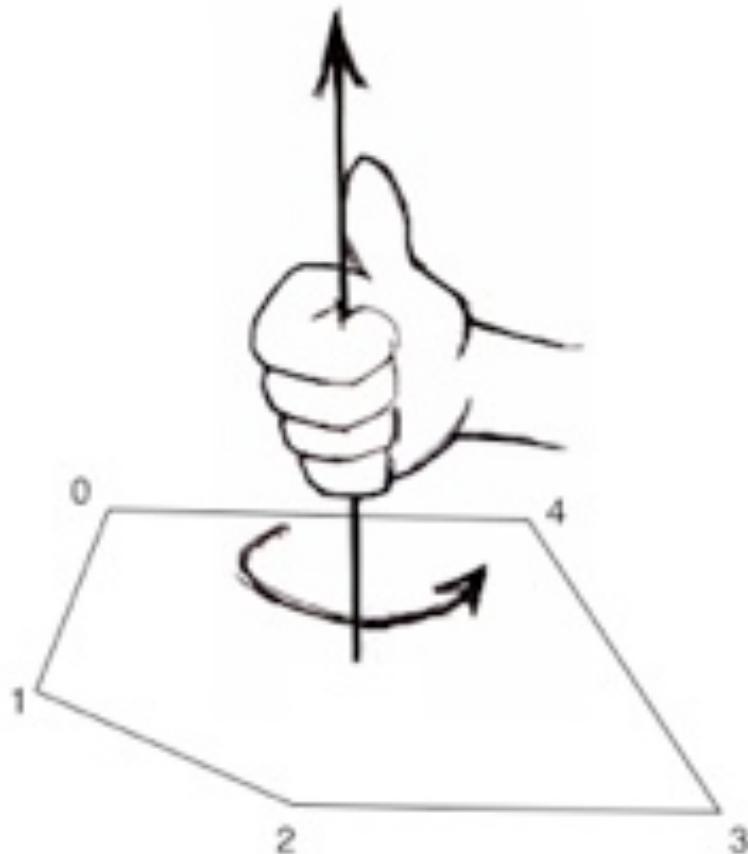
- A "rule of thumb" to determine the front side
  (= direction of the normal vector) for a polygon

- Please note: The relationship between vertex order and normal vector is just a *convention*!

  - Q: How can we see this from the previous slides?

Source: http://www.csse.monash.edu.au/~cema

Face-Vertex Meshes

http://en.wikipedia.org/wiki/File:Mesh_fv.jpg

Möbius Strip: Non-Orientable Surface
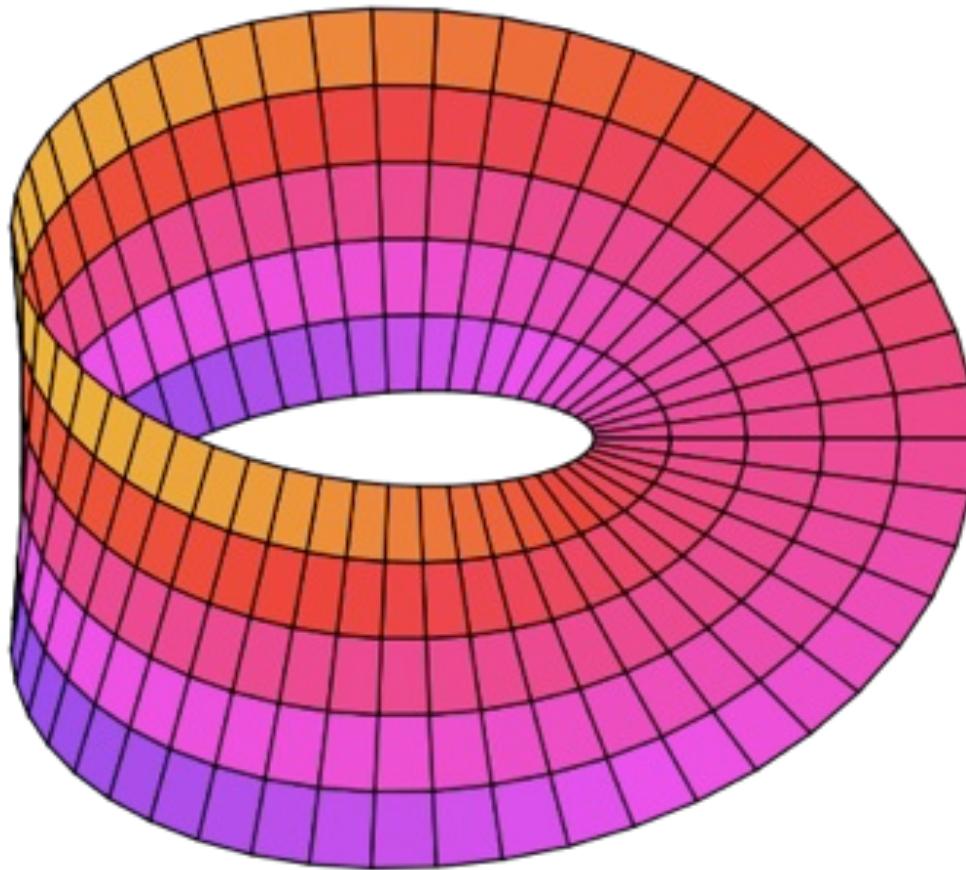
Complete object:
Does not have a
front and back side!

Polygon Meshes: Optional data

- Color per vertex or per face: produces colored models
- Normal per face:
  - Easy access to front/back information (for visibility tests)
- Normal per vertex:
  - Standard computation accelerated (average of face normals)
  - Allows free control over the normals
    - use weighted averages of normals
    - mix smooth and sharp edges (VRML/X3D: crease angles)
    - wait for shading chapter ;-)
- Texture coordinates per vertex
  - wait for texture chapter ;-)
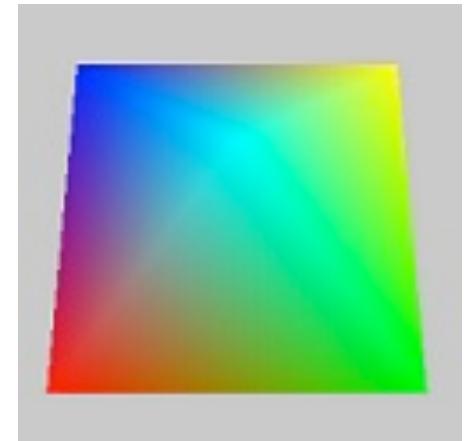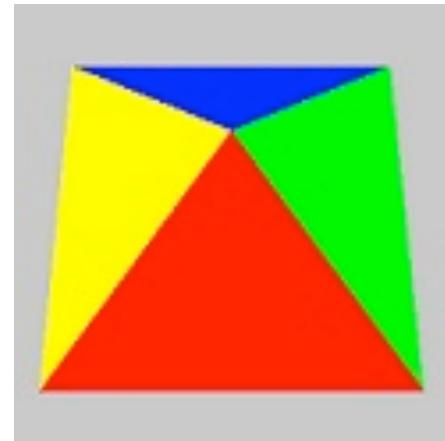
http://en.wikipedia.org/wiki/File:Triangle_Strip.png

Polygon Meshes: other descriptions

- Other representations for polygon meshes exist
  - optimized for analyzing and modifying topology
  - optimized for accessing large models
  - optimized for fast rendering algorithms
  - optimized for graphics hardware

- Example: triangle strip
  - needs N+2 points for N polygons
  - implicit definition of the triangles
  - optimized on graphics hardware
  - OpenGL / JOGL:
    ```
    gl.glBegin(GL2.GL_TRIANGLE_STRIP)
        gl.glVertex3d(-1, -1, 1);
        ...
    ```

Practical example: VRML IndexedFaceSet

Quiz: what is given by the following piece of VRML code??

```
geometry IndexedFaceSet {
  coord Coordinate {
    point [ -1 0 1, 1 0 1, -1 0 -1, 1 0 -1, 0 1 0 ]
  }
  coordIndex [ 0, 1, 4, -1,
               1, 3, 4, -1,
               3, 2, 4, -1,
               2, 0, 4, -1,
               1, 0, 2, 3, -1 ]
}
```

y

x

z
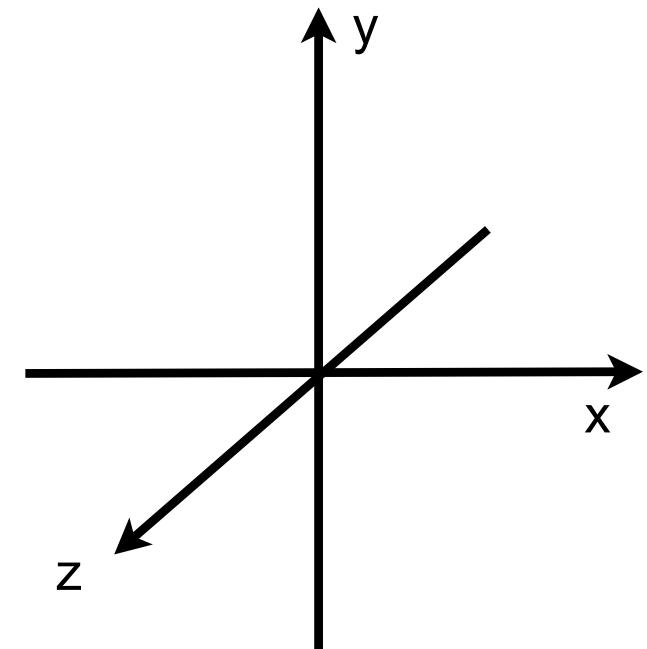
Approximating Primitives by Polygon Meshes

- Trivial for non-curved primitives...
- The curved surface of a cylinder, sphere etc. must be represented by polygons somehow (Tesselation).
- Not trivial, only an approximation and certainly not unique!
  - GLU utility functions for tesselation exist
- Goal: small polygons for strong curvature, larger ones for areas of weak curvature
  - This means ideally constant polygon size for a sphere
  - Where do we know this problem from??? Something playful...
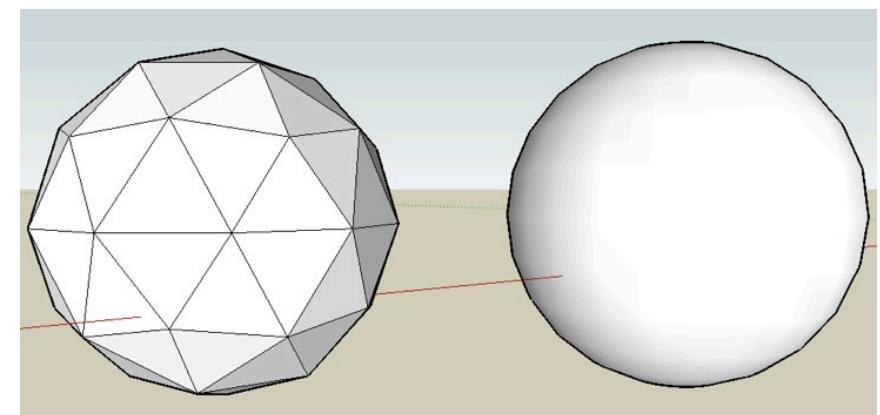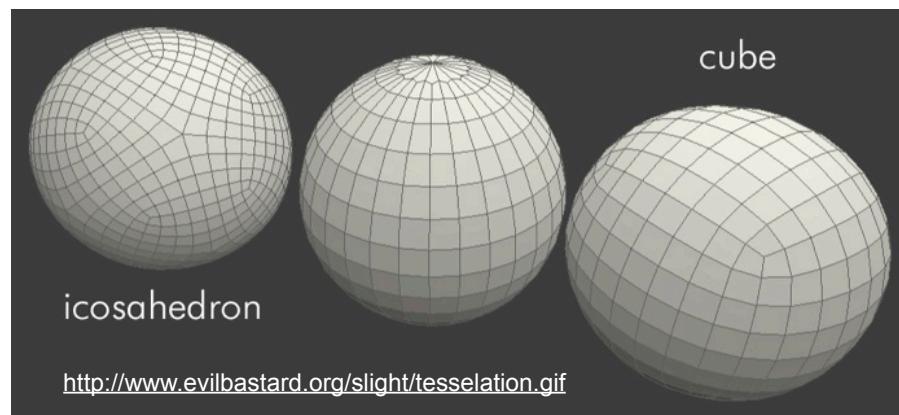
cube

icosahedron

http://www.evilbastard.org/slight/tesselation.gif

Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

Geometric Primitives

- Simplest way to describe geometric objects
- Can be used directly by some renderers (e.g., Ray tracing)
- Can be transformed into polygons easily (Tesselation)
- Can be transformed into Voxels easily
- Useful for creating simple block world models

- Supported in many frameworks of different levels
  - VRML/X3D, Java 3D
  - OpenGL, JOGL
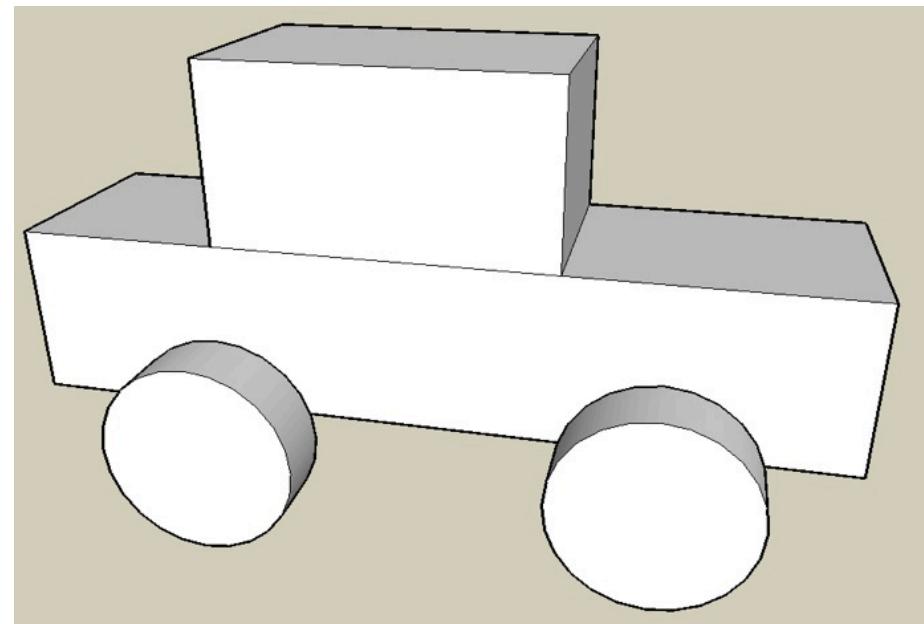
Box

- Described by (width, length, height)
- Origin usually in the center
- 8 points, 12 edges, 6 rectangles, 12 triangles
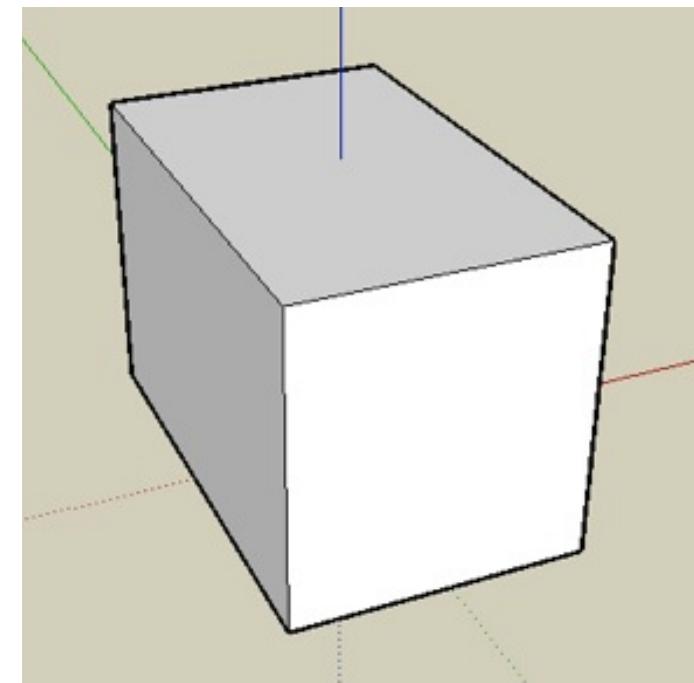
Pyramid, Tetrahedron *(Tetraeder)*

- Basis of pyramid = rectangle
- given by (width, length, height)
- 5 points, 8 edges, 6 triangles

- Basis of tetrahedron = triangle
- given by (width, length, height)
- 4 points, 6 edges, 4 triangles,
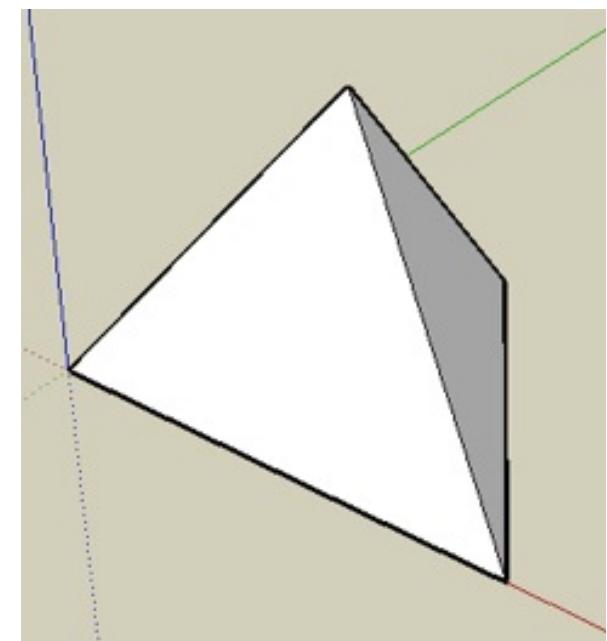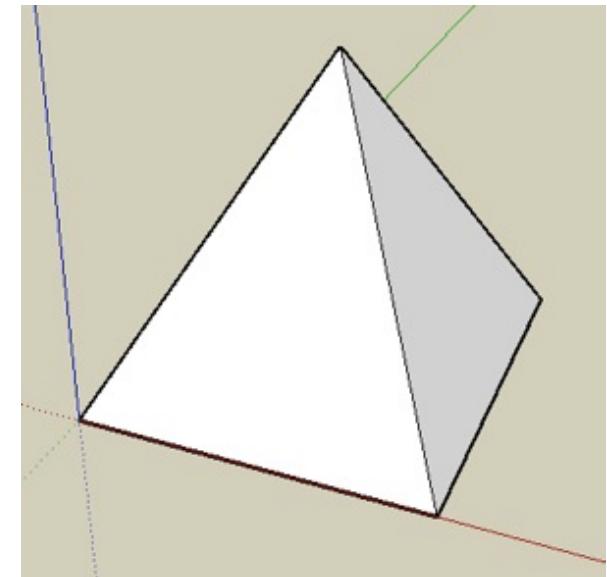
Generalization: Polyhedra

- Polyhedron *(Polyeder)*:
  - Graphical object where a set of surface *polygons* separates the interior from the exterior
  - Most frequently used and best supported by hardware: surface triangles
  - Representation: Table of
    - Vertex coordinates
    - Additional information, like surface normal vector for polygons

- Regular polyhedra: Five Platonic regular polyhedra exist
  - Tetrahedron *(Tetraeder)*
  - Hexahedron, Cube *(Hexaeder, Würfel)*
  - Oktahedron *(Oktaeder)*
  - Dodekahedron *(Dodekaeder)*
  - Icosahedron *(Ikosaeder)*

http://www.aleakybos.ch/

Cylinder, cone, truncated cone

- Cylinder given by (radius, height)
- Number of polygons dep. on tesselation

- Cone given by (radius, height)
- Number of polygons dep. on tesselation

- Truncated cone given by (r1, r2, height)
- Number of polygons dep. on tesselation

- Which of these would you rather have if you only had one available?
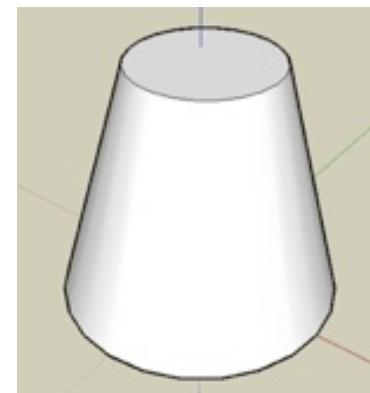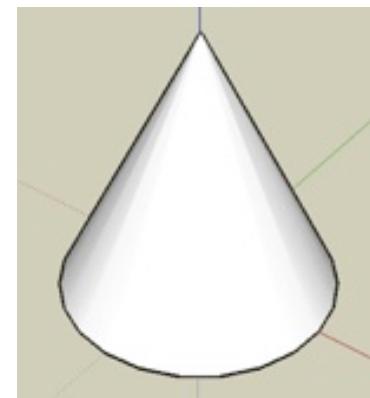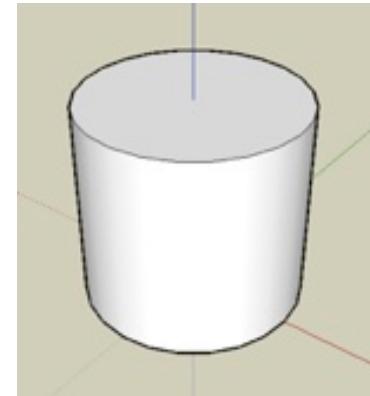
Sphere, Torus

- Sphere is described by (radius)
- Torus is defined by (radius1, radius2)
- Number of polygons dep. on tesselation
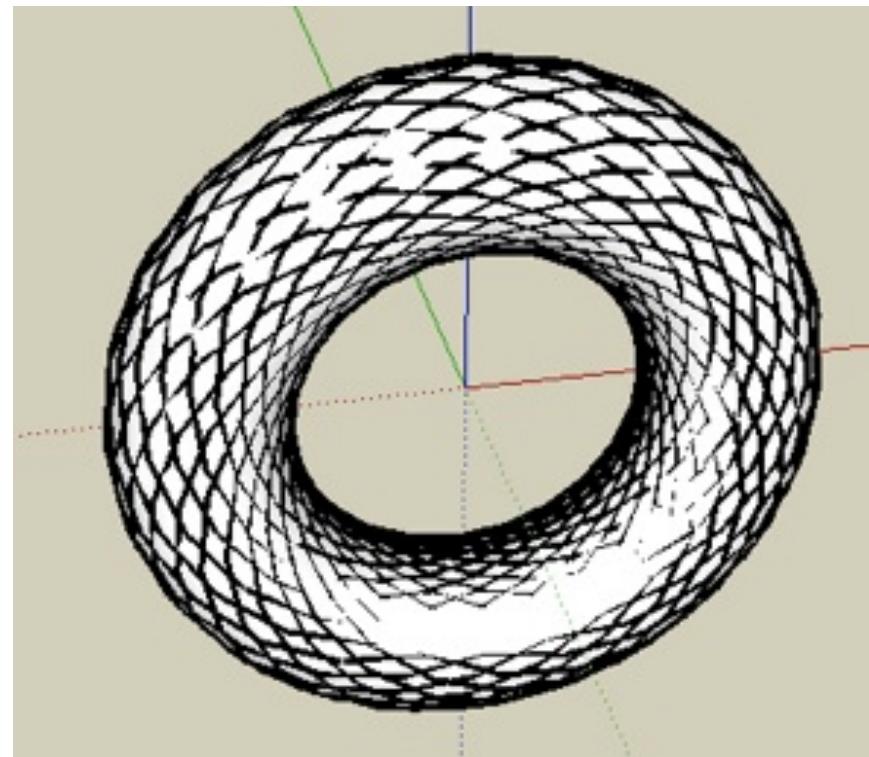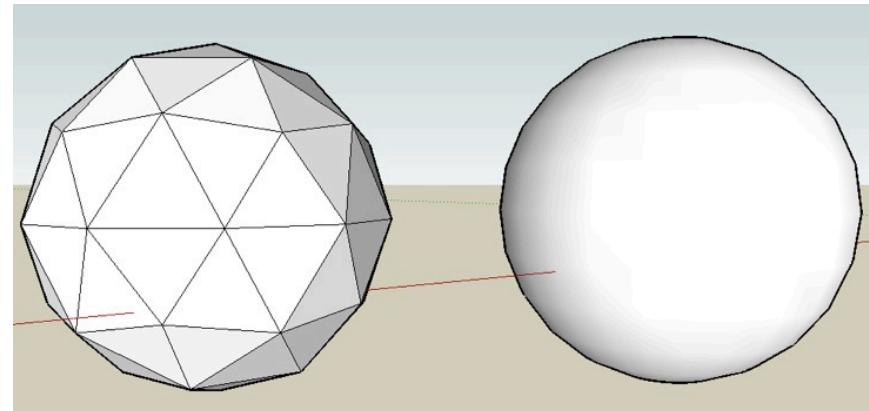
Geometric Primitives: Summary

- Not all of these exist in all graphics packages
- Some packages define additional primitives (dodecahedron, teapot...;-)

- Practically the only way to model VRML or X3D in a text editor
- Can give quite accurate models
- Extremely lean! Very few polygons

- Think of application areas even in times of powerful PC graphics cards!
  –
  –
  –
  –

GLUT Polyhedrons

```java
import com.jogamp.opengl.util.gl2.GLUT;

public void display(GLAutoDrawable drawable) {
    …
    final GLUT glut = new GLUT();
    glut.glutWireOctahedron();
    …
}
```
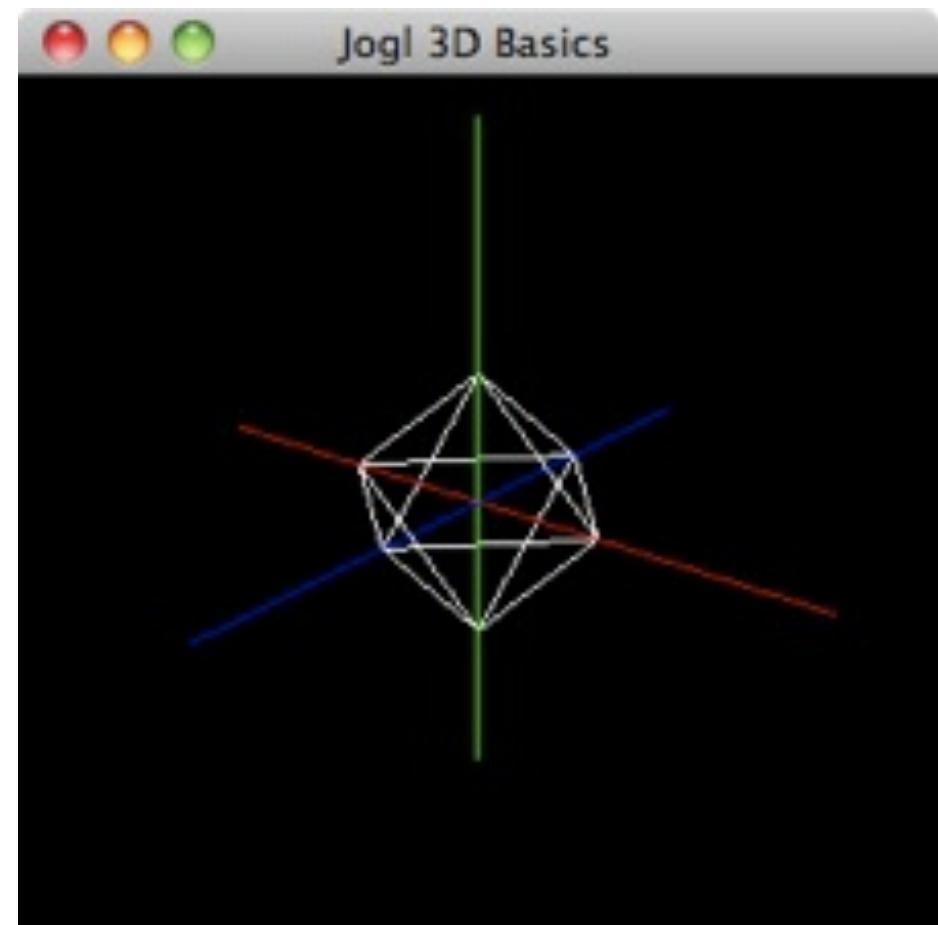
Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

Interpolation Curves, Splines

- Original idea: „Spline" used in ship construction to build smooth shapes:
  - Elastic wooden band
  - Fixed in certain positions and directions
  - Mathematically simulated by interpolation curves
  - Piecewise described by polynomials
- Different types exist
  - Natural splines
  - Bézier splines
  - B-Splines
- Control points may be on the line or outside of it.
 - All on the line for a natural spline
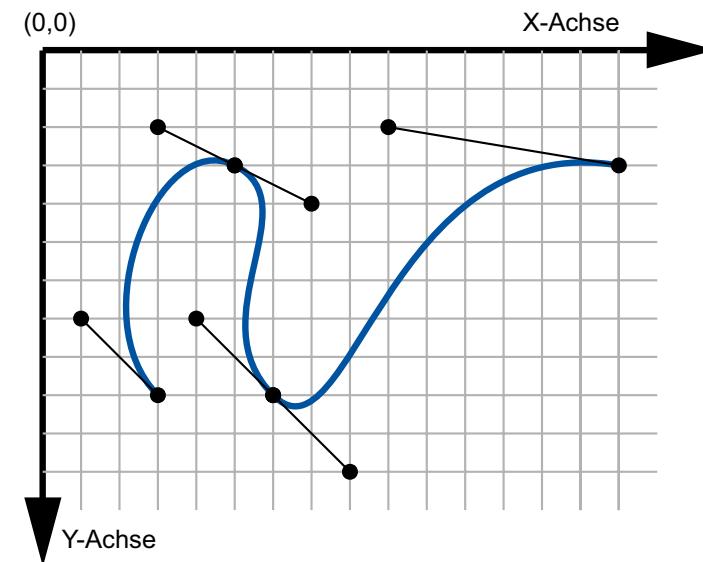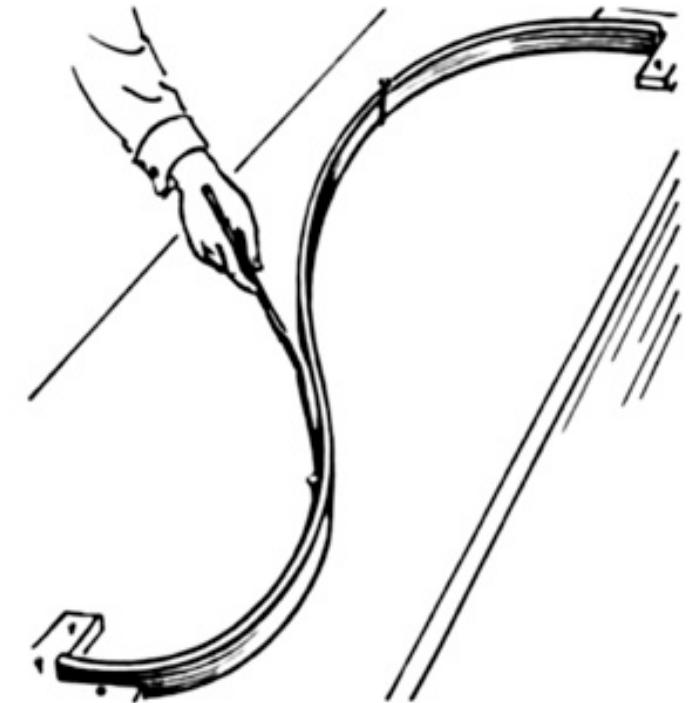
Bézier Curves (and de-Casteljau Algorithm)

- Bézier curves first used in automobile construction
  (1960s, Pierre Bézier – Renault, Paul de Casteljau – Citroën)

- Degree 1: straight line interpolated between 2 points

- Degree 2: quadratic polynomial

- Degree 3: cubic Bézier curve, described by cubic polynomial

- Curve is always contained in convex hull of points

- Algorithm (defines line recursively):
  - Choose t between 0 and 1
  - I1: Divide line between P1 and P2 as t : $(1-t)$
  - I2, I3: Repeat for all Ps *(one segment less!)*
  - J1, J2: Repeat for I1, I2, I3 (same t)
  - K: Repeat for J1, J2 *(single point!)*
  - Bézier curve: all points K for t between 0 and 1

- see http://goo.gl/811sV (Dominik Menke)
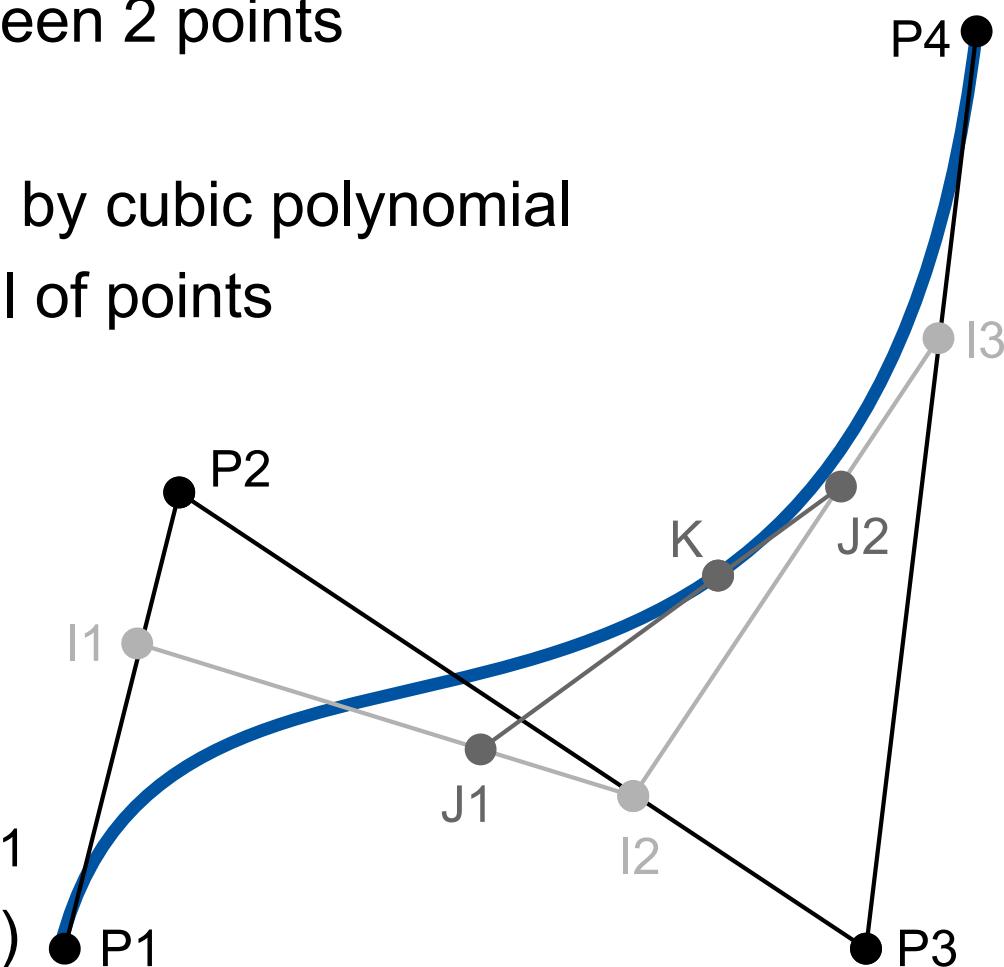
Bézier Patches

- Combine 4 Bézier curves along 2 axes

- Share 16 control points

- Results in a smooth surface

- Entire surface is always contained within the convex hull of all control points

- Border line is fully determined by border control points

- Several patches can be combined

  – connect perfectly if border control points are the same.

- Advantage: move just one control point to deform a larger surface...

- Other interpolation surfaces based on other curves

  – Generalization of Bézier idea: B-splines

  – Further generalization: Non-uniform B-splines

  – Non-uniform rational B-splines (NURBS) *(supported by OpenGL GLU)*
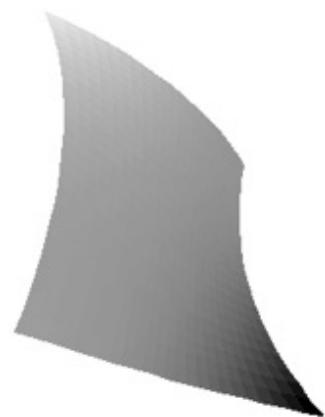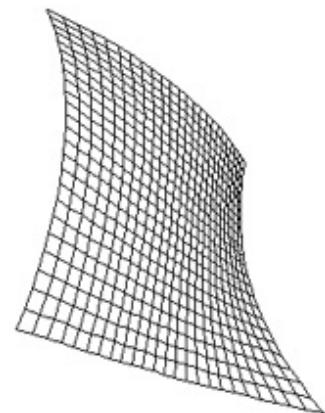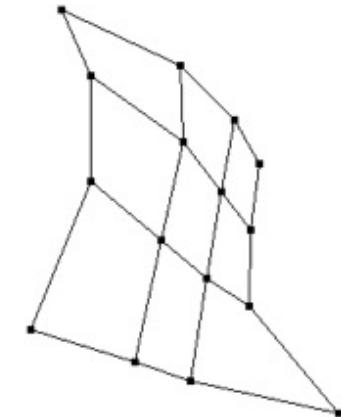
Interpolation in OpenGL (Bezier Example)

- Utah teapot
  - Martin Nevell, 1975
  - 306 vertices
  - 32 bicubic Bézier surface patches
- JOGL:

```
final GLUT glut = new GLUT();
glut.glutWireTeapot(1.5);
```
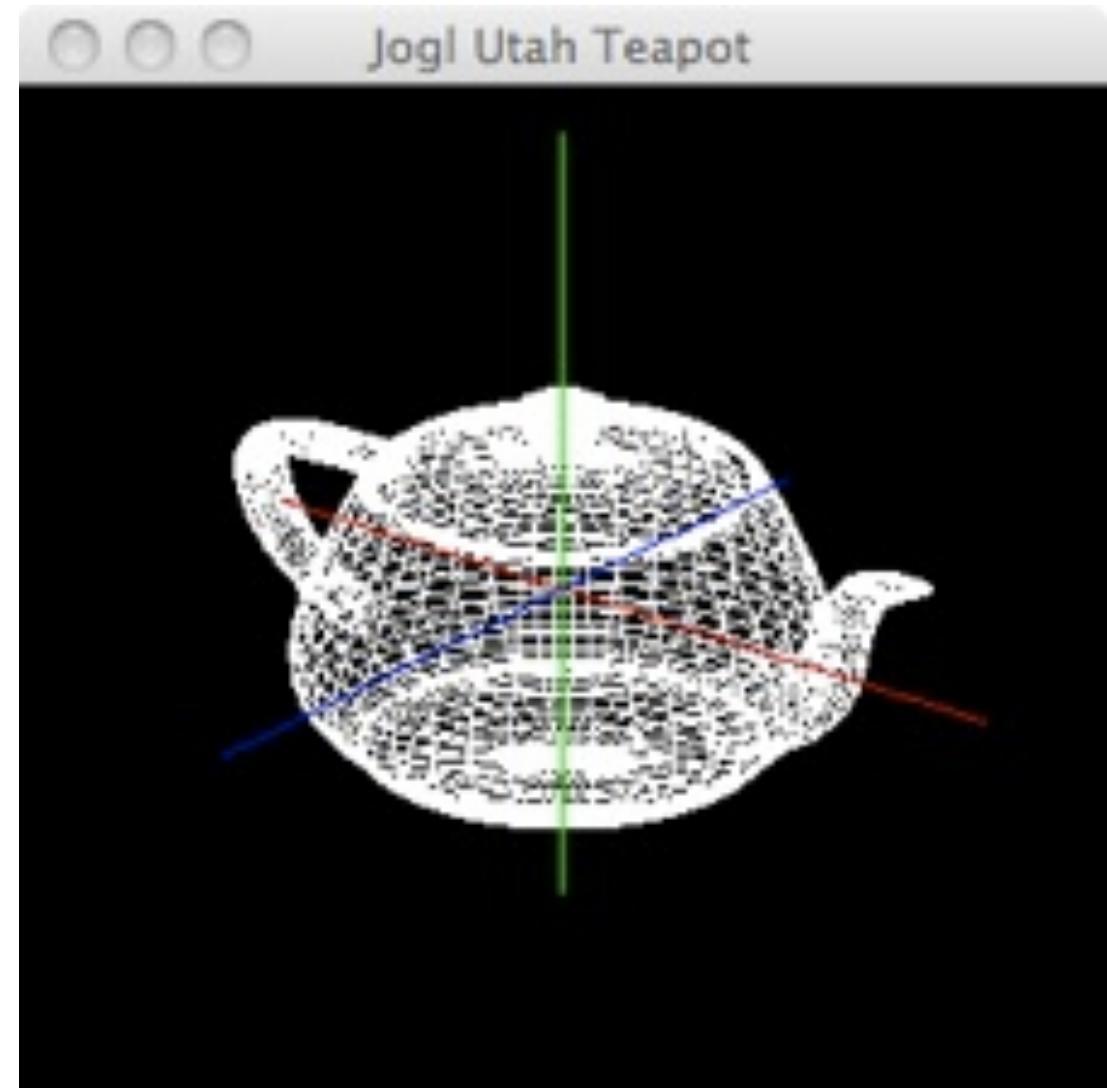
Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

Levels of Detail

- Assume you have a very detailed model
- from close distance, you need all polygons
- from a far distance, it only fills a few pixels
- How can we avoid drawing all polygons?
  - –
  - –
  - –
  - –
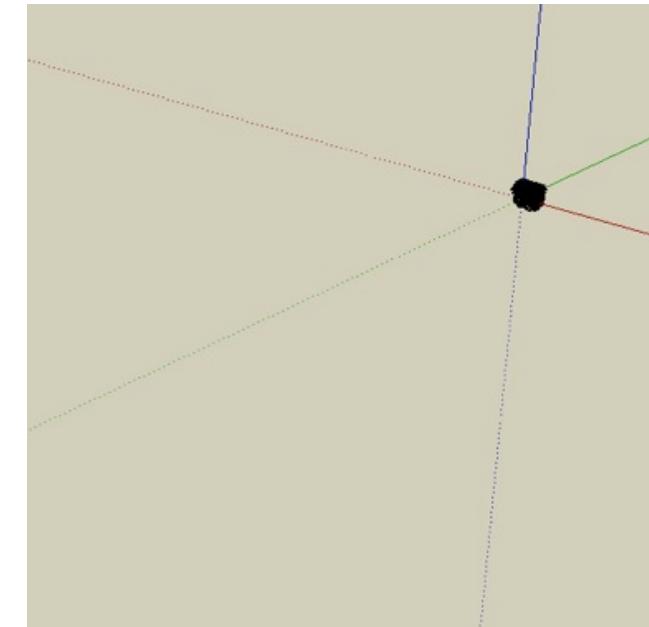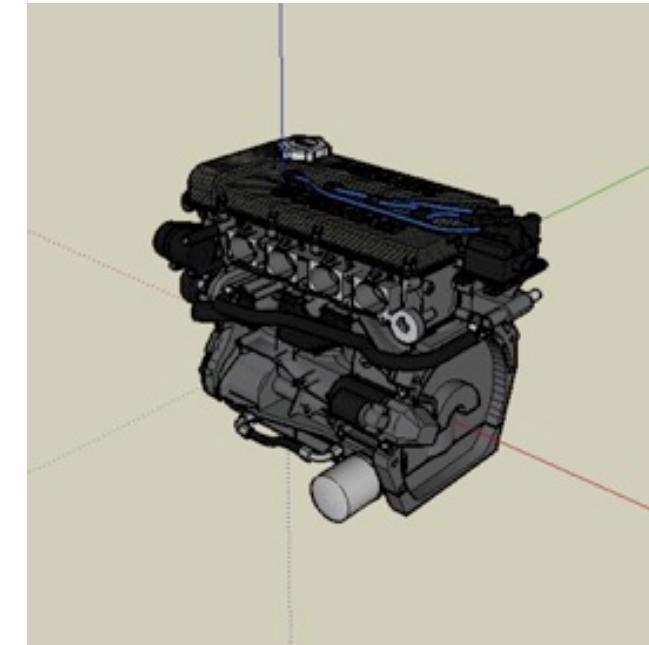
Mesh reduction

- Original: ~5.000 polygons
- Reduced model: ~1.000 polygons
- ==> about 80% reduction

- Very strong reductions possible, depending on initial mesh

- Loss of shape if overdone
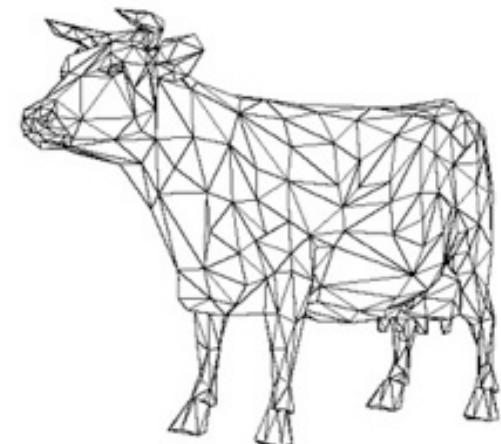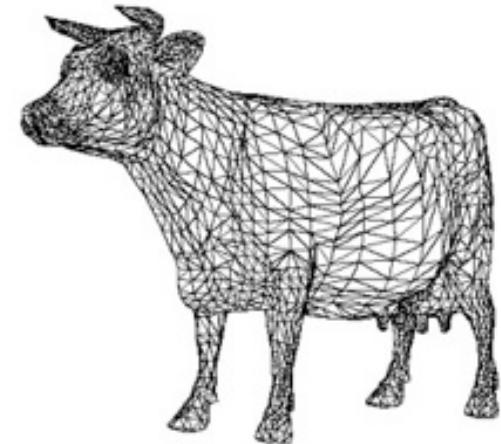
http://www.okino.com/conv/polygon_reduction/geoman2/polygon_reduction_tutorial1.htm

A method for polygon reduction

- Rossignac and Borell, 1992, „Vertex clustering"
- Subdivide space into a regular 3D grid
- For each grid cell, melt all vertices into one
  - Choose center of gravity of all vertices as new one
  - Triangles within one cell disappear
  - Triangles across 2 cells become edges (i.e. disappear)
  - Triangles across 3 cells remain
- Good guess for the minimum size of a triangle
  - edge length roughly = cell size
- Yields constant vertex density in space
- Does not pay attention to curvature
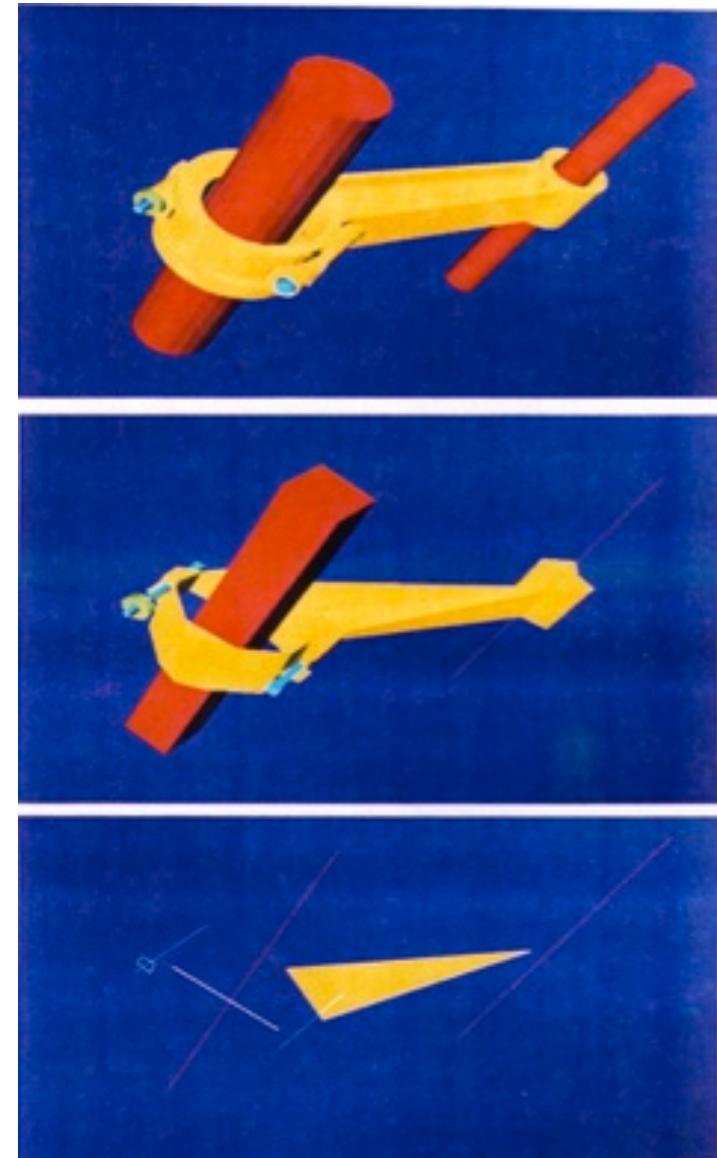
Billboard

- A flat object which is always facing you

- Very cheap in terms of polygons (2 triangles)

- Needs a meaningful texture

- Example (from SketchUp): guy in the initial empty world rotates about his vertical axis to always face you
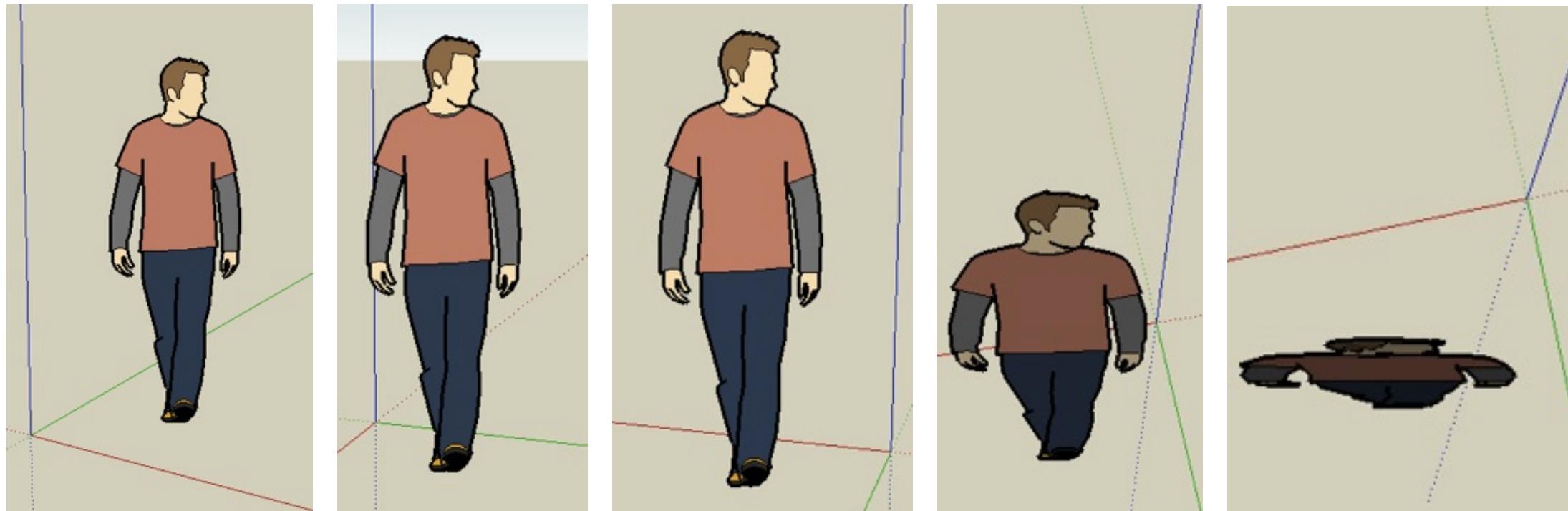
Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- **Constructive Solid Geometry (CSG)**
- Extrusion & Rotation
- Volume- and Point-based Graphics

Constructive Solid Geometry

- Basic idea: allow geometric primitives and all sorts of boolean operations for combining them

- Can build surprisingly complex objects

- Good for objects with holes (often the simplest way)

- Basic operations:
  - **Or**: combine the volume of 2 objects
  - **And**: intersect the volume of 2 objects
  - **Not**: all but the volume of an object
  - **Xor**: all space where 1 object is, but not both

- Think about:
  - wheels of this car
  - tea mug
 - coke bottle (Problems??)
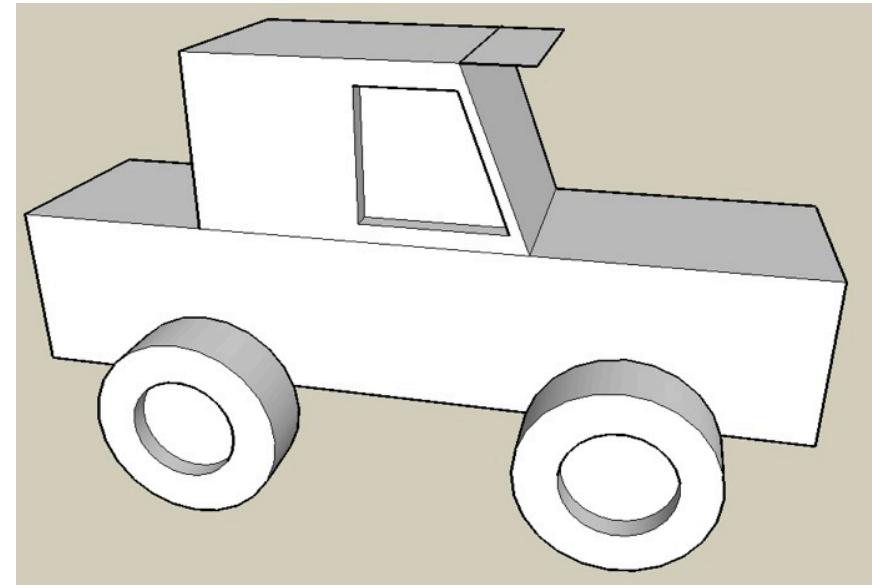
CSG: a complex Example

- rounded_cube =
  cube **And** sphere

- cross =
  cyl1 **Or** cyl2 **Or** cyl3

- result =
  rounded_cube **And** (**Not** cross)
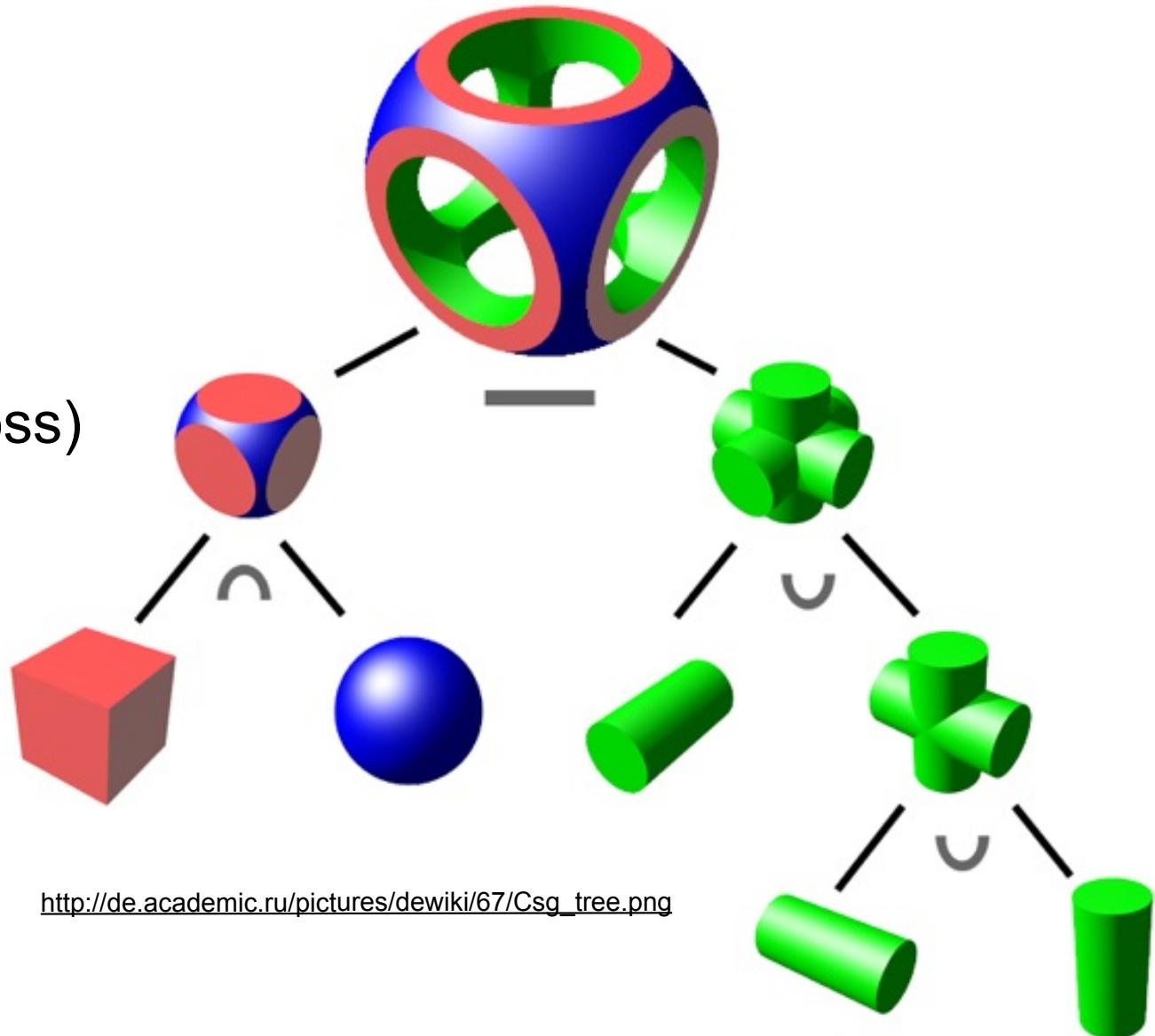
- Think: Are CSG operations associative?
  -

- ...commutative?
  -

http://de.academic.ru/pictures/dewiki/67/Csg_tree.png

Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

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
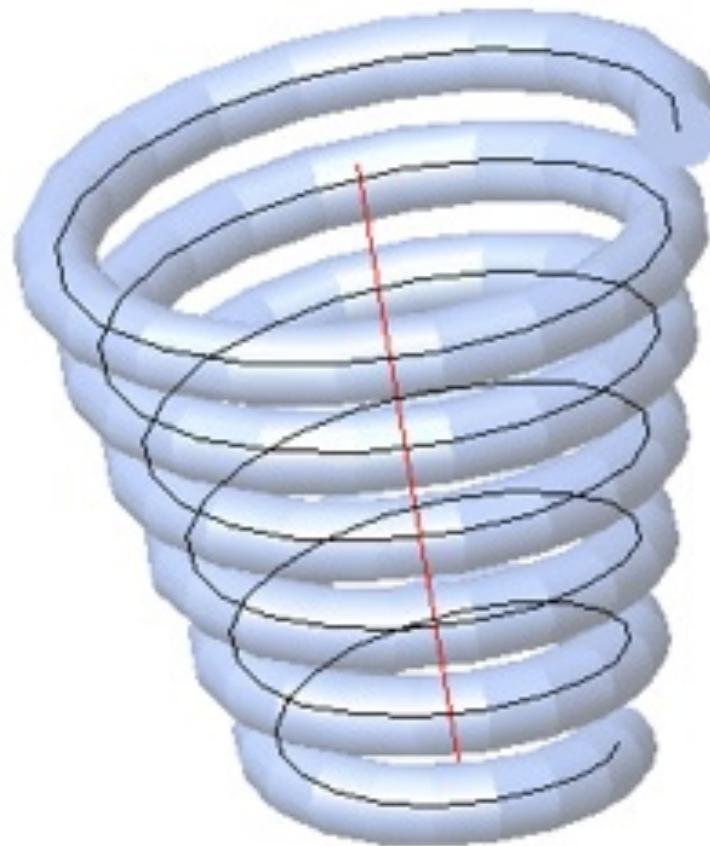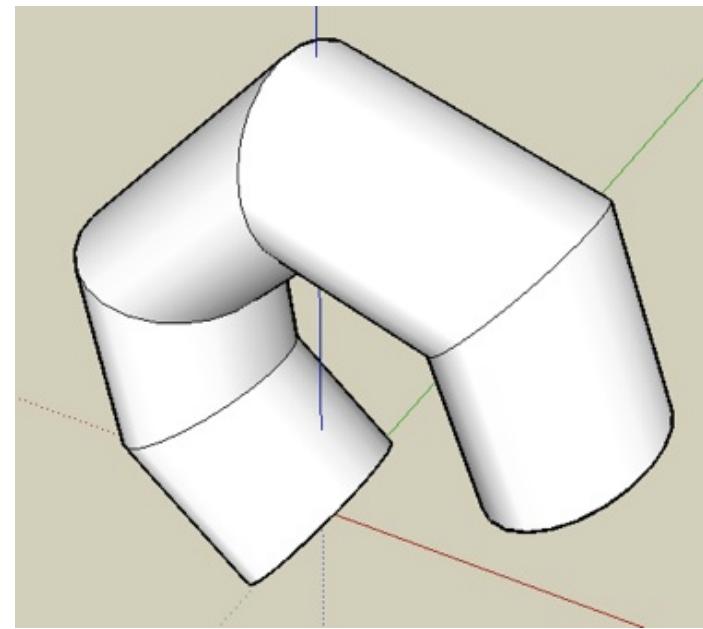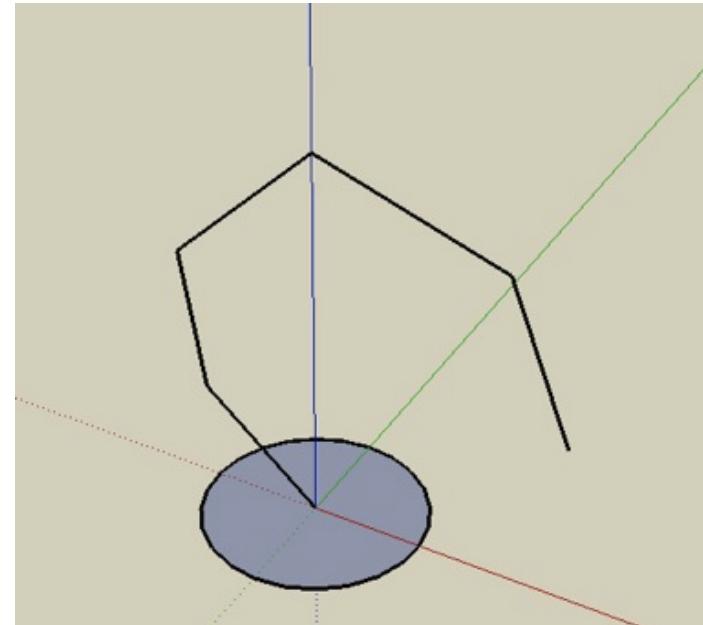
Rotation

- Rotate a 2D shape around an arbitrary axis

- Can be expressed by extrusion along a circle

- How can we model a vase?
  -
  -
  -

- How a Coke bottle?
  -
  -
  -
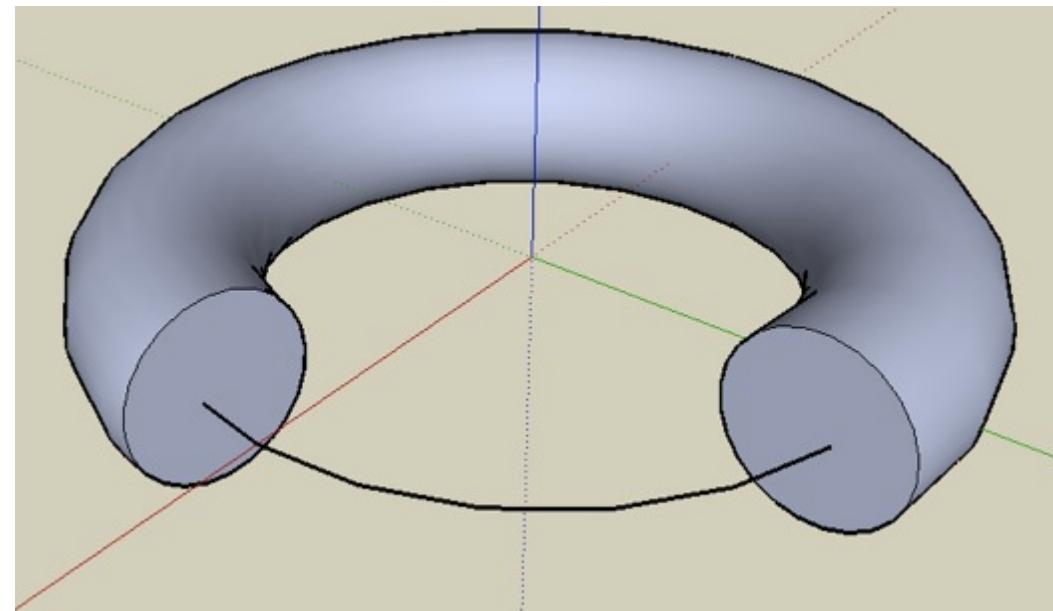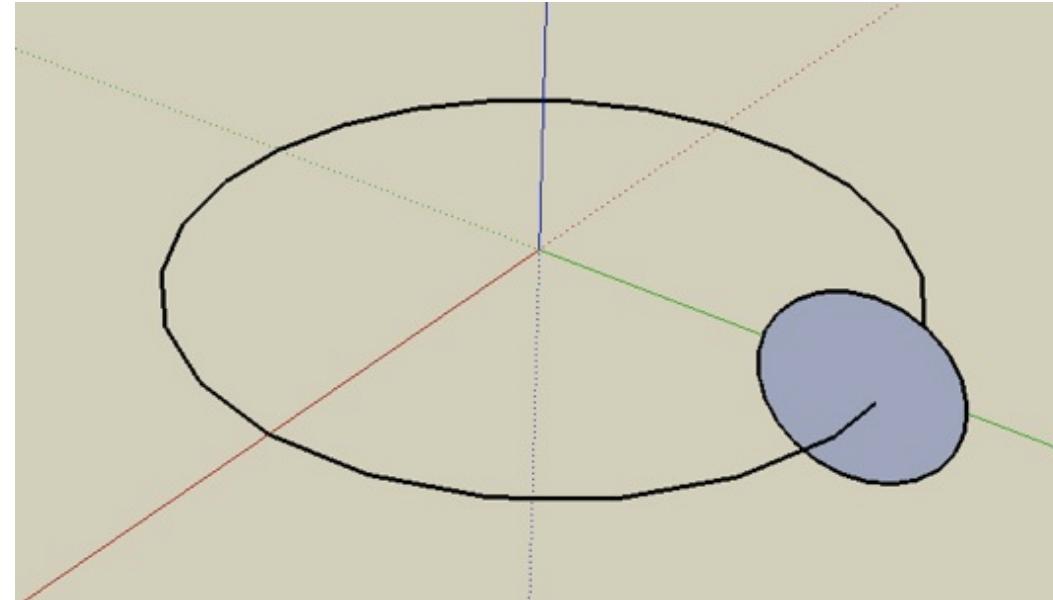
Chapter 4 - 3D Modeling

- Polygon Meshes
- Geometric Primitives
- Interpolation Curves
- Levels Of Detail (LOD)
- Constructive Solid Geometry (CSG)
- Extrusion & Rotation
- Volume- and Point-based Graphics

Voxel data

original 3d object

3d object converted to cubes

- „Voxel" = „Volume" + „Pixel", i.e., voxel = smallest unit of volume
- Regular 3D grid in space
- Each cell is either filled or not
- Memory increases (cubic) with precision
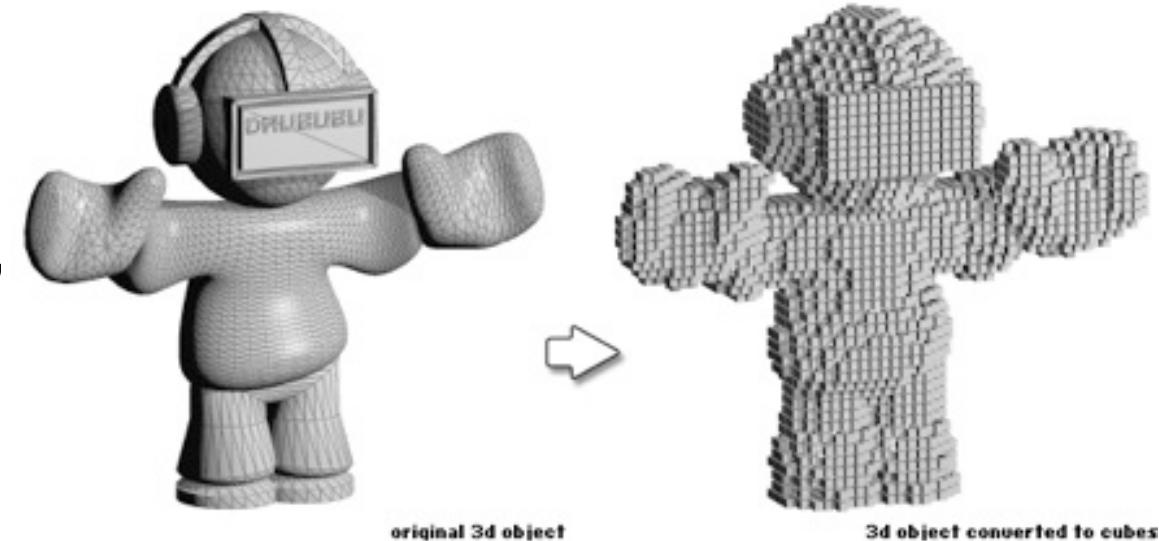
- Easily derived from CSG models
- Also the result of medical scanning devices
  - MRI, CT, 3D ultrasonic

- Volume rendering = own field of research
- Surface reconstruction from voxels

http://www.drububu.com/tutorial/voxels.html
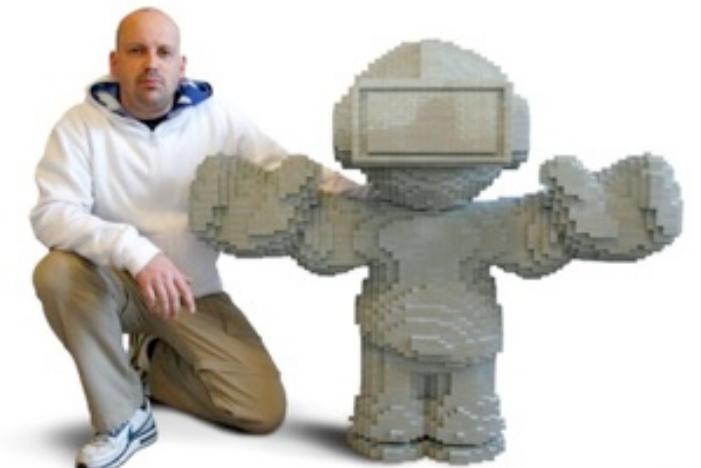
Point-based graphics

- Objects represented by point samples of their surface („Surfels")

- Each point has a position and a color

- Surface can be visually reconstructed from these points
    - purely image-based rendering
    - no mesh structure
    - very simple source data (x,y,z,color)

- Point-data is acquired e.g., by 3D cameras

- Own rendering techniques

- Own pipeline

- ==> own lecture ;-)

http://www.crs4.it/vic/data/images/img-exported/stmatthew_4px_full_shaded2.png

(C) 2004, CRS4 - Data courtesy of Standford University