CMT107 Visual Computing

III.1 Object Representation

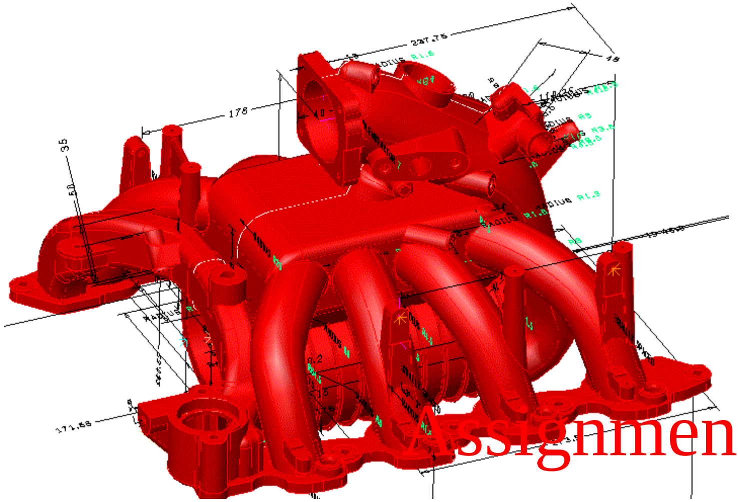
Xianfang Sun

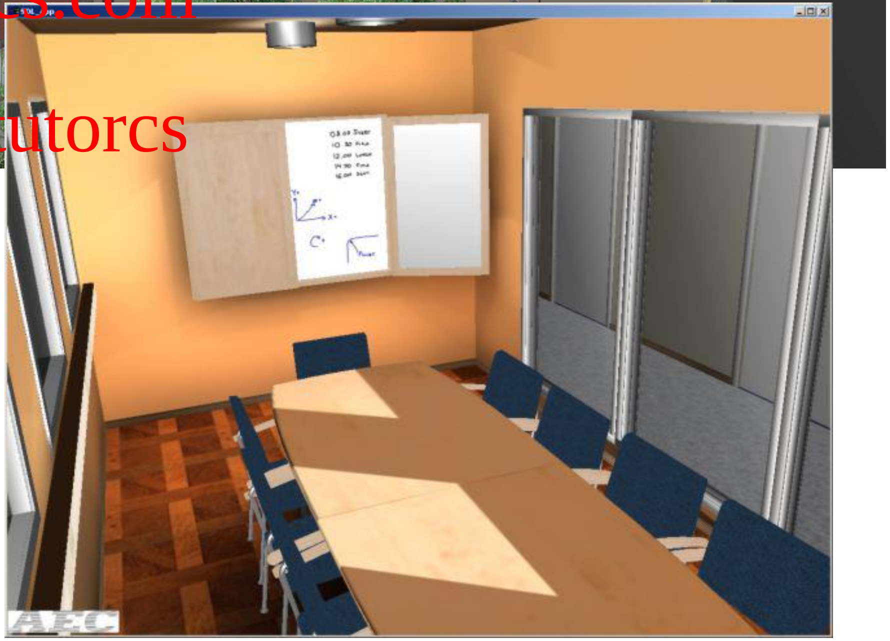
School of Computer Science & Informatics
Cardiff University

Overview

- Constructive solid geometry
- Boundary representation
- Mesh representation
  - Rendering meshes with OpenGL
- Volumetric representation: voxels

Example Models and Scenes





Geometric Modelling

- Need data-structures and algorithms to model shapes
  - *Scene* – description of the whole environment
  - *Model* – description of an object in the environment
  - Suitable for *creating, editing, analysing* and *rendering*

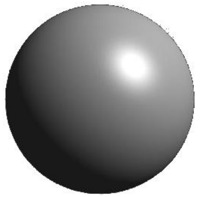
- Object representations

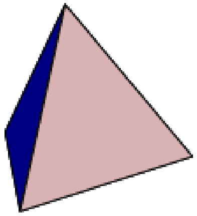
  - Constructive solid geometry (CSG)

  - Boundary representation (B-rep)
  - Mesh representation
  - Volumetric representation: voxels

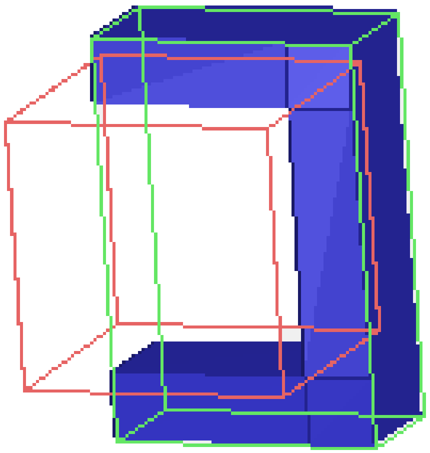
Constructive Solid Geometry

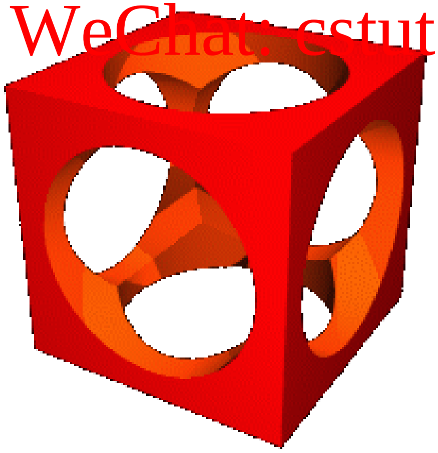
- Use set of volumetric primitives
  - Block, Tetrahedron, sphere, cylinder, cone, ...

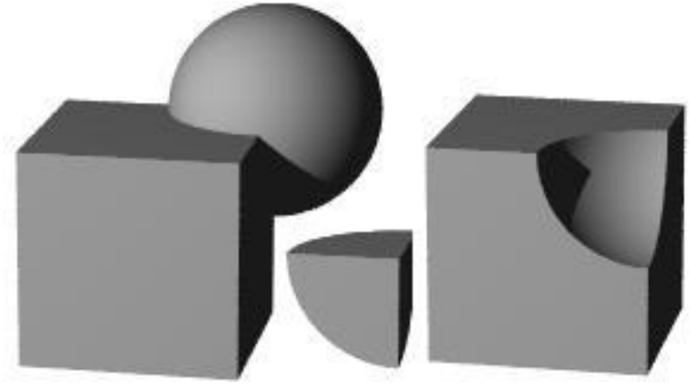




- Construct objects using Boolean operations
  - Union, intersection, difference

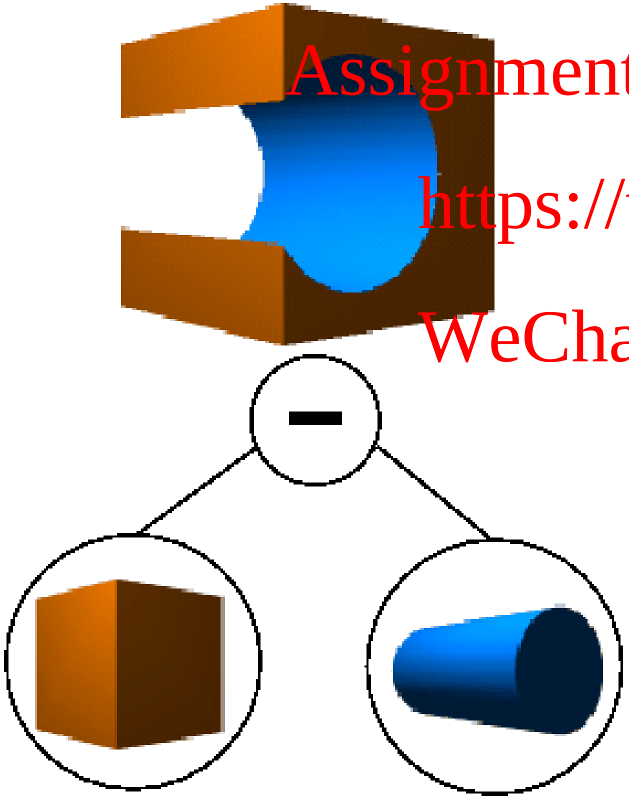


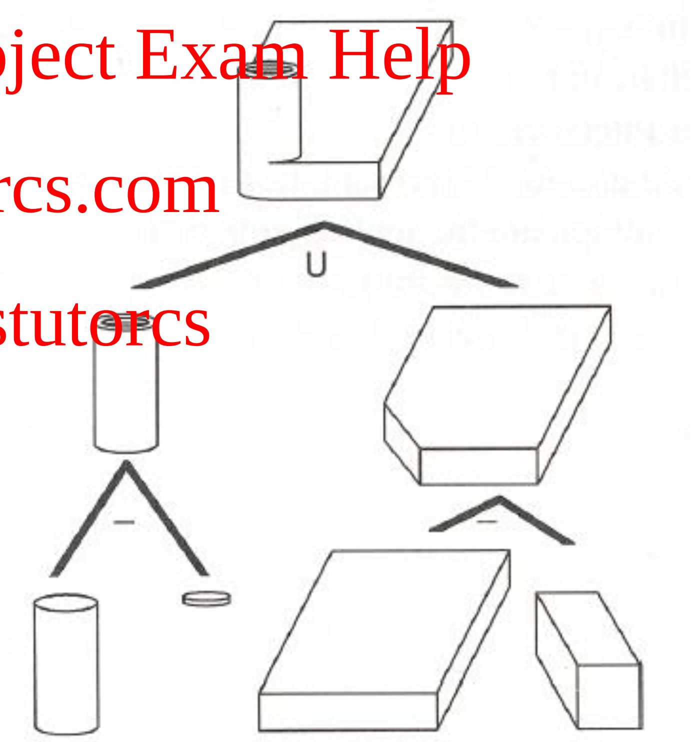




CSG Tree

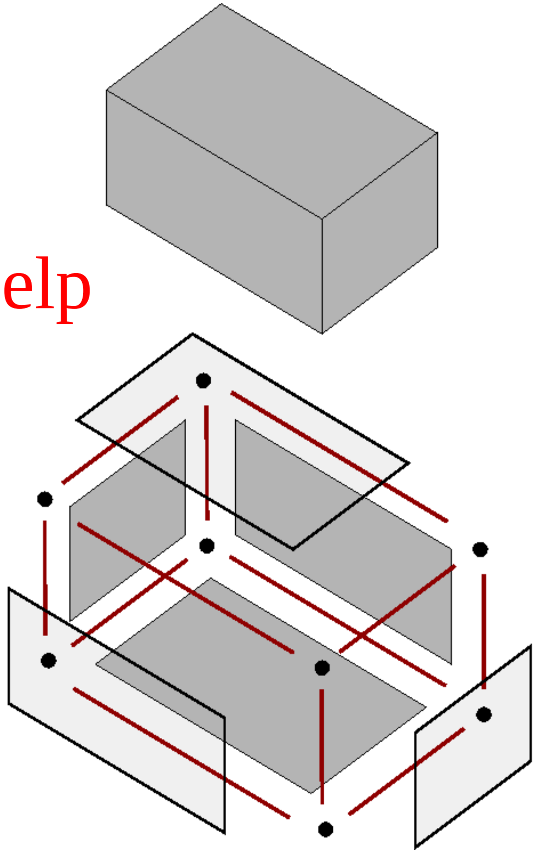
- CSG operations stored as tree (or sequence) of operations on primitives
- Common for CAD – *feature based modelling*





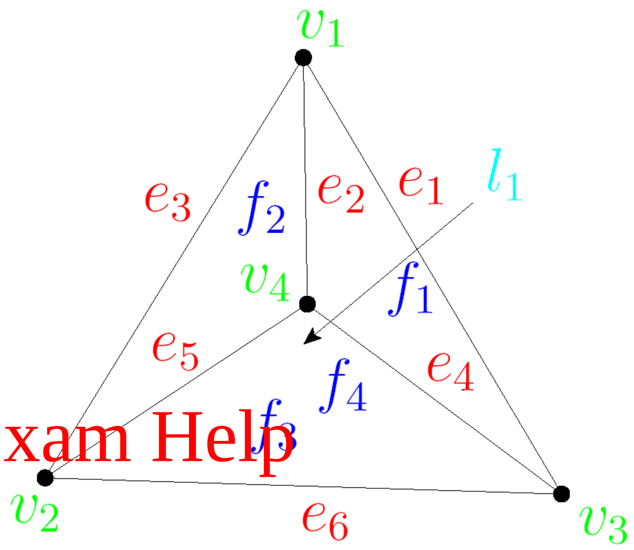
Boundary Representation

- Explicitly represent boundary of object:
  - Basic elements are (natural) *faces, edges, vertices* with a geometry (shape)
  - Also record topology (connectivity/ boundary relations) of elements
- Mathematically: an algebraic complex (topology) with a geometric realisation (geometry)
- Algorithmically: a graph data structure (topology) where nodes have shape (geometry) attributes



B-Rep: An Algebraic Complex

- *Cells* (elements) = $\{v_1, v_2, v_3, v_4, e_1, e_2, e_3, e_4, e_5, e_6, f_1, f_2, f_3, f_4, l_1\}$
- *Rank* (dimension) $=$ $\{(0, \{v_1, v_2, v_3, v_4\}),$ $(1, \{e_1, e_2, e_3, e_4, e_5, e_6\}),$ $(2, \{f_1, f_2, f_3, f_4\}), (3, \{l_1\})\}$



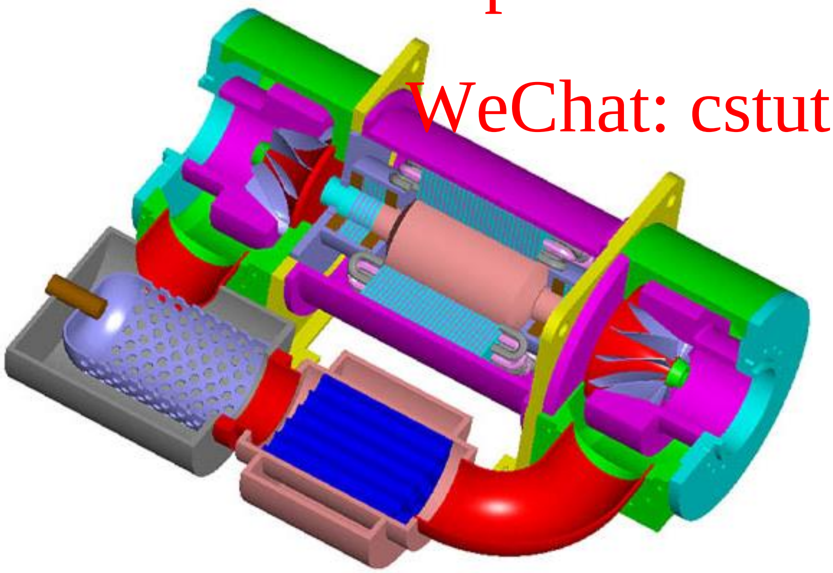
- *Bound* (topology) $= \{(e_1, \{v_1, v_3\}), (e_2, \{v_1, v_4\}),$ $(e_3, \{v_1, v_2\}), (e_4, \{v_3, v_4\}), (e_5, \{v_2, v_4\}), (e_6, \{v_2, v_3\}),$ $(f_1, \{e_1, e_2, e_4\}), (f_2, \{e_2, e_3, e_5\}), (f_3, \{e_1, e_3, e_6\}),$ $(f_4, \{e_4, e_5, e_6\}), (l_1, \{f_1, f_2, f_3, f_4\})\}$

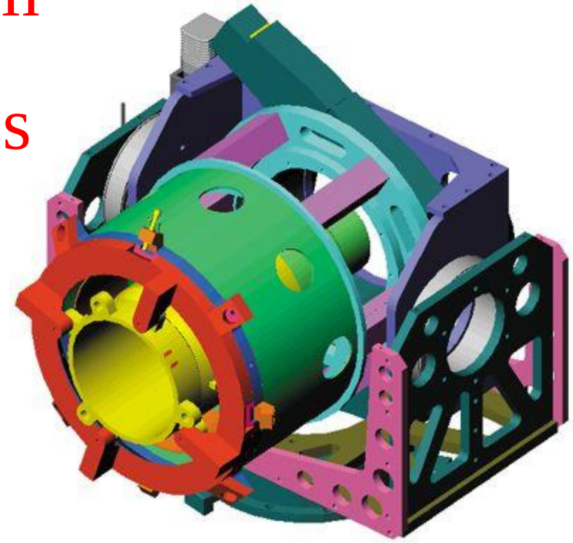
B-Rep Geometry

- Describe shape of each face, edge and vertex
  - Vertex geometry: position
  - Edge geometry: curve

    E.g. straight line, circle, ellipse, free-form curve, . . .
  - Face geometry: surface

    E.g. plane, sphere, cylinder, cone, torus, free-form, . . .





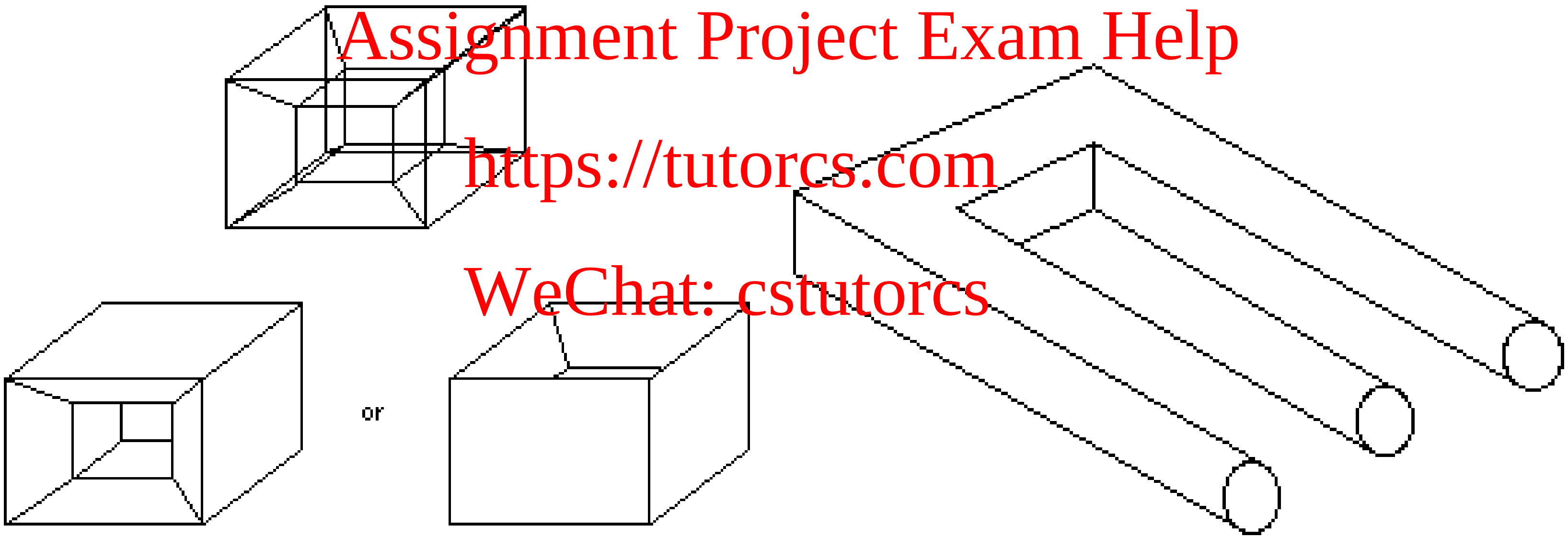
B-Rep Data Structure

- B-Rep graph data structure representing the topology:

BODY	Solid made of a list of LUMPS
LUMP	Connected volume, bounded by a list of SHELLS
SHELL	Connected surface, consisting of a list of FACES
FACE	Natural surface, bounded by a LOOP
LOOP	Connected curves, consisting of a list of COEDGES
COEDGE	Directed edge as part of a loop, consisting of an EDGE (also called half-edge)
EDGE	Natural edge, bounded by VERTICES
VERTEX	Boundary of an edge

B-Rep Issues

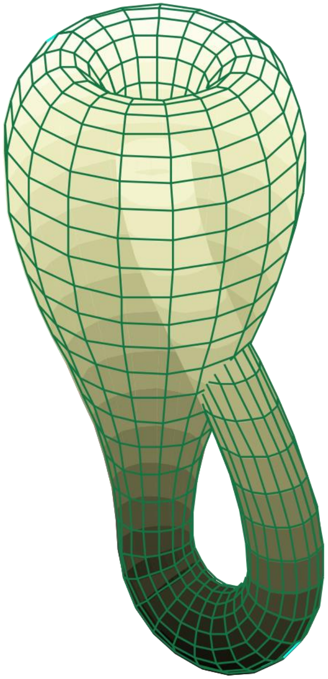
- Consistency of geometry and topology
  - No explicit way to ensure boundary relations are preserved by geometry
- Ambiguous and impossible models



- Topology allows us to determine impossible models
- Orientation and topology distinguish ambiguous models

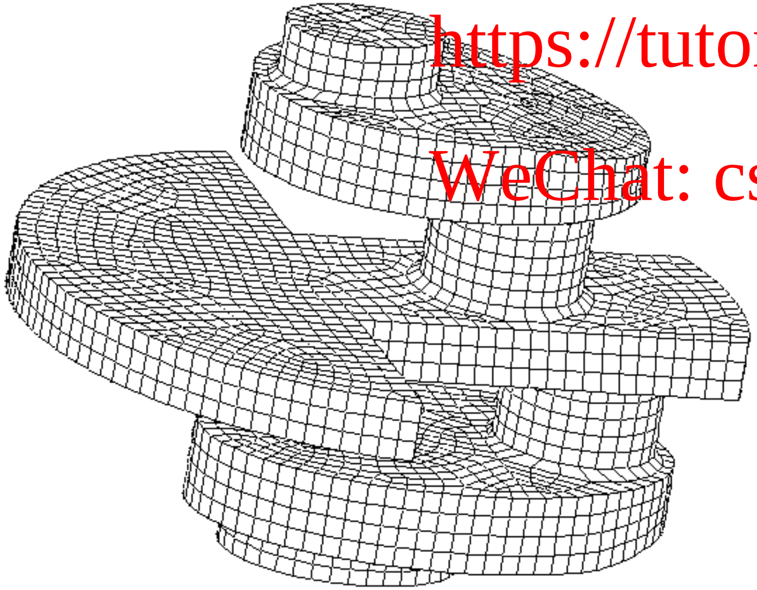
B-Rep Orientation

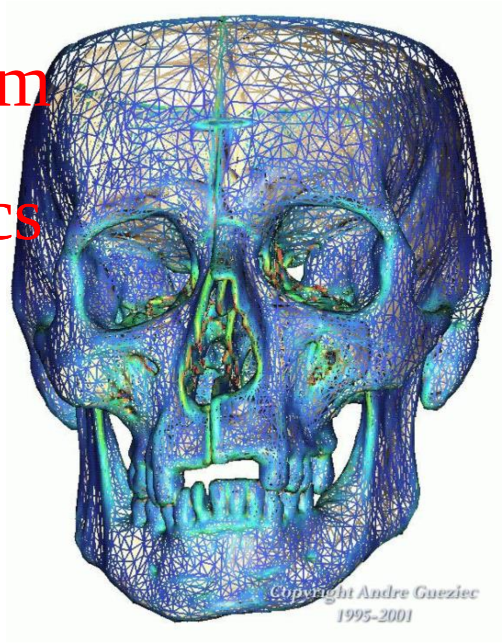
- Orient face: distinguish between inside and outside
  - Surface normals always point towards the outside
- Orient each loop
  - Move around each loop such that the inside lies to the left when viewed from outside the model
  - COEDGEs indicate direction of loop by ordering edge end-points
  - EDGE lies on two faces as indicated by two COEDGEs
- Non-manifold objects: EDGE can lie on more than two faces
  - Causes problems for orientation, etc. (so not allowed in standard B-rep)



Mesh Representation

- Describe model as a polygonal mesh (often triangular)
  - Collection of polygons (facets)
  - Similar, but simpler than B-rep
  - Linear approximation of object
  - Fast and quality good enough for real-time rendering





Polygons

- Polygons are specified by a sequence of vertices
- Polygons are not just line segments, but have an interior
  - Simple polygon: lines do not intersect
  - Convex polygon: given two points inside the polygon, the line segment joining the points lies inside polygon
  - Flat polygon: polygon lies in a plane
- Orientation / sidedness:
  - Polygons have a front and a back
  - If vertices are in anti-clockwise order on display, we see the front
    (default OpenGL convention; consistent with B-rep orientation)

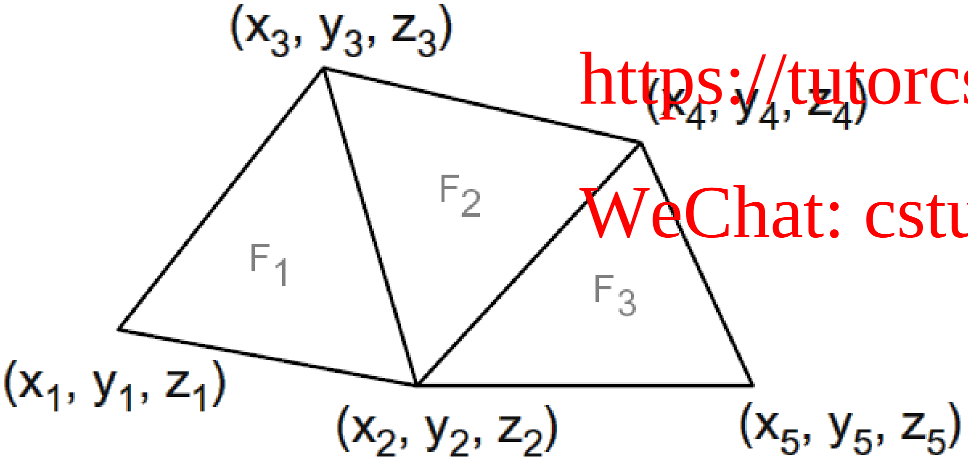
Polygon Normal

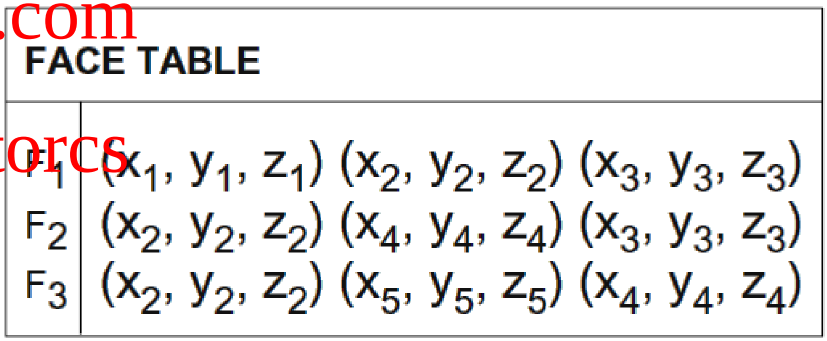
- If a polygon is simple, convex and flat, its normal can be calculate using any 3 non-collinear points p_l , p_m, and p_n
  - Suppose $l<m<n$
  - $v_1 = p_m - p_l$, $v_2 = p_n - p_m$,
  - $n = v_1 \times v_2$
  - normal n points outside the front.

- Polygon normal vector and the viewer direction vector can determine whether the viewer is looking at the front or back of the polygon
  - If the angle between normal vector and viewer direction vector are less than 90°, it's at the front
  - If the angle is great than 90°, it's at the back
  - If the angle is 90°, the viewer is on the polygon plane.

List of Faces

- Each face lists vertex coordinates
  - Redundant vertices
  - No adjacency or other structural information (topology)
  - Orientation from sequence of vertices

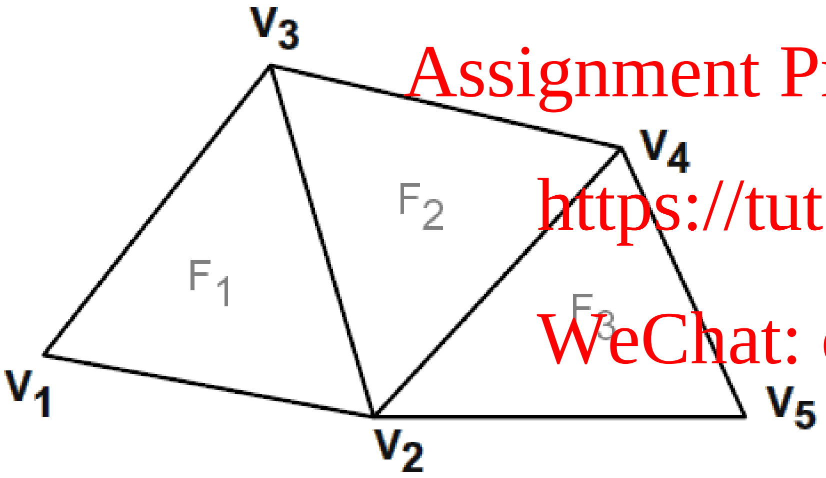




FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

- Each face lists vertex references
  - Shared vertices
  - No adjacency or other structural information (topology)
  - Orientation from sequence of vertices



VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

- Can add half-edges, shells, lumps, bodies for representing solids

Rendering Meshes with OpenGL

- Two simple OpenGL drawing functions:

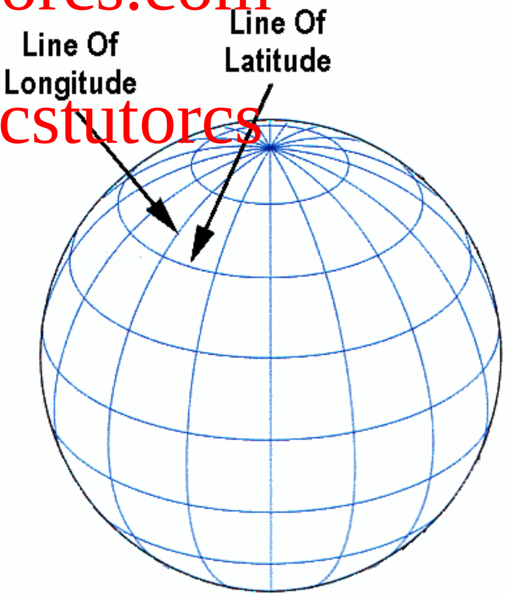
  - ✓ `glDrawArrays(mode,first,count);`
  - ✓ `glDrawElements(mode, count, type, indices);`

  - mode: GL_POINTS, GL_LINES, GL_TRIANGLES, etc.
  - first: the starting index in the enabled arrays.
  - count: the number of elements to be rendered
  - type: type of the values in indices, GL_UNSIGNED_BYTE, GL_UNSIGNED_SHORT, or GL_UNSIGNED_INT
  - indices: a pointer to the location where the indices are stored.

- glDrawArrays() is used for “List of Faces”
  - Example see CG02.java in the labs
- glDrawElements() is used for “Vertex and Face Tables”
  - Example see CG03.java in the labs

Modelling a Sphere

- A sphere can be modelled by covering the surface with triangles
  - use lines of longitude and latitude to divide the surface into triangles (around north and south poles) and quadrangles
  - each quadrangles is divided into two triangles for rendering by OpenGL



Spherical Coordinates

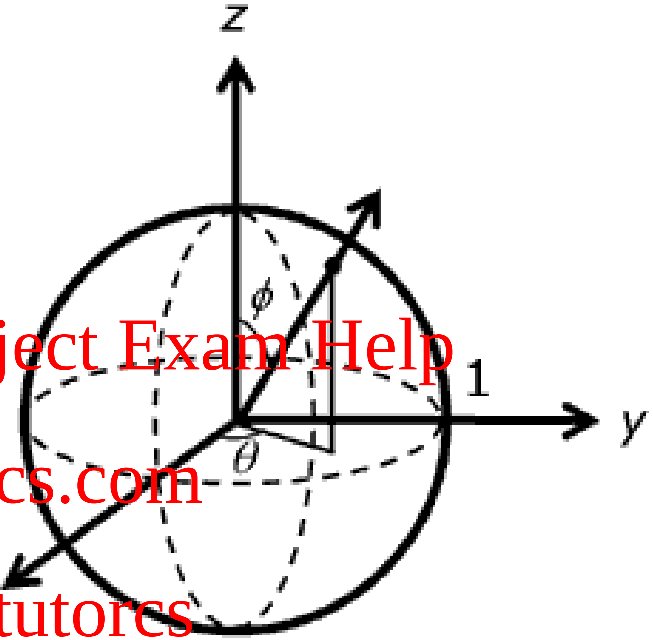
- Points on a unit sphere in spherical coordinates :

$$x(\phi,\theta) = \sin\phi\cos\theta$$

$$y(\phi,\theta) = \sin\phi\sin\theta$$

$$z(\phi,\theta) = \cos\phi$$

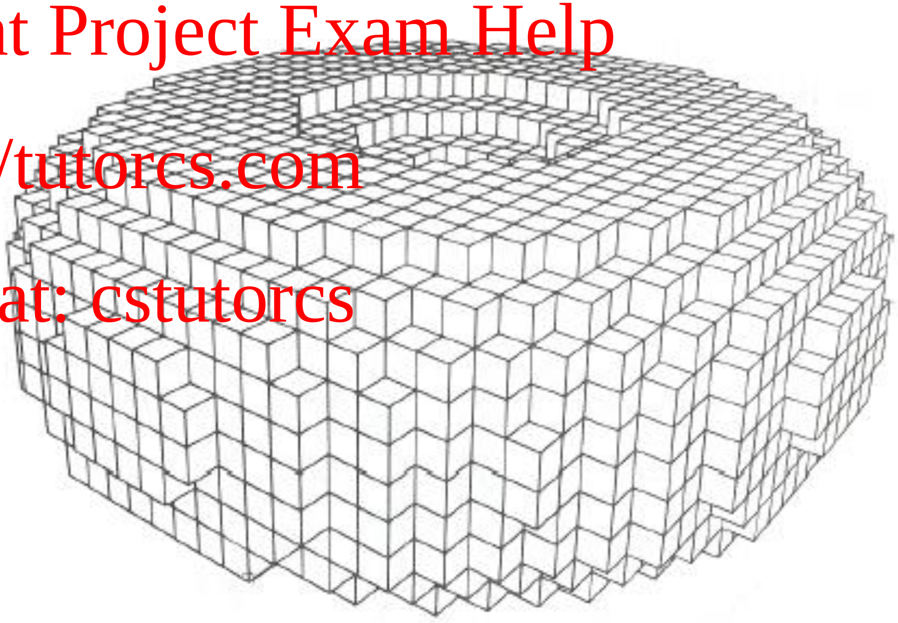
$$(\phi,\theta) \in [0,\pi]\times[0,2\pi]$$



- Maps each (ϕ,θ) on a point on the unit sphere (but be careful at the north and south poles)
- More details see sphere.java in the labs…

Volumetric Representation: Voxels

- Partition space into uniform 3D grid
  - Grid cells are called voxels (volume elements) (also see pixels)
- Store *properties* of solid object with each voxel
  - Occupancy
  - Colour
  - Density
  - Temperature
  - . . .



FvDFH Figure 12.20

Voxel Examples

Visible Human

(National Library of Medicine)

SUNY Stoney Brook

Voxel Issues

- Advantages:
  - Simple inside/outside test
  - Simple and robust boolean operations
  - Represent interior of the object
- Disadvantages:
  - Memory consuming

  (can use octree for hierarchical construction to save memory)
  - Non-smooth
  - Time consuming to manipulate and render

Summary

- Explain the following model representations:
  - constructive solid geometry
  - boundary representation
  - mesh representation
  - volumetric representation
  - How is the model represented?
  - Which data structures are used?
  - What are advantages/disadvantages of these representations?
- What is a simple / convex / flat polygon?
- What do we understand by the orientation of a polygon/loop/edge?