3D Graphics Programming Tools

theory and technical methods

Dr. Pengwei Hao
(p.hao@qmul.ac.uk)

Queen Mary
University of London

Learning Objectives

At the end of the course, you will be able to:

LO1. Understand 3D Computer Graphics' fundamental mathematical and computational principles (theory);

LO2. Describe rendering techniques for the creation of 3D Computer Graphics (technical theory);

LO3. Apply OpenGL programming principles (practical basics);

LO4. Generate and comment OpenGL code (practice);

LO5. Implement 3D graphics animations using a variety of programming tools (practical advanced)

Teaching Units

- Practical
  - Unit 1 OpenGL Basics
  - Unit 2 OpenGL Production
- Theoretical
  - Unit 3 Modelling, Transformations, Colours
  - Unit 4 Projection, Rasterisation

Queen Mary
University of London

Teaching Schedule

- The timetable (Nov-Dec)

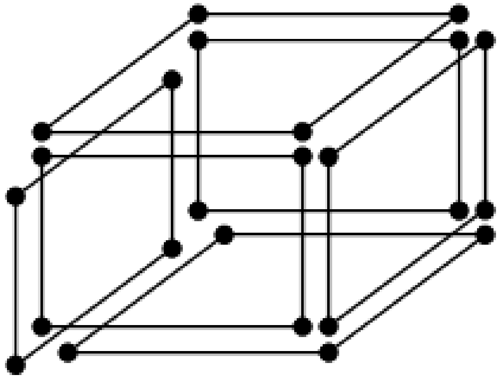
10	11	12	13	14	15	Class Days, Times and Room Number (3-437)			
01-Nov	08-Nov	15-Nov	22-Nov	29-Nov	06-Dec				
PH	PH	CS	PH	PH	CS	Telecom_M_G1		Telecom_M_G2	
Rec	Rec	Live	Rec	Rec	Live	Mon	16:35-17:20	Mon	19:20-20:05
Rec	Rec	Live	Rec	Rec	Live		17:25-18:10		20:10-20:55
Rec	Live	Live	Rec	Live	Live	Thu	16:35-17:20	Wed	19:20-20:05
Rec	OH	Live	Rec	OH	Live		17:25-18:10		20:10-20:55
	L3			L4					
					CW				

- Room: 3-437
- 1 hour live revision before 1 hour office hour
- Revision is on my taught materials of the unit
- Tutorials are delivered lively by Dr Chao Shu

Queen Mary
University of London

Content Overview : this unit

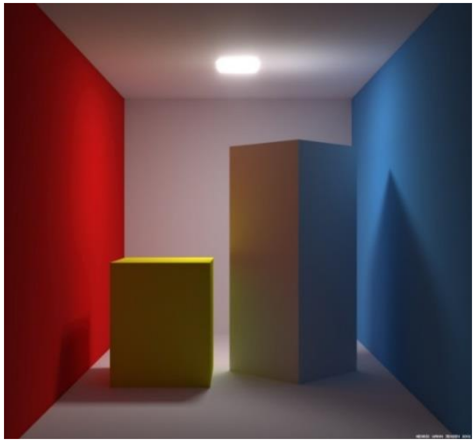
OBJECT MODELLING (LO1, LO2)

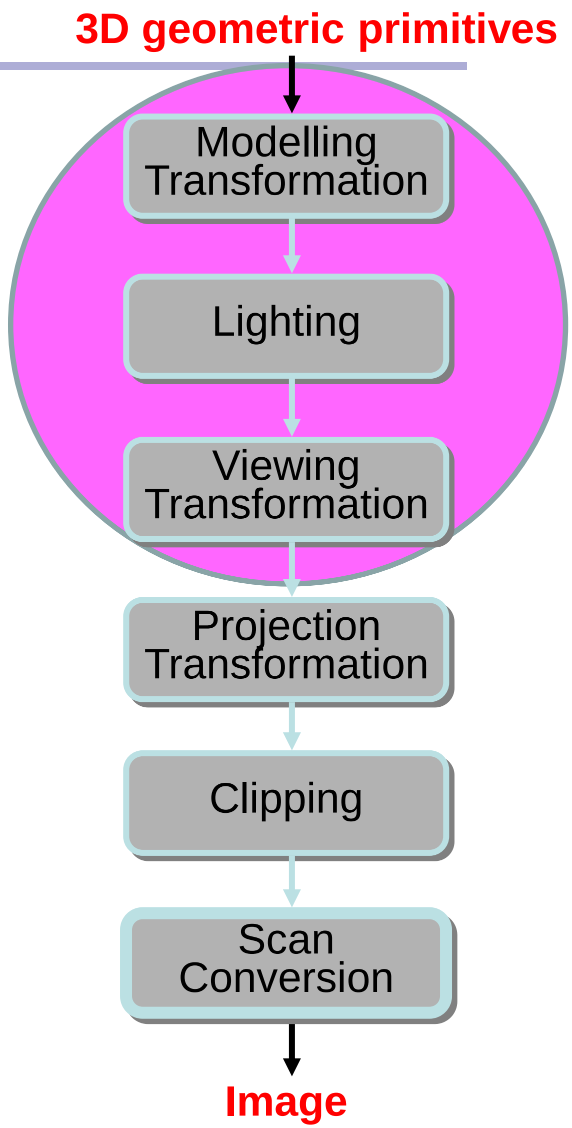


GEOMETRIC TRANSFORMS (modelling and viewing) (LO1, LO2)

COLOUR (LO1, LO2)

LIGHTING (LO1, LO2)





Queen Mary
University of London

3D Graphics Programming Tools

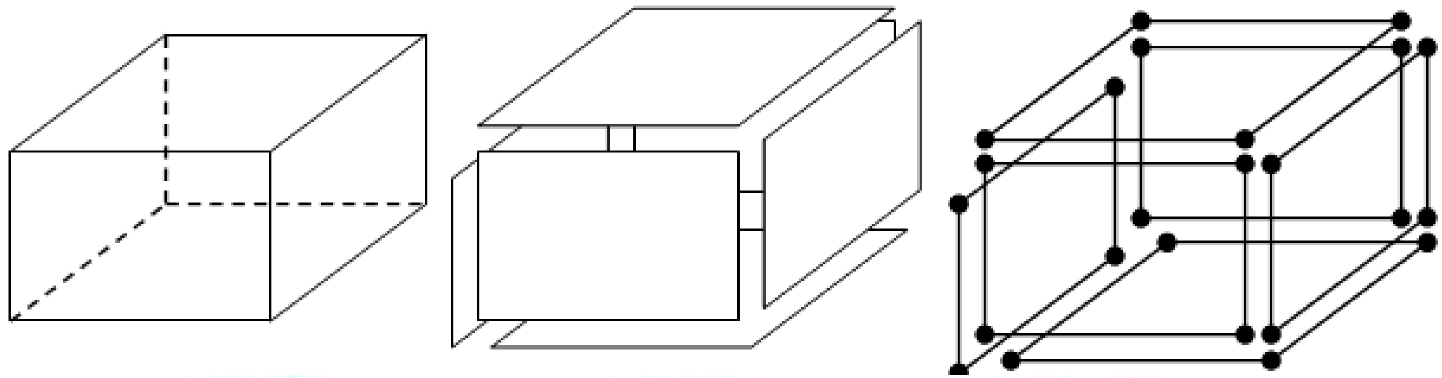
Object modelling

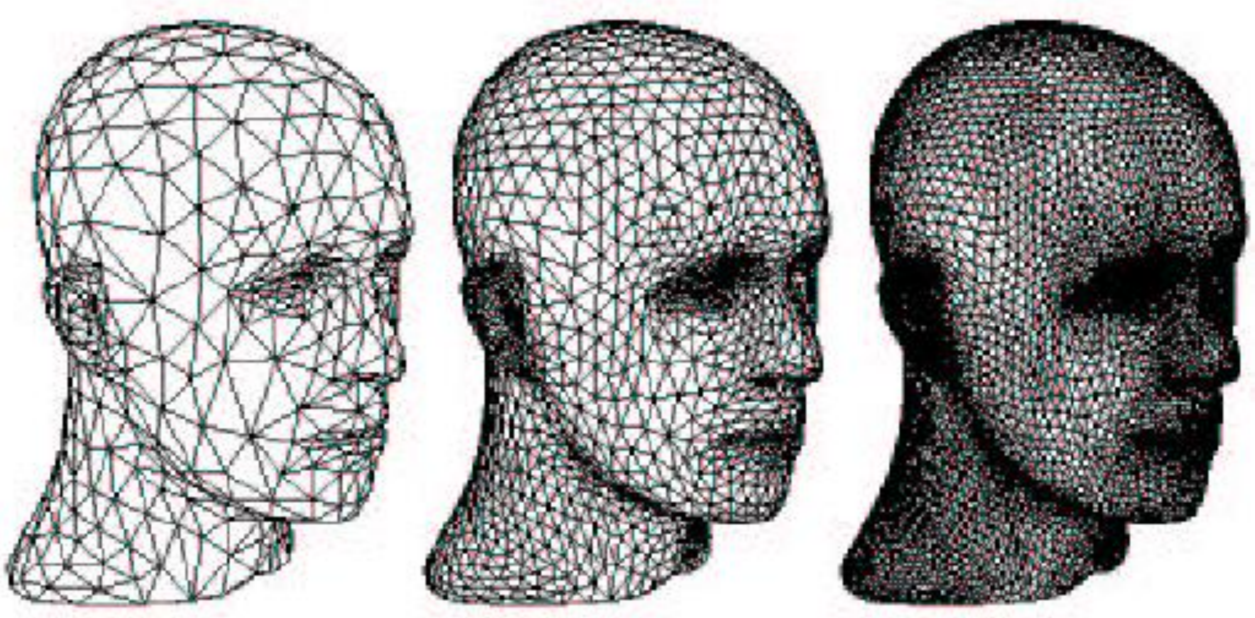
What do we mean by a solid object?

- finite
- three dimensional
- rigid
- closed
- finitely describable
- with a determinable boundary

Modelling

The generation of abstract descriptions of 3D objects is called geometric modelling.





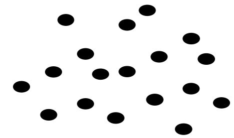
3D Graphics

Queen Mary
University of London

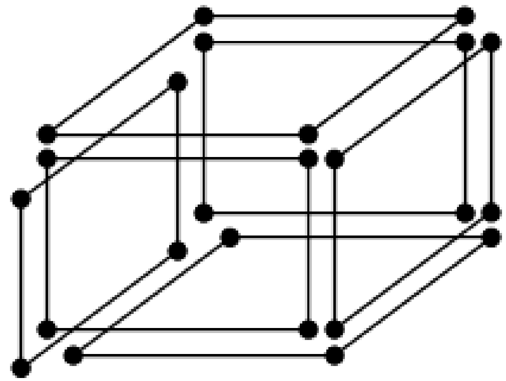
3D Graphics

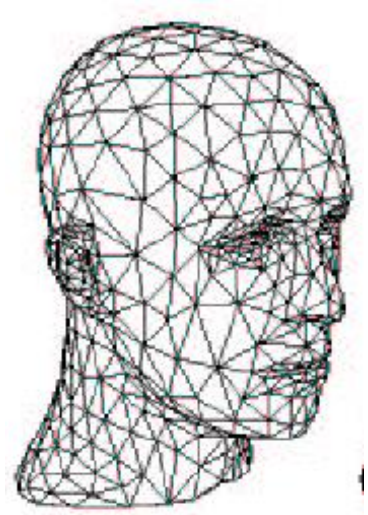
Geometric modelling

- Point-based

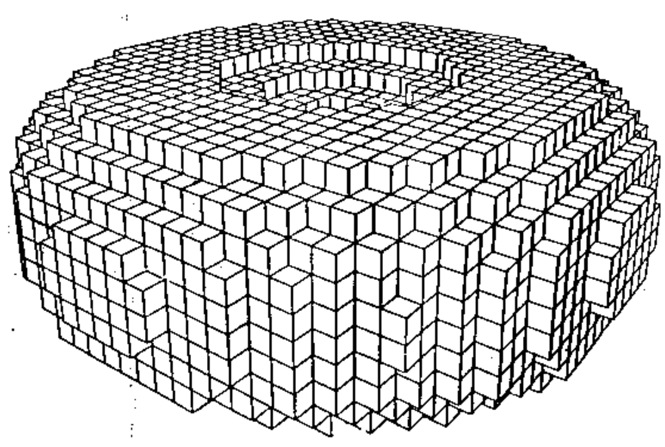


- Surface-based



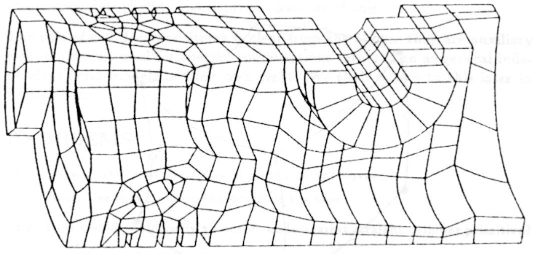


- Constructive

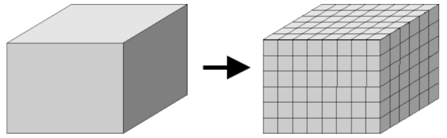


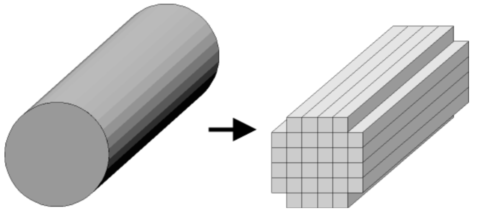
Spatial-Partitioning Representations

- **Cell decomposition** (voxels)

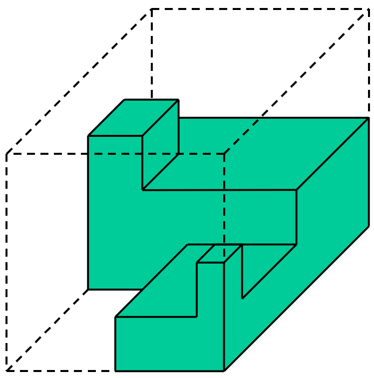


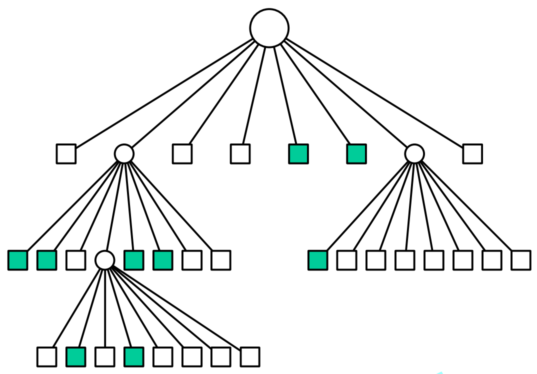
- **Spatial occupancy enumeration**



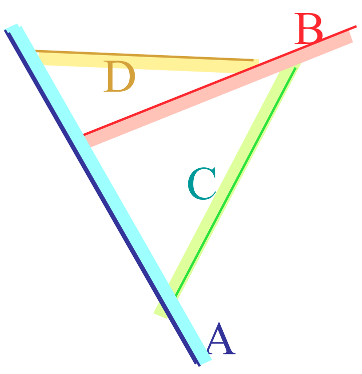


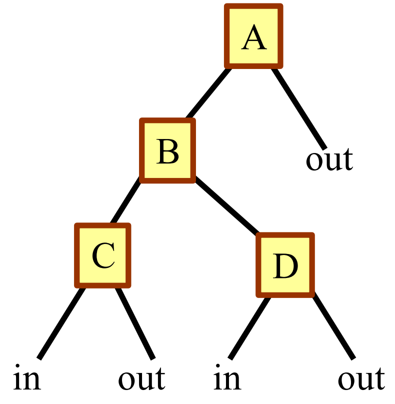
- **Octrees**



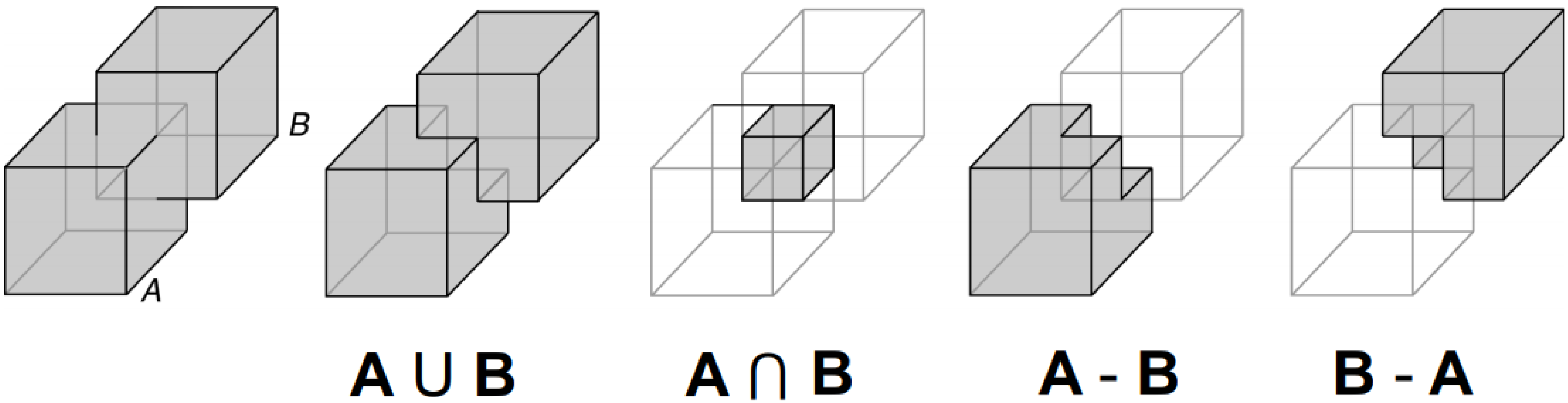


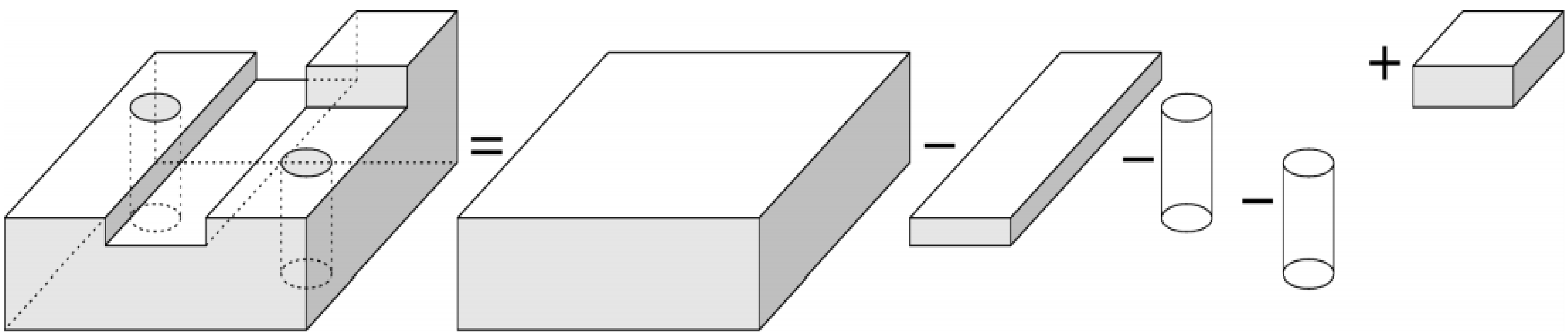
- **Binary Space Partition**





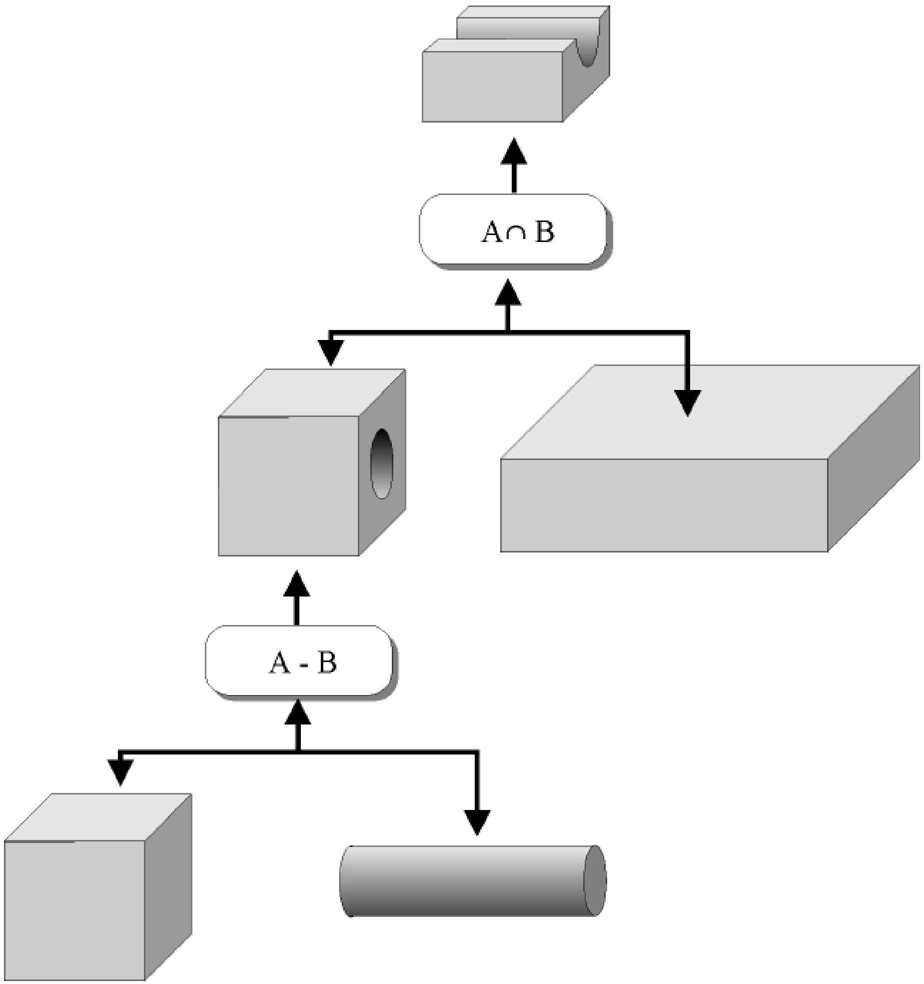
Boolean Set Operations

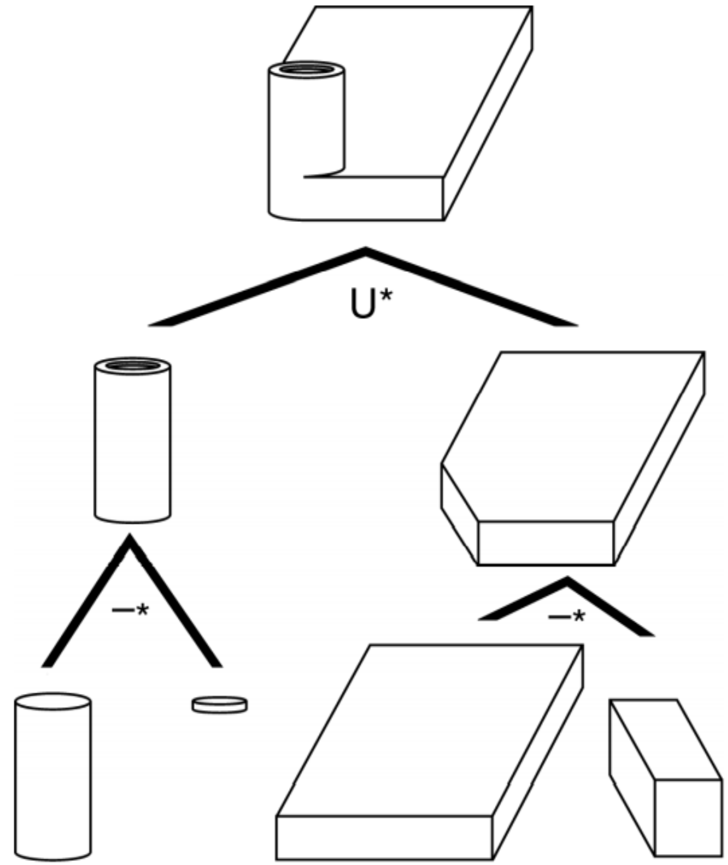




Queen Mary
University of London

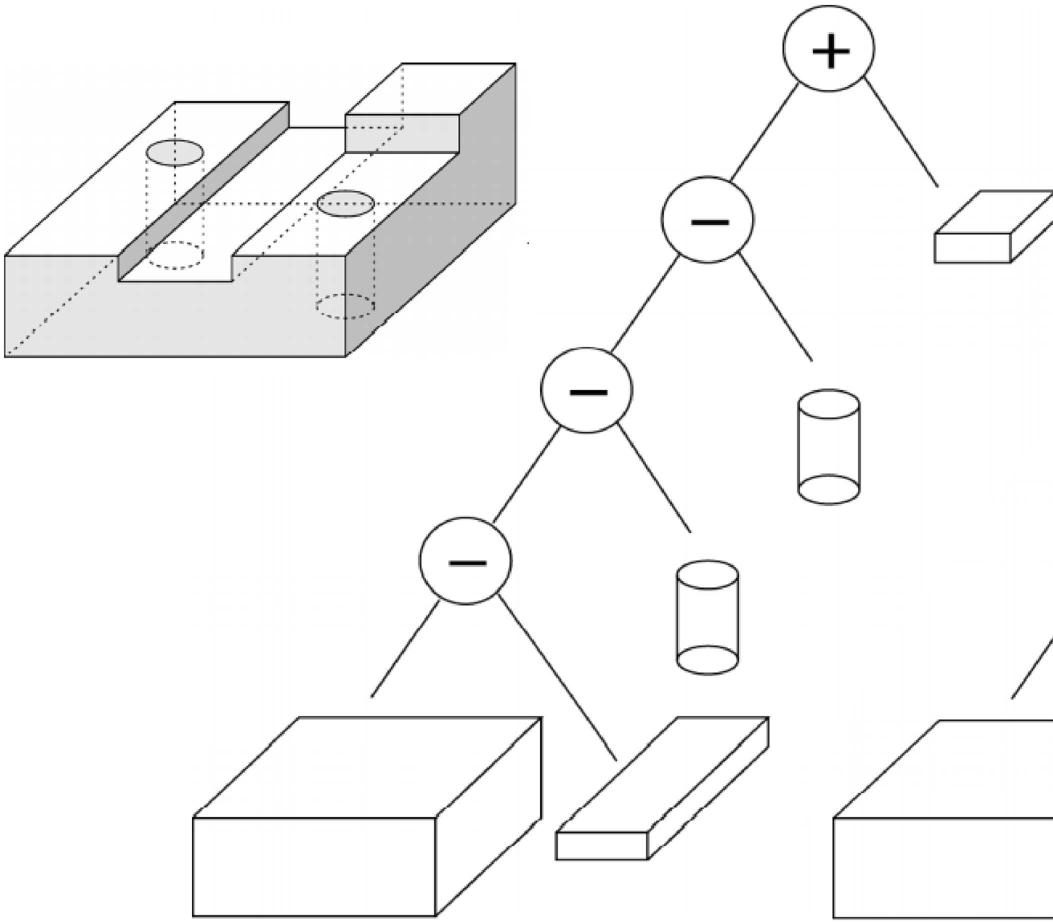
Constructive Solid Geometry (CSG)



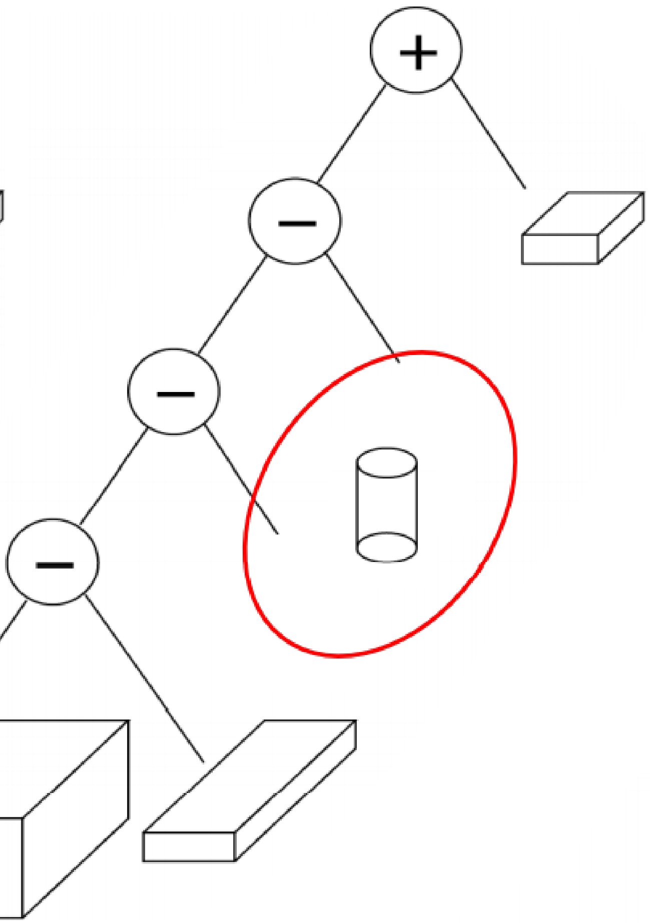


Queen Mary
University of London

CSG Tree and DAG



The induced binary CSG tree

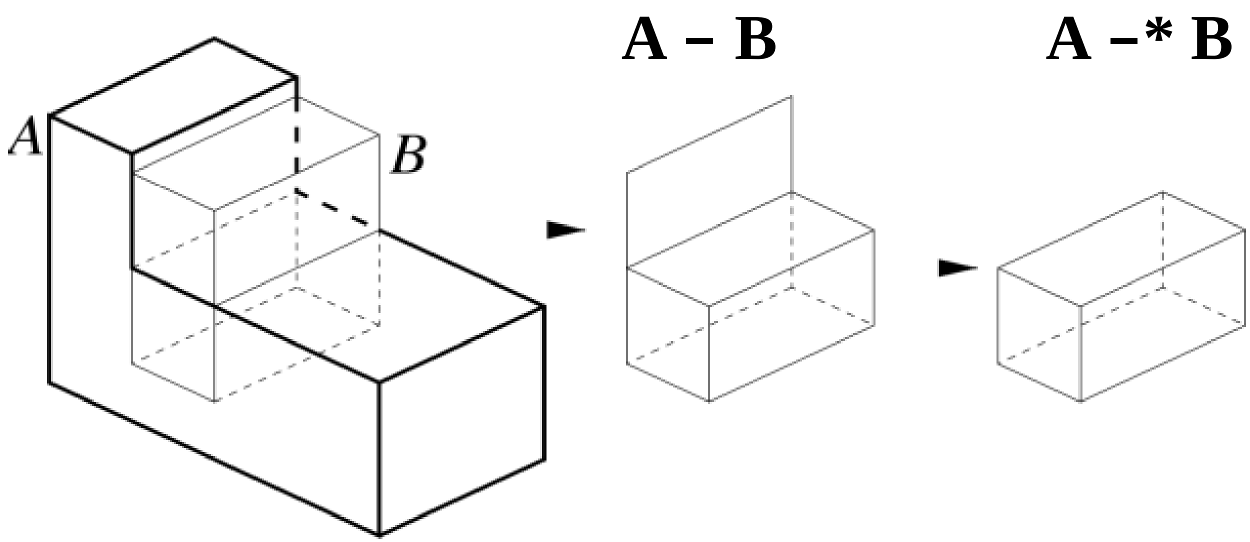


Directed acyclic graph (DAG)

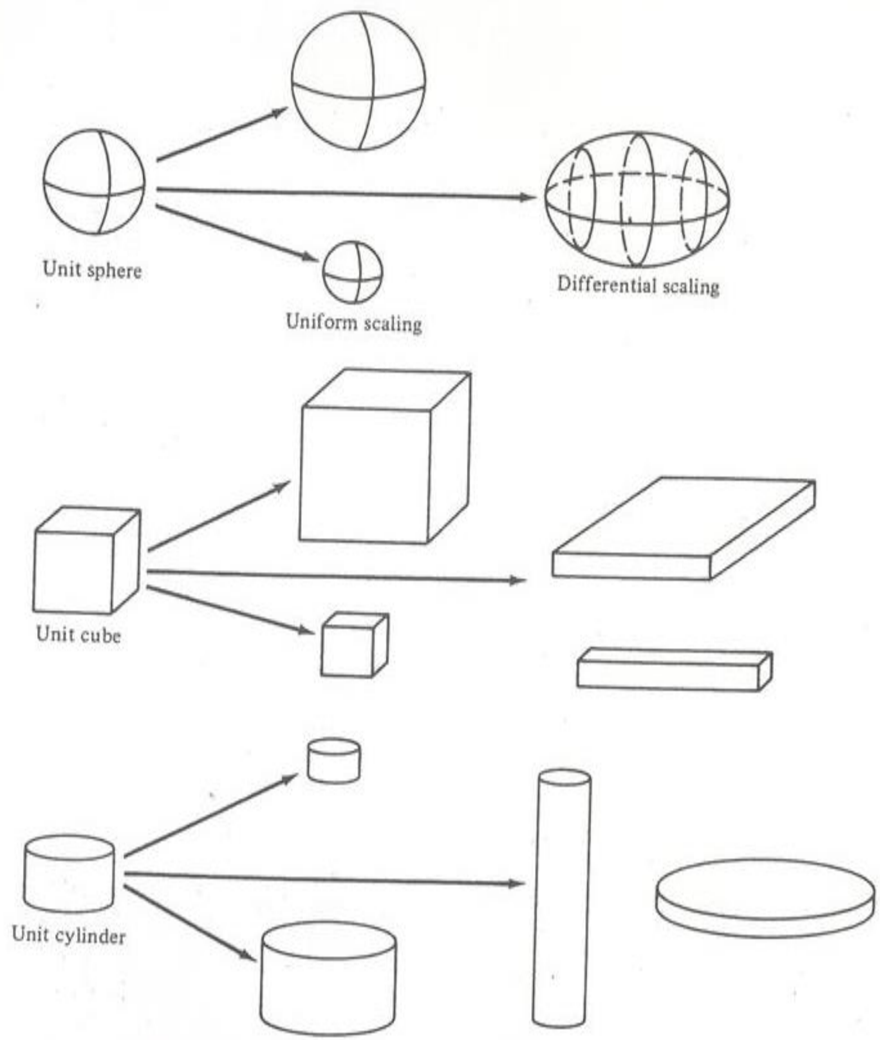
Queen Mary
University of London

Regularized Boolean Set Operations

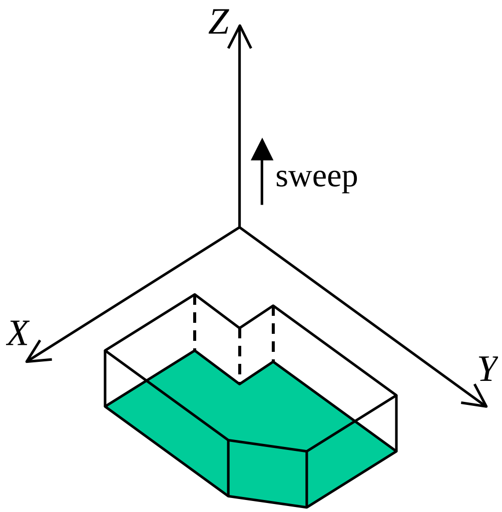
After Boolean set operations between solid objects, we need to remove the dangling faces (regularized)

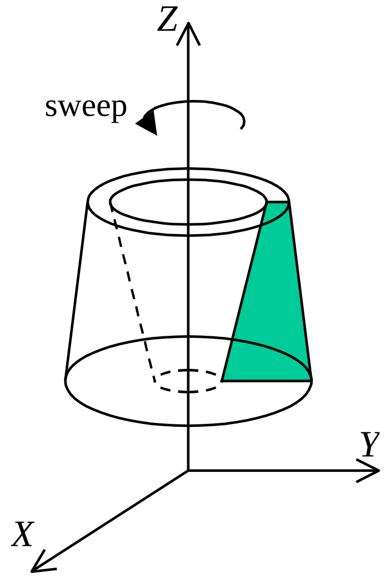


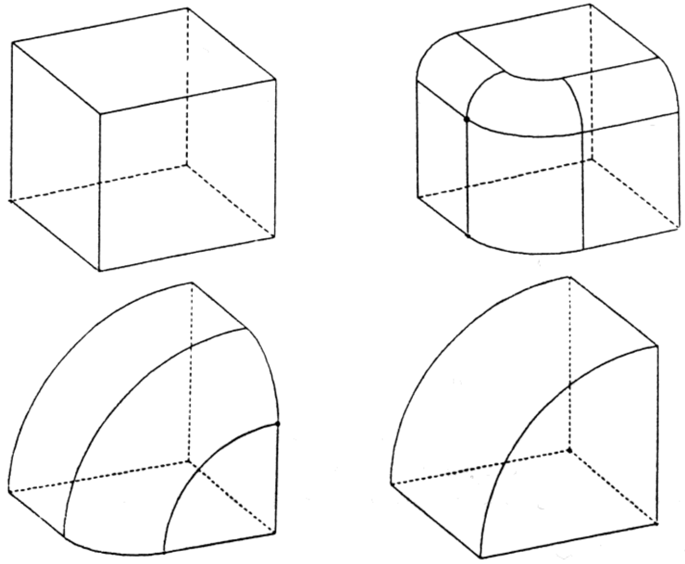
Primitive instancing



Sweep Representations

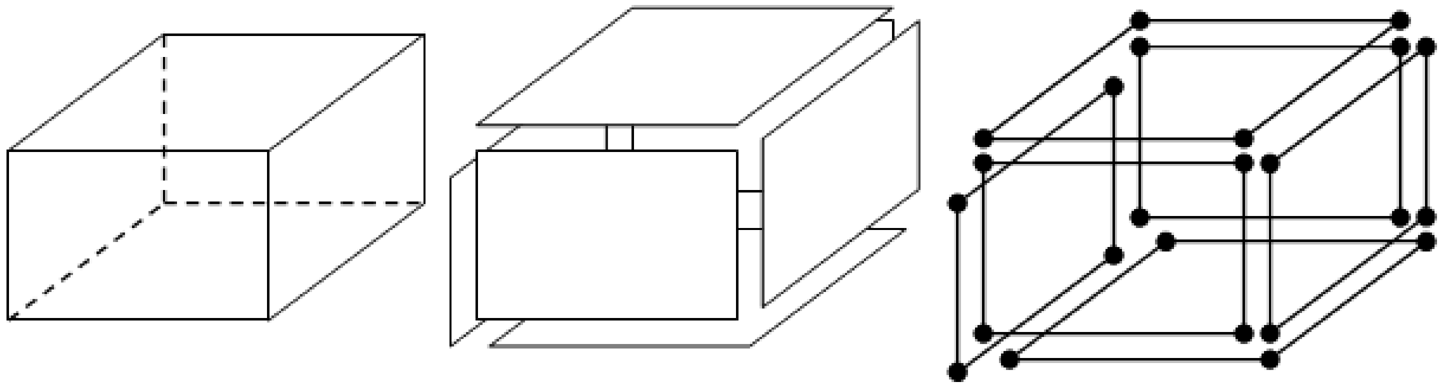




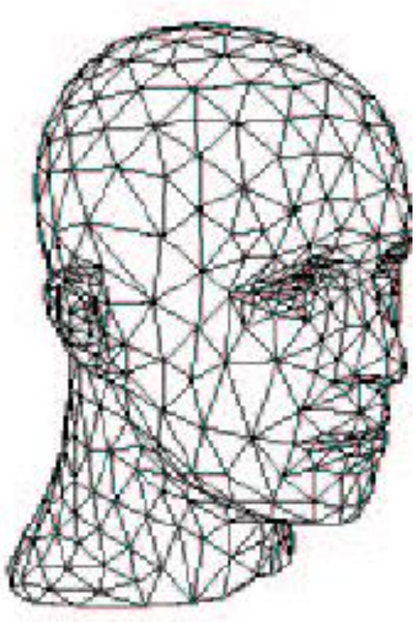


Surface modelling

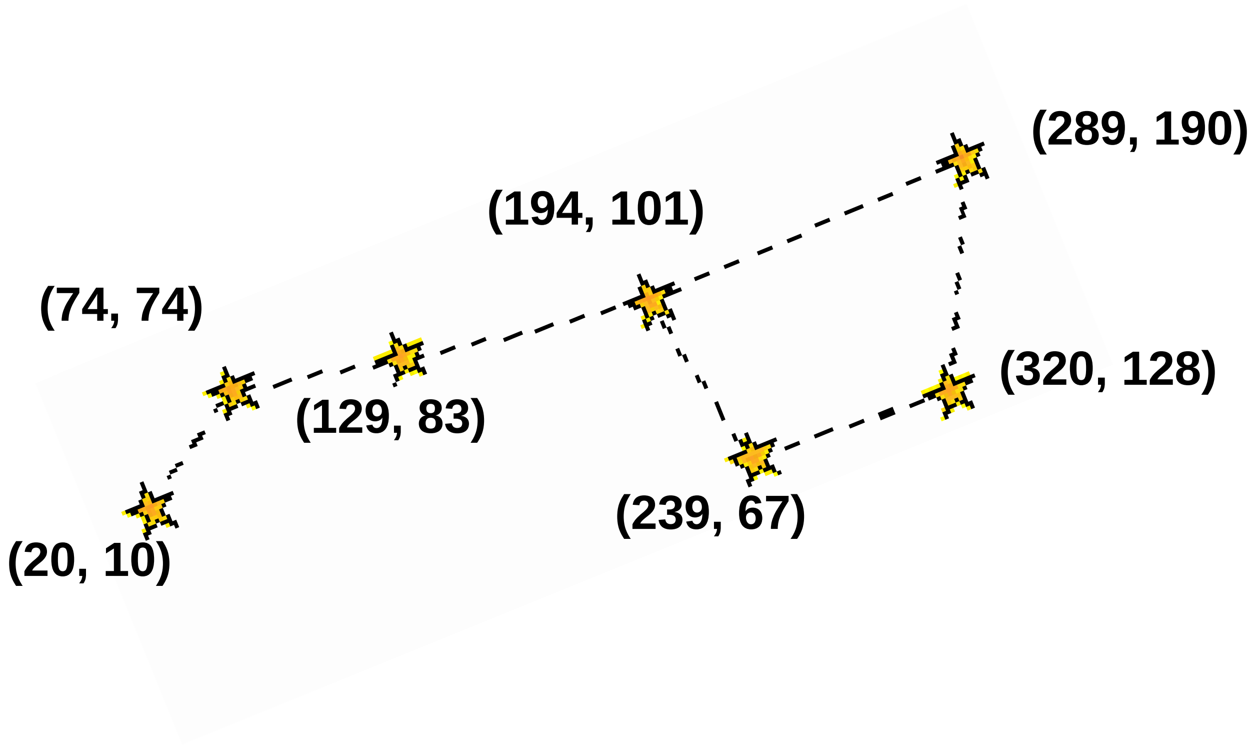
Boundary representations (b-reps)



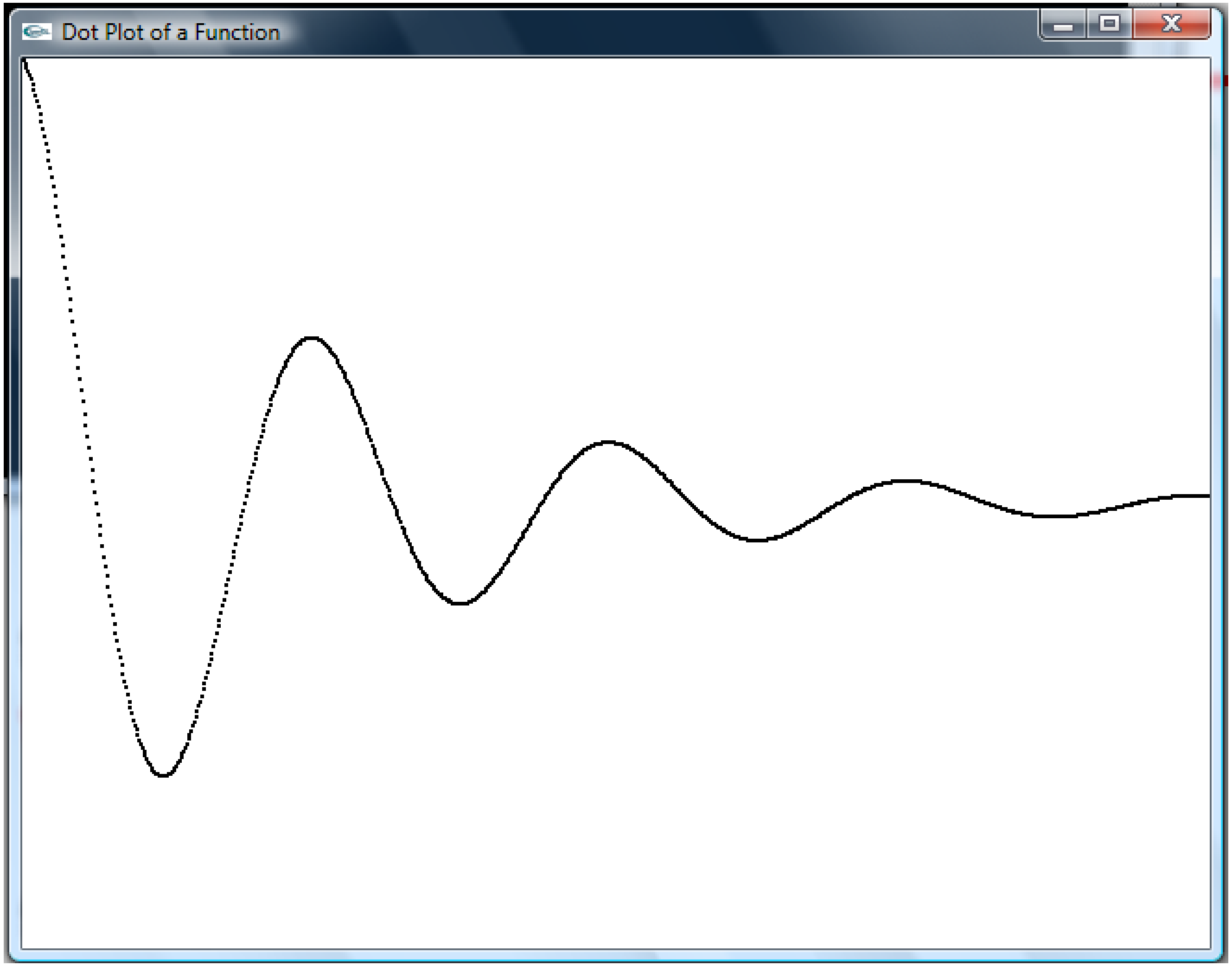
Polygon mesh



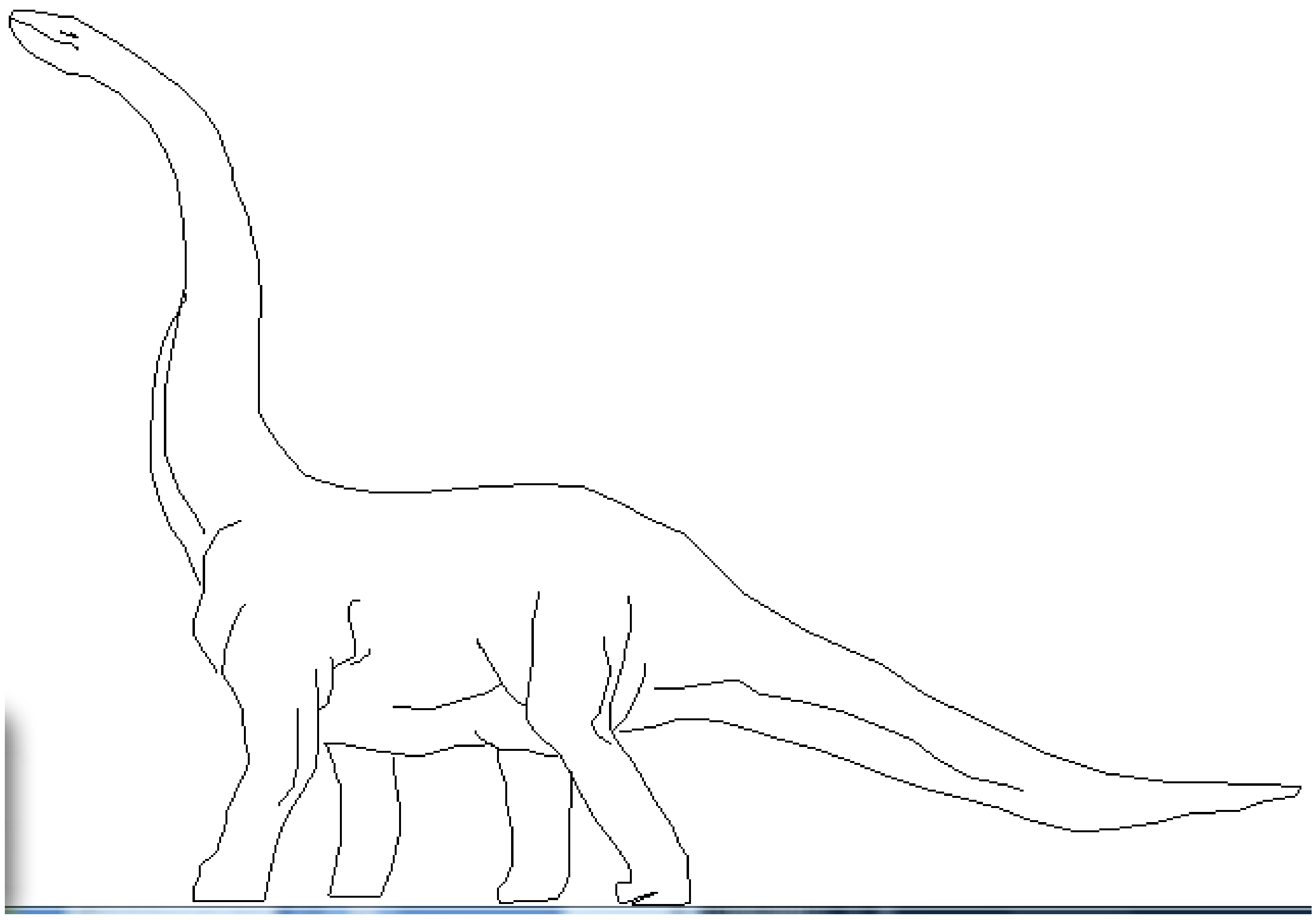
Point-based: the Big Dipper



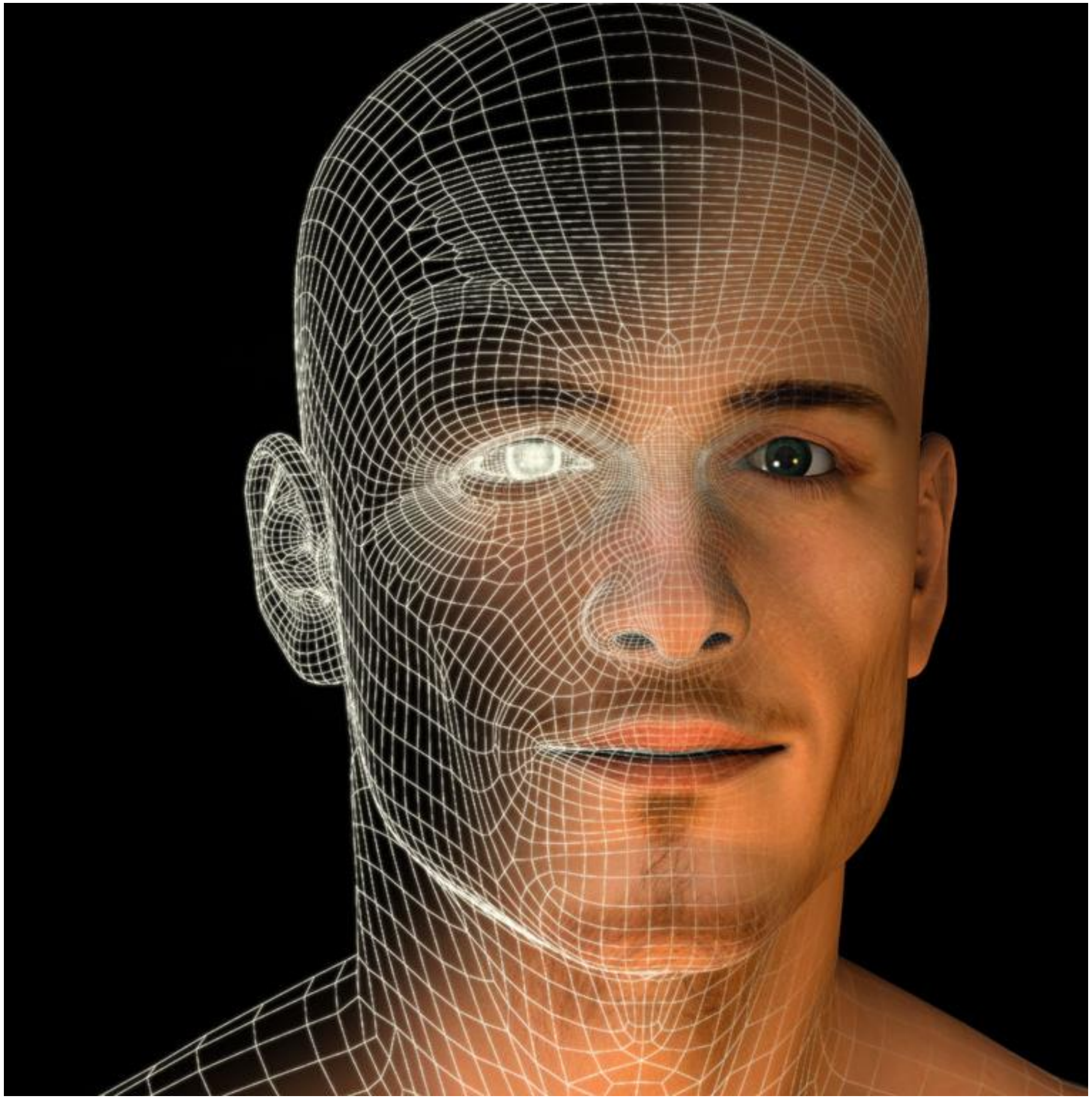
Dot plots



Drawing from a file



Polygon mesh

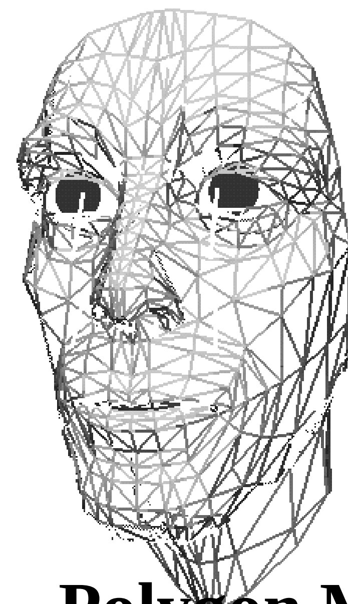


Queen Mary
University of London

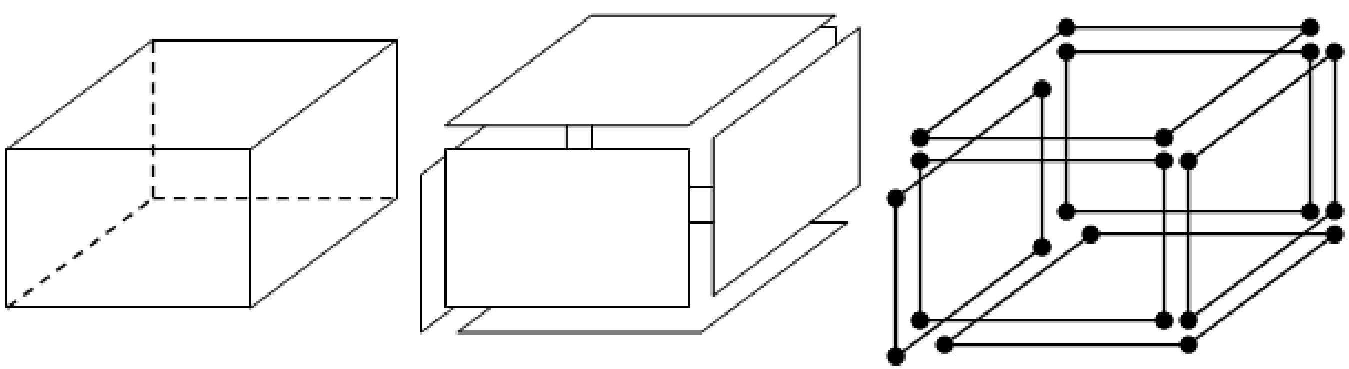
Representing polygon meshes

Polygon mesh (or “POLYHEDRON”):

- POLYHEDRON = geometric object with flat faces and straight edges
- Collection of polygons (faces), which are together connected to form the skin of the object
  - Faces → the boundary of solid objects/spaces
  - Edges → the boundary of faces
  - Vertices → the boundary of edges,
    or where three or more faces meet
  - Normals, texture coordinates, colours,
    shading coefficients, etc



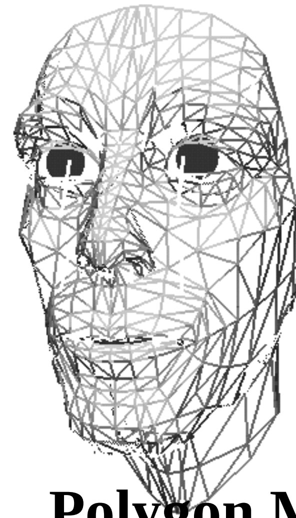
Polygon Mesh



Queen Mary
University of London

Why polygons?

- They are:
  - Easy to represent
  - Easy to transform
  - Easy to draw
- Especially if they are:
  - Flat
  - Convex
  - Simple

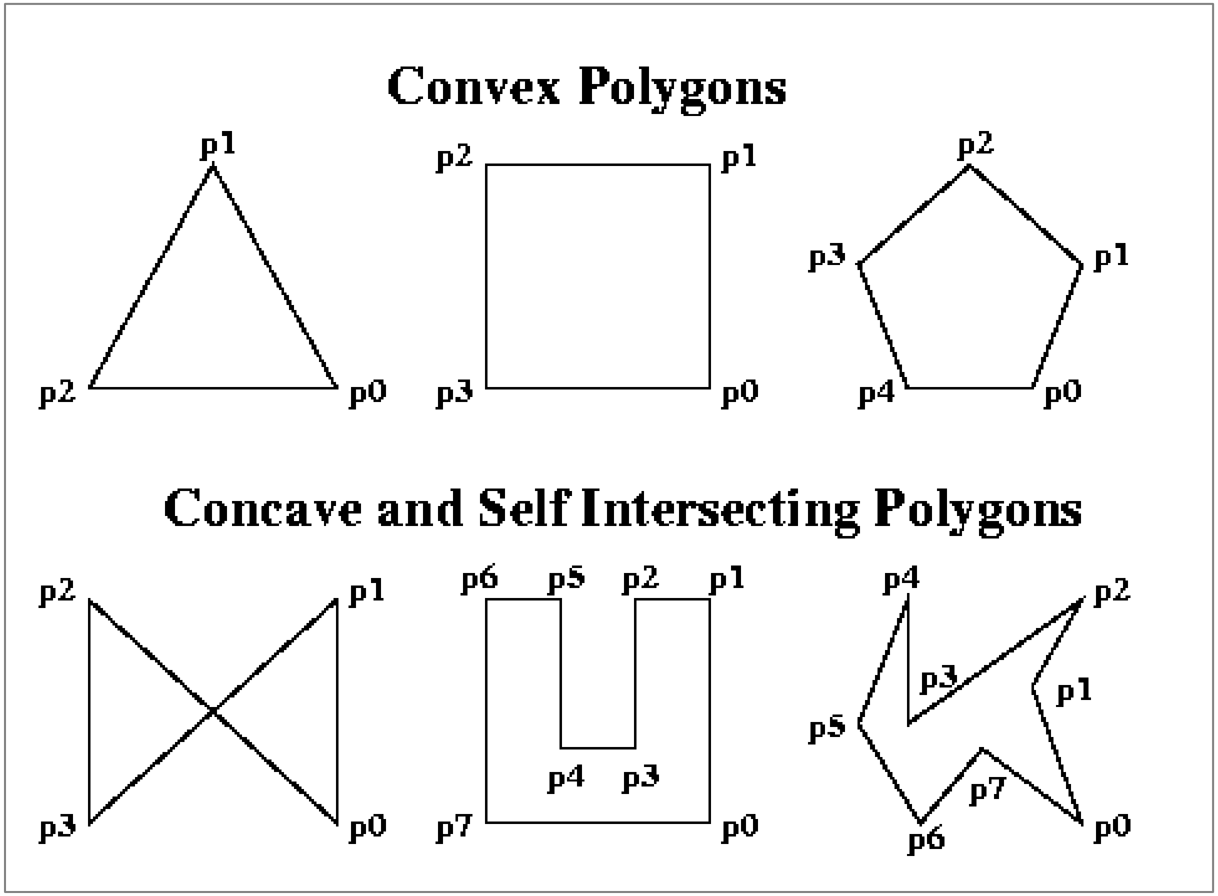


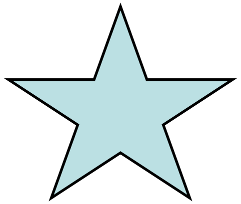
Polygon Mesh

Polygon Issues

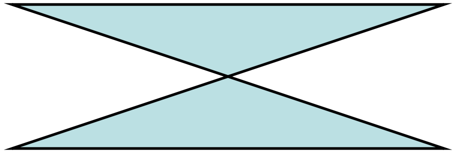
- OpenGL will only display polygons correctly that are
  - Simple: edges cannot cross
  - Convex: All points on line segment between two points in a polygon are also in the polygon
  - Flat: all vertices are in the same plane
- User program must check if above true
  - OpenGL will produce output if these conditions are violated but it may not be what is desired
- Triangles satisfy all conditions

Polygon Issues





nonconvex polygon

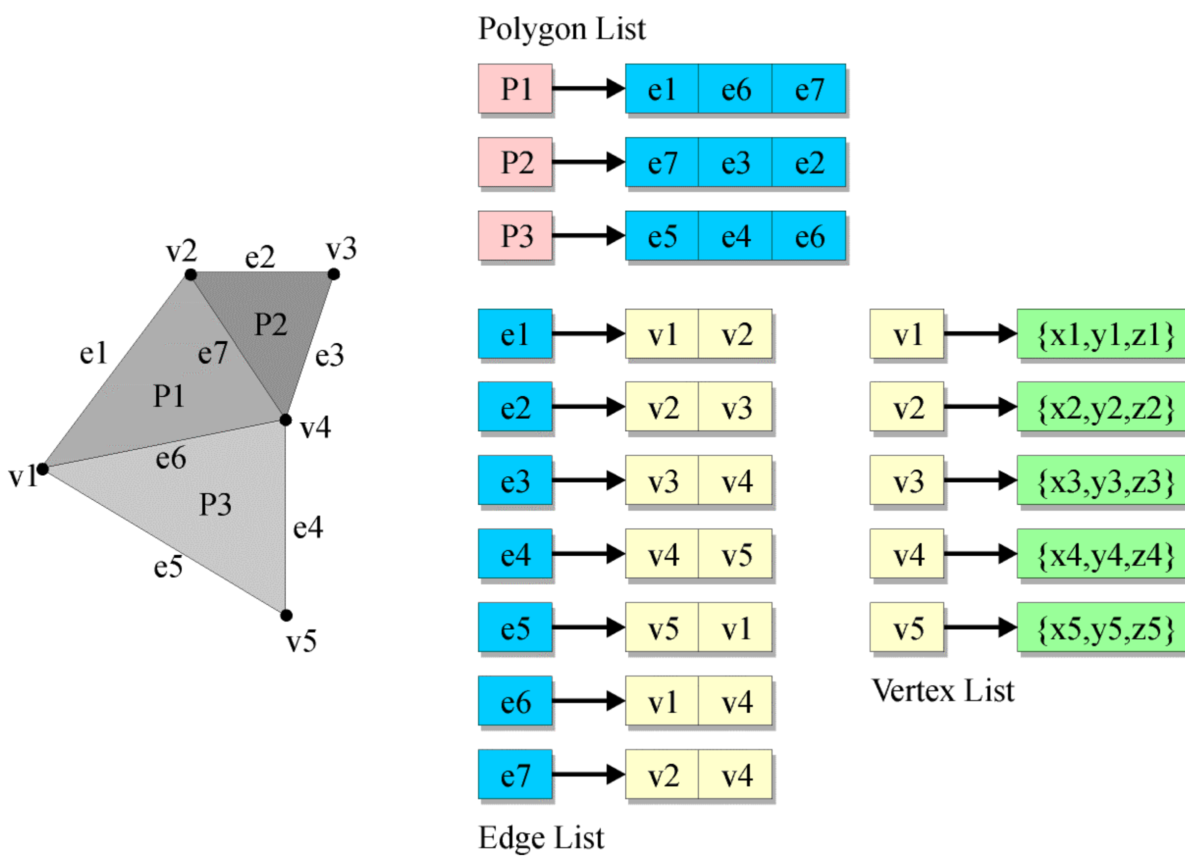


nonsimple polygon

Queen Mary
University of London

Representing polygon meshes

- Vertex list → locations of the vertices, <u>geometric</u> info
- Edge list → indexes into end vertices of edges, <u>topological</u> info
- Face list → indexes into vertices and normal lists, <u>topological</u> info
- Normal list → directions of the normal vectors, <u>orientation</u> info

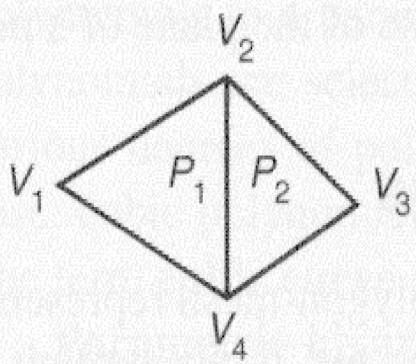


Queen Mary
University of London

Representing polygon meshes

- Euler's Formula: V – E + F = 2
  - V: # of vertices
  - E: # of edges
  - F: # of faces

- Vertex list and face list are enough



$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \ldots, (x_4, y_4, z_4))$$

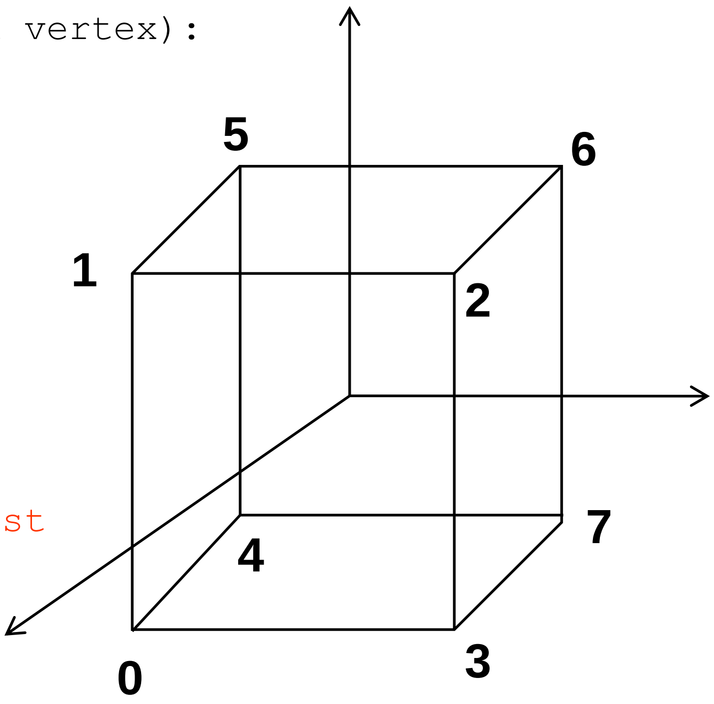
$$P_1 = (1, 2, 4)$$

$$P_2 = (4, 2, 3)$$

Polygon mesh example

A cube:

```
Coordinates of Vertices (3D, x, y, and z for each vertex):
-1 , -1 ,  1  # Vertex 0
-1 ,  1 ,  1  # Vertex 1
 1 ,  1 ,  1  # Vertex 2
 1 , -1 ,  1  # Vertex 3
-1 , -1 , -1  # Vertex 4
-1 ,  1 , -1  # Vertex 5
 1 ,  1 , -1  # Vertex 6
 1 , -1 , -1  # Vertex 7
```




```
Lists of 6 Faces (vertices are referenced to the
vertices above, -1 marks the end of the vertex list
of a face) :
0, 3, 2, 1, -1   # Front face=vertex 0,1,2,3
2, 3, 7, 6, -1   # Right side
3, 0, 4, 7, -1   # Bottom
1, 2, 6, 5, -1   # Top
4, 5, 6, 7, -1   # Back
5, 4, 0, 1, -1   # Left
```

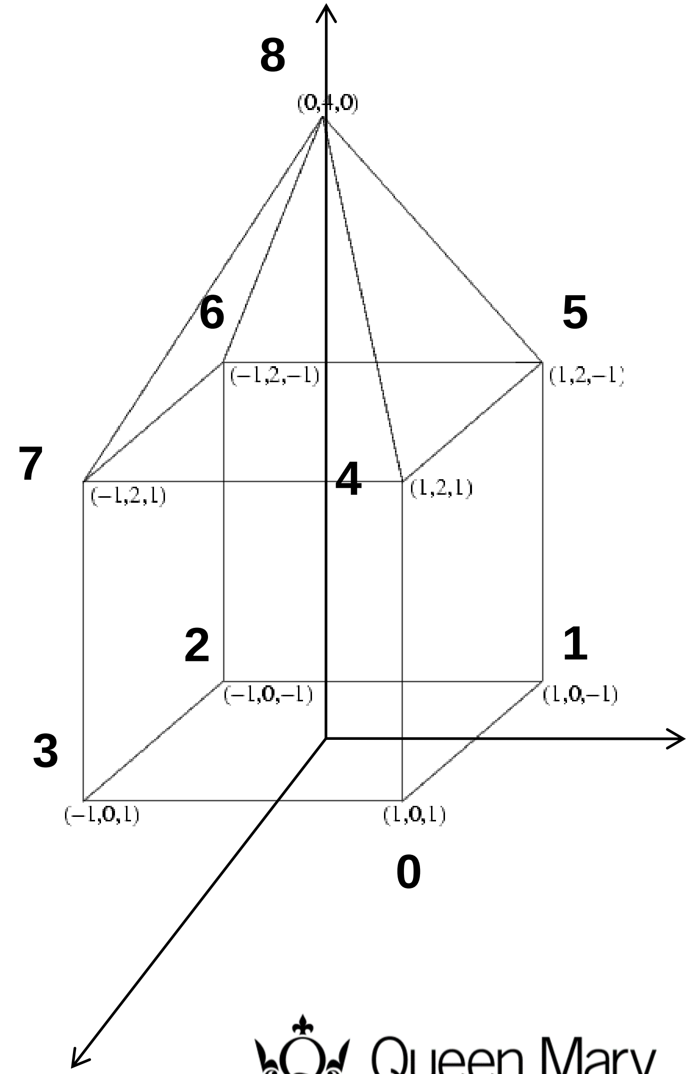
Polygon mesh example

A House:

Coordinates of Vertices:

```
 1   0    1     # Vertex 0
 1   0   -1     # Vertex 1
-1   0   -1     # Vertex 2
-1   0    1     # Vertex 3
 1   2    1     # Vertex 4
 1   2   -1     # Vertex 5
-1   2   -1     # Vertex 6
-1   2    1     # Vertex 7
 0   4    0     # Vertex 8
```

List of Faces?



Polygon mesh example

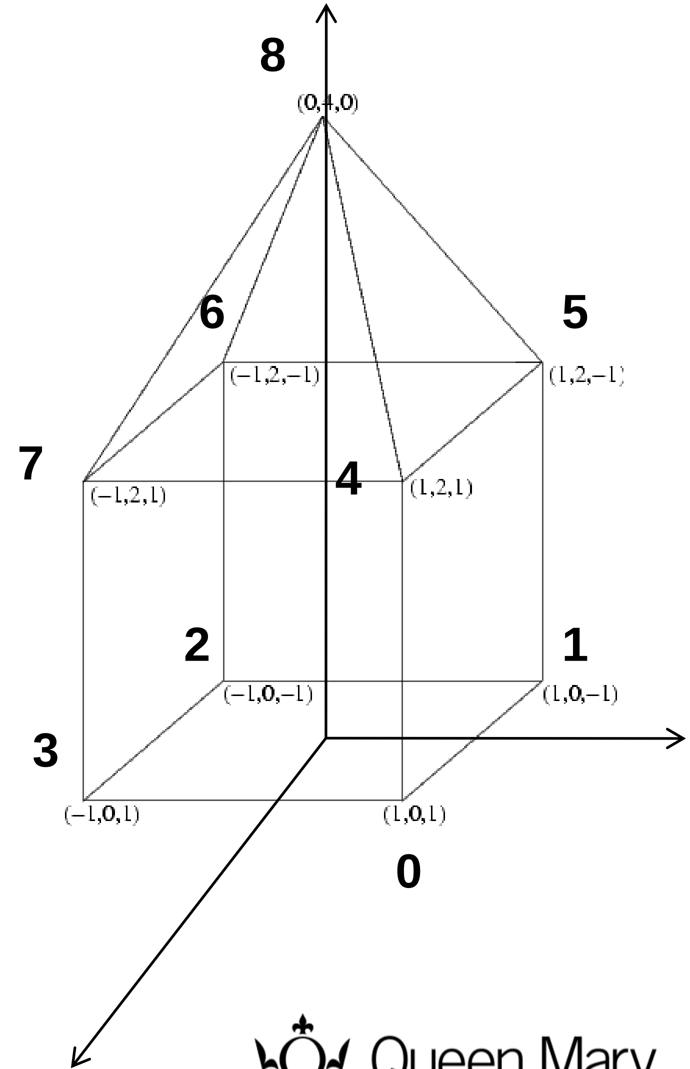
A House:

Coordinates of Vertices:

```
 1   0    1      # Vertex 0
 1   0   -1      # Vertex 1
-1   0   -1      # Vertex 2
-1   0    1      # Vertex 3
 1   2    1      # Vertex 4
 1   2   -1      # Vertex 5
-1   2   -1      # Vertex 6
-1   2    1      # Vertex 7
 0   4    0      # Vertex 8
```

List of Faces:

```
0, 1, 2, 3, -1
0, 1, 5, 4, -1
1, 2, 6, 5, -1
2, 3, 7, 6, -1
3, 0, 4, 7, -1
4, 5, 8, -1
5, 6, 8, -1
6, 7, 8, -1
7, 4, 8, -1
```



Queen Mary University of London

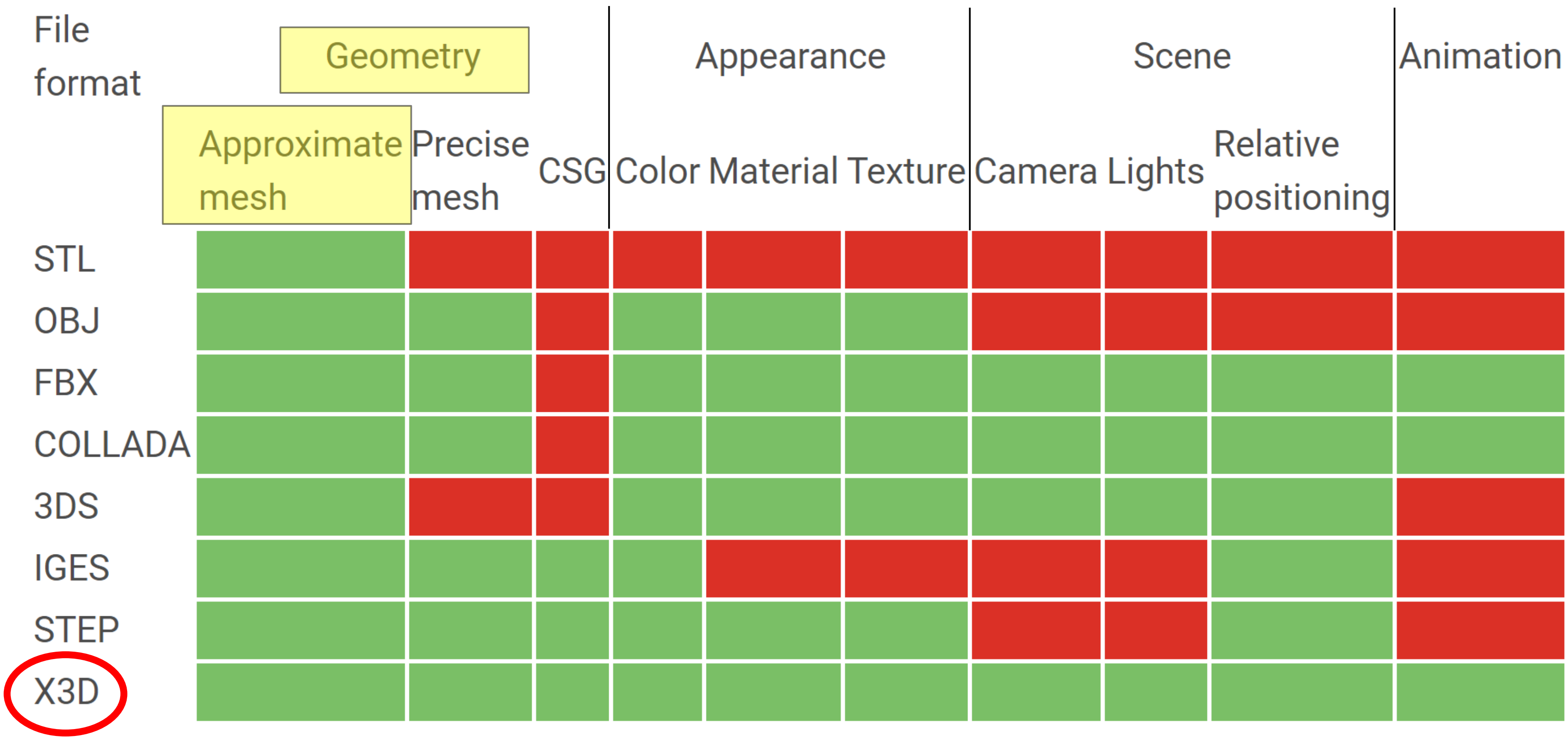
3D File Formats

- The basic purpose of a 3D file format is to store information about 3D models as plain text or binary data.
- There are hundreds of 3D file formats currently being used!
- Software manufacturers such as AutoDesk and Blender have their own proprietary format which is optimized for their software.
- To solve the problem of interoperability, *neutral* or *open source* formats were invented as intermediate formats for converting between two proprietary formats.
- Some popular formats: STL, OBJ, FBX, COLLADA, VRML and X3D, etc.

Queen Mary
University of London

3D File Formats

Green indicates *supported*, red indicated *not supported*



Queen Mary
University of London

STL (STereoLithography)

- STL is one of the most important neutral 3D file formats in the domain of 3D printing, rapid prototyping, and computer aided manufacturing.
- STL encodes the surface geometry of a 3D model approximately using a triangular mesh.

OBJ

- The OBJ file format is another neutral format widely used in the fields of 3D printing and 3D graphics.
- The OBJ file format supports both approximate and precise encoding of surface geometry (i.e. smooth curves and surfaces such as NURBS: *Non-Uniform Rational B-Spline)*, instead of polygons.
- When using the approximate encoding, it doesn't restrict the surface mesh to triangular facets (users can use polygons like Quadrilaterals).

FBX

- FBX is a proprietary file format which is widely used in the film industry and video games.
- Used as an exchange format which facilitates high fidelity exchange between 3DS Max, Maya, MotionBuilder, Mudbox and other proprietary software.

COLLADA

- Collada is a neutral file format used heavily in the video game and film industry.
- The file extension for the COLLADA format is .DAE
- The COLLADA format supports geometry, appearance related properties like color, material, textures, and animation.
- The COLLADA format stores data using the XML (Extensible Markup Language).

Queen Mary
University of London

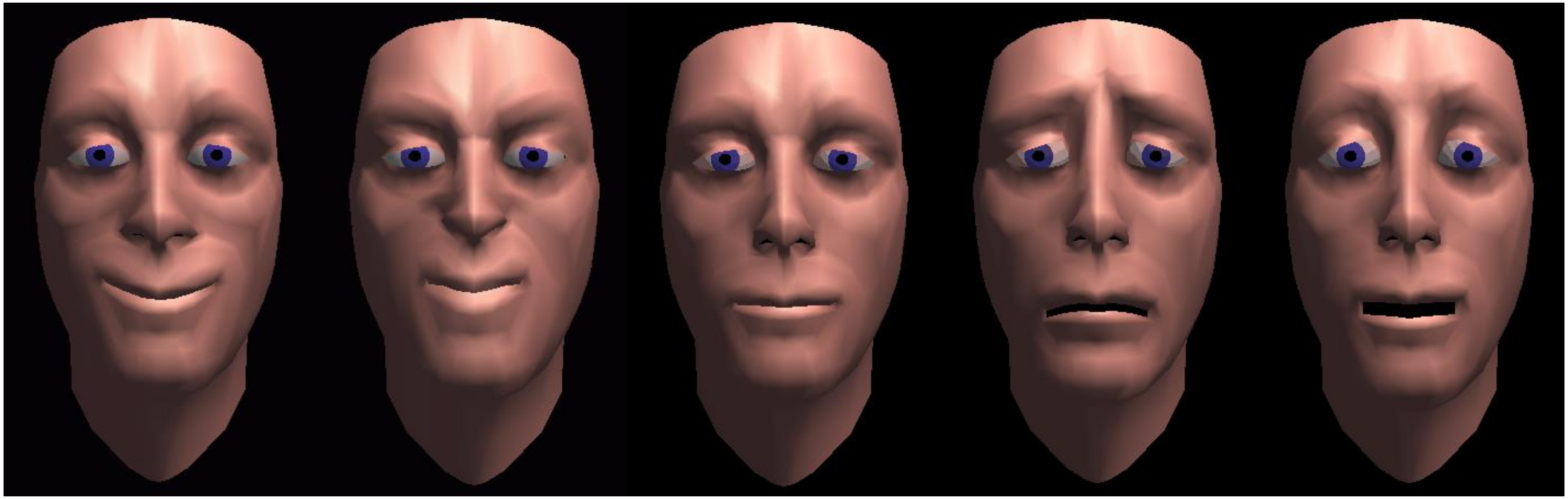
VRML and X3D

- VRML stands for Virtual Reality Modelling Language.
- It is a 3D file format that was developed for the web.
- The VRML format uses a polygonal mesh to encode surface geometry and can store appearance related information such as color, texture, transparency etc.
- It has been succeeded by X3D.

- X3D is an XML-based 3D file format.
- The X3D format adds NURBS encoding of the surface geometry, the capability of storing scene related information and support for animation.
- The goal of X3D is to become the standard 3D file format for the web.

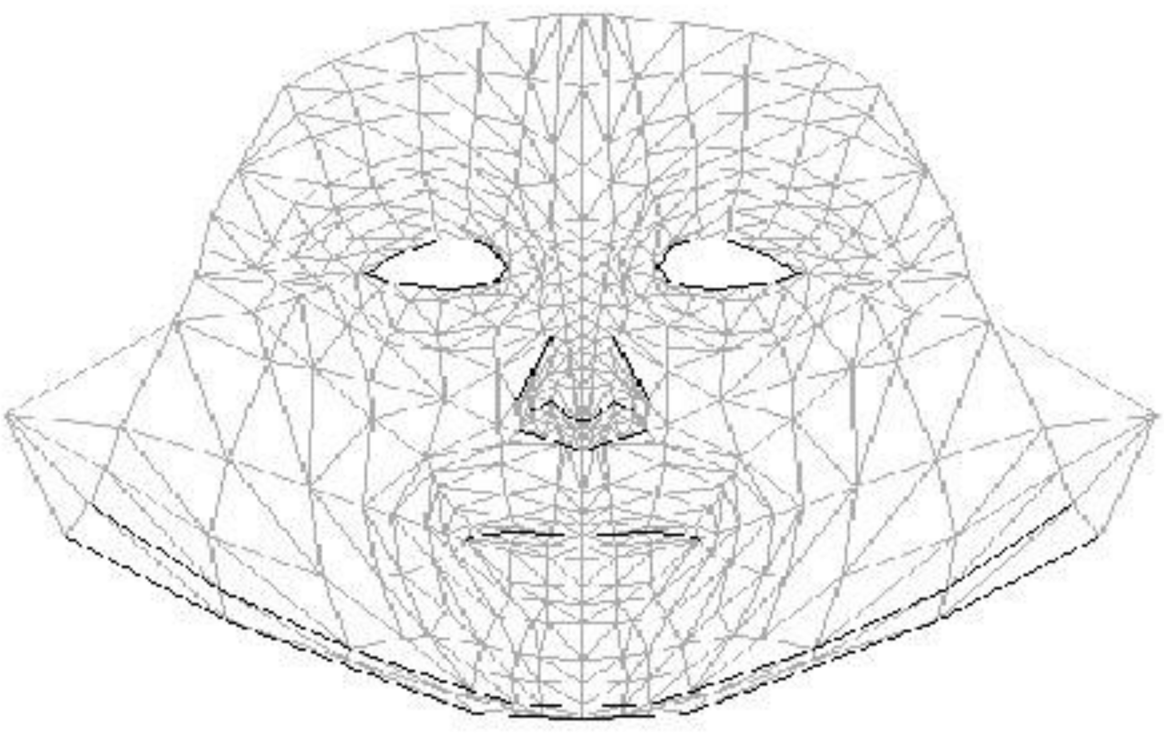
Building a face mesh

- Number of nodes
- Computational cost



*by: **Cem Yuksel**, BUPAM Bogazici University Pattern Analysis and Machine Vision Laboratory*

Generic Face Mesh and Mesh Adaptation

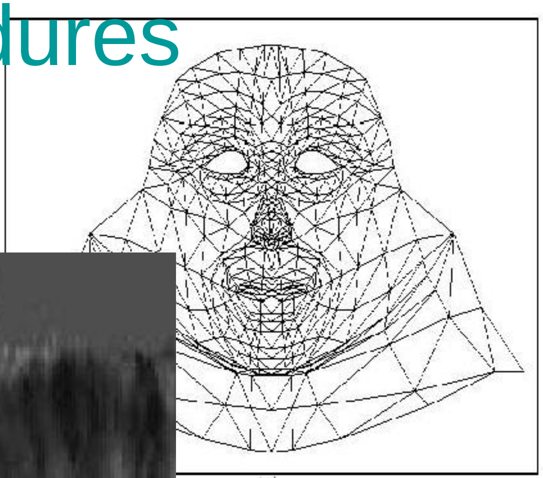


Advantages:

- Well-defined features
- Efficient Triangulation

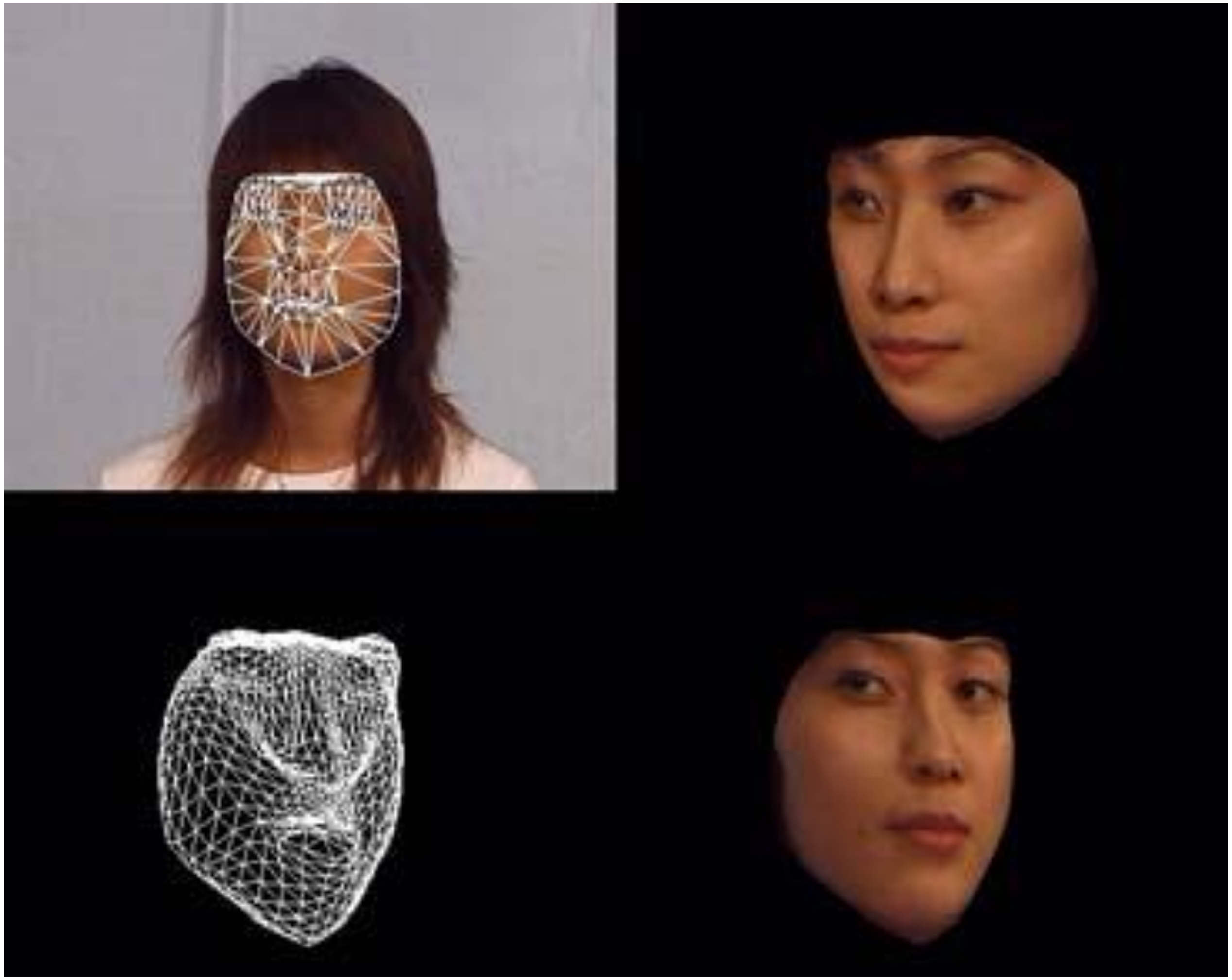
K. Waters. A muscle model for animating threedimensional facial expression. *Computer Graphics*, 1987.

Mesh Adaptation Procedures

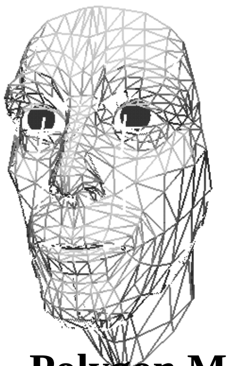


1. Locate nose tip
2. Locate chin tip
3. Locate mouth contour
4. Locate chin contour
5. Locate ears
6. Locate eyes
7. Activate spring forces
8. Adapt hair mesh
9. Conform to 3D

*by: **Cem Yuksel**, BUPAM Bogazici University Pattern Analysis and Machine Vision Laboratory*

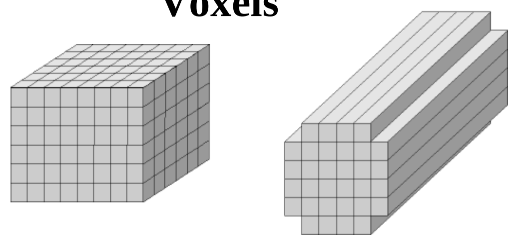


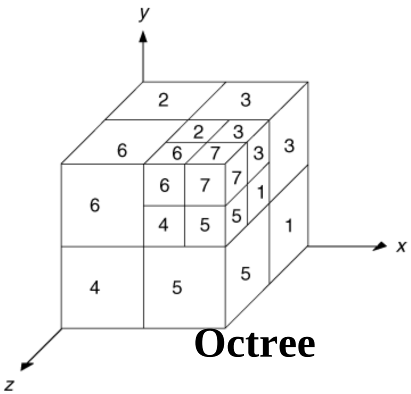
Object Modelling – Summary



Polygon Mesh

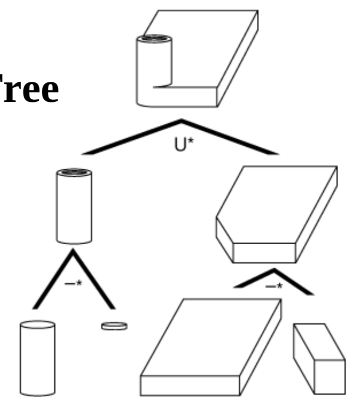
Voxels





Octree

CSG Tree



B-rep