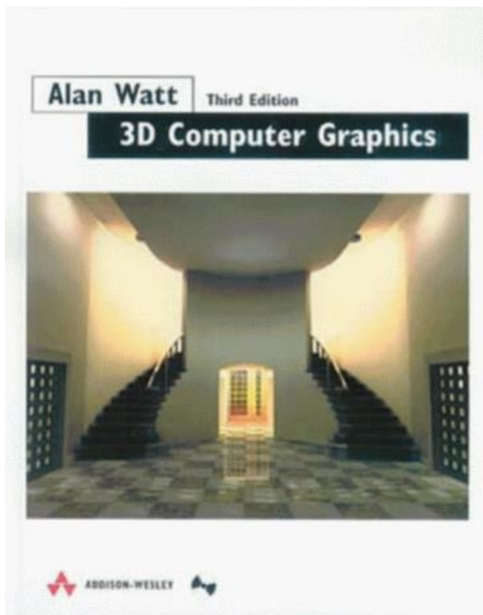
COM3503/4503/6503: 3D Computer Graphics

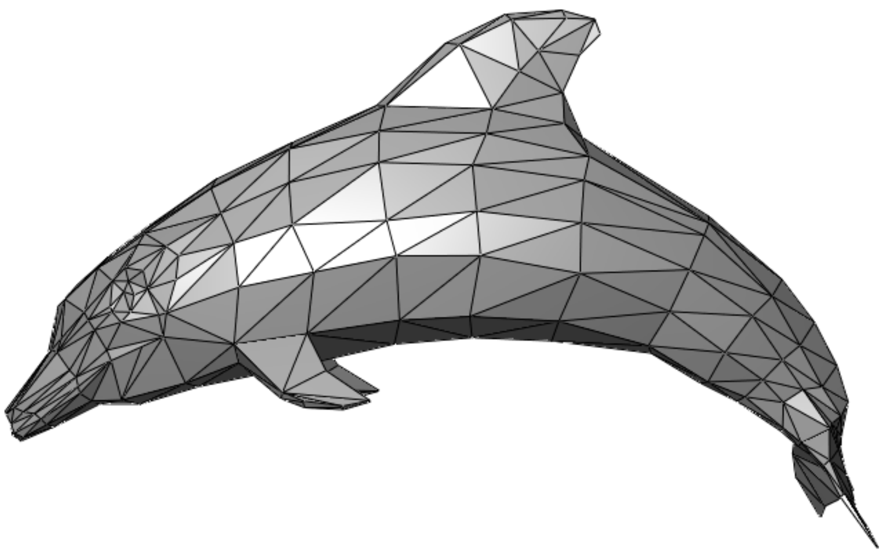
Lecture 5: Representation of 3D objects



Dr. Steve Maddock
Room G011, Regent Court
s.maddock@sheffield.ac.uk

1. Representation of 3D objects

- "…computer graphics was born when Ivan Sutherland had the idea that Geometry is Structured Data" (Kajiya, 92)
- Many alternative representations
  - Polygons, parametric patches, Constructive Solid Geometry, space subdivision, implicit representation, etc.

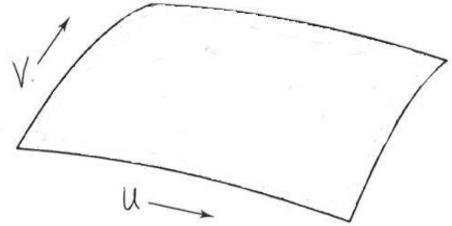


http://en.wikipedia.org/wiki/File:Dolphin_triangle_mesh.png

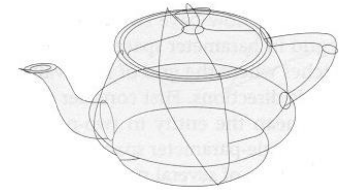
© Dr Steve Maddock, The University of Sheffield

2. Parametric patches

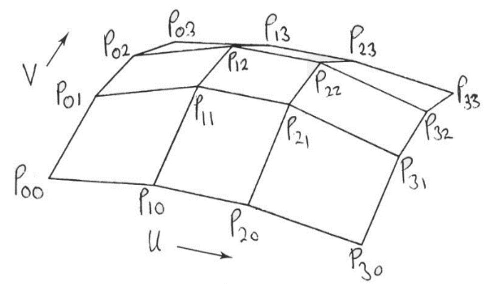
- Parametric patches are used widely in Computer-Aided Design (CAD)
  - ‘curvilinear quadrilaterals’
  - Bicubic patches most common
- Different representations:
  - Bezier, B-spline, NURBS, …
- 3 aspects:
  - Maths of the representation
  - Control structure to edit
  - Technique to visualise



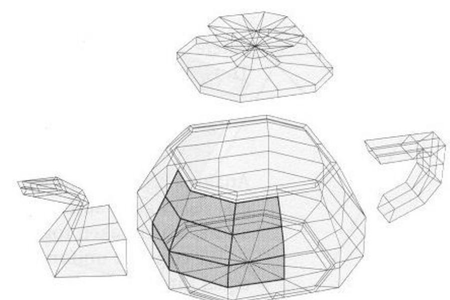
Single patch



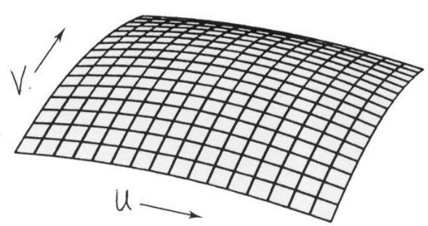
Teapot: 32 patches



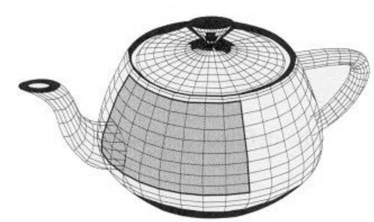
Control structure



Control structure (exploded view)



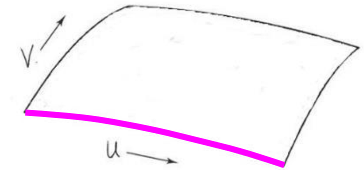
Visualised as polygons



Visualised as polygons

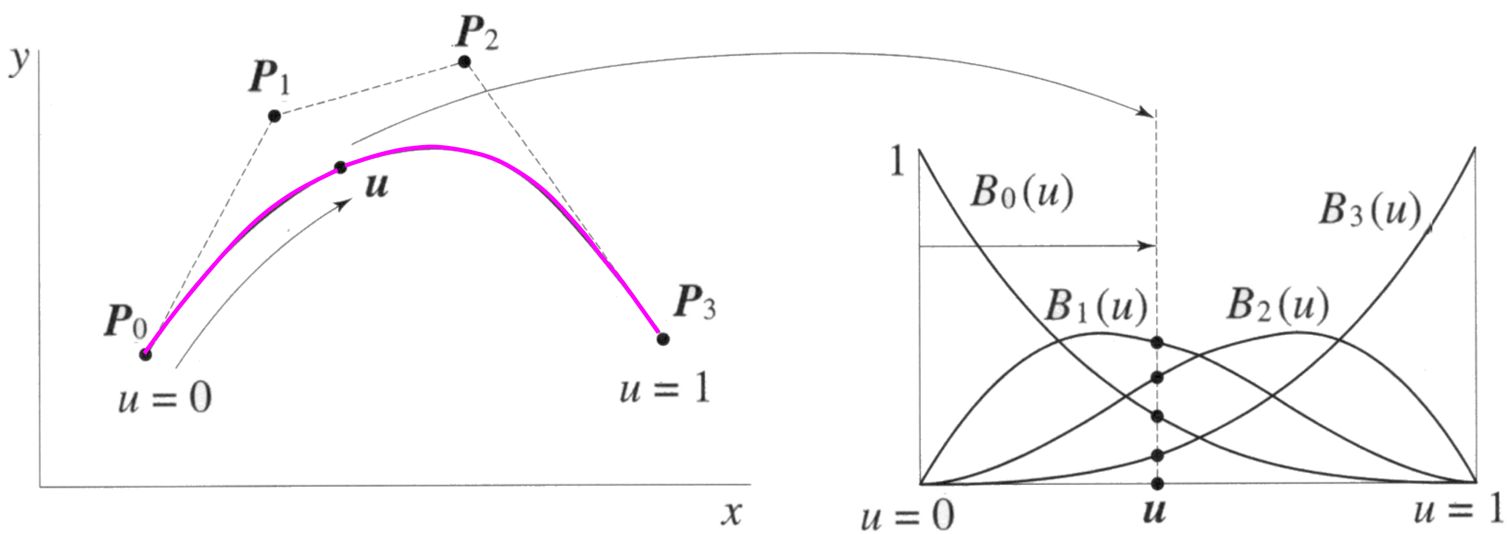
 © Dr Steve Maddock, The University of Sheffield

2.1 Maths of bicubic Bezier patches



- Consider one edge: cubic curve with 4 control points, P_i

$$\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i B_i(u) \qquad \mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$$



Space of curve   Space of basis functions

- Bi(u) are the Bernstein blending functions, which sum to 1 for 0≤u≤1

© Dr Steve Maddock, The University of Sheffield

2.1 Maths of bicubic Bezier patches

- For a bicubic patch:

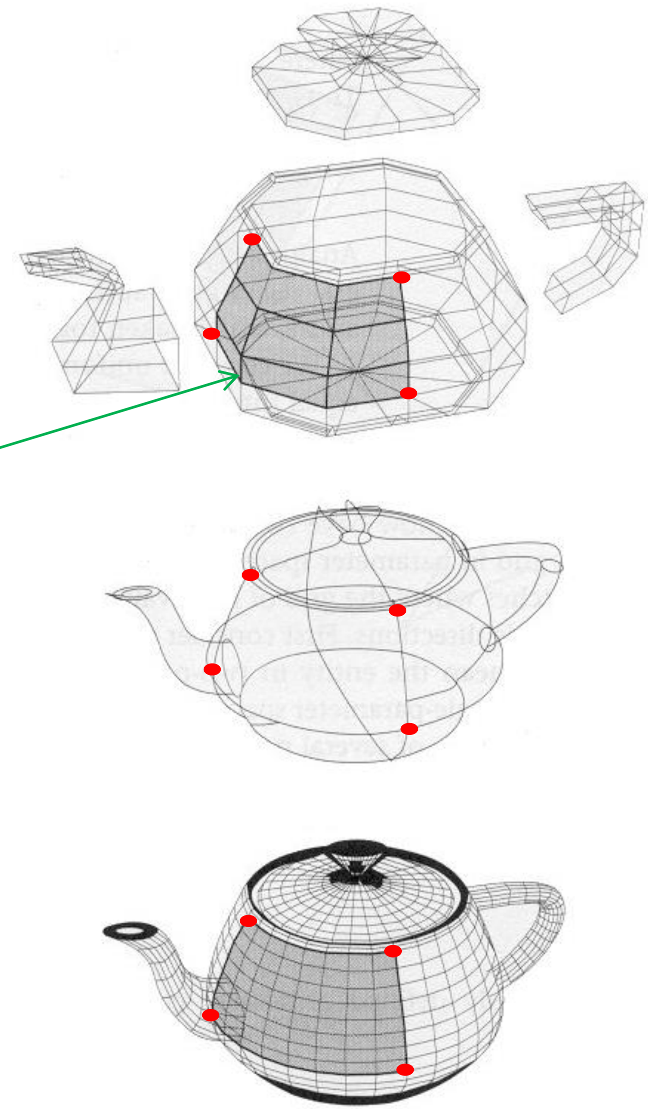
$$\mathbf{Q}(u,v) = \sum_{i=0}^{3} \sum_{j=0}^{3} \mathbf{P}_{ij} B_i(u)B_j(v)$$

$$0 \le u,v \le 1$$

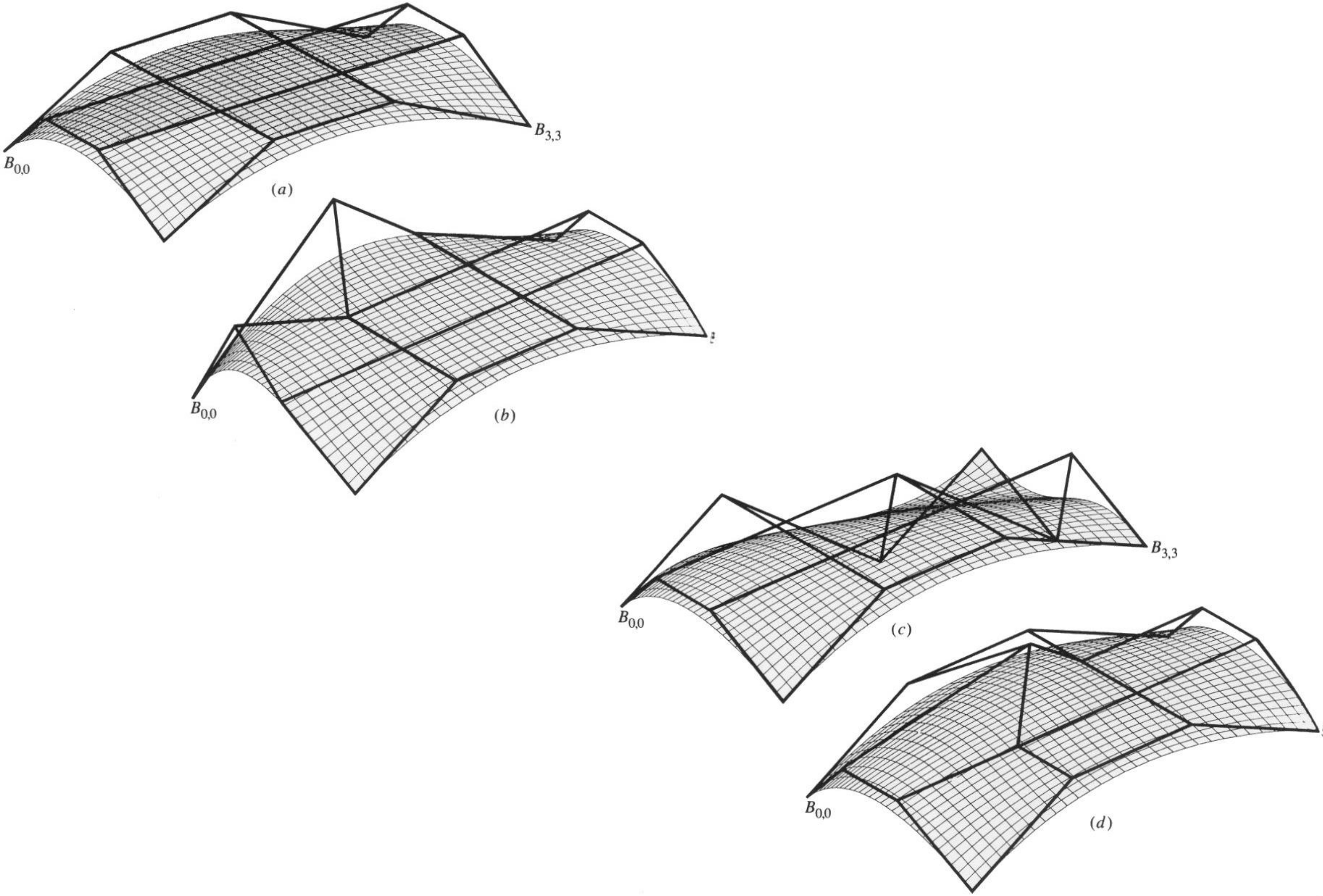
- 16 control points $\mathbf{P}_{ij}$ are needed to determine the shape of the cubic patch

- $\mathbf{Q}(u,v)$ only passes through the corner points

- Exact analytical form – good for CAD

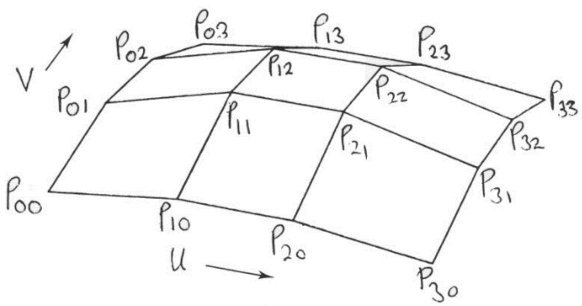


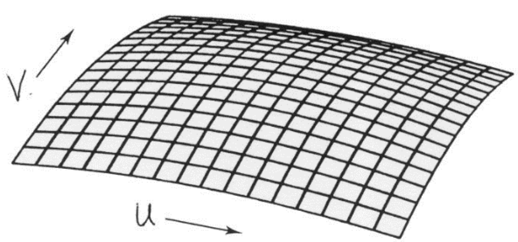
© Dr Steve Maddock, The University of Sheffield

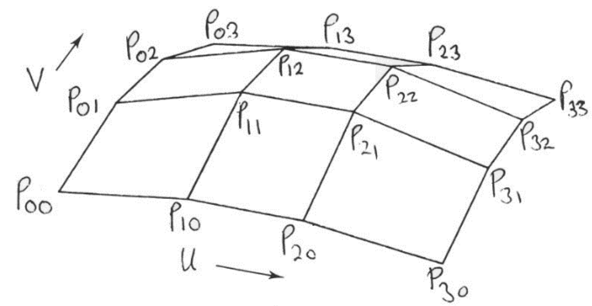

$B_{0,0}$
$B_{3,3}$
(a)
$B_{0,0}$
(b)
$B_{0,0}$
$B_{3,3}$
(c)
$B_{0,0}$
(d)

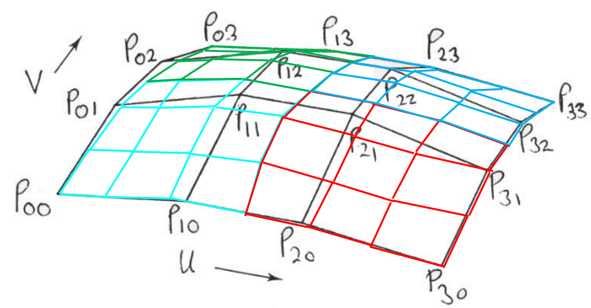
2.3 Displaying patches as polygons

- Incremental evaluation
  - Sample Q(u,v) in u and v to create a polygon network
- Recursive subdivision algorithm
  - Example: de Castlejau
  - Subdivide one patch into 4 patches





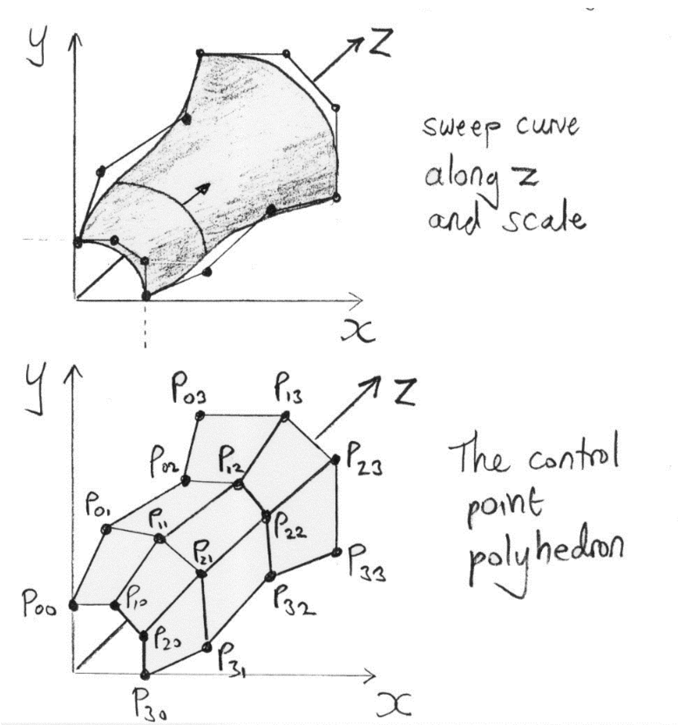


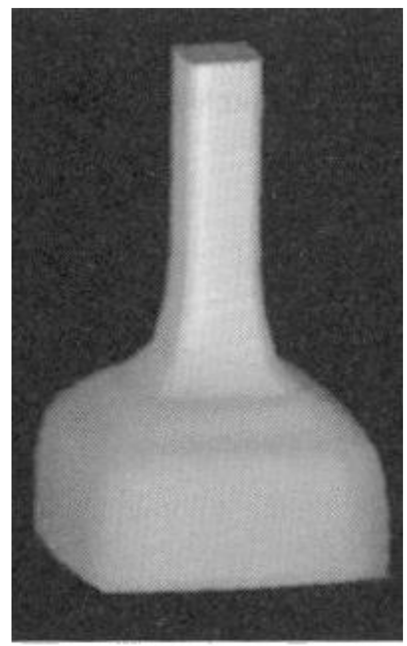


 © Dr Steve Maddock, The University of Sheffield

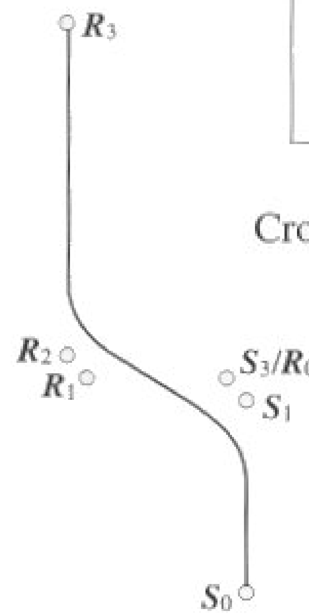
2.4 Creating patch models - alternatives

- Fit patches to a set of points using an optimization approach
- Model by editing control points
- Sweep a profile represented as a parametric curve along another parametric curve

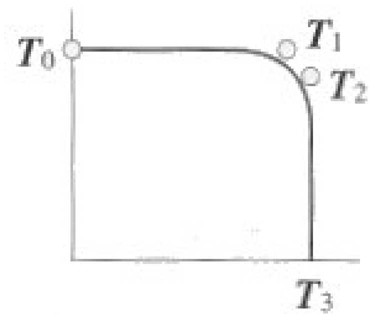




Object



Profile design



Cross-section design

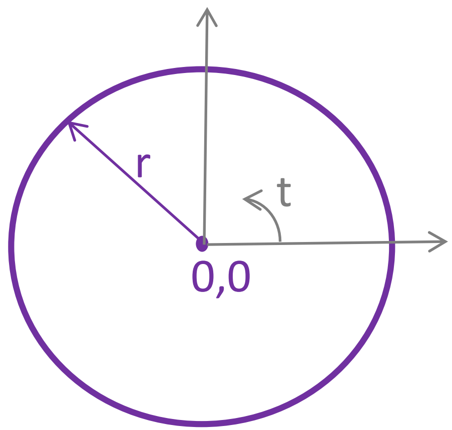
3. Implicit functions

$$x^2 + y^2 = r^2$$

or

$$F(x,y) = x^2 + y^2 - r^2 = 0$$

Implicit form



$$x = f(t) = r\sin(t)$$

$$y = g(t) = r\cos(t)$$

$$0 <= t <= 2\pi$$

Parametric form

- Points where $F(x,y) = 0$
- Easy to test if a point is inside ($F<0$) or outside ($F>0$)
- No systematic way to generate points on the surface
- (Circle with centre (a,b): $(x-a)^2 + (y-b)^2 = r^2$)

© Dr Steve Maddock, The University of Sheffield

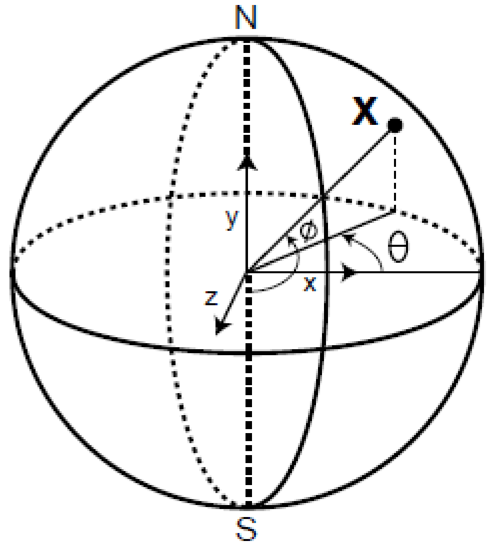
3. Implicit functions

$$x^2 + y^2 + z^2 = r^2$$

or

$$x^2 + y^2 + z^2 - r^2 = 0$$

Implicit form



$$x = r \sin\phi \cos\theta,$$
$$y = -r \cos\phi,$$
$$z = -r \sin\phi \sin\theta,$$

$$0 \le \theta \le 2\pi$$
$$0 \le \phi \le \pi$$

Parametric form

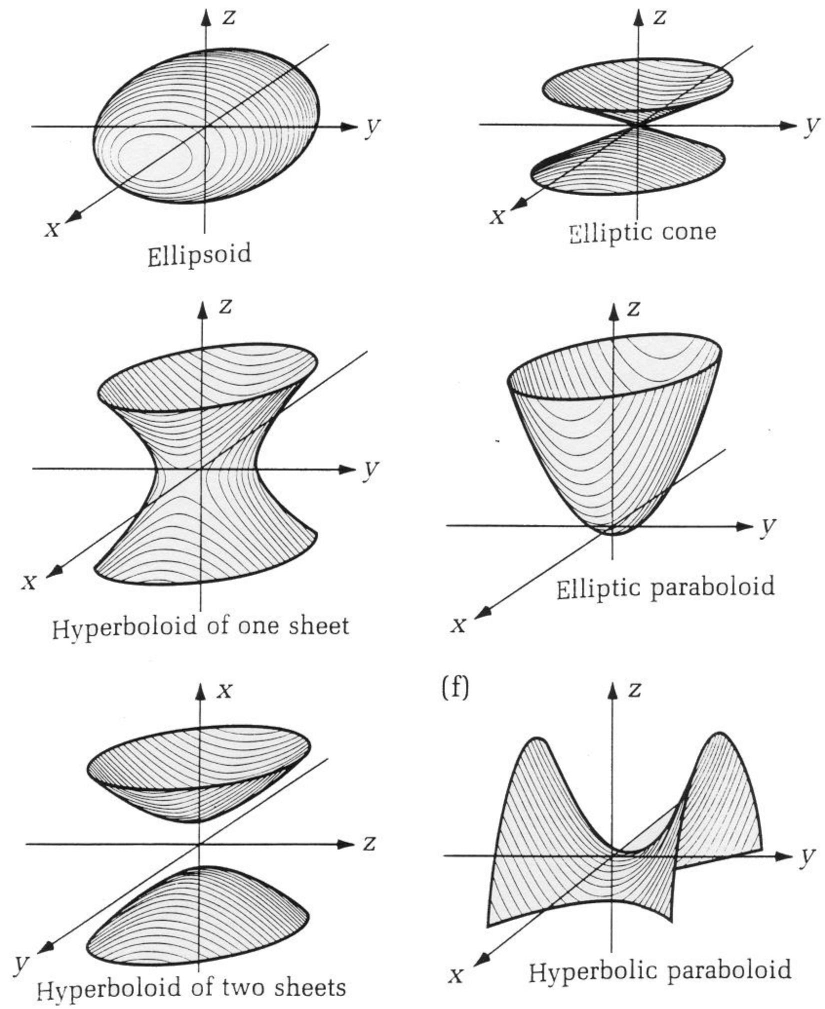
- Points where f(x,y,z) = 0
- Easy to test if a point is on either side of a surface
  - Polygons and parametric surfaces do not have this property
- Also easy for intersection tests (see ray tracing lecture)
- No systematic way to generate points on the surface

http://www.cs.clemson.edu/~dhouse/courses/405/notes/implicit-parametric.pdf

© Dr Steve Maddock, The University of Sheffield

3.1 Quadric surfaces

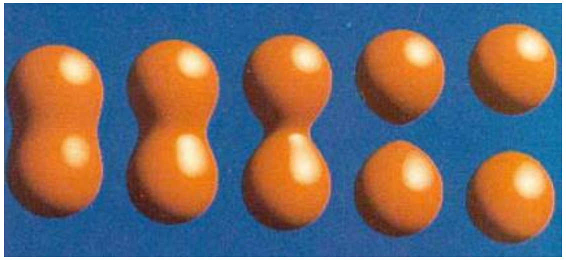
$$F(x,y,z) = ax^2 + ey^2 + hz^2$$
$$+ 2bxy + 2fyz + 2cxz$$
$$+ 2dx + 2gy + 2iz + j = 0$$



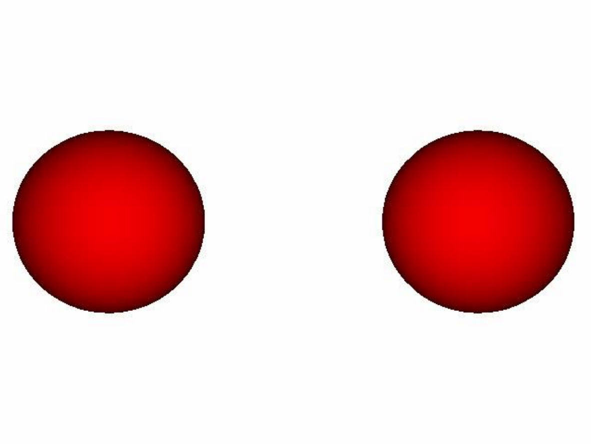
© Dr Steve Maddock, The University of Sheffield

3.2 Creating a surface from implicit functions

- Constant value surface or level set or isosurface
- Blobs, metaballs or soft objects
  - Good for modelling organic objects
  - Additive process to combine the individual blobs



J.F. Blinn, "A Generalization of Algebraic Surface Drawing." ACM Transactions on Graphics 1(3), July 1982, pp. 235–256.

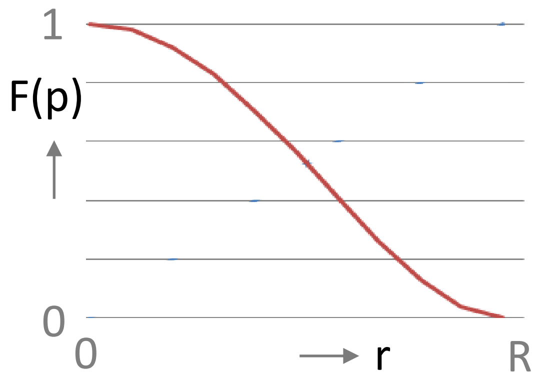


3.3 Blobs, metaballs and soft objects

- For a single blob:

$$f(p) = s\left(1 - \frac{r^2}{R^2}\right)^2$$

f(p) – *Iso* = 0



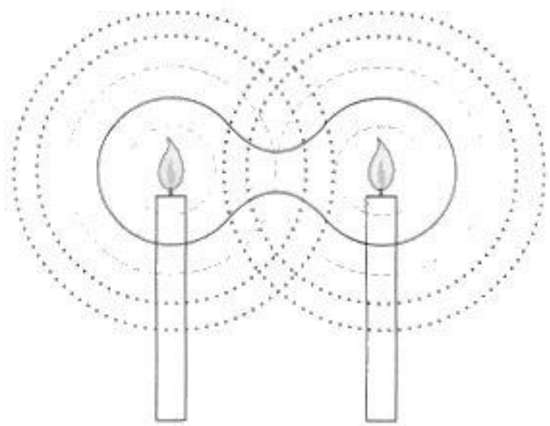
s = 1 (usually).

r = distance of p to centre of primitive

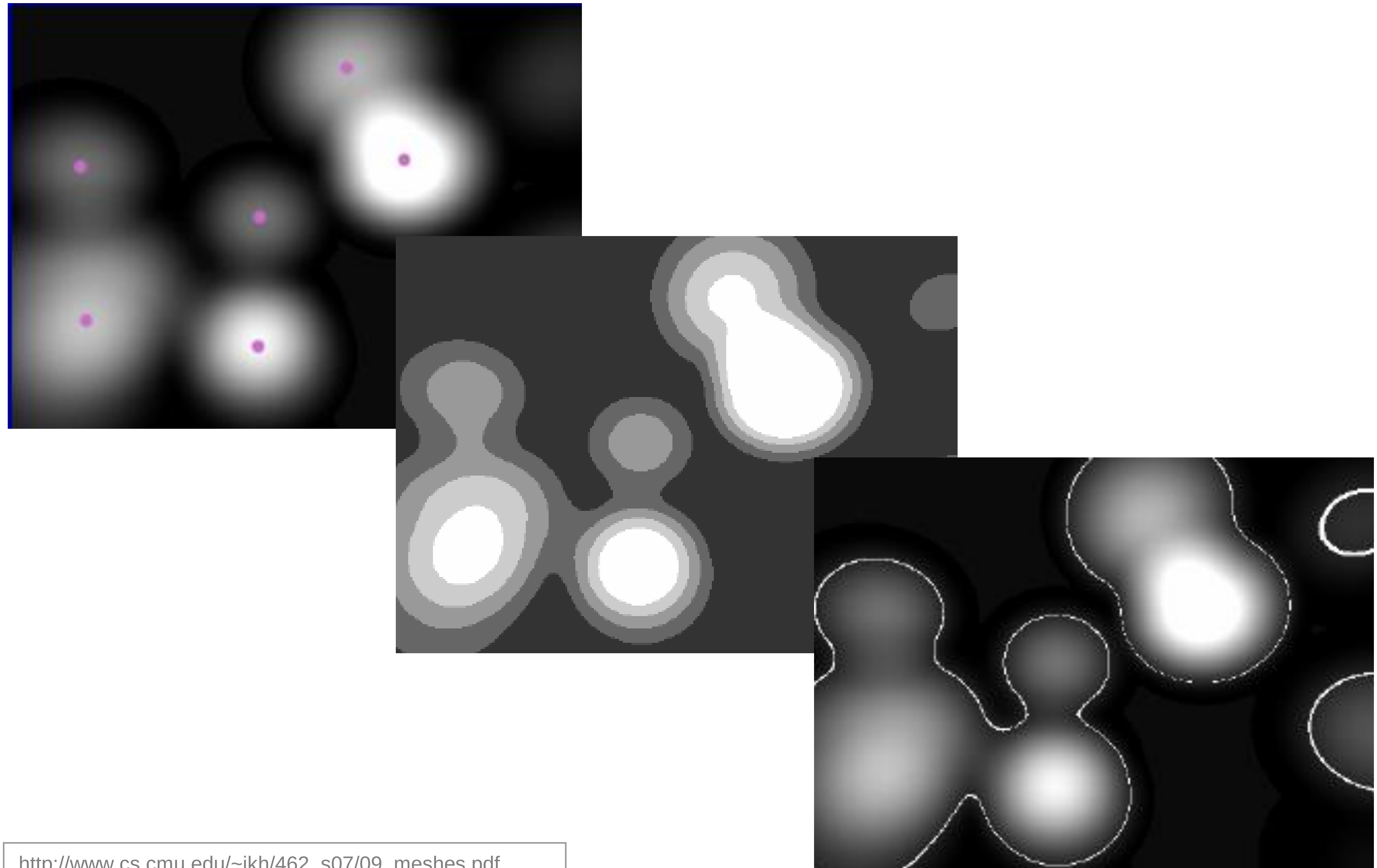
R = Radius of influence of a primitive

- Multiple blobs: Sum influences

$$F(p) = \sum_{i \in Influencing(p)} f_i(p)$$



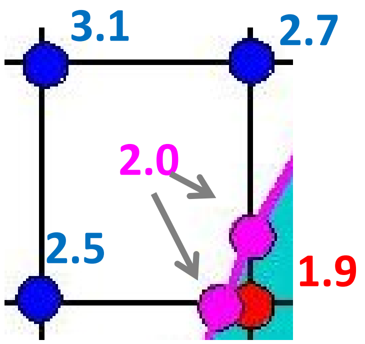
© Dr Steve Maddock, The University of Sheffield

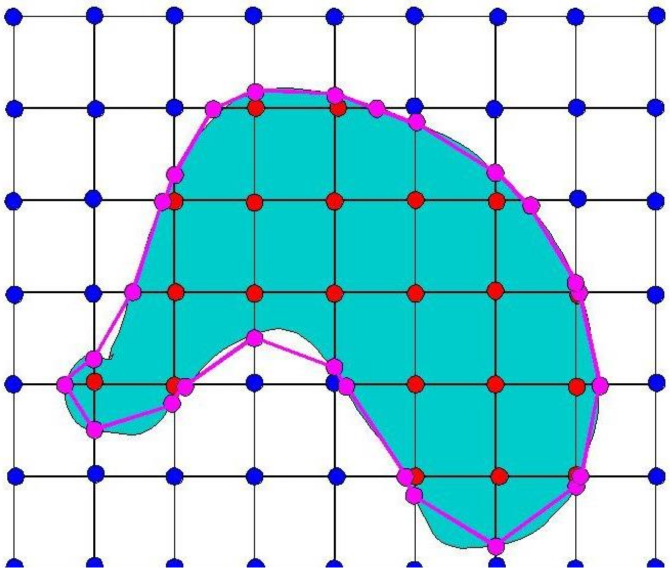


http://www.cs.cmu.edu/~jkh/462_s07/09_meshes.pdf,
Jessica K. Hodgins, CMU, 2002

3.3.1 Rendering blobs, metaballs and soft objects

- Consider first a 2D case
- At each grid point, evaluate the implicit function(s)
- Choose an isovalue - pink
  - Red points inside
  - Blue points outside
- Given values at grid points estimate where surface intersects the square
  - Pink points
- Join pink points to give estimation of surface
  - Accuracy (and speed) dependent on grid resolution and estimation process



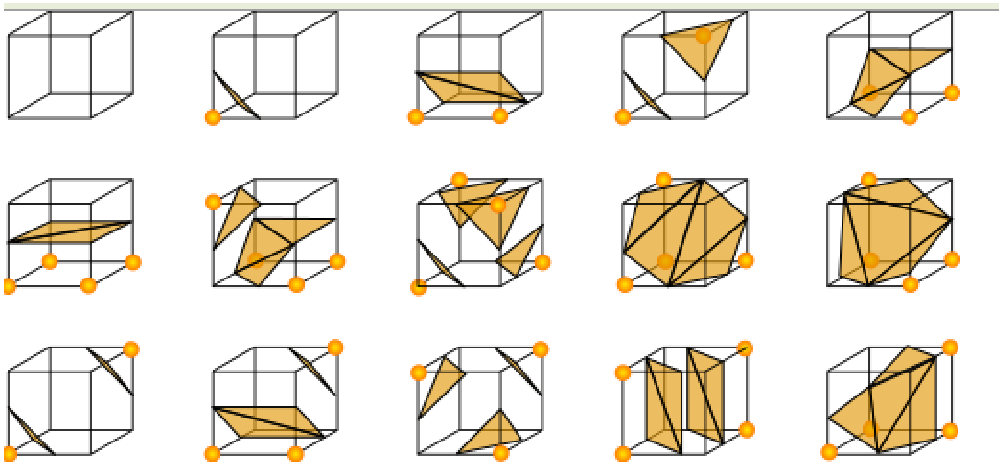


http://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html

© Dr Steve Maddock, The University of Sheffield

3.3.1 Rendering blobs, metaballs and soft objects

- Convert to polygons
  - Marching cubes algorithm [Lorensen and Cline, 1987]
  - Divide space into cubes
  - Evaluate implicit function at each cube vertex
  - Do root finding or linear interpolation along each edge
  - Polygonize on a cube-by-cube basis



- Ray tracing (see later lecture)
  - Render using mathematical representation
  - Easy intersection test

Lorensen, W. E. and Cline, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," Computer Graphics, vol. 21, no. 3, pp. 163-169, July 1987

© Dr Steve Maddock, The University of Sheffield

3.4 Some examples

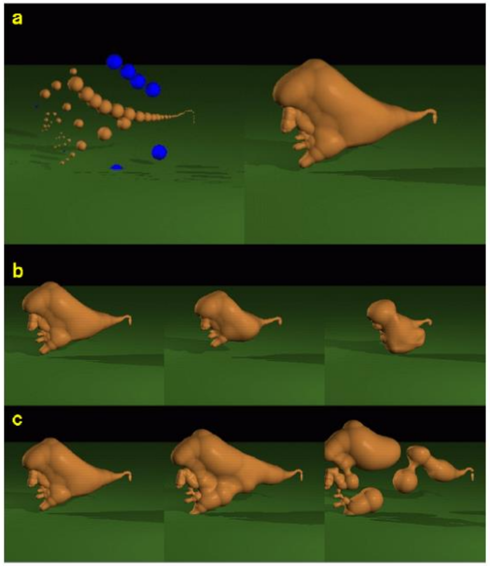
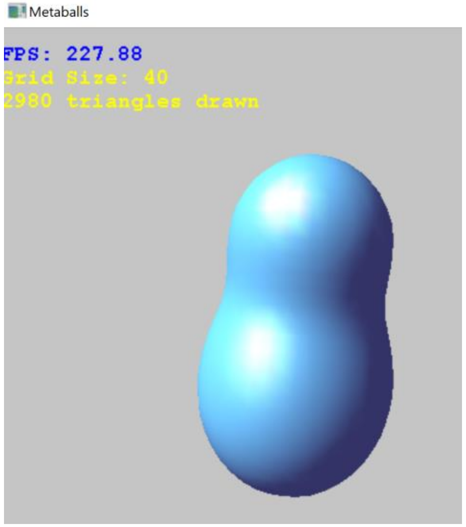


Figure 3.3: *a) A model inspired by the work of Salvador Dali: Lighter spheres represent positive primitives, darker spheres represent negative primitives, the radius of each sphere is proportional to the radius of influence of the corresponding primitive. b) Loosing appearance by unwanted blending c) Loosing appearance by coherence loss*

Opalach, 96

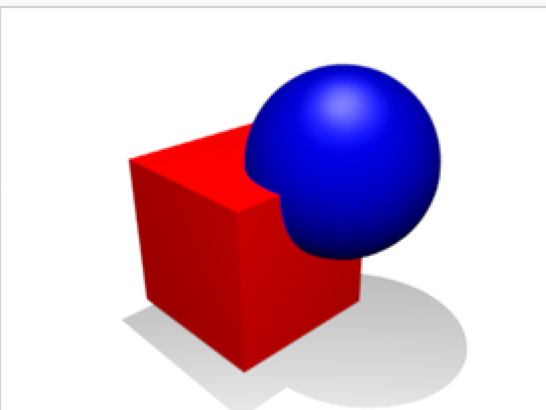
Terminator 3 Teaser trailer,
(Warner Bros, Sony Pictures)



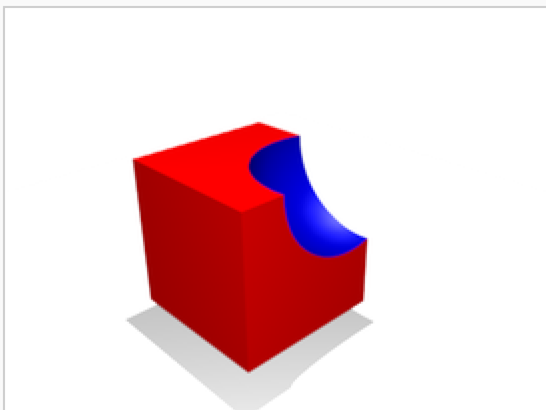
Metaballs,
www.paulsprojects.net

4. Constructive Solid Geometry (CSG)

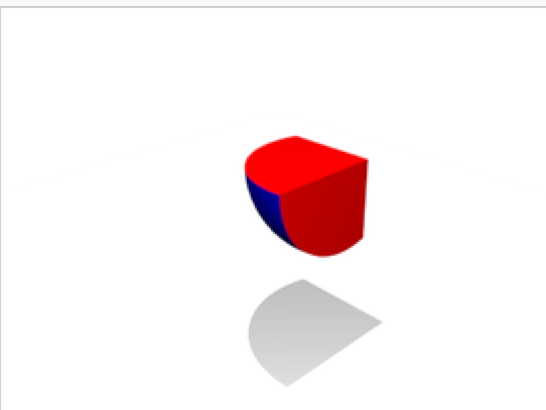
- Geometric primitives:
  - e.g. sphere, cone, cylinder, rectangular solids
- Combine using (regularized) Boolean set operators (and linear transformations)
  - union, difference, intersection



Union
Merger of two objects into one



Difference
Subtraction of one object from another



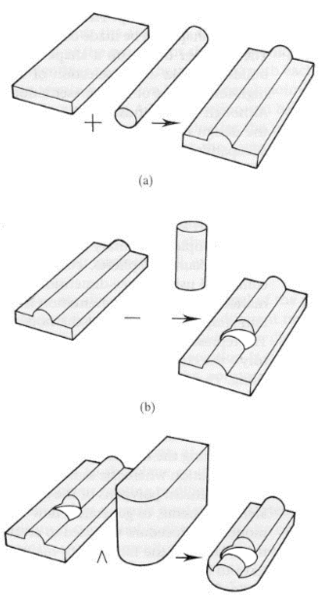
Intersection
Portion common to both objects

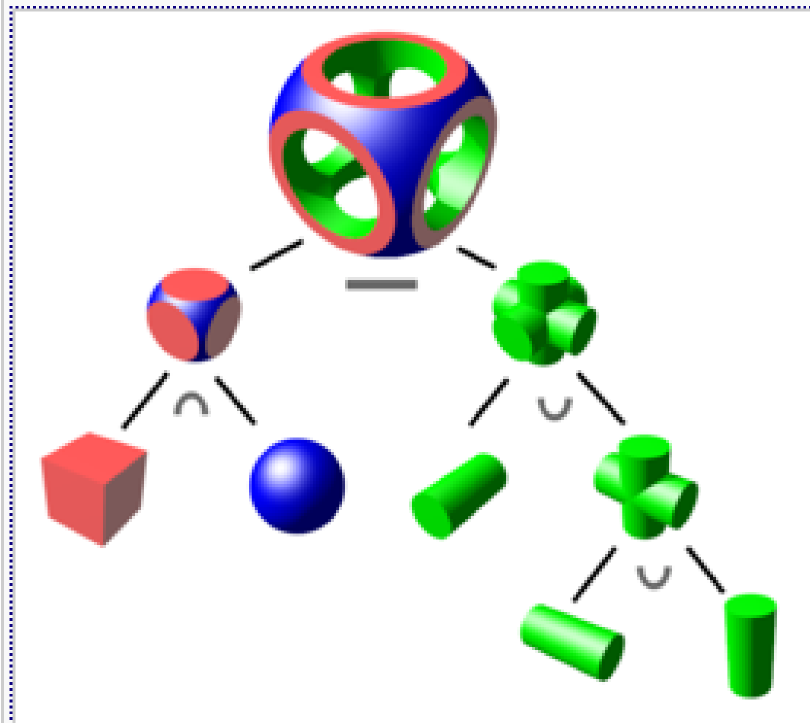
http://en.wikipedia.org/wiki/Constructive_solid_geometry

© Dr Steve Maddock, The University of Sheffield

4. Constructive Solid Geometry (CSG)

- Object model stored as a tree:
  - Leaves – primitives
  - Nodes – operators or linear transforms
- Machine shop parts are suited to representation using CSG



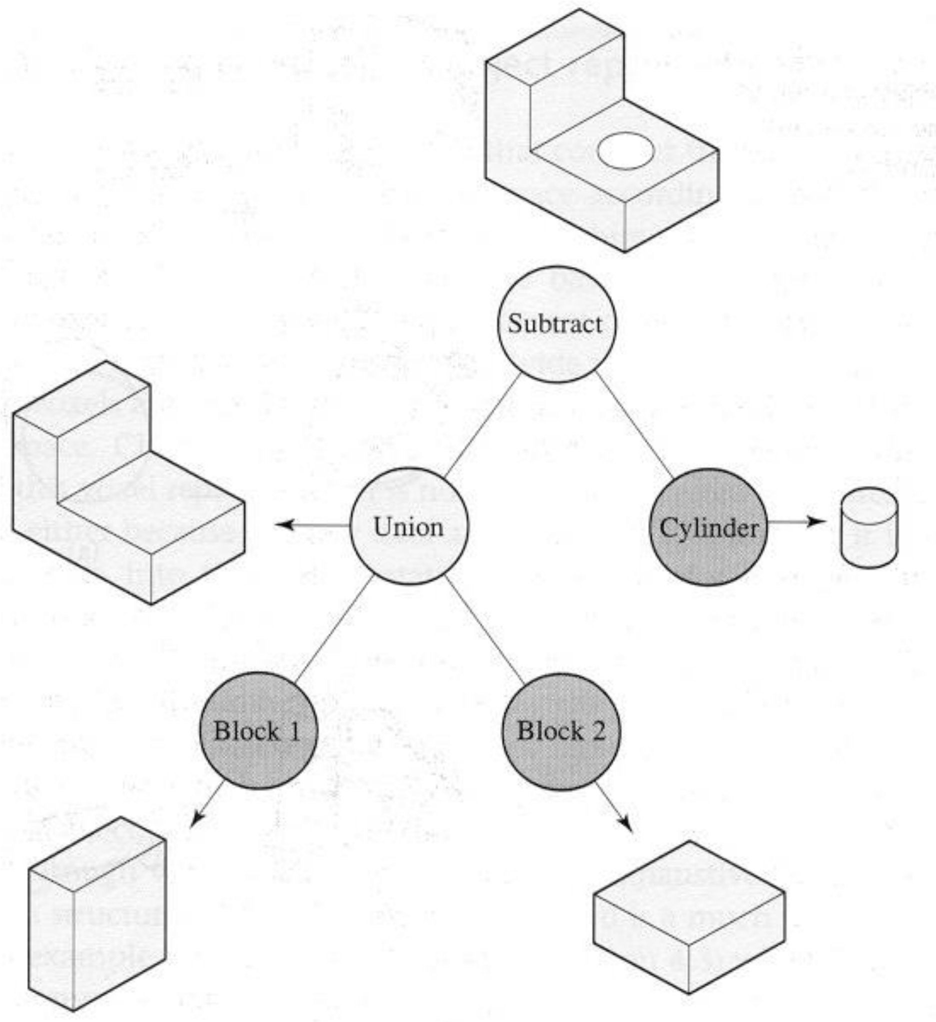


CSG objects can be represented by binary trees, where leaves represent primitives, and nodes represent operations. In this figure, the nodes are labeled ⋂ for intersection, ⋃ for union, and — for difference.

http://en.wikipedia.org/wiki/Constructive_solid_geometry

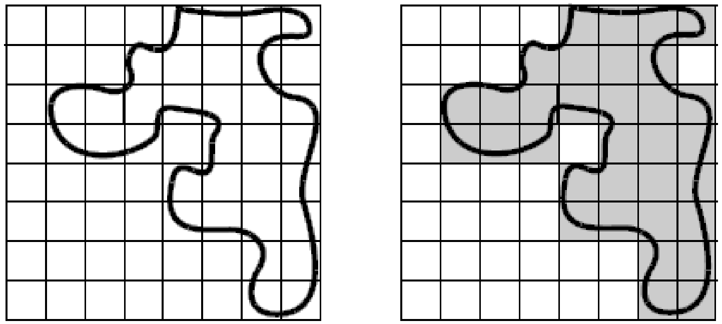
4. Constructive Solid Geometry (CSG)

- Advantages
  - Compactness
  - Modelling history is recorded – easy to do simple shape editing
  - Easy to validate representation
- Disadvantages
  - Unevaluated model – need to walk tree after change
  - Rendering time
  - Doesn't provide a unique representation



5. Space subdivision techniques

- Decompose space into regions that are labeled as being inside or outside the solid being modeled

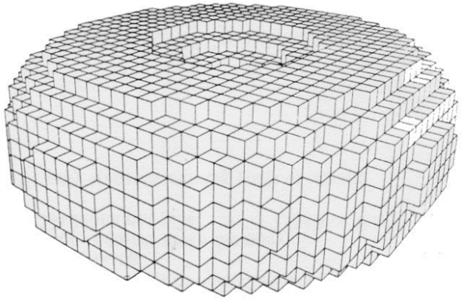


- 2D example – a grid of squares

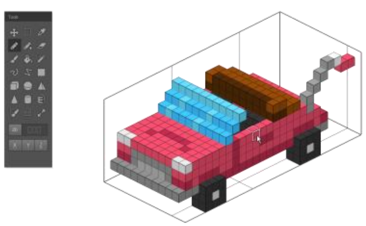
3D examples:

- 'Voxels' – equal subdivision of space into cubes
    - Voxels in Modelling
    - Voxels in Medical Imaging
- Octrees – hierarchical subdivision into cuboid areas
- Binary Space Partitioning (BSP) trees – hierarchical splitting of space using partitioning planes

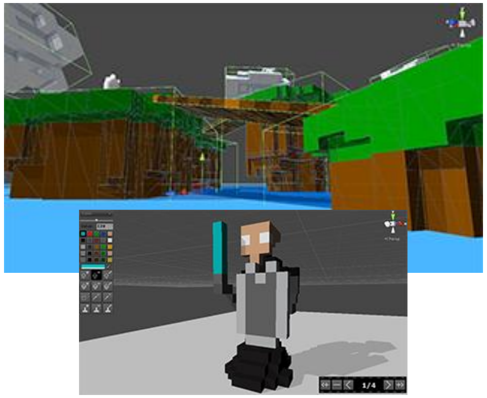
5.1.1 Voxels - cubes



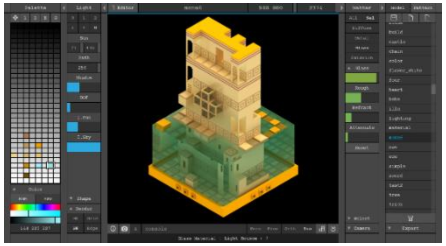
- Analogous to pixels
- Equal subdivision of space into cubes
- Cell is occupied or not – enumeration of space
- *Issues*: Approximation; Amount of data
- Have been used for terrain in games



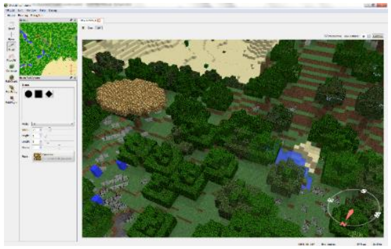
Quibicle
http://qubicle-constructor.com/



PicaVoxel
http://picavoxel.com/



MagicaVoxel
https://ephtracy.github.io/



MCEdit
http://www.mcedit.net

© Dr Steve Maddock, The University of Sheffield

Voxel (cubes) puzzle game: Sam Dickinson, dissertation project 2016/17

5.1.2 Voxels – scientific visualisation

- Medical imaging: data sets from MRI or CT scans are represented as voxels
- Essentially a value is stored for each pixel in a layer
  - A measurement is assumed to be at the point in the bottom left of a square pixel
  - no info between layers
  - 8 measurements(from two layers) make a voxel
- Marching cubes algorithm for surface reconstruction (Or volumetric ray casting)

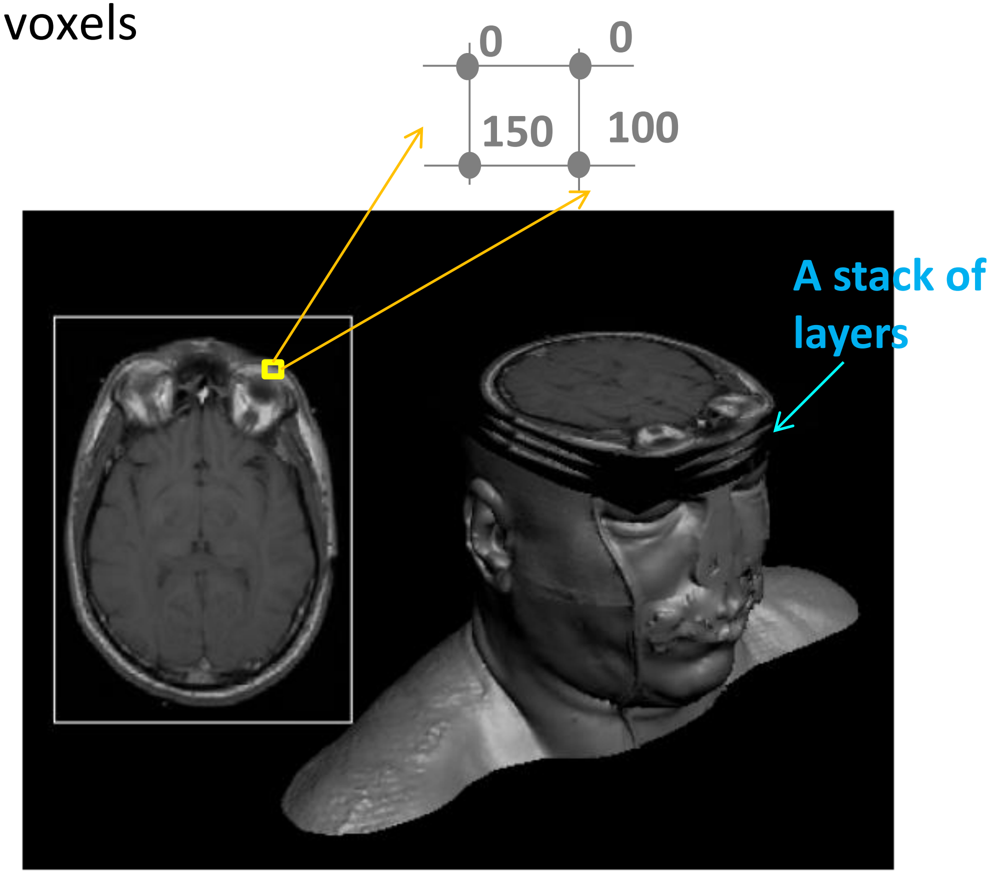
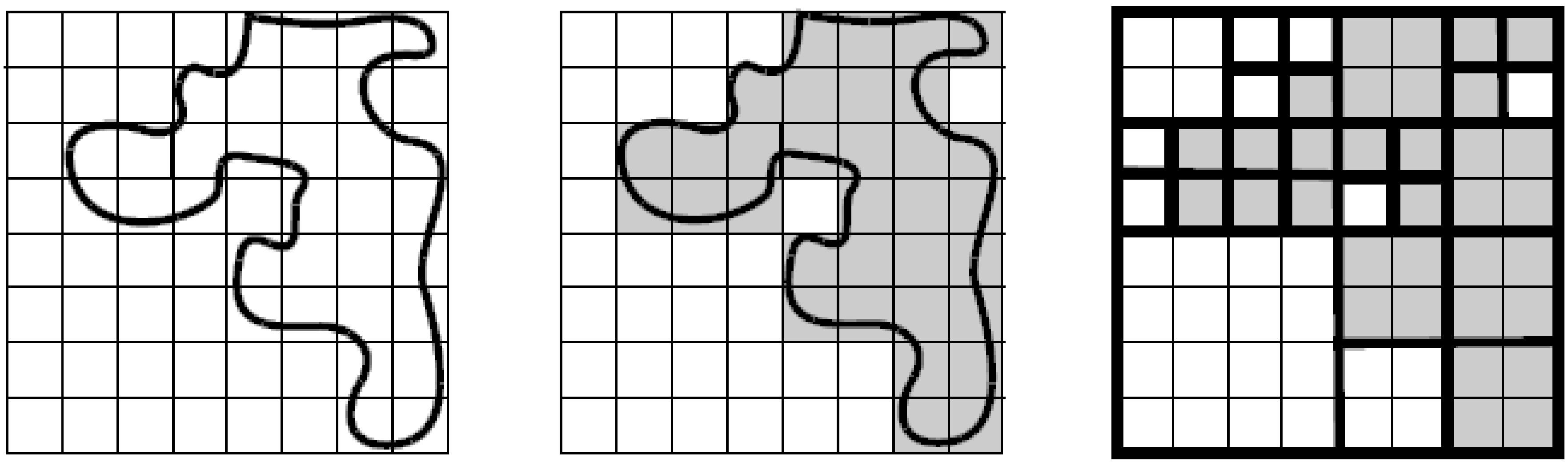


Figure 3.1: The magnetic resonance male dataset from the VHP project [4, 89]. Left: a typical MRI scan image. Right: the structure of the volume.

Salas, 2010

5.2 Quadtree

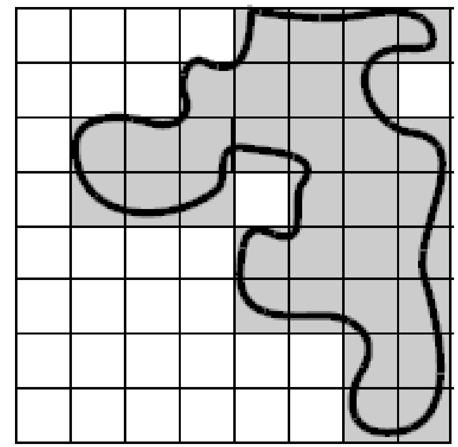
- Instead of a set of voxels all the same size, use adaptive resolution
- Don't subdivide voxels that are either fully occupied or fully empty

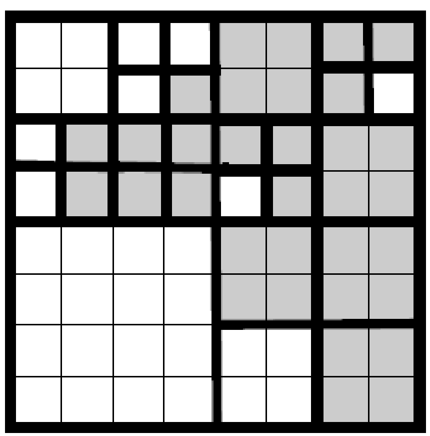


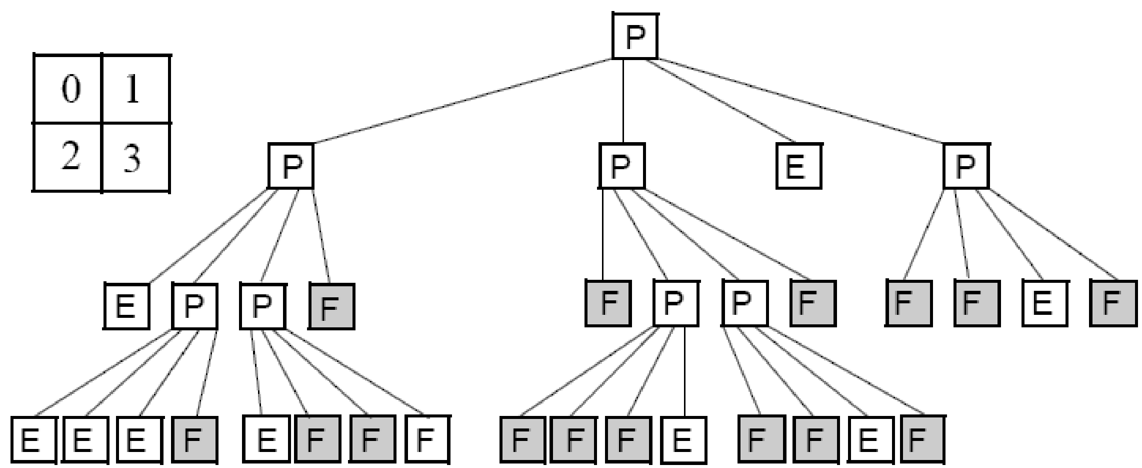
Lecture notes, William Smith, University of York

5.2 Quadtree

- Represent as a tree







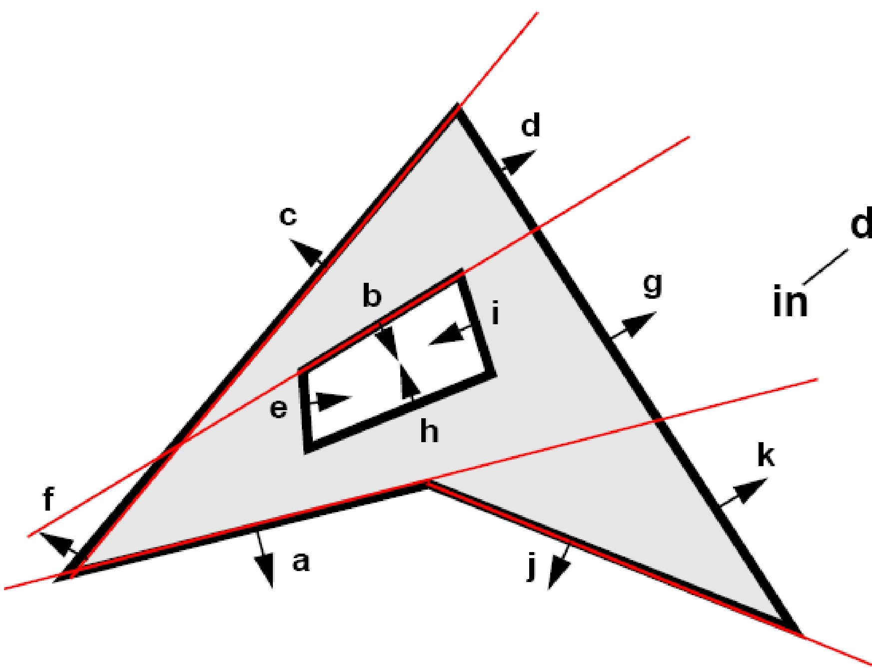
Lecture notes, William Smith, University of York

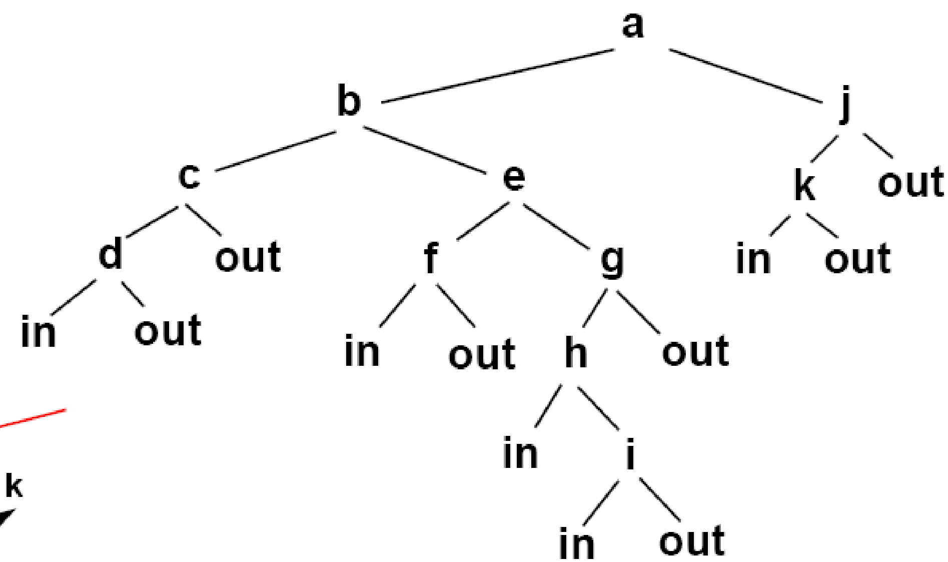
5.3 Octree

- Octree extends this to 3D – more often used as a secondary data structure to label occupancy of an area of space (see ray tracing lecture)

5.4 Binary space partitioning trees (BSP trees)

- Similar to octree
- Each node is a single partitioning plane that splits space into two
  - However, each plane is at an arbitrary orientation
- Again, more often used as a secondary data structure to label occupancy of an area of space
  - (see ray tracing lecture)





Lecture notes, William Smith, University of York

6. Representation of 3D objects

Factors to consider for each representation:

- Suitability for particular objects
    - e.g. accuracy, complexity
- Ease of creation
    - by hand, or procedurally, or by fitting to measurements
- Storage and processing costs
    - data structure, transmission (e.g. file access)
- Ease of rendering
    - may involve conversion to polygons;
- Interaction
    - editing, animation
- Geometric computations
    - distance, intersection, normal vectors, curvature

© Dr Steve Maddock, The University of Sheffield

7. Summary

- Parametric patches are like ‘curvilinear polygons’
  - More compact than polygons
  - Complications when joining patches together
- Implicit surfaces (in the form of blobs/metaballs) are good for representing organic objects
- CSG is good for describing machined objects
- Spatial enumeration techniques:
  - A representation of an object, e.g. voxels
- *Alternative view* of spatial enumeration techniques
  - Decompose space into regions and label what objects or parts of objects are in each region, i.e. bucket sort
  - Secondary data structures for optimizing other algorithms, e.g. collision detection and ray tracing

© Dr Steve Maddock, The University of Sheffield

Appendix A. A comparison of polygons and parametric patches

	Polygons	**Patches**
Intuitive editing	*Positive*: WYSIWYG	*Neutral*: although they can be intuitive for certain classes of object, there are problems with continuity and topology
Topology	*Positive*: any topology can be represented	*Neutral*: extra complexity if non rectangular patches used, but using just rectangular patches puts some limits on topology
Levels of detail	*Neutral*: can be generated but data at each level is separate, e.g. texture data	*Positive*: in general many LODs can easily be generated using subdivision, but some cost implications
Connection	*Positive*: easy to connect separate meshes by sharing vertices	*Neutral*: connectivity problems can cause knock-on effects for modelling

© Dr Steve Maddock, The University of Sheffield

Appendix A. A comparison of polygons and parametric patches

	Polygons	**Patches**
Smoothness	*Negative*: inherently angular; lots of polygons for smooth surface; silhouette problems	*Positive*: inherently smooth
Creases	*Positive*: easy to add sharp creases	*Neutral*: add complexity to connectivity information and to shading
Texturing	*Positive*: easily textured	*Positive*: readily textured and LODs are all textured as one process
Shading	*Positive*: hardware set up for fast rendering	*Positive*: easily shaded, but at extra cost in converting to polygons
Bandwidth	*Negative*: storage and processing costs escalate with number of polygons needed for smoothness & small-scale surface features	*Neutral*: less initial representation but extra cost in producing final representation

© Dr Steve Maddock, The University of Sheffield

Appendix B. The Utah teapot [created by Martin Newell, 1975]

- Less data that a polygon representation
  - 32 bicubic patches → 32* 9 control points (not 16 since borders are shared) → 288 control points (not always 9, so ~288)
  - Patches: 288 * 3 real numbers
  - Polygons: 2048 polygons (triangles) → 2048 vertices (for an approximate representation)
  - Polygons: 2048 * 3 real numbers