3D Object Representation

Intended Learning Outcomes

- Understand the concept of standard graphics object
- Able to mathematically manipulate and program in OpenGL two types of planar representation: tables and mesh
- Distinguish the concepts of parametric and non-parametric equations and understand the advantage of using the former in computer graphics
- Able to mathematically manipulate and program in OpenGL quadrics and super-quadrics

Standard Graphics Object

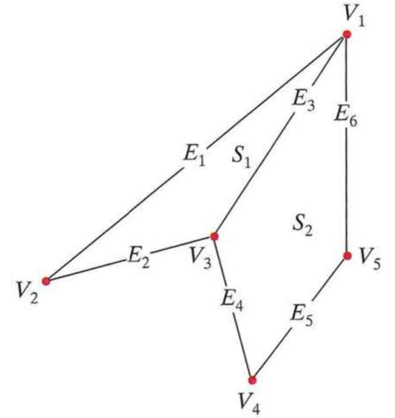
- **standard graphics object = a set of (planar) polygons**

- Complicated objects can be described by using many polygons

- Dedicated hardware are designed to speed up rendering of standard graphics objects.

Two methods for storing standard graphics objects

- Method 1: use table (vertex, edge, polygon, attribute)



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

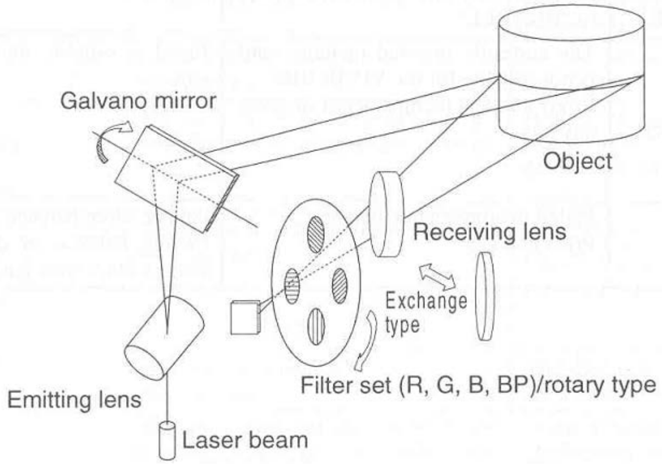
EDGE TABLE	
E_1:	V_1, V_2
E_2:	V_2, V_3
E_3:	V_3, V_1
E_4:	V_3, V_4
E_5:	V_4, V_5
E_6:	V_5, V_1

SURFACE-FACET TABLE	
S_1:	E_1, E_2, E_3
S_2:	E_3, E_4, E_5, E_6

Geometric data-table representation for two adjacent polygon surface facets, formed with six edges and five vertices.

- Method 2: Quadrilateral Mesh
  - A n x m array of vertex positions (X, Y, Z)
  - Represent a surface of (n-1) x (m-1) quadrilaterals
  - Each quadrilateral may be further subdivided into two triangles
  - Two ways to obtain data in the mesh
    - Way 1: By specifying an equation
    - Way 2: By 3D digitizer

3-D scanner



3D data obtained by triangulation

3D scanner is available in CityU Library:

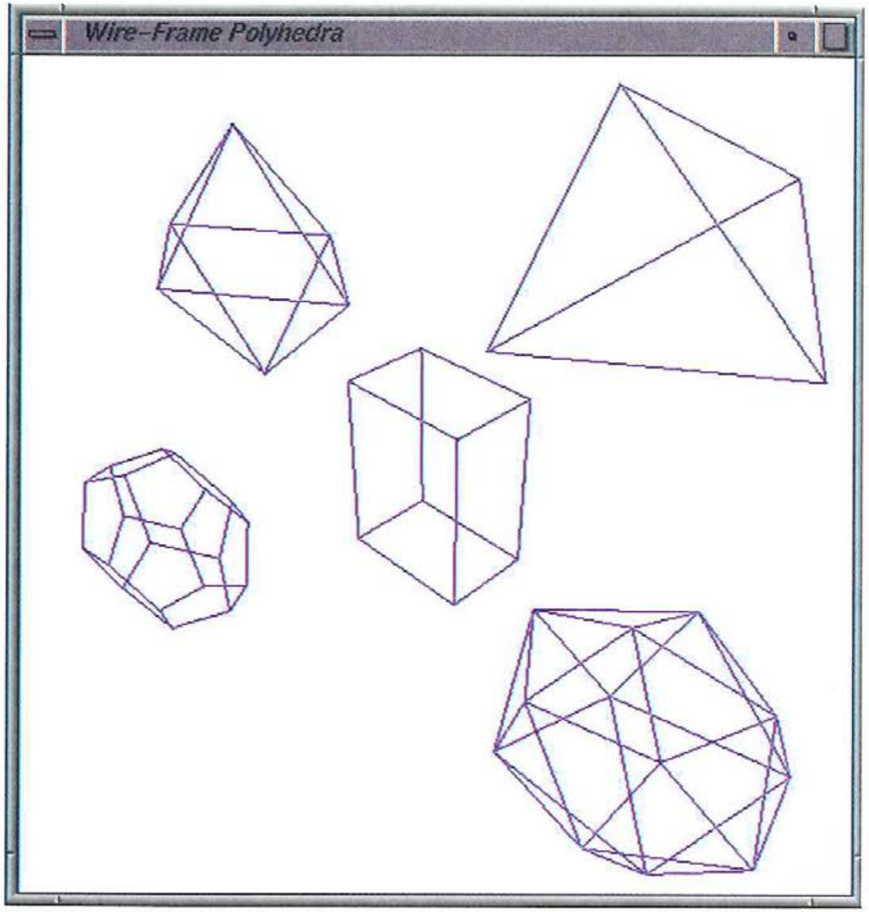
http://www.cityu.edu.hk/lib/about/facility/3d/index.htm

Glut functions

- *glutWire* as wireframe
- *glutSolid* as fill area polygon patches

glutSolidCube (edgelength);

- Tetrahedron, Cube, Octahedron, Dodecahedron, Icosahedron



A perspective view of the five GLUT polyhedra, scaled and positioned within a display window by procedure `displayWirePolyhedra`.

Mathematical Concepts for Plane

- Plane

$$aX + bY + cZ + d = 0$$

- Only 3 parameters define the plane, the fourth can be set to 1 or 0
- d = 1 does not pass through (0, 0, 0)
- d = 0 pass through (0, 0, 0)

Normal

- Important concept in lighting and shading

- Normal vector
  - vector $\perp$ to the plane
  - "Unit vector" - L2 norm is 1.

- Solving for Normal
  - Normal **n** = (a, b, c)

  - Select 3 vertices on the plane **V1**, **V2**, **V3**

$$\mathbf{n} = (V2 - V1) \times (V3 - V1)$$

Distinguishing “Inside” from “Outside”

- Useful for “collision detection”
- Use (a, b, c)

$aX+bY+cZ+d$	> 0	Outside
	$= 0$	On the plane
	< 0	Inside

- Use **V1**, **V2**, **V3**

V1, **V2**, **V3**	selected	CCW	=>	Outside
		CW	=>	Inside

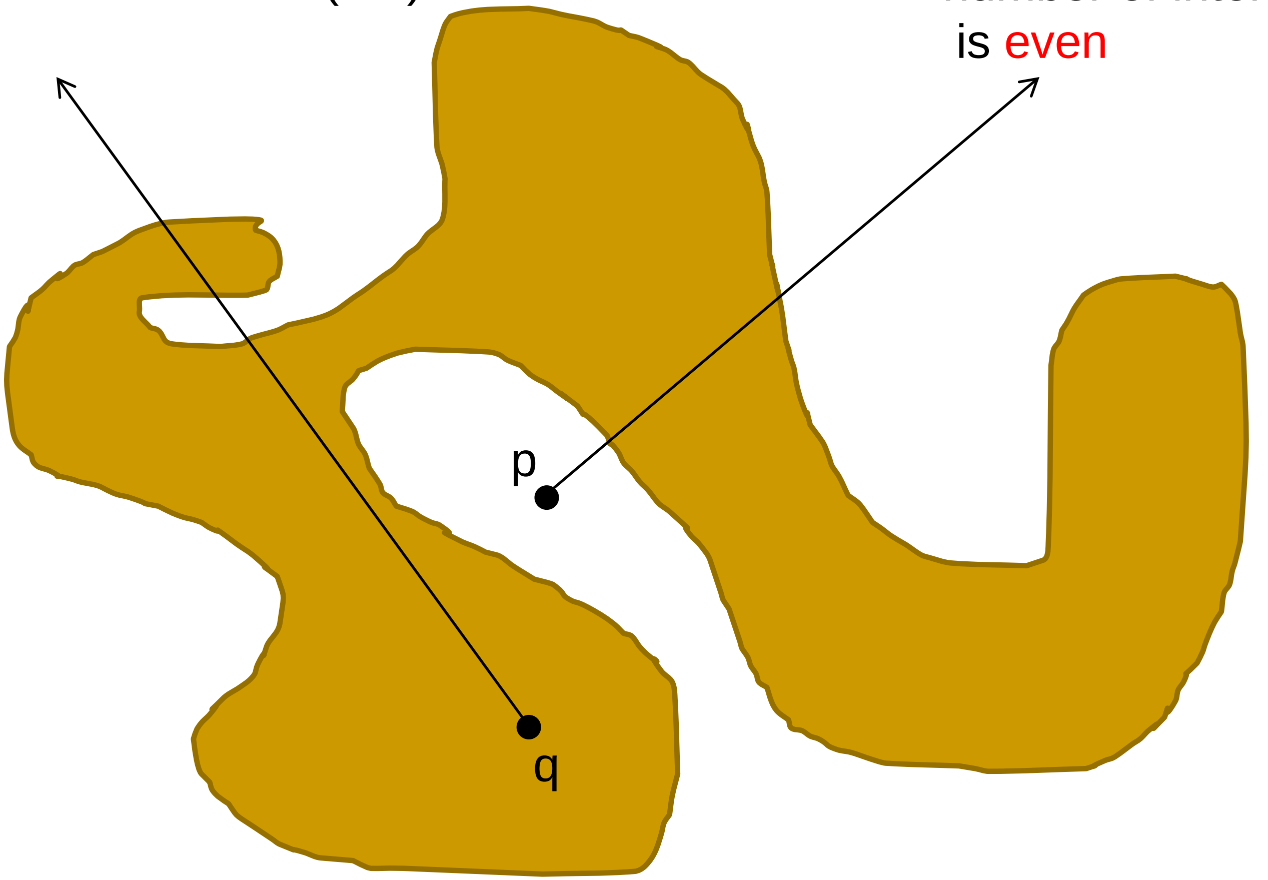
Inside-Outside Test

- To determine whether a pixel ***p*** is inside or outside an object ***S***
- Send a ray $p + t\ v$ which starts at the pixel, t is a scalar, ***v*** is an arbitrary direction vector
- Find all non-degenerate† intersections between the ray and ***S***
- *If the number of intersections is odd (even),* ***p*** *is inside (outside)* ***S***
- It is not easy to check non-degenerate intersections. One can solve this problem by sending out n rays in random directions and then use majority voting

† a degenerate intersection is one which the ray grazes the surface

Point q is inside as the number of intersections (= 3) is odd

Point p is outside as the number of intersections (= 2) is even



The yellow object is depicted as a 2D object but the technique can be applied to any n-dimensional object (n > 2)

Superquadrics

- 2D QUADRICS (conic section)

$$aX^2 + bY^2 + cXY + dX + eY + f = 0$$

- 3D QUADRICS

$$aX^2 + bY^2 + cZ^2 + dXY + eXZ + fYZ + gX + hY + iZ + k = 0$$

In 2D,

- Circle $$X^2 + Y^2 = r^2$$
- Ellipse $$\left(\frac{X}{a}\right)^2 + \left(\frac{Y}{b}\right)^2 = 1$$
- Parabola $$Y^2 = 4aX$$
- Hyperbola $$X^2 - Y^2 = r^2$$

In 3D

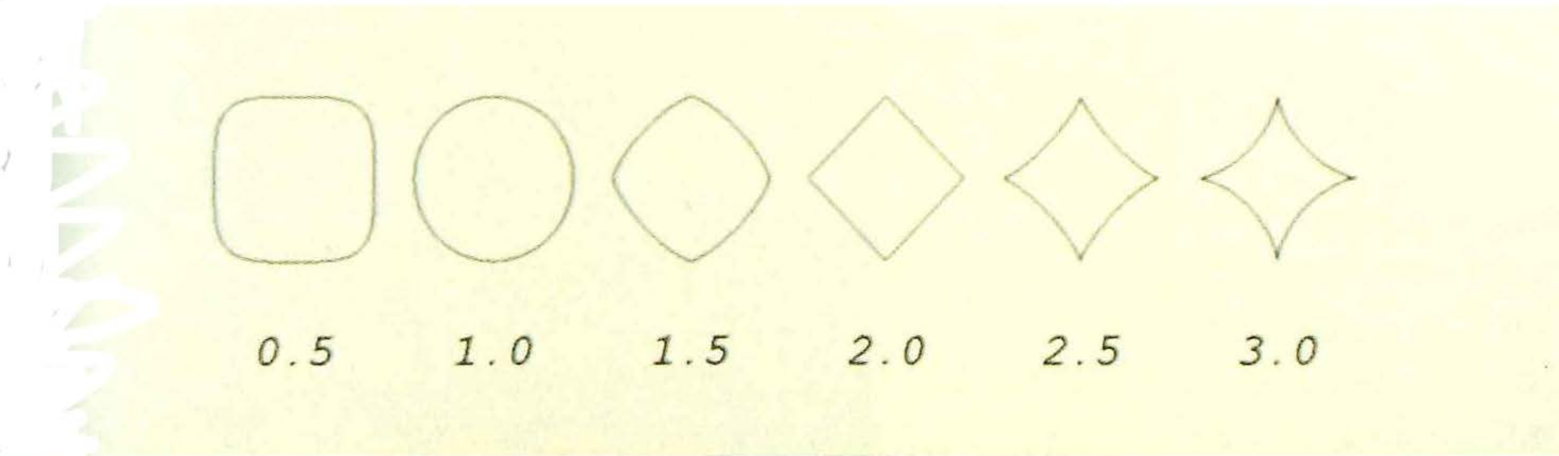
- Sphere $$X^2 + Y^2 + Z^2 = r^2$$
- Ellipsoid $$\left(\frac{X}{a}\right)^2 + \left(\frac{Y}{b}\right)^2 + \left(\frac{Z}{c}\right)^2 = 1$$
- Paraboloid ?
- Hyperboloid ?

(ans. to be discussed in tut.)

“Super”-quadrics

- Introduce to additional parameters s1 and s2
- Allow continuous transformation from “circle” to “square”   (Idiom)
- Example (2D)   “Super-ellipse”

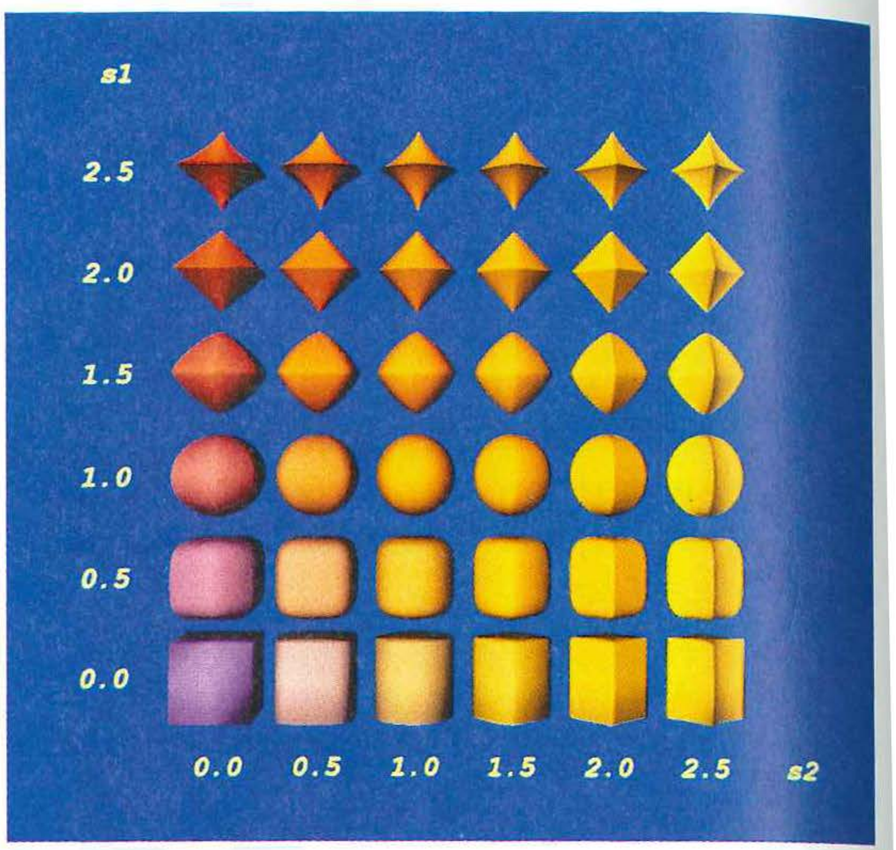
$$\left(\frac{X}{a}\right)^{\frac{2}{s}} + \left(\frac{Y}{b}\right)^{\frac{2}{s}} = 1$$



Superellipses plotted with values for parameter s ranging from 0.5 to 3.0 and with $r_x = r_y$.

Super-ellipsoid

$$\left[\left(\frac{X}{r_x}\right)^{2/s_2}+\left(\frac{Y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1}+\left(\frac{Z}{r_z}\right)^{2/s_1}=1$$



Superellipsoids plotted with values for parameters s_1 and s_2 ranging from 0.0 to 2.5 and with $r_x = r_y = r_z$.

Non-parametric and Parametric forms

- Non-parametric form
  - Z = f(X, Y)   or f(X, Y, Z) = 0
  - Used in mathematics

- Parametric form
  - Introduced two additional parameters u, v
  - X = f1 (u, v)   Y = f2 (u, v)   Z = f3 (u, v)
  - Used in CG

Parametric form of the super-ellipsoid

$$\left[\left(\frac{X}{r_x}\right)^{2/s_2}+\left(\frac{Y}{r_y}\right)^{2/s_2}\right]^{s_2/s_1}+\left(\frac{Z}{r_z}\right)^{2/s_1}=1$$

Non-parametric

$$X = r_x \cos^{s_1}\phi \cos^{s_2}\theta$$

$$Y = r_y \cos^{s_1}\phi \sin^{s_2}\theta$$

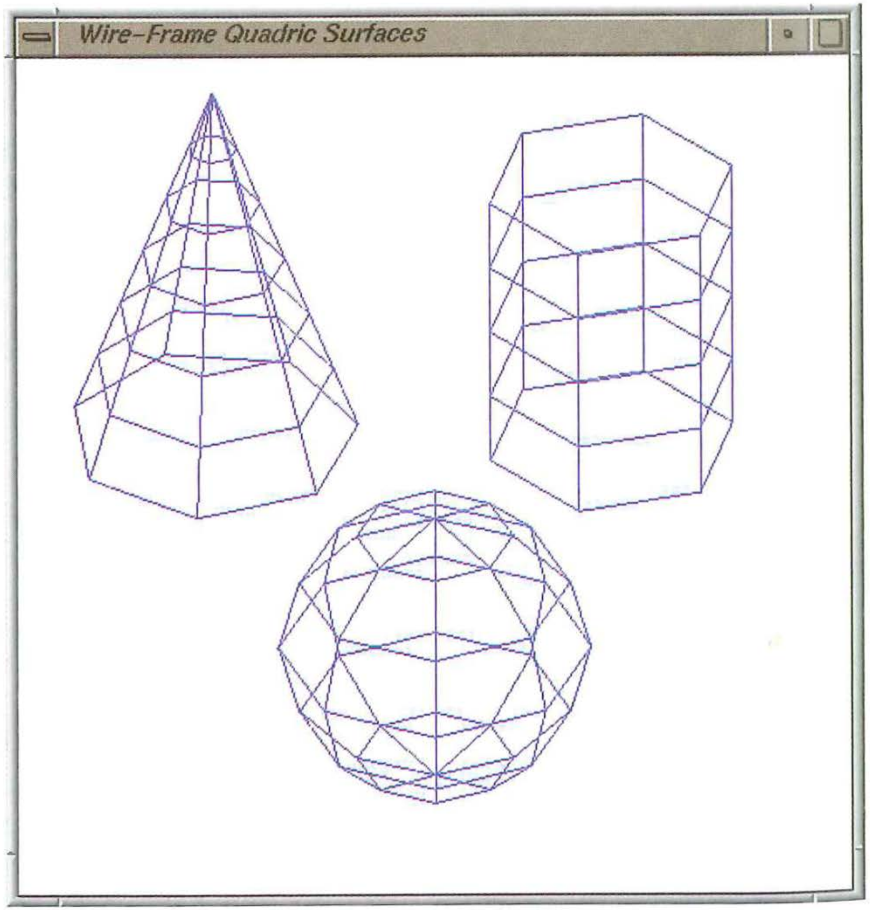
$$Z = r_z \sin^{s_1}\phi$$

Parametric

OpenGL functions

- Does not have superquadrics function
- Can display sphere, cone, cylinder
- Quadrilateral mesh

glutWireSphere (r, nLongitudes, nLatitudes)



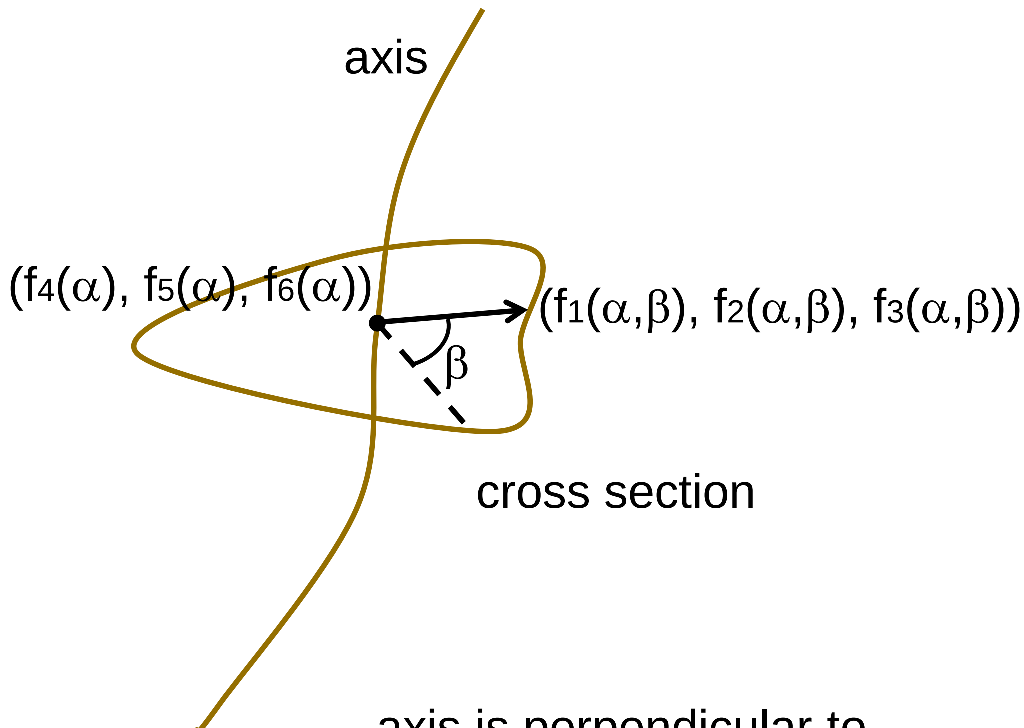
Display of a GLUT sphere, GLUT cone, and GLU cylinder, positioned within a display window by procedure `wireQuadSurfs`.

Generation of complicated shapes

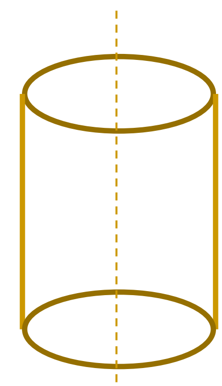
- Complicated shapes can be generated using quadrilateral mesh and parametric form
- Two examples are
  - Generalized Cylinder
  - Generalized Symmetry

Generalized Cylinder

real life example



primordial shape

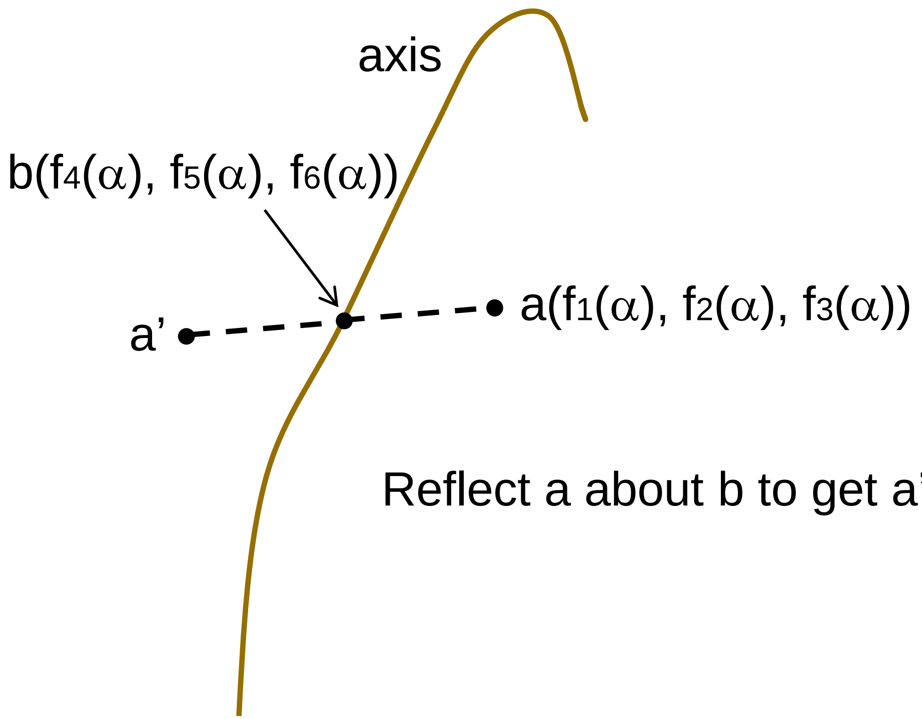


axis is perpendicular to
cross section

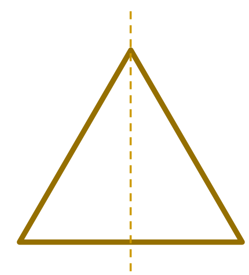
quadrilateral mesh parameterized by α and β

Generalized Reflectional Symmetry

real life example



primordial shape



quadrilateral mesh parameterized by α and β, with β varying linearly from a to a’

References

Ex: Practice using the index

For example, text

- OpenGL Line Functions Sec. 4-4
- Superquadrics: Sec. 13.4-13.5
- Parametric and non-parametric forms: A-8, A-9