POLY.MP: DEFINITION OF REGULAR CONVEX POLYHEDRA IN METAPOST

version 1.0, 8 April 1998 **Denis Roegel** (`roegel@loria.fr`)

This package defines the five regular convex polyhedra classes and must be used with the **3d** package.

Each object definition is parameterized by an instance identification.

All polyhedra are normalized (centered on the origin and inscriptible in a sphere of radius 1).

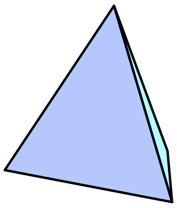
The polyhedra are defined with a certain number of points and faces. The points comprise both the vertices and the center of the object.

It might be useful to remind the Euler formula linking the number of faces f, vertices v and edges e in polyhedra with no hole in dimension 3: $f + v - e = 2$. This can be verified in the following table:

type	faces	vertices	edges
tetrahedron	4	4	6
cube	6	8	12
octahedron	8	6	12
dodecahedron	12	20	30
icosahedron	20	12	30

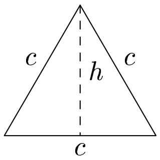
For more details, see the documentation enclosed in the distribution.

message `"*** poly, version 1.0 (c) D. Roegel, 8 April 1998 ***"`;



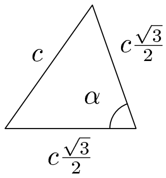
TETRAHEDRON (4 faces)

Construction of a tetrahedron:

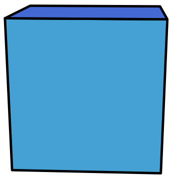
One face is 

with $h = c\sqrt{3}/2$

The angle α between two faces is given by $\sin(\alpha/2) = \dfrac{c/2}{c\sqrt{3}/2} = 1/\sqrt{3}$



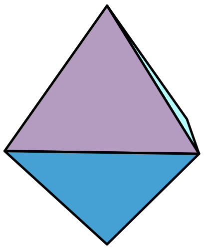

```
def set_tetrahedron_points(expr inst) =
    set_obj_point(1, 0, 0, 0);  set_obj_point(2, 1, 0, 0);
    set_obj_point(3, cosd(60), sind(60), 0); % 60 = 360/6
    sinan = 1/ sqrt(3);  cosan = sqrt(1 - sinan ** 2);
    an = 180 - 2 * angle((cosan, sinan));
    new_face_point(4, 1, 2, 3, an);
    normalize_obj(inst)(1, 2, 3, 4);
    set_obj_point(5, 0, 0, 0); % center of tetrahedron
enddef;

vardef def_tetrahedron(expr inst) =
    new_obj_points(inst, 5); % 4 vertices and the center
    new_obj_faces(inst, 4); % 4 faces in the definition
    set_tetrahedron_points(inst);
    set_obj_face(1, "1,2,4", "b4fefe");
    set_obj_face(2, "2,3,4", "b49bc0");
    set_obj_face(3, "1,4,3", "b4c8fe");
    set_obj_face(4, "1,3,2", "b4fe40");
enddef;
```

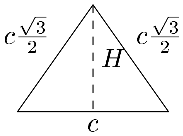
CUBE (HEXAHEDRON) (6 faces) 

The definition of a cube is straightforward.

def *set_cube_points*(**expr** *inst*) =
    set_obj_point(1, 0, 0, 0);
    set_obj_point(2, 1, 0, 0);
    set_obj_point(3, 1, 1, 0);
    set_obj_point(4, 0, 1, 0);
    set_obj_point(5, 0, 0, 1);
    set_obj_point(6, 1, 0, 1);
    set_obj_point(7, 1, 1, 1);
    set_obj_point(8, 0, 1, 1);
    normalize_obj(*inst*)(1, 7); % 1 and 7 are opposite vertices
    set_obj_point(9, 0, 0, 0); % center of cube
enddef;

vardef *def_cube*(**expr** *inst*) =
    new_obj_points(*inst*, 9); % 8 vertices and the center
    new_obj_faces(*inst*, 6); % 6 faces in the definition
    set_cube_points(*inst*);
    set_obj_face(1, "1,2,6,5", "b4fe40");
    set_obj_face(2, "2,3,7,6", "45d040");
    set_obj_face(3, "4,8,7,3", "45a114");
    set_obj_face(4, "1,5,8,4", "45a1d4");
    set_obj_face(5, "5,6,7,8", "4569d4");
    set_obj_face(6, "4,3,2,1", "112da1");
enddef;

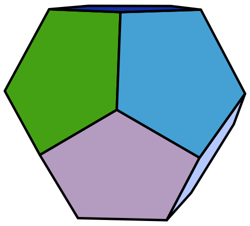
OCTAHEDRON (8 faces) 

A section of a pyramid is: 

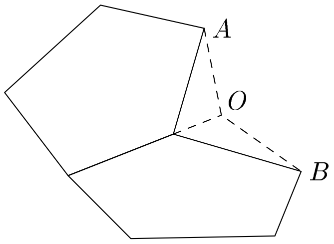
The height of this pyramid is given by $H^2 = (c\sqrt{3}/2)^2 - (c/2)^2 = c^2/2$, hence $H = c/\sqrt{2}$

```
def set_octahedron_points(expr inst) =
    set_obj_point(1, 0, 0, 0);
    set_obj_point(2, 1, 0, 0);
    set_obj_point(3, 1, 1, 0);
    set_obj_point(4, 0, 1, 0);
    set_obj_point(5, .5, .5, 1/ sqrt(2));
    set_obj_point(6, .5, .5, −1/ sqrt(2));
    normalize_obj(inst)(5, 6); % 5 and 6 are opposite vertices
    set_obj_point(7, 0, 0, 0); % center of octahedron
enddef;

vardef def_octahedron(expr inst) =
    new_obj_points(inst, 7); % 6 vertices and the center
    new_obj_faces(inst, 8); % 8 faces in the definition
    set_octahedron_points(inst);
    set_obj_face(1, "1,2,5", "b4fefe");
    set_obj_face(2, "2,3,5", "45d040");
    set_obj_face(3, "3,4,5", "4569d4");
    set_obj_face(4, "4,1,5", "b49bc0");
    set_obj_face(5, "6,1,4", "45a1d4");
    set_obj_face(6, "6,2,1", "b4c8fe");
    set_obj_face(7, "6,3,2", "b49b49");
    set_obj_face(8, "6,4,3", "112da1");
enddef;
```

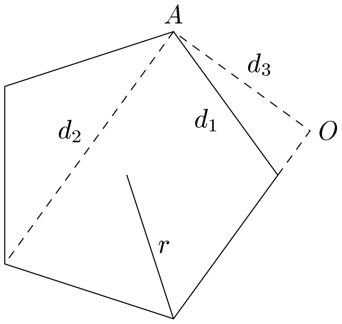
DODECAHEDRON (12 faces)



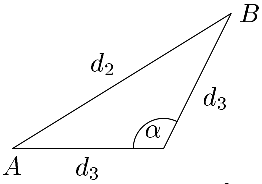
Two adjacent faces (pentagons) of the dodecahedron are as follows:



The angle between the faces is the angle between $\overrightarrow{OA}$ and $\overrightarrow{OB}$. $OA = OB = d_3$ as defined in



$d_1 = 2r\,\sin(\pi/5)$, $d_2 = 2r\,\sin(2\pi/5)$ and $d_3 = d_1\,\sin(2\pi/5) = 2r\,\sin(\pi/5)\,\sin(2\pi/5)$
The angle α is defined by the following conditions:

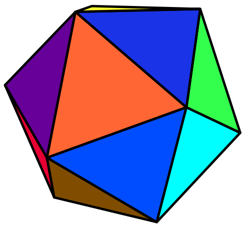


Thus $d_2^2 = d_3^2 + d_3^2 - 2d_3^2\,\cos\,\alpha$, which leads to $\cos\,\alpha = 1 - \dfrac{d_2^2}{2d_3^2} = -\dfrac{\cos(2\pi/5)}{2\,\sin^2(\pi/5)}$

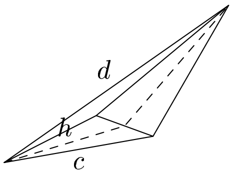
```
def set_dodecahedron_points(expr inst) =
    new_points(fc)(10); % face centers
    set_point(fc1, 0, 0, 0);  set_obj_point(1, 1, 0, 0);
    set_obj_point(2, cosd(72), sind(72), 0); % 72=360/5
    rotate_in_plane(3, fc1, 1, 2);
    rotate_in_plane(4, fc1, 2, 3);
    rotate_in_plane(5, fc1, 3, 4);
    cosan = - cosd(72)/(2 * sind(36) * sind(36));  sinan = sqrt(1 - cosan ** 2);
    an = 180 - angle((cosan, sinan));
    new_abs_face_point(fc2, fc1, 1, 2, an);
    new_abs_face_point(fc3, fc1, 2, 3, an);
    new_abs_face_point(fc4, fc1, 3, 4, an);
    new_abs_face_point(fc5, fc1, 4, 5, an);
    new_abs_face_point(fc6, fc1, 5, 1, an);
    rotate_in_plane(6, fc2, 2, 1);
    rotate_in_plane(7, fc2, 1, 6);
    rotate_in_plane(8, fc2, 6, 7);
    rotate_in_plane(9, fc3, 2, 8);
    rotate_in_plane(10, fc3, 8, 9);
    rotate_in_plane(11, fc4, 3, 10);
    rotate_in_plane(12, fc4, 10, 11);
    rotate_in_plane(13, fc5, 4, 12);
    rotate_in_plane(14, fc5, 12, 13);
    rotate_in_plane(15, fc6, 5, 14);
    new_abs_face_point(fc7, fc2, 6, 7, an);
    new_abs_face_point(fc8, fc3, 8, 9, an);
    new_abs_face_point(fc9, fc4, 10, 11, an);
    new_abs_face_point(fc10, fc5, 12, 13, an);
    rotate_in_plane(16, fc7, 6, 15);
    rotate_in_plane(17, fc7, 15, 16);
    rotate_in_plane(18, fc8, 7, 17);
    rotate_in_plane(19, fc9, 9, 18);
    rotate_in_plane(20, fc10, 11, 19);
    normalize_obj(inst)(2, 20); % opposite vertices
    set_obj_point(21, 0, 0, 0); % center of dodecahedron
    free_points(fc)(10);
enddef;
```

```
vardef def_dodecahedron(expr inst) =
    save cosan, sinan, an;
    new_obj_points(inst, 21); % 21 points in the definition
    new_obj_faces(inst, 12); % 12 faces in the definition
    set_dodecahedron_points(inst);
    set_obj_face(1, "5,4,3,2,1", "ff0fa1");
    set_obj_face(2, "8,7,6,1,2", "b40000");
    set_obj_face(3, "10,9,8,2,3", "b49b49");
    set_obj_face(4, "12,11,10,3,4", "b49bc0");
    set_obj_face(5, "14,13,12,4,5", "b4c8fe");
    set_obj_face(6, "6,15,14,5,1", "b4fefe");
    set_obj_face(7, "6,7,17,16,15", "b4fe40");
    set_obj_face(8, "8,9,18,17,7", "45d040");
    set_obj_face(9, "10,11,19,18,9", "45a114");
    set_obj_face(10, "11,12,13,20,19", "45a1d4");
    set_obj_face(11, "14,15,16,20,13", "4569d4");
    set_obj_face(12, "16,17,18,19,20", "112da1");
enddef;
```

ICOSAHEDRON (20 faces)

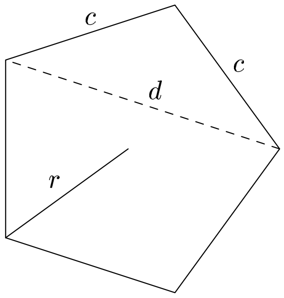


Two faces of an icosahedron are linked in the following way:



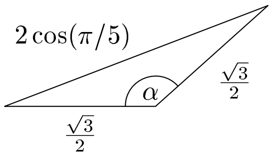
where $h = c\sqrt{3}/2$

d is also a diagonal in a pentagon:



$c = 2r\ \sin(\pi/5)$, $d = 2r\ \sin(2\pi/5) = 2c\ \cos(\pi/5)$

The angle between two faces is computed as follows:



We have $(2\ \cos(\pi/5))^2 = 2(\sqrt{3}/2)^2 - 2(\sqrt{3}/2)^2\ \cos\alpha$

Thus $cos\alpha = 1 - \frac{8}{3}\cos^2(\pi/5)$

```
def set_icosahedron_points(expr inst) =
    set_obj_point(1, 0, 0, 0);  set_obj_point(2, 1, 0, 0);
    set_obj_point(3, cosd(60), sind(60), 0); % 60=360/6
    cosan = 1 - 8/3 * cosd(36) * cosd(36);  sinan = sqrt(1 - cosan ** 2);
    an = 180 - angle((cosan, sinan));
    new_face_point(4, 1, 2, 3, an);
    new_face_point(5, 2, 3, 1, an);
    new_face_point(6, 3, 1, 2, an);
    new_face_point(7, 2, 4, 3, an);
    new_face_point(8, 3, 5, 1, an);
    new_face_point(9, 1, 6, 2, an);
    new_face_point(10, 3, 4, 7, an);
    new_face_point(11, 3, 7, 5, an);
    new_face_point(12, 1, 8, 6, an);
    normalize_obj(inst)(1, 10); % opposite vertices
    set_obj_point(13, 0, 0, 0); % center of icosahedron
enddef;

vardef def_icosahedron(expr inst) =
    save cosan, sinan, an;
    new_obj_points(inst, 13); % 12 vertices and the center
    new_obj_faces(inst, 20); % 20 faces in the definition
    set_icosahedron_points(inst);
    set_obj_face(1, "3,2,1", "b40000");
    set_obj_face(2, "2,3,4", "ff0fa1");
    set_obj_face(3, "3,7,4", "b49b49");
    set_obj_face(4, "3,5,7", "b49bc0");
    set_obj_face(5, "3,1,5", "b4c8fe");
    set_obj_face(6, "1,8,5", "b4fefe");
    set_obj_face(7, "1,6,8", "b4fe40");
    set_obj_face(8, "1,2,6", "45d040");
    set_obj_face(9, "2,9,6", "45a114");
    set_obj_face(10, "2,4,9", "45a1d4");
    set_obj_face(11, "9,4,10", "4569d4");
    set_obj_face(12, "4,7,10", "112da1");
    set_obj_face(13, "7,5,11", "b4fefe");
    set_obj_face(14, "5,8,11", "b49bc0");
    set_obj_face(15, "8,6,12", "45a114");
    set_obj_face(16, "6,9,12", "b49b49");
    set_obj_face(17, "8,12,11", "b40000");
    set_obj_face(18, "7,11,10", "45a1d4");
    set_obj_face(19, "12,10,11", "b4c8fe");
    set_obj_face(20, "9,10,12", "ff0fa1");
enddef;

endinput
```