Generating and Rendering Four-Dimensional Polytopes

John M. Sullivan, Geometry Supercomputer Project
Current affiliation: Dept. of Mathematics, Univ. of Illinois, Urbana

November, 1990

Abstract

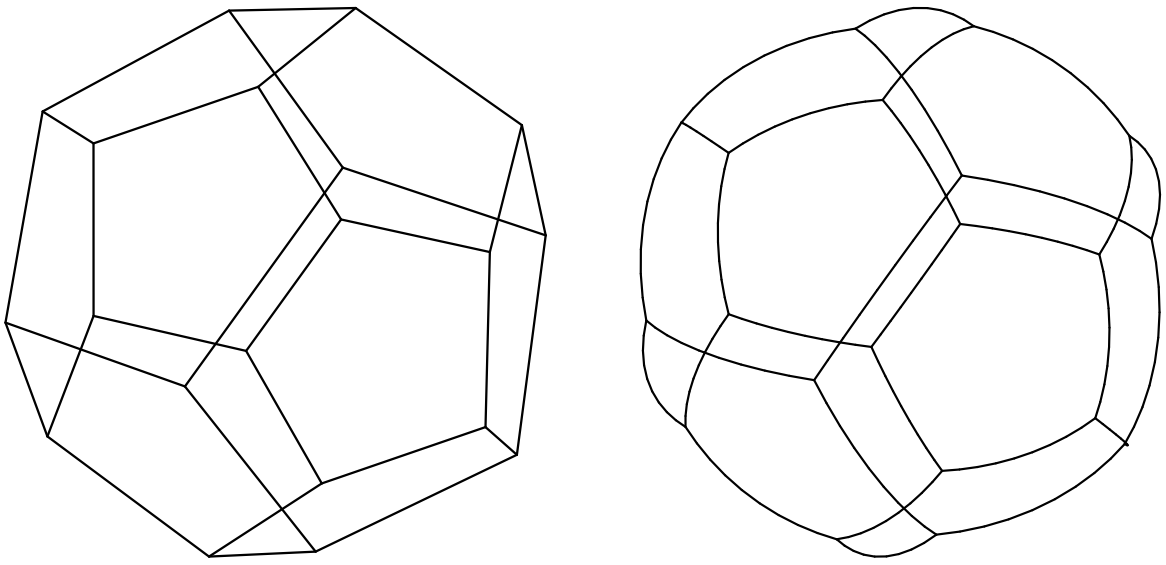
The regular polytopes in four dimensions can be generated easily by the judicious use of their symmetry groups, and can be rendered in three dimensions in stereographic projection. In this article we construct one such polytope, introducing the relevant group-theory concepts as needed. The result can be viewed with *Mathematica* graphics, or with a more sophisticated renderer such as RenderMan.

Regular Polytopes and Soap Bubbles

The regular polyhedra—the tetrahedron, the cube, the octahedron, the dodecahedron and the icosahedron, collectively known as the platonic solids—have long been appreciated for their beauty and symmetry. The first three of them have analogues in all dimensions higher than three; they are called the simplex, the hypercube and the orthoplex. Only in four dimensions do we find counterparts for the icosahedron and dodecahedron, and even a sixth polytope, the 24-cell, with no counterpart in any other dimension. (*Polytopes* are the generalization of polyhedra to higher dimensions. Their *cells* correspond to a polyhedron's faces: for a four-dimensional polytope, cells are three-dimensional "faces," and so on. An excellent reference on regular polytopes is [Cox].)

In the tetrahedron, cube and dodecahedron, three faces meet around every vertex. The three lines emerging from each vertex are at equal angles; if we project them to the circumscribed sphere, they become arcs of great circles, meeting three to a vertex, at 120° angles (Figure 1). Because these edges meet at 120° angles and are "straight" (they are shortest paths along the surface of the sphere), this configuration can be physically realized by soap films: one might imagine, for example, two concentric spherical glass shells separated by a thin air cushion, with thin strips of soap film perpendicular to the shells in the pattern of Figure 1(b).

Figure 1(b) hides the fact that the spherical dodecahedron is really a two-dimensional object, since it lies on the surface of a sphere. All the information contained in this figure can be preserved if we map the sphere to the plane, just as cartographers do, and consider the resulting edges in the planar map. A convenient map is the so-called *stereographic projection*, illustrated in Figure 2(a). A point on the sphere—say, the north pole—is chosen as the center of projection, and points on the sphere are mapped to points on the equatorial plane along lines from the center of projection.



(a) (b)

Figure 1: Projecting a dodecahedron in space (a) onto the circumscribed sphere gives a spherical dodecahedron (b).

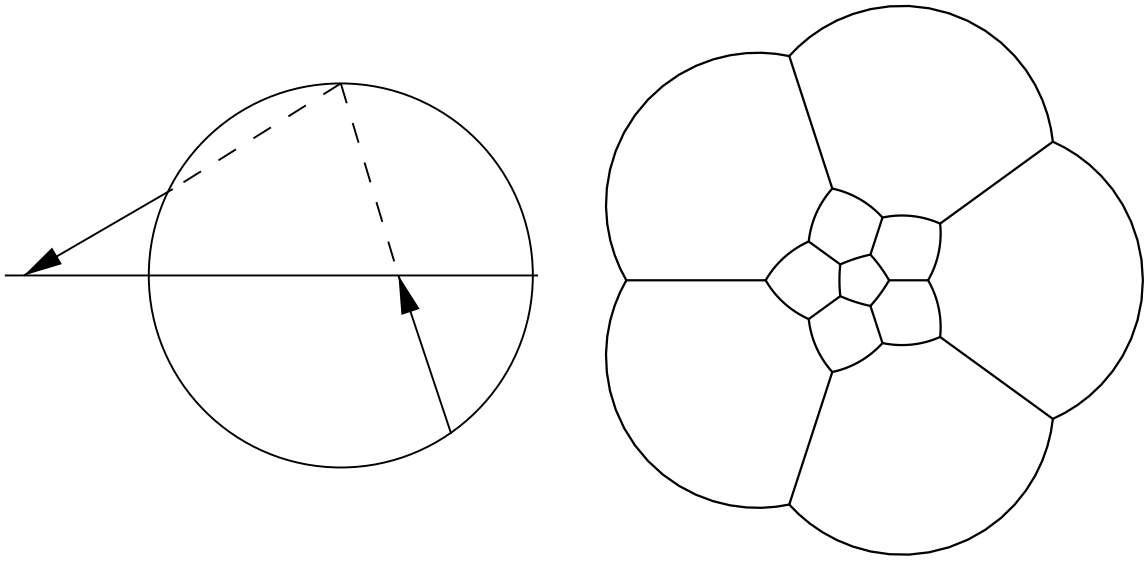
Stereographic projection preserves angles and takes circles to circles, so the projected image of the spherical dodecahedron, shown in Figure 2(b), is a figure that is still realizable by means of soap bubbles (this time between flat sheets of glass, for example).

Soap films have the shapes they do because surface tension tries to minimize their area. The transverse effect of surface tension on a surface is proportional to its *mean curvature*. In a bubble cluster this force is balanced by any difference in pressure between the two sides of the film. For a planar cluster (trapped between two plates) this means that each film must have constant curvature, that is, must describe an arc of a circle or a line segment. For soap bubbles in space, it means that the surfaces must have constant mean curvature; the most common shape having this property is the sphere.

Furthermore, where different sheets meet, they must meet three at a time, at equal angles. This condition arises intuitively from the need to balance tension forces at such a junction [Tho, Boy], but its necessity was only recently proven mathematically [Tay]. Taylor also showed that the triple junctions can only meet in fours, at tetrahedral angles.

In three out of the six four-dimensional regular polytopes, each edge is shared by three cells and each corner by four. These polytopes can be visualized in terms of soap bubbles in our familiar three-dimensional Euclidean space, by a process similar to the one described above: central projection onto the circumscribed sphere, followed by stereographic projection onto Euclidean space of one lower dimension. The result is a structure that meets the necessary conditions for the geometry of a soap bubble cluster.

The color plate at the end illustrates the result of this process for the four-dimensional analogue of the dodecahedron. This polytope is known as the 120-cell or the dodecaplex, because it is made of 120 dodecahedral "faces." Its vertices can be thought of as lying on the three-sphere, the set of all points (w, x, y, z) at unit distance from the origin. Since every dodecahedron is bounded by



(a) (b)

Figure 2: Stereographic projection (a) allows us to display the spherical dodecahedron of Figure 1(b) as a planar, though distorted, object (b).

twelve pentagons, each separating two dodecahedra, there are 720 pentagons in all. Each edge of a pentagon is shared by three pentagons, meeting at equal angles; there are thus 1200 edges. These edges meet four at a time at 600 vertices, each shared by four dodecahedra.

Each of the 120 cells is rendered under stereographic projection as a dodecahedron whose faces are pieces of spheres, meeting at 120° angles. While stereographic projection preserves angles, it distorts distances (it is a simple geometric fact, and a nuisance to cartographers, that no map from the sphere to the plane can preserve distances). Thus, in Figure 2(b), the inner pentagons are much smaller than the outer ones. Similarly, in our cover illustration, bubbles near the center are much too small to be seen. To better understand the picture, we organize the 120 cells into layers.

Invisible in the center is a single dodecahedron, which bulges a tiny bit (a dodecahedron with flat faces has dihedral angles of approximately 116.565°, so increasing them to 120° requires curving the faces only slightly). We surround it with a shell of twelve dodecahedra, one touching each face of the central cell. This shell (shown in Figure 5) has dimples above each vertex of the central cell, so there is room for a second shell of twenty. The third shell again has twelve cells, above the faces of the central cell. All of these bubbles are too small to be discerned in the cover picture. There follows a middle layer of 30 cells; the smallest clearly visible bubbles in the center of the picture belong to this layer.

The remaining three layers mirror the first three, in reverse. From the outside in, we have the twelve largest, round bubbles, then twenty smaller ones, with noticeable outward points, then again twelve, after which we encounter the middle layer again. The last dodecahedron, mirroring the tiny one in the center, is infinite: like the outermost pentagon in Figure 2(b), it contains the center of projection, which has no finite image. In four dimensions, of course, it is just like any other dodecahedron: it is surrounded by its twelve neighbors, which are the largest bubbles in the projection.

In the remainder of this article, we develop the *Mathematica* code used to create the cover picture, starting with the simpler case of a dodecahedron. We introduce the necessary geometrical properties of the dodecahedron and of the 120-cell as we go along; at the end, we mention some more advanced mathematical properties of the 120-cell. Finally, we discuss the rendering of the cover picture in some detail.

The Dodecahedron

Mathematica comes with two packages defining the dodecahedron, `Graphics/Polyhedra.m` and `Geometry/Polytopes.m`. The first includes information about the connectedness of the vertices, but just lists the coordinates as real numbers; the second generates the vertices intelligently, but doesn't know how they are arranged. We will write our own code to get the advantages of both, and to prepare the ground for the 120-cell.

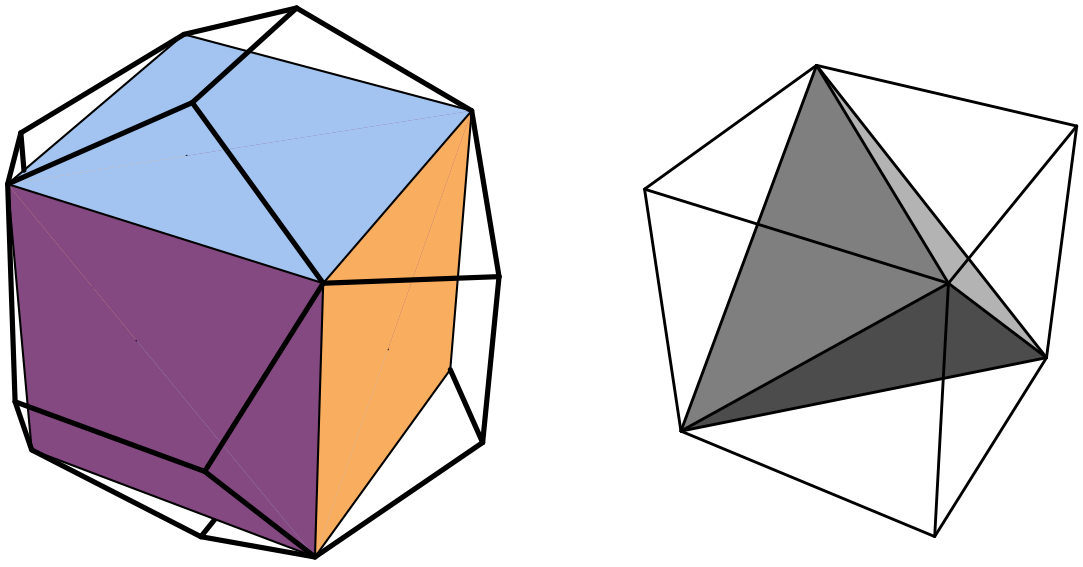
The dodecahedron has twenty vertices, thirty edges, and, as its name implies, twelve faces. It has a large amount of symmetry: altogether, there are 120 symmetry transformations (rigid motions or reflections) that take the dodecahedron to itself. We will use these symmetries to generate the geometry. More discussion of the symmetry of the dodecahedron and related figures can be found in [LGM].

To better understand this *symmetry group*, let's examine those of some simpler objects. A line segment can be taken to itself in two ways: we can either fix it, or switch the ends. This group of two elements is called Z_2. A regular polygon with k edges has $2k$ symmetries, because a given edge can be taken into any of the k edges, in either of the two orientations allowed by Z_2. Thus a pentagon has ten symmetries, while a square has eight. The symmetry group of a k-sided regular polygon is called a dihedral group, and denoted by D_k.

To count the symmetries of the dodecahedron, note that any pentagonal face can be taken to any other one, in the ten ways allowed by the symmetries of the pentagon. Thus the symmetry group has order 120; it turns out to coincide with S_5, the group of permutations of five objects. To see this, consider Figure 3(a), which shows how a cube can be inscribed in the dodecahedron, the edges of the cube being diagonals of the dodecahedron's faces. Notice how we can get the dodecahedron by pulling one line segment away from each of the six faces of the cube. The three pairs of opposite and parallel edges pulled out in this way look special in this figure, but because all edges are really equivalent, we see that there are five such sets of six edges, each corresponding to an inscribed cube.

Each vertex of the dodecahedron occurs in two of these five cubes, corresponding to the two ways of drawing a pentagon diagonal from that vertex. We should instead look only at a tetrahedron within each cube, as shown in Figure 3(b): then each vertex is in only one tetrahedron and the twenty vertices are divided into five sets of four. These five sets are acted on by the symmetry group of the dodecahedron, and in fact they can be permuted in any way, showing that the group is indeed S_5 as claimed earlier. (If we consider instead the action on the five cubes, we get each *even* permutation twice, because central inversion fixes each cube.)

The trick of considering an inscribed cube also leads to some symmetry subgroups that are easier to work with than the full symmetry group of the dodecahedron. If we take the cube of Figure 3 to be centered at the origin, with edges parallel to the coordinate axes, its symmetries are very easy to express in coordinates. In fact, the vertices have all coordinates ± 1; any permutation of



(a) (b)

Figure 3: One of the five ways to inscribe a cube in a dodecahedron (a), and one of the two ways to inscribe a tetrahedron in a cube (b).

the coordinates is a symmetry, as are sign changes of the various coordinates, which correspond to reflections in the coordinate planes.

Whenever we have a group G of permutations of n objects, and some group H of states that each object can be in (for example, $H = Z_2 = \{\pm 1\}$, changing the sign of a coordinate), we can form a (semi-direct) product $G \ltimes H^n$, by allowing G to permute the states as it permutes the objects. The symmetry group of the cube (permutations and sign changes of three coordinates) is therefore $S_3 \ltimes Z_2^3$. For a hypercube of any dimension n, it is $S_n \ltimes Z_2^n$.

This group contains two subgroups of *index two* (that is, of half the size), which are easy to confuse. Any permutation has a parity, even or odd, and so does a sign change in Z_2^n—just count the number of signs switched. The first subgroup of index two of $S_n \ltimes Z_2^n$ is the subgroup of rotational symmetries of the hypercube: it contains only even permutations with even sign changes, and odd permutations with odd sign changes. This is because odd permutations, like odd sign changes, reverse handedness. For a three-dimensional cube, the group of rotations happens to equal S_4, the permutations of four objects, acting (for example) on the four body diagonals of the cube. It contains some elements that are not symmetries of the dodecahedron, such as rotations of order four.

The second subgroup of index two allows only even permutations, no matter how many signs change. This group can be written as $A_n \ltimes Z_2^n$, where A_n is the usual name for the *alternating group* of all even permutations. In three dimensions, $A_3 = C_3$ corresponds to three-fold rotations around the axis from $(-1,-1,-1)$ to $(1,1,1)$. The index-two group, then, can be written $C_3 \ltimes Z_2^3$; unlike the subgroup of the previous paragraph, it does contain only symmetries of the dodecahedron.

To obtain the full symmetry group of the dodecahedron, we might add to this group some symmetry that does not preserve the cube—for example, an order-five rotation preserving a face of the dodecahedron. But writing such a transformation in coordinates is messy, and quite inefficient

for our purposes, as we don't need the full symmetry group S_5 to generate the dodecahedron. If we applied S_5 to one vertex, we would get every vertex six times; one face would generate each face ten times.

On the other hand, applying just a subgroup to one vertex need not generate all the vertices. Our subgroup for instance divides the vertices into two types, called *orbits*: those that lie on the cube have eight images under this group, and those that don't have twelve. (The difference is caused by the fact that the subgroup is not normal in S_5.) Starting with one vertex of each type and applying our subgroup $A_3 \ltimes Z_2^3$, we get all vertices with fewer duplications than we would have if we used the entire symmetry group. This subgroup has no special significance for the dodecahedron: we choose it for its simplicity in our coordinate system.

To obtain coordinates for the two initial vertices, we take the vertices of the cube to be $(\pm 1, \pm 1, \pm 1)$; thus the circumscribed sphere has radius $\sqrt{3}$. Looking at Figure 3(a), we see that a typical vertex of the second type, with coordinates $(0, a, b)$ such that $0 < a < 1 < b$, is characterized by two conditions: it should lie on the sphere of radius $\sqrt{3}$ as well, and the distance between it and its image $(a, b, 0)$ under a coordinate permutation should be the diagonal of a pentagon, or the side of the cube. We can find a and b using *Mathematica*:

```
dist2[x_List,y_List] := (x-y).(x-y);
solns = Solve[ dist2[{0,a,b},{0,0,0}]==3
            && dist2[{0,a,b},{a,b,0}]==4 ];
```

The answer is messy, but if you apply `N` you may recognize that $b = \tau$ and $a = \tau - 1 = 1/\tau$, where $\tau = \frac{1}{2}(1 + \sqrt{5})$ is the golden ratio. Throughout this article we will use `tau` instead of `GoldenRatio` in our code, to save typing. To inform *Mathematica* of the numerical value of τ, we set

```
N[tau,n___] ^:= N[GoldenRatio,n]
```

Because τ satisfies a quadratic equation, $\tau^2 = \tau + 1$, any rational expression in τ can be reduced to some rational linear combination of 1 and τ. Because we will be making much use of τ, we would like to teach *Mathematica* to make these simplifications. The following rules clearly follow from the equation for τ^2:

```
taurepl = {
  (1+tau)^x_ -> tau^(2x),
  tau^(n_Integer) ->
    If[n > 1, tau^(n - 1) + tau^(n - 2),
    If[n < 0, tau^(n + 2) - tau^(n + 1),
    tau^n]]
};
```

We could use `AlgebraicRules[tau^2==tau+1]` to generate a rule quite like this entire set, but it would not work well with expressions other than polynomials in `tau`. Instead, we use the following function, which repeatedly applies `taurepl` and often will completely reduce an expression in `tau` to the standard form:

```
tauexp[x_] :=
  FixedPoint[
```

```
    Map[Expand,
        # /. {a_^b_ :> Factor[a]^b} //. taurepl,
        {0,Infinity}] &,
    x];
```

With this function, for instance, we can have *Mathematica* simplify the result `solns` above for us, without having to look at numerical approximations:

```
First[solns] /. Sqrt[5]->(2tau-1) // tauexp;
```

Now we are ready to generate the dodecahedron. We will implement a group as a function that takes a point in space and returns a list of all its images under the group. We can write functions for cyclic and dihedral groups that will also work later for the four-dimensional case, and a general function `applysigns` that can produce a group from any list of sign changes. The built-in function `Permutations` implements the full symmetric group in our sense. The functions `C3` and `Z23` implement the groups $C_3 = A_3$ and Z_2^3.

```
cyclic[l_List] := NestList[RotateLeft, l, Length[l]-1];
dihedral[l_List] := Join @@ cyclic /@ {l, Reverse[l]}

applysigns[s_][x_] := (Distribute /@ (x #))& /@ s;
signs[n_Integer] :=
  Flatten[Array[ (-1)^{##}&, Table[2,{n}], 0], n-1]

Z23 = applysigns[signs[3]];
C3 = cyclic;
```

If `gp` is a group function, `images[gp]` is another function, which takes a list of points and returns all images of all the points, with duplicates removed. To get the list `v` of vertices of the dodecahedron, we take the images of the two initial vertices under C_3, then the images of the resulting vertices under Z_2^3:

```
images[gp_][xs_List] := Union @@ gp /@ xs;
dodecV0 = {{1,1,1},{0,tau-1,tau}};
dodecV = images[Z23] @ images[C3] @ dodecV0;
```

We can find the edges and faces of the dodecahedron similarly. We represent these, respectively, as lists of two and five vertices. A group can be applied to such lists by applying it to each vertex. Given a group function `gp` as above, `dilist[gp]` is a new function that works on lists. We have to be careful about removing duplicates: in our representation of a face, only the cyclic order of vertices matters, not which one comes first. Thus, before applying `Union`, we put each face in standard order, using the function `diSort`, which chooses the lexicographically least among all dihedral rearrangements of its argument, just as the usual `Sort` can be thought of as choosing the least among all rearrangements.

```
dilist[gp_] [f_List] := Union [ diSort /@ Transpose[gp /@ f] ];
diSort[f_List] := First @ Sort @ dihedral[f];
dicompose[G_,H_][f_List] := Union @@ dilist[G] /@ dilist[H] @ f;
half3cube = dicompose[C3,Z23];
```

In Figure 3(a) we see that there are two kinds of edges: those touching a corner of the cube, and those pulled out from the cube faces. These are two orbits under the half cube symmetry group we are using. The faces, however, all look the same in the figure—they are all in one orbit.

```
dodecE0 = dodecV0;
dodecE1 = {{tau-1,tau,0}, {1-tau,tau,0}};
dodecE = Join[half3cube[dodecE0], half3cube[dodecE1]];
dodecF0 = {{1,1,1}, {0,tau-1,tau},
    {0,1-tau,tau}, {1,-1,1}, {tau,0,tau-1}};
dodecF = half3cube[dodecF0];
```

Now `dodecE` is a list of the thirty distinct edges of the dodecahedron. To draw Figure 1(a), it is enough to say

```
SetOptions[Graphics3D,Boxed->False];
Show[Graphics3D[Line /@ dodecE]];
```

Figure 1(b), the spherical dodecahedron, uses the same data, but we must render each edge as an arc of great circle, by projecting out onto the circumscribing sphere. Here again we write general code that will be useful later:

```
tosphere[x_?VectorQ] := x / Sqrt[x.x];
interp1[p_,q_,lambda_] := lambda p + (1-lambda) q;
divide[edge_List,n_Integer] := tosphere /@
  Table[interp1 @@ Append[edge,t], {t,0,1,1/n}];
segments[l_List] :=
  Rest @ Transpose @ ({#,RotateRight[#]}&) @ l;
sphere[edge_List] :=
  Line /@ segments[divide[edge,9]];
Show[Graphics3D[Join @@ sphere/@dodecE]];
```

Note that the construction `interp1 @@ Append[edge,t]` applies `interp1` to the two vertices of `edge` and the number `t`. Another way to write this would be `interp1[Sequence@@edge,t]`. `Sequence`s are strange objects, hard to print out with `FullForm`, but nonetheless useful (see [Mae], for example).

The additional *Mathematica* code needed for the stereographic projection of Figure 2(b) is the following:

```
stereo[S][x_?VectorQ] := 2 Rest[x]/(1+First[x]);
stereo[pole_?VectorQ][x_?VectorQ] :=
  Block[{yy=Array[y,Length[pole]],t},
    yy/. First@Solve[{yy==t x+(1-t)pole, yy.pole==0},
                Prepend[yy,t]] ]

stereop[pole_][edge_List] :=
  Line @ (stereo[pole] /@ divide[edge//N,9]);
```

```
mean[x_List] := (Plus @@ x) / Length[x];
center = tosphere[mean[dodecF0]]//N;

Show[Graphics3D[stereop[center] /@ dodecE],
        ViewPoint->center];
```

Here the function `stereo` does stereographic projection from a given `pole`. To get a symmetric projection of the dodecahedron, we to project from the center of one face, rather than the usual south pole. Our function `stereo`, applied from an arbitrary pole, leaves its result in a plane in the higher-dimensional space; we must set the `ViewPoint` to see the picture properly. Called with the symbol `S` as the pole, it projects from the south pole $(-1, 0, \ldots, 0)$. The special code here is much faster, and truly projects to a lower dimension; we will need this for projections from four-space.

Generating the 120-cell

Every regular polytope can be described by a small graph (in fact a chain), called its Coxeter diagram. For a regular polygon, the Coxeter diagram consists of a single edge, labeled with the number of sides of the polygon; one of the diagram's two nodes is distinguished, and is generally marked with ◉. The Coxeter diagram of a higher-dimensional polytope is obtained by extending the diagram of one of the polytope's top-dimensional cells or faces with an edge, labeled with the number of these cells meeting at a time. The new edge is added at the non-distinguished end of the chain.

Thus the Coxeter diagram of the dodecahedron is ◉$\overset{5}{\bullet}\!\!-\!\!\bullet\overset{3}{}$, since the faces are pentagons meeting three around a vertex, while the diagram of the 120-cell is ◉$\overset{5}{\bullet}\!\!-\!\!\bullet\overset{3}{}\!\!-\!\!\bullet\overset{3}{}$, because its cells are dodecahedra meeting three around an edge. The numbers from the Coxeter diagram are often used to name the polytopes; in [Cox], the 120-cell is called $\{5, 3, 3\}$.

Coxeter diagrams are useful because they give a recipe to generate the polytope's symmetry group using reflections: take one mirror for each node, including the distinguished one, and place them so that mirrors not adjacent in the diagram are at right angles to each other, while any pair connected by an edge labeled k forms an angle π/k. If we start with the Coxeter diagram for an n-dimensional polyhedron, which has n nodes, we find that the mirrors can be arranged in a unique way in n-space, while they would not fit in lower dimensions. For all three-dimensional polyhedra, we can describe the three mirrors by starting with a triangle consisting of a vertex, the midpoint of an incident edge, and the center of an adjacent face. The mirrors are the planes through the origin and the three edges of this triangle. If you look toward the origin from within the cone enclosed by the mirrors, you see figures with the appropriate symmetry, somewhat as if looking into a kaleidoscope.

The symmetry group of the 120-cell, then, can be built up from four reflections in four-space. But again, the entire group is too big to work with—it has order 120^2, since given one of the 120 cells, it can be taken to any other (or itself), in any one of 120 ways (this being the order of the dodecahedral symmetry group). Note that this method of computing the order of a symmetry group, which we have used before, is reminiscent of the Coxeter mirror construction, in that we build from the symmetry group of top-dimensional cells.

We would like to orient the 120-cell in such a way that some subgroup of its symmetries is easily implemented. We can start by putting one dodecahedral cell (oriented as before in its own three dimensions) along the additional w coordinate axis, that is, around the north pole of the three-sphere. We will not use the unit sphere, but, as for the dodecahedron, some larger sphere, to make the numbers nicer. So the coordinates of the first dodecahedron are

```
verts = Prepend[#,a]& /@ dodecV;
```

for some a. One way to find a is to check that distances along diagonals are equal: if we reflect this dodecahedron across one of its faces, we find 15 new vertices of the 120-cell, and these must be at the right distances from the original ones. The following code finds the reflected vertices: `normal` finds a normal vector to the hyperplane passing through the origin and its $n-1$ arguments (using an extension of the determinant formula for cross products). Once we make it a unit normal `u`, `reflect[u]` will reflect other vectors through that hyperplane.

```
normal[s__] :=
  Det[{s,#}]& /@ IdentityMatrix[Length[{s}]+1];
reflect[u_List] [x_List] := x - 2 (x.u) u;
refl1 = reflect[
  tosphere[ normal @@ verts[[{8,12,20}]] ] //tauexp ];
rverts = refl1 /@ verts //tauexp;
dist2[ verts[[16]],rverts[[16]] ] == 4;
```

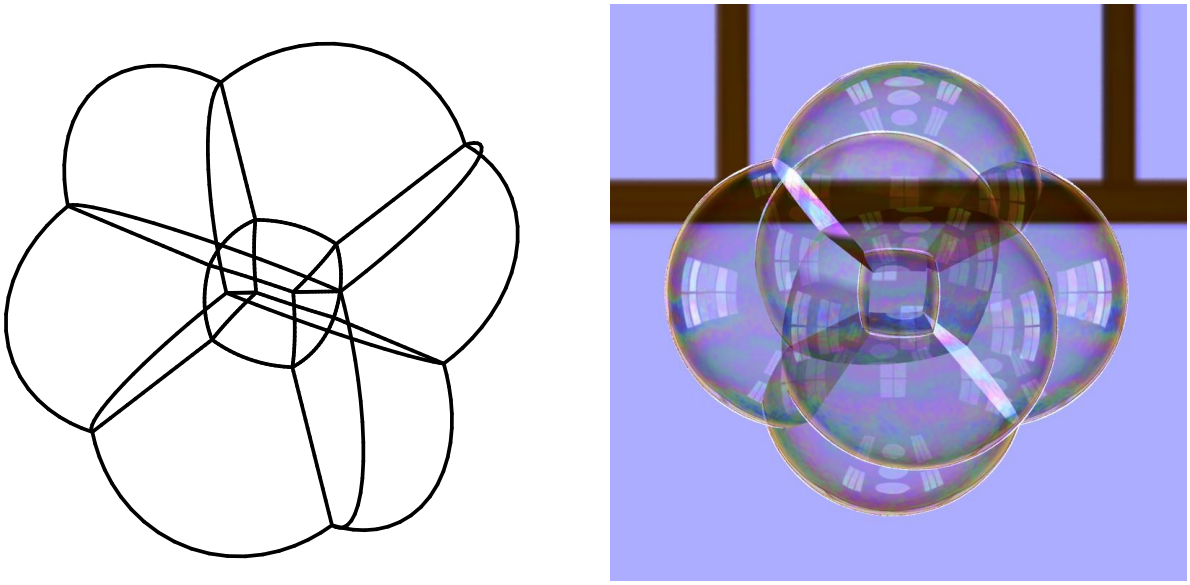
Solving this equation for a gives the right answer, although not in the simple form $1+2\tau$. (Even applying `tauexp`, we do not get this expression, since we have not taught the system how to take square roots like $\sqrt{34-21\tau} = 5-3\tau$.) Entering the correct value of a, we should simplify the expressions for `refl1` and `refl2` again:

```
a = 1 + 2tau;
refl1 = refl1 // tauexp ;
rverts = rverts // tauexp;
```

This one reflection `refl1`, together with the original dodecahedral symmetries, is enough to generate the entire 120-cell, because it gives reflections in all other faces. We could generate all the vertices from the ones we have so far by repeated applications of `refl1` and our dodecahedral symmetry subgroup. But this would proceed slowly, following the layers we mentioned in the description of the cover illustration.

Instead we will again use a relation to the cubic symmetry group. We don't have a picture like Figure 3(a) to guide us; in fact we are writing the code in order to be able to generate such pictures. But we might imagine that, just as in three dimensions, half of the symmetries of the hypercube are symmetries of the 120-cell. The hypercube has symmetry group $S_4 \ltimes Z_2^4$ as mentioned before—this includes all permutations and sign changes of the coordinates. We will implement the subgroup with only even permutations.

```
Z24 = applysigns[signs[4]];
compose[G_,H_][x_List] := Join @@ G /@ H[x];
C2a[x_List] := {x,RotateLeft[x,2]};
```



(a) (b)

Figure 4: The stereographic projection of the hypercube can be rendered by drawing the edges (a) or the faces (b).

```
C2b[x:{a_,b_,c_,d_}] := {x,{b,a,d,c}};
V = compose[C2a,C2b];  (* Klein four-group *)
A3[x_] := Prepend[#,First[x]]& /@ cyclic[Rest[x]];
A4 = compose[A3,V];
half4cube = dicompose[A4,Z24];
```

In fact, even though this is only half of the symmetry group of the 4-cube, it is enough to generate its vertices, edges and faces, given one of each.

```
hypercubeV0 = {1,1,1,1};
hypercubeV = images[A4] @ images[Z24] @ hypercubeV0;
hypercubeE0 = Take[hypercubeV,-2];
hypercubeE = half4cube[hypercubeE0];
hypercubeF0 = hypercubeV[[{-1,-2,-4,-3}]];
hypercubeF = half4cube[hypercubeF0];
```

We can draw these edges with the same stereographic projection routines used for the dodecahedron, since they were written to work in arbitrary dimensions. Figure 4(a) was generated with

```
Show[Graphics3D[stereop[S] /@ hypercubeE]];
```

The faces were used with the graphics code described later to produce Figure 4(b), which shows the hypercube as a soap bubble.

The hypercube has eight cubic cells, one along each direction of each coordinate axis in four-space. In building the 120-cell, we have placed a dodecahedral cell in the center of one of these cubes, somewhat as in Figure 3(a), but with a much smaller dodecahedron. This cell has eight images

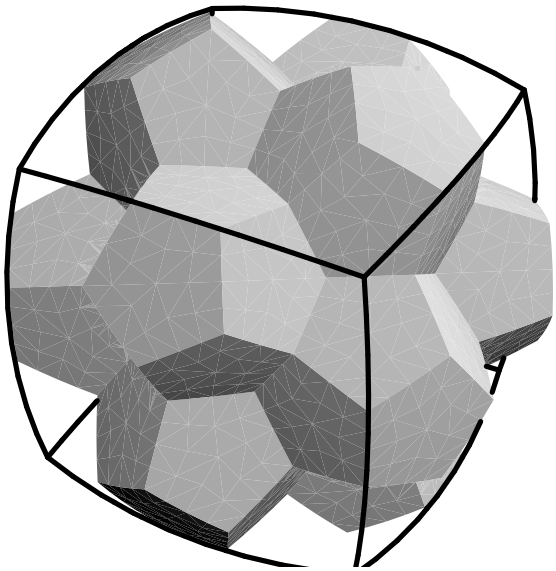


Figure 5: A cluster of thirteen bubbles from the 120-cell nearly fills out one cube of the hypercube bubble.

under our symmetry subgroup $A_4 \ltimes Z_2^4$, one in the center of each cube; these form one of the orbits of cells under the subgroup. The $96 = 8 \times 12$ cells obtained from these eight by reflection in their faces form a second orbit, because we know that all faces of the original dodecahedron are equivalent under under the subgroup, and similarly all eight cells of the hypercube are equivalent. (Notice that we have already found a representative of this second orbit, when we computed $a = 1 + 2\tau$.)

So far we have eight clusters of thirteen cells, one in each cube. Each cluster consists of the central dodecahedron and its twelve neighbors. Together, they nearly fill out their cube, as shown in Figure 5; in fact, some of the outer edges follow the cube edges. We talked before about such a cluster having dimples over the vertices of the central dodecahedron. Eight of these are at the corners of the cube, leaving holes there. And in fact, the third orbit of cells of the 120-cell consists of sixteen dodecahedra at the corners of the hypercube, filling these holes. (The other dimples get filled by dodecahedra from adjacent clusters; because of a 90° twist, they don't meet other dimples.)

Thinking about this picture, we can find the seven orbits of vertices. The first two are the vertices of the first eight dodecahedra; they have largest coordinate $1 + 2\tau$. Directly out from each vertex of the central dodecahedron are the vertices in the dimples. These form the next two orbits, those vertices with largest coordinate $\tau + 2$; those near hypercube corners are one orbit and those near hypercube faces are the other. The vertices along the hypercube's edges form the fifth orbit. The last two orbits are the vertices in the saddle-shaped "passes" between two dimples; again some of these are nearer hypercube vertices than others. Note that every vertex is a vertex of one of the middle 96 dodecahedra. Since these are all equivalent under our subgroup, among the vertices `rverts` of one such dodecahedron, we can find a representative of every orbit; in the order mentioned above these are

```
d120v0 = rverts[[{12, 6, 16, 4, 18, 5, 3}]];
```

Examing the result of this command, we see that the vertices have all been expressed easily in terms

of τ; they are $(1,1,1,1+2\tau)$, $(0,\tau-1,\tau,1+2\tau)$, $(\tau,\tau,\tau,\tau+2)$, $(0,1,\tau+1,\tau+2)$, $(0,0,2\tau,2\tau)$, $(\tau-1,\tau+1,\tau+1,\tau+1)$, and $(1,\tau,\tau+1,2\tau)$.

We want to generate a picture of the two-dimensional faces of the 120-cell, rather than its edges, so we won't bother to find the complete list of edges. But we note that they occur in sixty sets of twenty parallel edges each. The sixty direction vectors and their negatives are the images under `half4cube` of the three vectors

```
d120edgevec0 = {{0,0,0,2tau-2},
    {0,1,tau-1,2-tau}, {tau-1,tau-1,tau-1,tau-1}};
```

Note that each set of parallel edges contains edges from several different orbits under our symmetry group; also an orbit under the group will contain edges from different sets, since the group can rotate an edge into one that is not parallel.

The faces are in six orbits under our subgroup: the faces of the first eight dodecahedra, the faces of the last sixteen, two kinds of faces between neighbors in a cluster, and two kinds between adjoining clusters. Again, each type is a face of one of the middle 96 dodecahedra, so we can again find representatives of all of them within the first reflected dodecahedron. We use the faces `dodecF` of the original dodecahedron, and the reflection `refl1`.

```
dod0 = Map[Prepend[#,a]&, dodecF, {2}];
refldod = (tauexp /@ refl1 /@ #)& /@ dod0;
d120f0 = Append[Take[refldod,5], refldod[[-3]]];
```

We now apply the group to our orbit representatives, obtaining a list of the 600 vertices and one of the 720 cells:

```
d120v = images[Z24] @ images[A4] @ d120v0;
d120f = Join @@ half4cube /@ d120f0;
```

We are finally ready to create some graphics. We use the same technique of division, pushing to the sphere, and stereographic projection that we used for Figure 2(c). We must subdivide each face in order to render it (approximately) as a piece of a sphere; this is done by repeated application of `subdivide`, which divides a polygon into quadrilaterals by adding the midpoints of the edges and a point inside the polygon. The depth of subdivision should depend on how large and curved the face will appear; this, in turn, depends on how close the face is to the pole. Therefore the `output` function depends on two parameters: `subd`, giving thresholds for additional subdivisions, and `func`, a function used to render each quadrilateral. The arguments that will be supplied to `func` are the quadrilateral's vertices and the normal vectors there, already projected into three-space.

```
subdivide[x_] := Block[{y=mean[x]},
  {y,mean[Take[#,2]],#[[2]],mean[Take[Drop[#,1],2]]}& /@
    cyclic[x]];

output[func_,subd_List][f_List] := Block[
  {n=normal @@ Take[N@f,3], qs=subdivide[N@f],
     m=First@mean[N@f]},
  Scan[ If[m<#, qs = Join @@ subdivide /@ qs]&, subd];
```

```
  call[func, n] /@ Map[tosphere,qs,{2}]];

call[func_,nrml_List][ptch_List] :=
  func[stereo[S] /@ ptch, Dstereo[nrml] /@ ptch];
Dstereo[v_][x_] := stereo[S][
  Prepend[Rest[v]-Rest[x]First[v]/(1+First[x]), First[x]]
];
```

The function `Dstereo` is the derivative of the stereographic projection `stereo[S]`, which we use to move tangent vectors into three-space.

To generate *Mathematica* graphics, we can set `func` to `Polygon[#1]&` (which ignores its second argument since the *Mathematica* renderer cannot handle normal vectors). We can test this out first on the hypercube, which is a much smaller data set, and then try the 120-cell:

```
hypercube = Graphics3D[
  output[Polygon[#1]&,{1,0}] /@ hypercubeF];
d120 = Graphics3D[
  output[Polygon[#1]&,{}] /@ d120f];
```

The first picture works nicely with `Show`, and on a fast workstation we can even do more subdivisions than suggested here, though without transparency, we cannot see much of the result. On the other hand, *Mathematica* alone can hardly do justice to the second picture: if we subdivide only once, by using `subd={}` as above, the display process is entertaining but the end result is impossible to interpret, while a more ambitious subdivision scheme tends to cause the author's workstation to run out of memory. We will describe in more detail below how the cover illustration and Figure 4(b) were rendered.

The third polytope whose stereographic projection has soap bubble geometry is the simplex. It is easy to model, as it has only five vertices, each three of which form a face.

```
simplexV = Append[cyclic[{3tau-1,1-tau,1-tau,1-tau}],
  {-2,-2,-2,-2}];
alltuples[l_List,1] := List/@l;
alltuples[l_List,n_Integer] := {l} /; Length[l]<=n;
alltuples[l_List,n_Integer] :=
    Join[Prepend[#,l//First]& /@ alltuples[Rest[l],n-1],
         alltuples[Rest[l],n]];
simplexF = alltuples[simplexV,3];
simplex = Graphics3D[
  output[Polygon[#]&,{0,0}] /@ simplexF];
```

This picture is not quite symmetric; the north pole is not at the center of a cell of our simplex, so in the projection, one of the four finite cells is smaller than the other three.

Of the remaining four-dimensional polytopes, the 600-cell and the orthoplex are made of tetrahedra, and are dual to the 120-cell and the hypercube, respectively. Duality means that, for instance, each vertex of the orthoplex coincides with the center of mass of a face of of the hypercube. Thus we could compute these two polytopes by adapting the code above. But neither they nor the final polytope, the self-dual 24-cell, project stereographically to clusters of soap bubbles.

Other Properties of the 120-cell

We have already described the 120-cell in terms of layers around one dodecahedron at the north pole. We get three layers surrounding it, then thirty cells in the equatorial two-sphere, and finally three more layers converging to the antipodal dodecahedron. This description uncovers some of the symmetry of the 120-cell that is preserved in our three-dimensional picture.

But in the three-sphere, the 120-cell has much more symmetry than we can see in stereographic projection. There are sixty different layerings like the one just described, since we could start from any bubble. Only one looks symmetric in our projected picture, but all are equivalent.

For any two bubbles, some rotation of the three-sphere carries the first bubble to the second. In fact, there is a nice motion (along the *Hopf fibration* of the sphere) that takes a dodecahedron across one of its faces to the adjacent dodecahedron, with a twist of one tenth of a turn. Repeating this ten times, we are back to the original dodecahedron: the dodecahedra occur in straight chains of ten. Because when we return to the original cell we have the original orientation, these dodecahedra are the fundamental regions for a covering group. A single dodecahedron with opposite faces identified by this motion gives an interesting mathematical space [KS], a *homology sphere* discovered by Poincaré. The 120-cell can be interpreted as the universal cover of this space.

Ten dodecahedra in a chain as described above form a tube along an equator of the three-sphere. These ten are surrounded by a layer of 50, filling out a thicker toroidal tube; the remaining 60 cells are again a chain of ten surrounded by 50 neighbors. This shows how the three-sphere is the union of two solid tori, whose common boundary is a torus, flat in four-space except for the small-scale dimples. The layers of fifty can themselves be divided into chains of ten, making twelve chains in all. All are equivalent under symmetries the 120-cell, and they are arranged so that each pair links once. A motion along the Hopf fibration takes each dodecahedron to the next along its chain of ten. Although each equator is taken to a geometric circle under stereographic projection, these circles and the chains along them are not easy to find in the cover illustration. Since each dodecahedron has six pairs of opposite faces, it is in six different chains of ten, and there are six different ways to express the 120-cell as the union of twelve such chains.

In the first kind of layering we mentioned, there was a middle layer of thirty cells on an equatorial two-sphere. These dodecahedra are positioned on the sphere like the edges of a usual dodecahedron. They lie in six chains of ten, along different equators of the sphere; these follow the zig-zag equators of ten edges in a dodecahedron, visible in Figure 3(a). We mentioned earlier that the edges of the 120-cell occur in sets of twenty parallel edges. Each such set of twenty occurs in one of these equatorial two-spheres; at each of the twenty vertices of our usual dodecahedron, three of the thirty dodecahedra in the layer meet, and the edge between them is in this family.

We made use of one other way of considering the 120 cells, starting with eight at the vertices of an orthoplex, that is, in the cells of a hypercube. If we put a shell of twelve around each of these, only sixteen cells are still unaccounted for. These sixteen are at the vertices of the hypercube, the points furthest from the original eight. We might hope to see understand this description by comparing the cover illustration with Figure 4(b). Each of the cells of the hypercube bubble is filled with a cluster of thirteen dodecahedra as in Figure 5. The infinite region and twelve largest bubbles of the 120-cell lie in the infinite region of the hypercube bubble. The last sixteen dodecahedra surround the junction points of the hypercube cluster; they include eight of the twenty relatively large pointed bubbles on the cover.

A Word About the Rendering

Our brains are very good at interpreting the three-dimensional structure of what we see. Motion and stereoscopic effects help in this process, but can be hard to recreate with computer graphics. In rendering a still image, we should try to include as many clues as possible, by making the image realistic.

The rendering of the 120-cell in *Mathematica* is unsatisfactory on two counts: one cannot subdivide the curved faces finely enough to give the impression of a smooth surface; and most of the detail is occluded by the large outer bubbles. To view this structure properly, we must make the walls transparent. Even simple "flat" transparency doesn't make a good picture, because although the internal structure is visible, we cannot interpret it well—perhaps because we have no experience seeing objects with such optical properties.

Soap films show mirror-like reflections of objects around them, without responding at all to the diffuse light used in most computer renderings. The curved highlights from reflected windows show us the shape of the surface. Also, soap film, like glass, is more transparent when viewed straight on. This *Fresnel effect* is essential in suggesting the roundness of the bubbles. One final visual feature of soap film, discussed extensively in [Boy], is the streaks of brilliant colors. These arise from optical interference between light rays reflected off the inner and outer surfaces of the film.

The RenderMan system, a high-quality renderer from Pixar, is convenient for modeling these features. It includes a general programming language for writing a *shader*, that is, a function that, given the orientation of the surface and the eye position, returns the fraction of light of different colors transmitted and reflected. The renderer combines this information in the proper order to simulate the many layers of transparency.

In our case the shader simply implements the basic laws of electromagnetics. Fresnel's equations give the fraction of light transmitted across a single interface between materials of different optical density. A soap film has two such interfaces separated by a distance roughly equal to the wavelength of light, and light may bounce internally off these any number of times before emerging. Thus we sum a geometric series to get the final reflection coefficients, using complex numbers to track phase shifts. The angle of view is mainly what determines the overall transparency (by the Fresnel effect) and the thickness of the soap film is largely responsible for the rainbow colors. However, these effects do not have to be programmed separately, as they are both determined by the same equations. This rendering is described in more detail in [AS].

It would be complicated to model the fluid swirls in real soap film and the draining under gravity which, together, account for most of the variations in thickness. Thus in our pictures, the thickness of the soap film is chosen randomly with bandwidth-limited noise. We created an artificial environment, with several windows and ceiling lights, to provide highlights in the bubbles; the end result is quite striking.

The interface between *Mathematica* and RenderMan is simple: we merely call `output` with `func` set to a function that prints the vertices and normals into a file, in the appropriate format.

Even better results can be obtained by giving RenderMan large bicubic patches, instead of quadrilaterals; less subdivision is needed for the same degree of realism, and the renderer can compute the normals. We can compute a mesh of any fineness from a quadrilateral with our interpolation routine:

```
interp2[{a_,b_,c_,d_},s_,t_] :=
```

```
  interp1[interp1[a,b,s], interp1[d,c,s], t];
mesh[quad_,n_] := Table[Table[interp2[quad,s,t],
  {s,-1/n,(n+1)/n,1/n}], {t,-1/n,(n+1)/n,1/n}];
```

Because we apply these functions in four-space, before projecting to the sphere or down to three-space, each face is flat, and this linear interpolation (and even slight extrapolation around the edges) gives exactly the right answers. This is the technique used to generate the cover illustration.

Acknowledgements

I would like to thank Pat Hanrahan for first suggesting to me the use of the Fresnel effect to render soap bubbles, Silvio Levy for helpful suggestions on the code and text for this article, and Toby Orloff for his help in designing the room around the soap bubbles.

References

[AS] Frederick J. Almgren, Jr. and John M. Sullivan. Visualization of soap bubble geometries. *Leonardo* (1992). To appear.

[Boy] C. V. Boys. *Soap Bubbles: Their Colors and the Forces Which Mold Them*. Dover, 1959.

[Cox] H. S. M. Coxeter. *Regular Polytopes*. Dover, 1973.

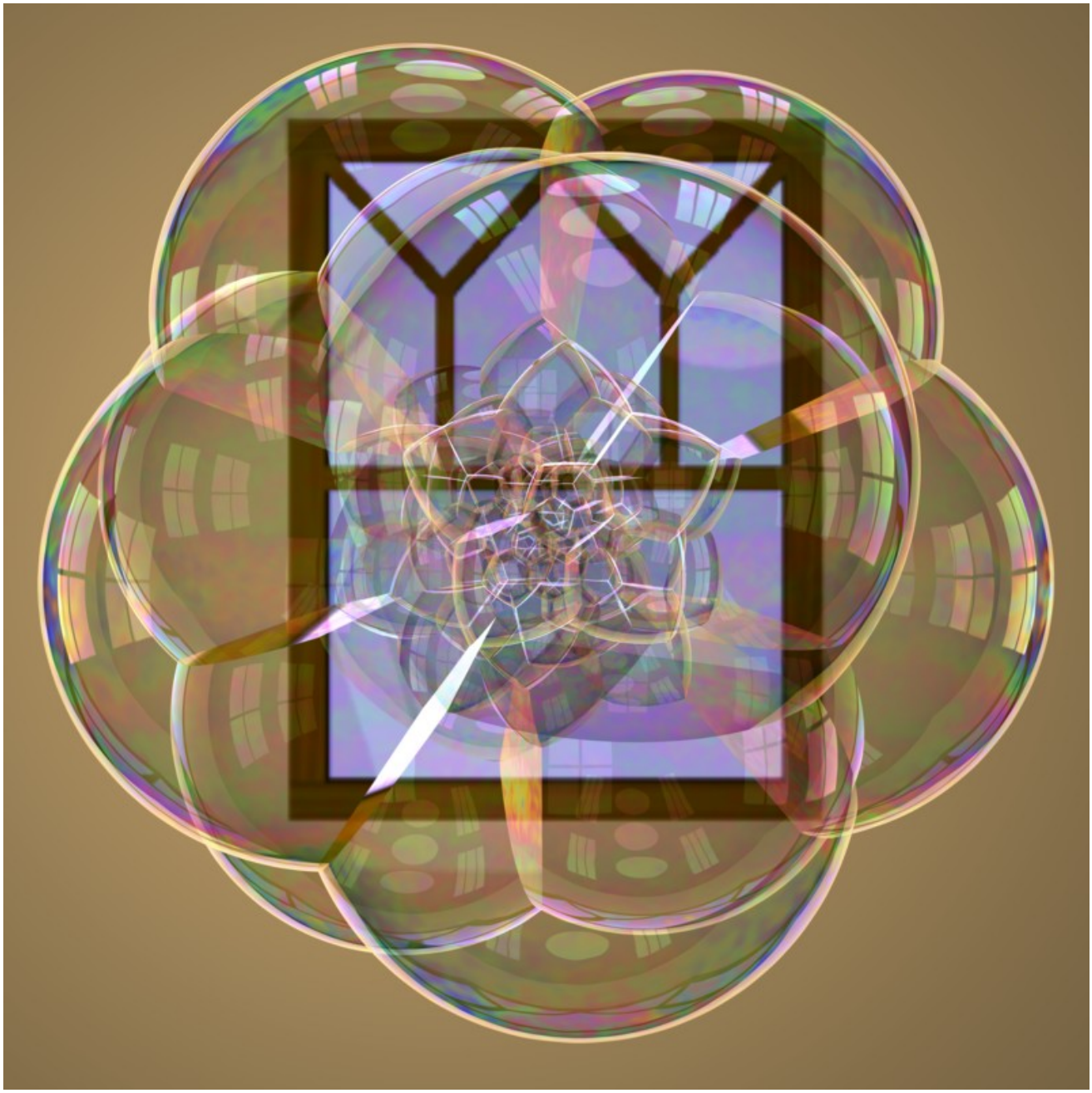
[KS] R. C. Kirby and M. G. Scharlemann. Eight faces of the Poincaré Homology 3-Sphere. In *Geometric Topology*, pages 113–146. Academic Press, 1979.

[LGM] Arthur L. Loeb, Jack C. Gray, and Philip R. Maillinson. On the Icosahedron, the Pentagonal Dodecahedron, and the Rhombic Triancontahedron. *Symmetry* **1**(1990), 29–36.

[Mae] Roman Maeder. *Programming in Mathematica*. Addison Wesley, 1990.

[Tay] J. E. Taylor. The structure of singularities in soap-bubble-like and soap-film-like minimal surfaces. *Ann. of Math.* **103**(1976), 489–539.

[Tho] D. W. Thompson. *On Growth and Form*. Cambridge Univ. Press, 1961. Abridged edition edited by J. T. Bonner.



Color Plate: The stereographic projection of the dodecaplex (or 120-cell) has the geometry of a bubble cluster. Here it is rendered as if it were made of soap films.