Geometry and Symmetry

Chapter 3

Barry H Dayton

https : // barryhdayton.space

This is a continuation of my Geometry and Symmetry book. This is not self contained but assumes material from Chapter 2 of my Geometry and Symmetry book, Chapters1-2 of https://barryhdayton.space/GSChapters1-2.pdf.

Transformation Function Groups on the Sphere

We will start with the sphere in R^3 as there are interesting finite groups. These groups are associated with the 5 platonic solids so we will discuss these. We will concentrate on rigid motions, these are sometimes synonymous with isometries, here we use the term do describe direct isometries, that is ones where the orthogonal part has determinant 1. Since we are limiting to finite groups this means that we are only considering rotations. While mathematically your left foot is a reflection of your right foot there is no way physically to make it into a left foot. So we are only considering those transforms which can be physically made on a ball.

We will use the following code for a set of points often so we give it a name, the spherical centroid.

```
In[ ]:= spcentroid[P_]:=If[Norm[Total[P]]>.0001,Normalize[Total[P]],{0,0,0}]
```

3.1 Rotation Groups of the Platonic Solids

We will consider our polygons to be spherical, that is all vertices are on the unit sphere and the centroid is the origin. Rotations will have axes through the center and will intersect the unit at 2 points, however it is sufficient to pick one of these points and use notation

```
In[ ]:= RotationTransform[θ, p]
```

where θ is an angle in radians and p is a point on the unit sphere.

3.1.1 The Tetrahedron

Mathematica has the data available to find the coordinates of a regular tetrahedron inscribed in the unit sphere

```
In[ ]:= Tet1 = Normalize[#] & /@ N[CanonicalizePolyhedron[Tetrahedron[1]]][[1]]
Out[ ]= {{0., 0., 1.}, {-0.471405, -0.816497, -0.333333},
  {-0.471405, 0.816497, -0.333333}, {0.942809, 0., -0.333333}}
```

Notice the centroid is the origin.

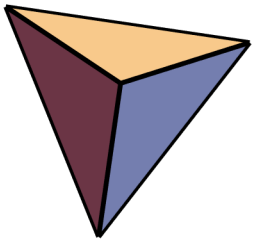
```
In[ ]:= spcentroid[Tet1]
Out[ ]= {0, 0, 0}
```

```
In[ ]:= Table[Norm[Tet1[[i]] - Tet1[[j]]], {i, 3}, {j, i + 1, 4}]
Out[ ]= {{1.63299, 1.63299, 1.63299}, {1.63299, 1.63299}, {1.63299}}
```

all the sides are of equal length so this is a regular tetrahedron inscribed in the unit sphere.

```
In[ ]:= Graphics3D[{EdgeForm[{Black, Thickness[.01]}], Tetrahedron[Tet1]},
  Boxed → False, ImageSize → Small]
```

Out[ ]=



We see there is a 3 - fold rotation about the axis from the vertex through the origin. For example, note that from time to time we will redefine various Greek letters.

```
In[ ]:= κ = N[RotationTransform [2 Pi / 3, {0, 0, 1}]]
        λ = RotationTransform [2 Pi / 3, Tet1[[2]]]
```

Out[ ]= TransformationFunction $\left[\left(\begin{array}{ccc|c} -0.5 & -0.866025 & 0. & 0. \\ 0.866025 & -0.5 & 0. & 0. \\ 0. & 0. & 1. & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$

Out[ ]= TransformationFunction $\left[\left(\begin{array}{ccc|c} -0.166667 & 0.866025 & -0.471405 & 0. \\ 0.288675 & 0.5 & 0.816497 & 0. \\ 0.942809 & 1.11022 \times 10^{-16} & -0.333333 & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$

```
In[ ]:= κ @ Tet1
```

```
Out[ ]= {{0., 0., 1.}, {0.942809, 0., -0.333333},
          {-0.471405, -0.816497, -0.333333}, {-0.471405, 0.816497, -0.333333}}
```

This just permutes the vertices . λ will do the same. So these are symmetries of the tetrahedron.

We check that they generate a finite group

```
In[ ]:= Gtet = finiteTransGroup [⟨| 1 → κ, 2 → λ |⟩, {2.316, -1.347, .3712}, 4]
```

» number of group elements calculated 12

```
Out[ ]= {{1}, {2}, {1, 1}, {1, 2}, {2, 1}, {2, 2}, {1, 1, 1},
          {1, 1, 2}, {1, 2, 1}, {1, 2, 2}, {1, 1, 2, 1}, {1, 1, 2, 2}}
```

The orders of the elements can be calculated by the following function

```
In[ ]:= orderAssoc [G_, tas_, tp_, n_] := ⟨| Table[k → orderTF[TasTF[k, tas], tp, n], {k, G}] |⟩
```

```
In[ ]:= ordAssTet = orderAssoc [Gtet, ⟨| 1 → κ, 2 → λ |⟩, {2.316, -1.347, .3712}, 4]
```

```
Out[ ]= ⟨| {1} → 3, {2} → 3, {1, 1} → 3, {1, 2} → 3, {2, 1} → 3, {2, 2} → 3, {1, 1, 1} → 1,
          {1, 1, 2} → 2, {1, 2, 1} → 2, {1, 2, 2} → 2, {1, 1, 2, 1} → 3, {1, 1, 2, 2} → 3 |⟩
```

So we have the identity and 8 3-fold rotations, two, a 1/3 turn and a 2/3 turn, about each vertex and 3 half turns. One of the half turns is

```
In[ ]:= ϕ121 = TasTF[{1, 2, 1}, ⟨| 1 → κ, 2 → λ |⟩]
```

Out[ ]= TransformationFunction $\left[\left(\begin{array}{ccc|c} -0.666667 & 0.57735 & -0.471405 & 0. \\ 0.57735 & -1.11022 \times 10^{-16} & -0.816497 & 0. \\ -0.471405 & -0.816497 & -0.333333 & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$

To find a non - zero point on the axis do

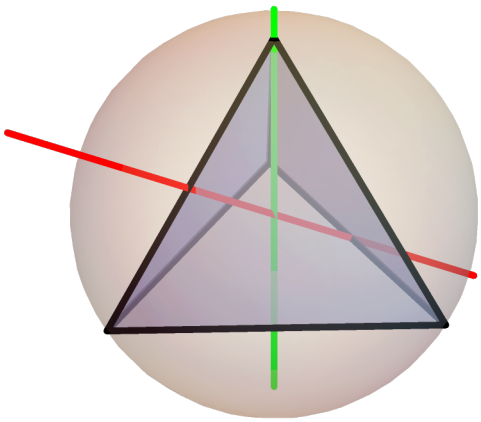
```
In[ ]:= v121 = NSolveValues [ϕ121[{x, y, z}] == {x, y, 1}, {x, y, z}][[1]]
```

```
Out[ ]= {-0.707107, -1.22474, 1.}
```

To check that this actually gives a fixed point

```
In[ ]:= ϕ121 @ v121
Out[ ]= {-0.707107 , -1.22474 , 1.}

In[ ]:= Show[ContourPlot3D [x ^ 2 + y ^ 2 + z ^ 2 == 1, {x, -1.1, 1.1}, {y, -1.1, 1.1},
          {z, -1.1, 1.1}, Mesh → None , ContourStyle → Directive[Opacity[.3], LightYellow ]],
        Graphics3D [{{Red , Thickness [.01], InfiniteLine [{{0, 0, 0}, v121}]},
          {Green , Thickness [.01], InfiniteLine [{{0, 0, 0}, {0, 0, 1}}]}, {LightBlue , Opacity[.6],
           EdgeForm [{Thickness [.01], Black}], Tetrahedron [Tet1]}}], Boxed → False , Axes → False]
```

Out[ ]=



The green line passes through the vertex {1,0,0} and the centroid of the opposite, bottom, side. The red line passes through the midpoint of one of the vertical sides and the midpoint of the opposite horizon - tal side. So we conclude, you might want to check, that the other 3-fold turns go through a vertex centroid of a side whereas the other two half turns also go through the midpoint of a vertical side and midpoint of the opposite sides.

Looking at the tetrahedron you might have noticed the half turn symmetries but they are compositions of the obvious symmetries. This is one reason why finding the full group is useful.

Above we used the **Mathematica** primitive to plot the tetrahedron . Later we want to be able to draw more complicated polyhedra or spherical paneling. We can use our routine **groupAssoc** to do this. The trick here, where all sides, panels, are congruent by a group transformation, is to find the vertices and centroid of one panel. By using the centroid as our test point **groupAssoc** will pick out one transformation to take that centroid to the centroid of each panel. Then we can write a routine to draw the polyhedron. In the case of the tetrahedron we start with the lower panel

```
In[ ]:= tPanel1 = Take[Tet1, -3]
Out[ ]= {{-0.471405, -0.816497, -0.333333},
  {-0.471405, 0.816497, -0.333333}, {0.942809, 0., -0.333333}}

In[ ]:= centtPanel1 = spcentroid[tPanel1]
Out[ ]= {0., 0., -1.}

In[ ]:= spcentroid[tPanel1]
Out[ ]= {0., 0., -1.}

In[ ]:= tCentA = groupAssoc[Gtet, <|1 → κ, 2 → λ|>, centtPanel1]
Out[ ]= <|{1} → {0., 0., -1.}, {2} → {0.471405, -0.816497, 0.333333},
  {1, 2} → {0.471405, 0.816497, 0.333333}, {2, 2} → {-0.942809, 0., 0.333333}|>
```

Our panels are then

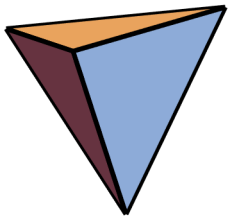
```
In[ ]:= tPanels = Table[TasTF[key, <|1 → κ, 2 → λ|>]@tPanel1, {key, Keys[tCentA]}]
Out[ ]= {{{0.942809, 0., -0.333333}, {-0.471405, -0.816497, -0.333333},
    {-0.471405, 0.816497, -0.333333}}, {{-0.471405, -0.816497, -0.333333},
    {0.942809, -1.11022 × 10^-16, -0.333333}, {2.77556 × 10^-17, 5.55112 × 10^-17, 1.}},
   {{0.942809, -5.55112 × 10^-17, -0.333333}, {-0.471405, 0.816497, -0.333333},
    {-5.55112 × 10^-17, 0., 1.}}, {{-0.471405, -0.816497, -0.333333},
    {-1.11022 × 10^-16, 5.55112 × 10^-17, 1.}, {-0.471405, 0.816497, -0.333333}}}
```

The values are the vertices of the 4 panels . We might decide to color the panels with the following fixed colors

The following simple Graphics3D directives will give our tetrahedron .

```
In[ ]:= Graphics3D[Table[{EdgeForm[{Black, Thickness[.01]}], Polygon[tPanels⟦i⟧]}, {i, 4}],
  ImageSize → Small, Boxed → False]
```

Out[ ]=

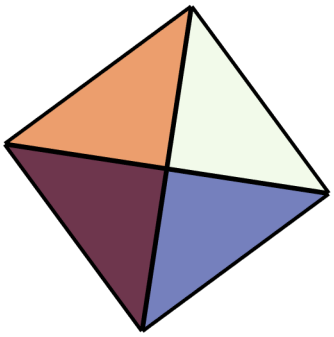


3.1.2 The Octahedron

The spherical octahedron is quite simple. From **Mathematica**

```
In[ ]:= Oct = CanonicalizePolyhedron [Octahedron[Sqrt[2]]][[1]]
Out[ ]= {{0, 1, 0}, {1, 0, 0}, {0, -1, 0}, {-1, 0, 0}, {0, 0, 1}, {0, 0, -1}}

In[ ]:= Graphics3D [{EdgeForm[{Black, Thickness[.01]}], Octahedron[Sqrt[2]]},
          Boxed → False, ImageSize → Small, ViewPoint → Above]
```

Out[ ]=



We see a 4-fold rotation at each vertex. Taking two of these we get rotations (re-initialize Rotation Transforms in each section as I may reuse names.)

```
In[ ]:= μ = RotationTransform [Pi / 2, {0, 0, 1}]
Out[ ]= TransformationFunction [( 0  -1  0 | 0 )
                                ( 1   0  0 | 0 )
                                ( 0   0  1 | 0 )
                                ( 0   0  0 | 1 )]

In[ ]:= ν = RotationTransform [Pi / 2, {1, 0, 0}]
Out[ ]= TransformationFunction [( 1  0   0 | 0 )
                                ( 0  0  -1 | 0 )
                                ( 0  1   0 | 0 )
                                ( 0  0   0 | 1 )]
```

Our group generated by these transformations is, suppressing the result for now

```
In[ ]:= GOct = finiteTransGroup [⟨| 1 → μ, 2 → ν |⟩, {2.316, -1.347, .3712}, 5];
» number of group elements calculated  24
```

The order of GOct, 24, is actually a well known number. (See [Yale])

We would like to find the orders of the rotations so we find the order association

```
In[ ]:= GAOord = orderAssoc [GOct, ⟨| 1 → μ, 2 → ν |⟩, {2.316, -1.347, .3712}, 6]
Out[ ]= ⟨| {1} → 4, {2} → 4, {1, 1} → 2, {1, 2} → 3, {2, 1} → 3, {2, 2} → 2, {1, 1, 1} → 4, {1, 1, 2} → 2,
  {1, 2, 1} → 2, {1, 2, 2} → 2, {2, 1, 1} → 2, {2, 2, 1} → 2, {2, 2, 2} → 4, {1, 1, 1, 1} → 1,
  {1, 1, 1, 2} → 3, {1, 1, 2, 1} → 3, {1, 1, 2, 2} → 2, {1, 2, 1, 1} → 3, {1, 2, 2, 2} → 3,
  {2, 1, 1, 1} → 3, {2, 2, 2, 1} → 3, {1, 1, 1, 2, 1} → 4, {1, 2, 1, 1, 1} → 4, {1, 2, 2, 2, 1} → 2 |⟩
```

Counting by hand or using Select we find that there are six rotations of order 4, eight of order 3, nine of order 2 and one, the identity, of order 1.

The order 4 rotations come from rotation at the vertices, the vertices, by inspection form 3 antipodal pairs so the rotations of each pair have the same axis, so each axis will support a group of 4 rotations,

two mutually inverse rotations of order 4, the common square which has order 2 and the identity. So these explain the six order 4 rotations and 3 of the order 2 rotations. The rotations of order 3 have axes going through the centroids of the sides. Again the 8 sides make up 4 antipodal pairs so their will be a 3 axes each with a subgroup of 3 elements one being the identity and the other 2 of order 3. So this explains the order 3 rotations.

The order 2 rotations are of two types, the 3 squares of order 4 rotations and 6 others. One of these is given by the list {1,1,2}. We can find the axis by

In[ ]:=
```
ξ = TasTF[{1, 1, 2}, ⟨| 1 → μ, 2 → ν |⟩]
```

Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

We can find a non-zero point on the axis by

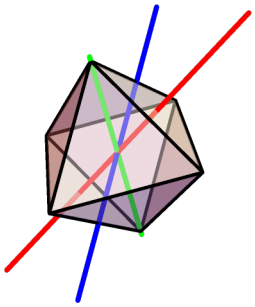
In[ ]:=
```
NSolveValues[ξ@{x, y, z} == {x, y, 1}, {x, y, z}]
```

Out[ ]=
```
{{0., 1., 1.}}
```

The line through {0, 1, 1} and {0, 0, 0} intersects two antipodal edges of the octahedron . Since there are 12 edges in the octahedron they pair up in 6 antipodal pairs. This exp1ains the 6 remaining order 2 rotations and the third kind of rotation. Here is a picture of the three different rotations and their axes.

In[ ]:=
```
Graphics3D[{{Opacity[.5], EdgeForm[{GrayLevel[0], Thickness[0.01`]}], Octahedron[√2]},
  {Green, Thickness[.015], InfiniteLine[{{0, 0, 0}, {0, 0, 1}}]},
  {Red, Thickness[.015], InfiniteLine[{{0, 0, 0}, {1/3, 1/3, 1/3}}]},
  {Blue, Thickness[.015], InfiniteLine[{{0, 0, 0}, {0, 1/3, 1/3}}]}},
 Boxed → False, ImageSize → Small]
```

Out[ ]=



The green axis is an example of an axis of an order 4 rotation, it intersects two antipodal vertices. The red line passes through the centroids of two sides giving an order 3 rotation. And the blue line passes through the midpoints of 2 edges giving a half turn.

As with the tetrahedron we can use our group to draw the octahedron without resorting to Mathemati - ca's primitive. One panel has centroid.

```
In[ ]:= oPanel1Cent = {1. / 3, 1. / 3, 1. / 3}
Out[ ]= {0.333333, 0.333333, 0.333333}
```

We find unique rotations sending this centroid to the other side centroids.

```
In[ ]:= OctCentA = groupAssoc[GOct, ⟨|1 → μ, 2 → ν|⟩, oPanel1Cent]
Out[ ]= ⟨|{1} → {-0.333333, 0.333333, 0.333333}, {2} → {0.333333, -0.333333, 0.333333},
  {1, 1} → {-0.333333, -0.333333, 0.333333}, {1, 2} → {0.333333, 0.333333, 0.333333},
  {2, 2} → {0.333333, -0.333333, -0.333333}, {1, 2, 2} → {0.333333, 0.333333, -0.333333},
  {2, 1, 1} → {-0.333333, -0.333333, -0.333333},
  {1, 1, 2, 2} → {-0.333333, 0.333333, -0.333333}|⟩
```

The panel we started with has vertices

```
In[ ]:= oPanel1 = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
```

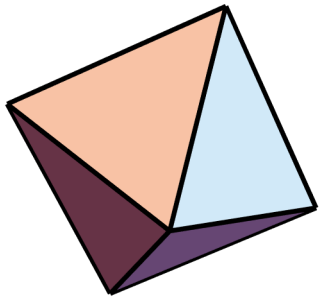
We could have done this by hand but using the above rotation our panels are

```
In[ ]:= oPanels = Table[TasTF[key, ⟨|1 → μ, 2 → ν|⟩]@oPanel1, {key, Keys[OctCentA]}]
Out[ ]= {{{0, 1, 0}, {-1, 0, 0}, {0, 0, 1}}, {{1, 0, 0}, {0, 0, 1}, {0, -1, 0}},
  {{-1, 0, 0}, {0, -1, 0}, {0, 0, 1}}, {{0, 1, 0}, {0, 0, 1}, {1, 0, 0}},
  {{1, 0, 0}, {0, -1, 0}, {0, 0, -1}}, {{0, 1, 0}, {1, 0, 0}, {0, 0, -1}},
  {{-1, 0, 0}, {0, 0, -1}, {0, -1, 0}}, {{-1, 0, 0}, {0, 1, 0}, {0, 0, -1}}}
```

To draw the octahedron

```
In[ ]:= Graphics3D[Table[{EdgeForm[{Black, Thickness[.01]}], Polygon[oPanels⟦i⟧]}, {i, 8}],
  Boxed → False, ImageSize → Small]
```

Out[ ]=



3.1.3 The Hexahedron AKA the Cube.

The spherical cube is given by

```
In[ ]:= hexs = {{-1, 1, 1}, {1, 1, 1}, {1, -1, 1}, {-1, -1, 1},
          {1, -1, -1}, {1, 1, -1}, {-1, 1, -1}, {-1, -1, -1}} / Sqrt[3.]
Out[ ]= {{-0.57735 , 0.57735 , 0.57735}, {0.57735 , 0.57735 , 0.57735},
         {0.57735 , -0.57735 , 0.57735}, {-0.57735 , -0.57735 , 0.57735},
         {0.57735 , -0.57735 , -0.57735}, {0.57735 , 0.57735 , -0.57735},
         {-0.57735 , 0.57735 , -0.57735}, {-0.57735 , -0.57735 , -0.57735}}
```

It is easily seen that the rotations μ, ν of the octagon are also symmetries of the hexahedron.

```
In[ ]:= μ @ hexs
Out[ ]= {{-0.57735 , -0.57735 , 0.57735}, {-0.57735 , 0.57735 , 0.57735},
         {0.57735 , 0.57735 , 0.57735}, {0.57735 , -0.57735 , 0.57735},
         {0.57735 , 0.57735 , -0.57735}, {-0.57735 , 0.57735 , -0.57735},
         {-0.57735 , -0.57735 , -0.57735}, {0.57735 , -0.57735 , -0.57735}}
```

and the same for ν. The group GOct is actually the full group of symmetries of the hexahedron. In this case we are not just talking isomorphism but has the exact same transformation functions.

So rather than use the Mathematica primitive Hexahedron we can directly draw the cube . The top panel is

```
In[ ]:= hPanel1 = {{-1, 1, 1}, {1, 1, 1}, {1, -1, 1}, {-1, -1, 1}} / Sqrt[3.]
Out[ ]= {{-0.57735 , 0.57735 , 0.57735}, {0.57735 , 0.57735 , 0.57735},
         {0.57735 , -0.57735 , 0.57735}, {-0.57735 , -0.57735 , 0.57735}}
```

with centroid

```
In[ ]:= hPanel1Cent = spcentroid [hPanel1]
Out[ ]= {0. , 0. , 1.}
```

The group association associated with this centroid is

```
In[ ]:= HexCent = groupAssoc [GOct , <| 1 → μ , 2 → ν |> , {0, 0, Sqrt[3.] / 3}]
Out[ ]= <| {1} → {0. , 0. , 0.57735}, {2} → {0. , -0.57735 , 0.}, {1, 2} → {0.57735 , 0. , 0.},
          {2, 2} → {0. , 0. , -0.57735}, {1, 1, 2} → {0. , 0.57735 , 0.}, {1, 1, 1, 2} → {-0.57735 , 0. , 0.} |>
```

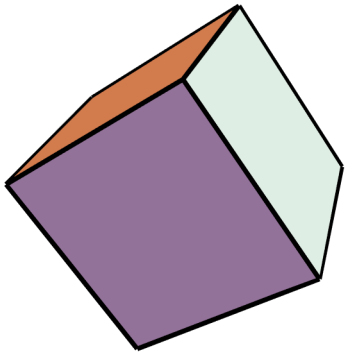
It is interesting that this routine can decide given this different test point that the cube has 6 sides, not 8. We can now calculate all panels

In[ ]:=
```
hPanels = Table[TasTF[key, ⟨| 1 → μ, 2 → ν |⟩]@hPanel1, {key, Keys[HexCent]}]
```

Out[ ]=
```
{{{-0.57735, -0.57735, 0.57735}, {-0.57735, 0.57735, 0.57735},
   {0.57735, 0.57735, 0.57735}, {0.57735, -0.57735, 0.57735}},
  {{-0.57735, -0.57735, 0.57735}, {0.57735, -0.57735, 0.57735},
   {0.57735, -0.57735, -0.57735}, {-0.57735, -0.57735, -0.57735}},
  {{0.57735, -0.57735, 0.57735}, {0.57735, 0.57735, 0.57735},
   {0.57735, 0.57735, -0.57735}, {0.57735, -0.57735, -0.57735}},
  {{-0.57735, -0.57735, -0.57735}, {0.57735, -0.57735, -0.57735},
   {0.57735, 0.57735, -0.57735}, {-0.57735, 0.57735, -0.57735}},
  {{0.57735, 0.57735, 0.57735}, {-0.57735, 0.57735, 0.57735},
   {-0.57735, 0.57735, -0.57735}, {0.57735, 0.57735, -0.57735}},
  {{-0.57735, 0.57735, 0.57735}, {-0.57735, -0.57735, 0.57735},
   {-0.57735, -0.57735, -0.57735}, {-0.57735, 0.57735, -0.57735}}}
```

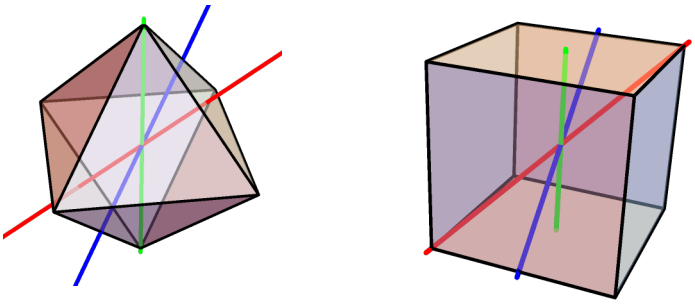
In[ ]:=
```
Graphics3D[Table[{EdgeForm[{Black, Thickness[.01]}], Polygon[hPanels⟦i⟧]}, {i, 6}],
 Boxed → False, ImageSize → Small]
```

Out[ ]=

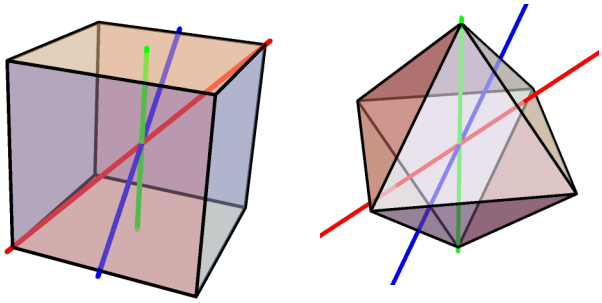


Since all the rotations are the same they have the same orders and axes here as for the octagon. In the following graphics the green, red and blue axes are the same

In[ ]:=



Out[ ]=



but now the red axes connect opposite vertices instead of opposite side centroids while the green now connect the side centroids instead of opposite vertices. The blue lines which are half turns still connect midpoints of edges.

3.1.4 The Icosahedron

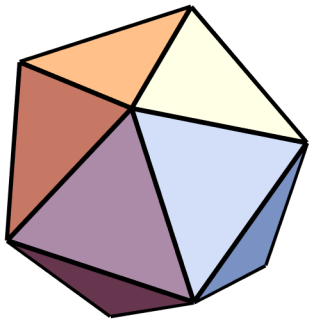
We can get the spherical icosahedron from **Mathematica** by

In[ ]:=
```
s = 1 / 0.9510565162951536` ;
iCos = CanonicalizePolyhedron [Icosahedron [s]][[1]]
```

Out[ ]=
```
{{0. , 0. , -1.}, {0. , 0. , 1.}, {-0.894427 , 0. , -0.447214}, {0.894427 , 0. , 0.447214},
 {0.723607 , -0.525731 , -0.447214}, {0.723607 , 0.525731 , -0.447214},
 {-0.723607 , -0.525731 , 0.447214}, {-0.723607 , 0.525731 , 0.447214},
 {-0.276393 , -0.850651 , -0.447214}, {-0.276393 , 0.850651 , -0.447214},
 {0.276393 , -0.850651 , 0.447214}, {0.276393 , 0.850651 , 0.447214}}
```

```
In[ ]:= Graphics3D[{EdgeForm[{Black, Thickness[.01]}], Icosahedron[s]},
         Boxed → False, ImageSize → Small]
```

Out[ ]=



There is an obvious 5-fold rotation about any vertex so we choose

```
In[ ]:= σ = N[RotationTransform[2 Pi / 5, {0, 0, 1}]]
```

Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 0.309017 & -0.951057 & 0. & 0. \\ 0.951057 & 0.309017 & 0. & 0. \\ 0. & 0. & 1. & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$$

Picking adjoining vertices about vertex {0, 0, 1} we get a face, panel, by

```
In[ ]:= iPanel1 = {iCos[[2]], iCos[[4]], iCos[[11]]}
```

```
Out[ ]= {{0., 0., 1.}, {0.894427, 0., 0.447214}, {0.276393, -0.850651, 0.447214}}
```

The centroid of this panel is

```
In[ ]:= iPanel1Cent = spcentroid[iPanel1]
```

```
Out[ ]= {0.491123, -0.356822, 0.794654}
```

All sides of an icosahedron are equilateral triangles so we have another symmetry

```
In[ ]:= τ = RotationTransform[2 Pi / 3, iPanel1Cent]
```

Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} -0.138197 & -0.951057 & 0.276393 & 0. \\ 0.425325 & -0.309017 & -0.850651 & 0. \\ 0.894427 & 0. & 0.447214 & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$$

This enables us to find the group of rotations of the icosahedron . This is well known, there are 60 rotations, but we can give it as an actual group of Transformation Functions.

```
In[ ]:= τ = RotationTransform[2 Pi / 3, iPanel1Cent]
```

Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} -0.138197 & -0.951057 & 0.276393 & 0. \\ 0.425325 & -0.309017 & -0.850651 & 0. \\ 0.894427 & 0. & 0.447214 & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$$

The transformation group of the icosahedron is then

```
In[ ]:= GIcos = finiteTransGroup [⟨| 1 → σ, 2 → τ|⟩ , {2.316 , -1.347 , .3712}, 10];
```

» number of group elements calculated 60

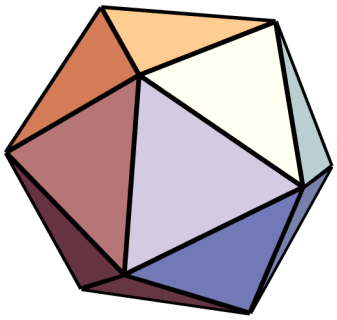
This agrees with the known order. We can now calculate the vertices of the 20 panels by

```
In[ ]:= ICosCentA = ⟨| groupAssoc [GIcos , ⟨| 1 → σ, 2 → τ|⟩ , iPanel1Cent ]|⟩ ;

In[ ]:= Length[iCosCentA ]

Out[ ]= 0

In[ ]:= iPanels = Table[TasTF[key , ⟨| 1 → σ, 2 → τ|⟩ ]@ iPanel1 , {key , Keys[ICosCentA ]}];
```

We can now draw the icosahedron

```
In[ ]:= Graphics3D [Table[{EdgeForm [{Black , Thickness [.01]}], Polygon[iPanels⟦i⟧]}, {i , 20}],
         Boxed → False , ImageSize → Small]
```

Out[ ]=



We can calculate the orders of the rotations and find that there are 24 rotations of order 5. This corresponds with the fact that there are 6 pairs of antipodal vertices each pair giving a five fold rotation. There are then 4 distinct non-identity rotation for each of the 6 axes.

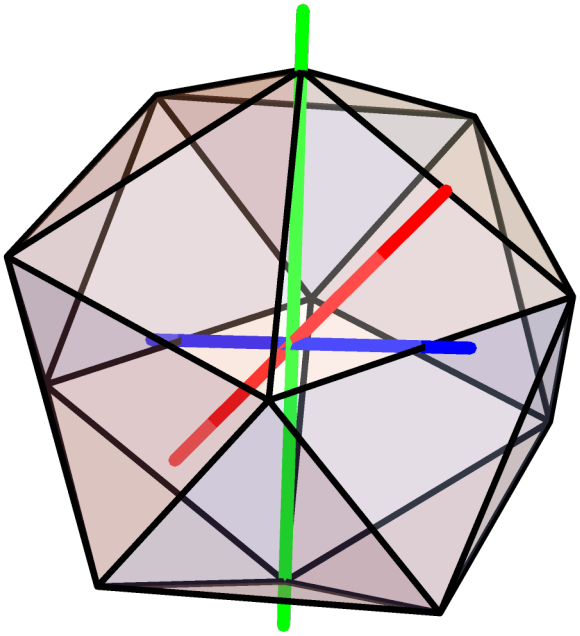
There are 20 order 3 rotations . There are 20 faces giving 10 pairs of antipodal faces. The axis through the centroids of each pair gives 10 axes. About each axis there are 2 non-identity rotations, actually inverses of each other. This explains the 10×2 = 20 order 3 rotations.

Finally there are 30 edges, again forming 15 antipodal pairs. Connecting the midpoints of members of a pair gives a 2-fold, half-turn, rotation with axis between the midpoints. There is only one per pair giving 15 axes and 15 rotations.

Adding the identity, his explains the 24 + 20 + 15 + 1 = 60 members of the rotation group.

The three types of rotation axis are shown below . The green axis is a 5-fold axis, the red is a threefold axis and the blue is a half-turn axis

Out[ ]=



We noted above that there were 6 axes for 5 fold rotations, 10 for 3 fold rotations and 15 for 2-fold rotations, altogether 31. Since our icosahedron is inscribed in the unit sphere we can think of each axis as being a line between two antipodal points on a sphere. There would then be 62 of these. In my earlier non-technical paper entitled *Soccer Balls* I illustrated these points as they were useful in designing panels for soccer balls, but I never gave coordinates. For later in this chapter these points may come in handy so I catalog them now in full precision.

In[ ]:=

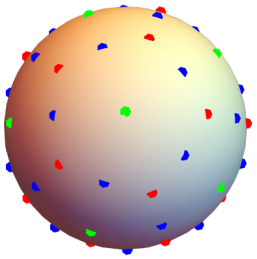
```
axisPoints5 =
{{0, 0, 1}, {-0.7236067977499789` , 0.5257311121191335` , 0.4472135954999581` }, {0, 0,
     -1}, {-0.8944271909999157` , 1.3352805415370576`*^-16 , -0.4472135954999577` },
   {0.7236067977499789` , -0.5257311121191335` , -0.4472135954999581` },
   {0.7236067977499789` , 0.5257311121191335` , -0.4472135954999581` },
   {-0.276393202250021` , -0.8506508083520395` , -0.4472135954999581` },
   {0.8944271909999157` , -1.3352805415370576`*^-16 , 0.4472135954999577` },
   {-0.7236067977499789` , -0.5257311121191335` , 0.4472135954999581` },
   {0.276393202250021` , -0.8506508083520399` , 0.4472135954999575` },
   {0.276393202250021` , 0.8506508083520395` , 0.4472135954999581` },
   {-0.276393202250021` , 0.8506508083520399` , -0.4472135954999575` }};
```

In[ ]:=
```
axisPoints3 =
{{0.18759247408507987` , 0.5773502691896258` , -0.7946544722917661` },
   {0.6070619982066862` , 0, -0.7946544722917661` },
   {0.1875924740850798` , -0.5773502691896257` , -0.7946544722917662` },
   {0.30353099910334314` , 0.9341723589627159` , -0.18759247408507984` },
   {-0.7946544722917661` , 0.5773502691896258` , -0.18759247408507979` },
   {-0.30353099910334297` , 0.9341723589627156` , 0.18759247408508006` },
   {0.30353099910334297` , -0.9341723589627156` , -0.18759247408508004` },
   {-0.7946544722917661` , -0.5773502691896258` , -0.18759247408507992` },
   {-0.30353099910334314` , -0.9341723589627158` , 0.18759247408507987` },
   {-0.1875924740850798` , -0.5773502691896257` , 0.7946544722917661` },
   {0.4911234731884231` , -0.35682208977309005` , 0.7946544722917661` },
   {-0.1875924740850798` , 0.5773502691896257` , 0.7946544722917662` },
   {-0.9822469463768461` , 0, 0.18759247408507998` },
   {0.49112347318842303` , 0.35682208977308993` , 0.7946544722917661` },
   {0.7946544722917661` , -0.5773502691896258` , 0.18759247408507973` },
   {0.9822469463768461` , 0, -0.18759247408507998` },
   {-0.6070619982066862` , 0, 0.7946544722917661` },
   {-0.4911234731884231` , 0.35682208977309016` , -0.7946544722917661` },
   {-0.49112347318842303` , -0.35682208977308993` , -0.7946544722917661` },
   {0.7946544722917661` , 0.5773502691896257` , 0.18759247408507987` }};
```

In[ ]:=
```
axisPoints2 =
{{0.5257311121191336` , 0, 0.85065080835204` },
   {0.1624598481164532` , 0.5` , 0.8506508083520399` },
   {0.6881909602355866` , 0.49999999999999994` , 0.5257311121191336` },
   {-0.42532540417601994` , 0.30901699437494734` , 0.8506508083520401` },
   {-0.42532540417601994` , -0.30901699437494734` , 0.8506508083520401` },
   {-0.2628655560595666` , 0.8090169943749473` , 0.5257311121191337` },
   {-0.8506508083520399` , 0, 0.5257311121191337` },
   {-0.26286555605956663` , -0.8090169943749473` , 0.5257311121191336` },
   {0.16245984811645328` , -0.5000000000000001` , 0.85065080835204` },
   {0.6881909602355869` , -0.5000000000000001` , 0.5257311121191335` },
   {0.9510565162951536` , -0.30901699437494745` , 0},
   {0.5877852522924731` , -0.8090169943749475` , 0},
   {0.9510565162951536` , 0.30901699437494734` , 0},
   {0.5877852522924731` , 0.8090169943749476` , 0},
   {-0.5257311121191336` , 0, -0.85065080835204` },
   {-0.1624598481164532` , -0.5` , -0.8506508083520399` },
   {-0.6881909602355866` , -0.49999999999999994` , -0.5257311121191336` },
   {0.42532540417601994` , -0.30901699437494734` , -0.8506508083520401` },
   {0.42532540417601994` , 0.30901699437494734` , -0.8506508083520401` },
   {0.2628655560595666` , -0.8090169943749473` , -0.5257311121191337` },
   {0.8506508083520399` , 0, -0.5257311121191337` },
   {0.26286555605956663` , 0.8090169943749473` , -0.5257311121191336` },
   {-0.16245984811645328` , 0.5000000000000001` , -0.85065080835204` },
   {-0.6881909602355869` , 0.5000000000000001` , -0.5257311121191335` },
   {-0.9510565162951536` , 0.30901699437494745` , 0},
   {-0.5877852522924731` , 0.8090169943749475` , 0},
   {-0.9510565162951536` , -0.30901699437494734` , 0},
   {-0.5877852522924731` , -0.8090169943749476` , 0}, {0, 1, 0}, {0, -1, 0}};
```

```
In[ ]:= Show[ContourPlot3D [x ^ 2 + y ^ 2 + z ^ 2 == 1, {x, -1.1, 1.1},
          {y, -1.1, 1.1}, {z, -1.1, 1.1}, ContourStyle → LightYellow , Mesh → None],
        Graphics3D [{PointSize [.025], {Green , Point[axisPoints5 ]}, {Red , Point[axisPoints3 ]},
          {Blue , Point[axisPoints2 ]}}], Boxed → False , Axes → None]
```

Out[ ]=



3.1.5 The Dodecahedron

Just like the octahedron and Hexahedron share symmetry groups so do the icosahedron and dodecahe - dron. The vertices of the dodecahedron are the centroids of the icosahedron and the the centroids of faces of the dodecahedron. They share centroids of the 30 edges, but not the edges themselves.

We can immediately draw this

```
Out[ ]= {{-0.187592 , -0.57735 , 0.794654},
          {0.491123 , -0.356822 , 0.794654}, {0.491123 , 0.356822 , 0.794654},
          {-0.187592 , 0.57735 , 0.794654}, {-0.607062 , 0, 0.794654}}
```

```
In[ ]:= dPanel1
```

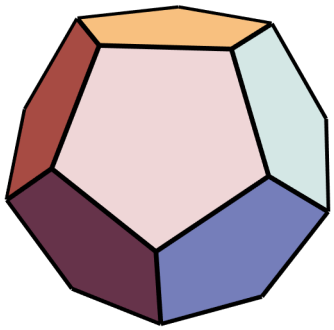
```
Out[ ]= {{-0.187592 , -0.57735 , 0.794654},
          {0.491123 , -0.356822 , 0.794654}, {0.491123 , 0.356822 , 0.794654},
          {-0.187592 , 0.57735 , 0.794654}, {-0.607062 , 0, 0.794654}}
```

```
In[ ]:= DcentA = groupAssoc [GIcos , ⟨| 1 → σ, 2 → τ|⟩ , {0, 0, 1}];
```

```
In[ ]:= dPanels = Table[TasTF[key , ⟨| 1 → σ, 2 → τ|⟩]@dPanel1 , {key , Keys[DcentA]}];
```

In[ ]:=
```
Graphics3D[Table[{EdgeForm[{Black, Thickness[.01]}], Polygon[dPanels[[i]]]}, {i, 12}],
 Boxed → False, ImageSize → Small]
```

Out[ ]=



The rotations are of the 3 types. One axis type goes through opposite centroids of panels, a second goes through opposite vertices and the third goes between opposite edge midpoints.

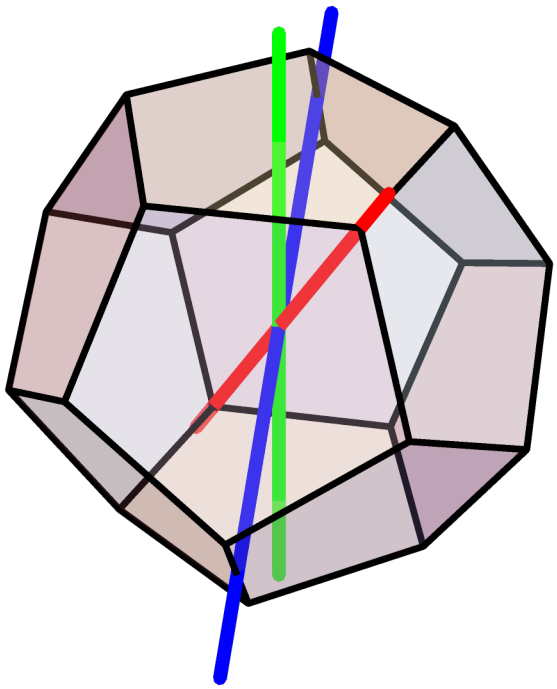
In[ ]:=
```
vert = axisPoints3[[11]]
mide = axisPoints2[[6]]
```

Out[ ]= {0.491123, -0.356822, 0.794654}

Out[ ]= {-0.262866, 0.809017, 0.525731}

In[ ]:=
```
Graphics3D[
 {Table[{Opacity[.3], EdgeForm[{Black, Thickness[.01]}], Polygon[dPanels[[i]]], {i, 12}],
  {Green, Thickness[.02], Line[{{0, 0, 1.2}, {0, 0, -1.2}}]},
  {Red, Thickness[.02], Line[{1.3 vert, -1.3 vert}]},
  {Blue, Thickness[.02], Line[{1.3 mide, -1.3 mide}]}}, Boxed → False, ImageSize → Medium]
```

Out[ ]=



3.2 Paneling the Sphere -- Soccer Balls

In my general audience paper Soccer Balls barryhdayton.space/SoccerBalls.pdf I promised to give the actual code used. This subsection will do this, however it will not be necessary to read that paper first. Before beginning I give several routines which will allow me to work on the sphere instead of polygons.

The main routine is

In[ ]:=
```
Options[sphereLn] = {len → False, del → .1};
sphereLn[p_, q_, OptionsPattern []] := Module[{r, t, d, length},
  If[.96 > Norm[p] || Norm[p] > 1.02 || .98 > Norm[q] || Norm[q] > 1.04,
   Echo["sphereLn error,points must be on sphere of radius 1"];
   Abort[]];
  d = OptionValue[del];
  r = Norm[p];
  T = Table[r (p + t (q - p)) / Norm[(p + t (q - p))], {t, 0, 1, d}];
  length = Total[Table[Norm[T[[i + 1]] - T[[i]]], {i, Length[T] - 1}]];
  If[OptionValue[len] == False, Return[T]];
  If[OptionValue[len] == True, Return[length]];
  Echo[length, "length"]; T]

spLn[p_, q_] := sphereLn[p, q]
spLen[p_, q_] := sphereLn[p, q, len → True]
```

This routine returns a list of points del apart from p to q . The actual line will be Line[spLn[p,q]] inside a Graphics3d routine.

This has two short forms, spLn[p,q] gives the spherical line from p to q, spLen[p,q] gives the length of that line as a curve in 3-space.

We have a special routine to find the midpoint of a spherical line . Warning: if the option **del** of **sphereLn** is used then the new middle point of **sphereLn** should be used.

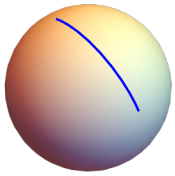
In[ ]:=
```
spMid[a_, b_] = spLn[a, b][[6]];
```

Since I will have many graphics with a sphere a quick syntax will be

In[ ]:=
```
StandardSphere := ContourPlot3D[x^2 + y^2 + z^2 == 1, {x, -1.2, 1.2}, {y, -1.2, 1.2},
  {z, -1.2, 1.2}, Mesh → None, ContourStyle → LightYellow, Axes → False]
```

In[ ]:=
```
Show[StandardSphere, Graphics3D[{Blue, Thickness[.005], Line[spLn[{1, 0, 0}, {0, 0, 1}]]}],
 Boxed → False, ImageSize → Small]
```

Out[ ]=



The length is approximately

In[ ]:=
```
spLen[{1, 0, 0}, {0, 0, 1}]
```

Out[ ]=
```
1.56899
```

For a better approximation compare the precise answer

In[ ]:=
```
sphereLn[{1, 0, 0}, {0, 0, 1}, del → 1.*^-5 , len → True]
```

Out[ ]=
```
1.5708
```

In[ ]:=
```
1.570796326776746`
```

Out[ ]=
```
1.5708
```

In[ ]:=
```
N[Pi / 2]
```

Out[ ]=
```
1.5708
```

In[ ]:=
```
1.5707963267948966`
```

Out[ ]=
```
1.5708
```

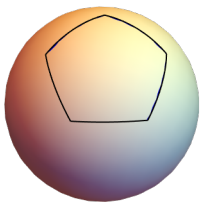
We also wish to draw and color a spherical polygon . If we just want the outline we have the following routine which is used inside the graphic primitive **Polygon.**

In[ ]:=
```
spOutline[P_] := Module[{P1, n},
  n = Length[P];
  P1 = Append[P, P[[1]]];
  Flatten[Table[spLn[P1[[j]], P1[[j + 1]]], {j, n}], 1]]
```

For example dPanel1 on the spherical decahedron is

In[ ]:=
```
Show[StandardSphere , Graphics3D[{Blue, Thickness[.02], Polygon[spOutline[dPanel1]]}],
 Boxed → False, ImageSize → Small]
```

Out[ ]=



To fill polygons have subroutines

In[ ]:=
```
subdivide3[{a_, b_, c_}] := Module[{d, s12, s21, s13, s31, s23, s32},
  d = spcentroid[{a, b, c}];
  s12 = {a, d, spMid[a, b]};
  s21 = {b, d, spMid[b, a]};
  s13 = {a, d, spMid[a, c]};
  s31 = {c, d, spMid[c, a]};
  s23 = {b, d, spMid[b, c]};
  s32 = {c, d, spMid[c, b]}; {s12, s21, s13, s31, s23, s32}]

subdividePolygon1[P_] := Module[{n, P1, cP, Q},
  n = Length[P];
  If[n == 3, Return[subdivide3[P]]];
  P1 = Append[P, P[[1]]];
  cP = spcentroid[P];
  Table[{P1[[j]], cP, P1[[j + 1]]}, {j, n}]
 ]
```

In[ ]:=
```
spPolygon[P_] := subdividePolygon[subdividePolygon[subdividePolygon[P]]]
```

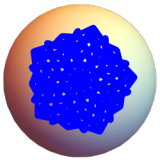
In[ ]:=
```
subdividePolygon[P_] := Module[{k, n, P1, cP},
  If[Depth[P] == 3, Return[subdividePolygon1[P]]];
  Flatten[Table[subdividePolygon1[P[[i]]], {i, Length[P]}], 1]]
```

For small polygons the following may be sufficient

In[ ]:=
```
spPolygon2[P_] := subdividePolygon[subdividePolygon[P]]
```

In[ ]:=
```
Show[StandardSphere , Graphics3D[
  {Blue, EdgeForm[{Blue, Thickness[.02]}], Thickness[.02], Line[spPolygon[dPanel1]]}],
 Boxed → False, ImageSize → Small, ViewPoint → Above]
```

Out[ ]=



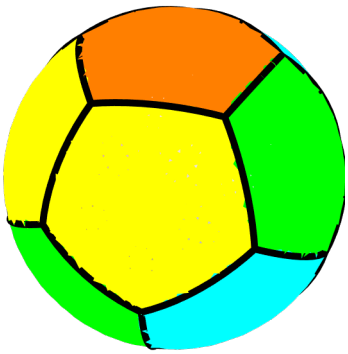
To color the entire spherical dodecahedron

```
In[ ]:= dcol = Join[{Orange , Yellow , Green , Cyan},
          {Orange , Yellow , Green , Cyan}, {Orange , Yellow , Green , Cyan}]
```

```
Out[ ]= {■, ■, ■, ■, ■, ■, ■, ■, ■, ■, ■, ■}
```

```
In[ ]:= Graphics3D [Table[
          {Thickness [.02], EdgeForm [{dcol⟦i⟧, Thickness [.03]}], Polygon [spPolygon [dPanels⟦i⟧]],
           Line[1.02 spOutline [dPanels⟦i⟧]]}, {i , 12}], Boxed → False , ImageSize → Small]
```

Out[ ]=



For the purpose of paneling soccer balls dodecahedral paneling is often used, especially by **Nike** .

In[ ]:=

3.1 More on the group GIcos, the Copa America Ball

All the elements of the group GIcos, other than the identity, are given by a power of a RotationTransfor - m[θ,a] where a is any one of the 62 axis points given in 3.1.4 and $\theta = 2\pi/n$ where n is the order of the group element, $n = 2$, 3 or 5. There are of course duplicates since these 62 points are composed of 31 pairs of antipodal points. If b is antipodal to α the RotationTransform[θ,β] is the inverse of Rotation - Transform[θ,α], same θ. Since these rotations have finite order the inverse is a power.

Further, note that the group GIcos also acts on this set of axis Points, in fact on each of the 3 subsets separately. This makes the set useful for constructing panelings of the sphere. An example follows.

A strange soccer ball was designed by Puma to be used in the 2024 Copa America. The important thing is the group GIcos which has a 5 fold rotation. Since we are only accepting direct isometries which can be physically applied to a soccer ball, there are no reflections. Thus our panels must have a rotational symmetry but need not be symmetric. The Copa America ball, made by Puma, takes advantage of this as the ball is designed to have symmetry group GIcos.

Construction of a replica ball is a good example of the use of our explicit description of GIcos.

We recall in 3.1.4

```
In[ ]:= σ = N[RotationTransform [2 Pi / 5, {0, 0, 1}]];
```

We start with a 3 - fold axis point

```
In[ ]:= a = axisPoints3 ⟦10⟧
Out[ ]= {-0.187592 , -0.57735 , 0.794654}
```

We then construct a specific, but somewhat arbitrary point b using a rotation in GIcos..

```
In[ ]:= b1 = spLn[axisPoints3 ⟦10⟧, axisPoints5 ⟦10⟧][4];
        τ = RotationTransform [2 Pi / 3, axisPoints3 ⟦10⟧];
        b = τ@b1
Out[ ]= {-0.137372 , -0.422788 , 0.895756}
```

Now we rotate that point about a half turn in GIcos

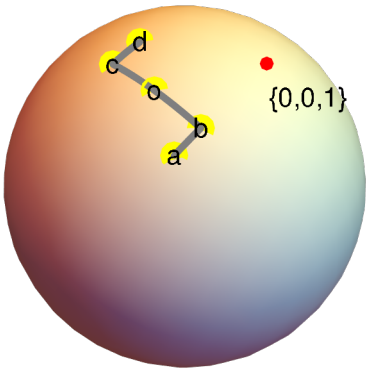
```
In[ ]:= o = axisPoints2 ⟦5⟧;
In[ ]:= ρ = RotationTransform [Pi , o];
        c = ρ@b
Out[ ]= {-0.671641 , -0.164995 , 0.722271}
```

and end at another 3 - fold axis point

```
In[ ]:= d = d = axisPoints3 ⟦17⟧
Out[ ]= {-0.607062 , 0, 0.794654}
```

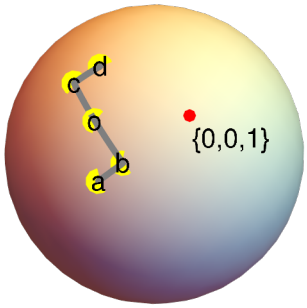
In[ ]:=
```
Show[StandardSphere , Graphics3D [{{Yellow , PointSize [.04],
    Point[{axisPoints3 ⟦10⟧, axisPoints2 ⟦5⟧, axisPoints3 ⟦17⟧, b, c}]},
   {Black , Text[Style["a", 14], {axisPoints3 ⟦10⟧}], Text[Style["o", 14], axisPoints2 ⟦5⟧],
    Text[Style["d", 14], axisPoints3 ⟦17⟧], Text[Style["b", 14], b],
    Text[Style["c", 14], c], Text[Style["{0,0,1}", 14], Normalize [{.3 , .1 , 1}]]},
   {Gray , Thickness [.01], Line[spLn[a, b]], Line[spLn[b, c]], Line[spLn[c , d]]},
   {Red , PointSize [.02], Point[{0 , 0 , 1.1}]}}], Boxed → False]
```

Out[ ]=

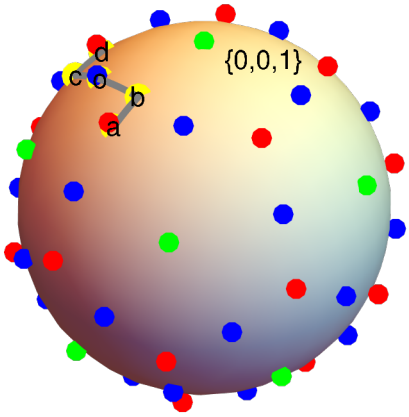


We compare with the panel on the Puma Copa America ball

In[ ]:=



Out[ ]=



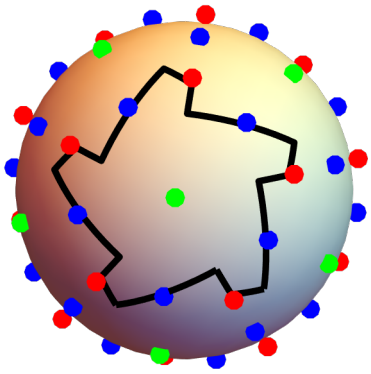
Now we have our base panel

In[ ]:=
```
capPanel0 = Reverse[RecurrenceTable [{s[i + 1] == σ@s[i], s[1] == {a, b, c, d}}, s, {i, 5}], 1];
```

In[ ]:=
```
ncapPanel0 = Normalize[#] & /@ Flatten[capPanel0 , 1];
```

In[ ]:=
```
Show[StandardSphere ,
 Graphics3D [{{Green , PointSize [.03], Point[1.03 axisPoints5 ]}, {Blue , PointSize [.03],
    Point[1.03 axisPoints2 ]}, {Red , PointSize [.03], Point[1.05 axisPoints3 ]},
  {Green , PointSize [.03], Point[1.03 axisPoints5 ]}, {Blue , PointSize [.03],
    Point[1.03 axisPoints2 ]}, {Red , PointSize [.03], Point[1.05 axisPoints3 ]},
  {Black , Thickness [.01], Line[spOutline [ncapPanel0 ]]}}], Boxed → False]
```

Out[ ]=



We note that the group association we need is just that of the dodecahedron so we panel our ball.

In[ ]:=
```
DcentA = groupAssoc [GIcos , ⟨| 1 → σ, 2 → τ |⟩ , {0, 0, 1}];
capPanels = Table[TasTF[Keys[DcentA]⟦i⟧, ⟨| 1 → σ, 2 → τ |⟩]@ncapPanel0 , {i , 12}];
```

In[ ]:=
```
Show[StandardSphere , Graphics3D[
  {Black , Thickness[.01], Table[Line[spOutline[capPanels⟦i⟧]], {i, 12}]}], Boxed → False]
```

Out[ ]=

3.2.1 The traditional Soccer Ball

This ball motivated this chapter. Originally these were hand sewed with 32 panels, 12 black pentagons and 20 white hexagons. Here, as in the rest of this section, there will be, unlike the platonic solids, two or more spherical polygonal shaped panels.

A rotation transform will be a panel symmetry if it takes panels to like panels. The set of panel symmetries of a paneled sphere is clearly a group. In this case, if embedded correctly in 3-space the group of panel symmetries will be the group GIcos. As we will see, there are many possible paneling with this group.

Here is how we can plot a traditional soccer ball using **Mathematica** .

We use σ, τ from section 3.1.4, let

```
In[ ]:= hexCent = {0.3902734644166457` , -0.28355026945067996` , 0.6314757303333052` }
Out[ ]= {0.390273 , -0.28355 , 0.631476}
```

```
In[ ]:= σ = N[RotationTransform [2 Pi / 5, {0, 0, 1}]];
        τ = RotationTransform [2 Pi / 3, hexCent];
```

These generated the group

```
In[ ]:= GIcos = finiteTransGroup [<| 1 → σ, 2 → τ |> , {2.316 , -1.347 , .3712}, 10];
  » number of group elements calculated  60
```

Sparing the reader the details we define a pentagon

```
In[ ]:= PentPanel0 = {{0.3432786130319566` , 1.0904807614249159`*^-17 , 0.9392336204772782` },
          {0.10607892523233595` , -0.32647736182880455` , 0.9392336204772784` },
          {-0.27771823174831395` , -0.20177410616759894` , 0.9392336204772789` },
          {-0.2777182317483142` , 0.2017741061675989` , 0.9392336204772782` },
          {0.10607892523233588` , 0.3264773618288047` , 0.9392336204772782` }}
Out[ ]= {{0.343279 , 1.09048 × 10^-17, 0.939234},
         {0.106079 , -0.326477 , 0.939234}, {-0.277718 , -0.201774 , 0.939234},
         {-0.277718 , 0.201774 , 0.939234}, {0.106079 , 0.326477 , 0.939234}}
```

and a hexagon

```
In[ ]:= HexPanel0 = {{0.3432786130319566` , 1.0904807614249159`*^-17 , 0.9392336204772782` },
          {0.10607892523233595` , -0.32647736182880455` , 0.9392336204772784` },
          {0.21215785046467175` , -0.6529547236576094` , 0.7270757700126067` },
          {0.5554364634966286` , -0.6529547236576096` , 0.5149179195479349` },
          {0.7926361512962496` , -0.32647736182880477` , 0.5149179195479349` },
          {0.6865572260639138` , 4.11934439063236`*^-16 , 0.7270757700126069` }}
Out[ ]= {{0.343279 , 1.09048 × 10^-17, 0.939234}, {0.106079 , -0.326477 , 0.939234},
         {0.212158 , -0.652955 , 0.727076}, {0.555436 , -0.652955 , 0.514918},
         {0.792636 , -0.326477 , 0.514918}, {0.686557 , 4.11934 × 10^-16, 0.727076}}
```

These are left unchanged by σ, τ respectively.

```
In[ ]:= σ @ PentPanel0
Out[ ]= {{0.106079 , 0.326477 , 0.939234},
         {0.343279 , 1.11022 × 10^-16, 0.939234}, {0.106079 , -0.326477 , 0.939234},
         {-0.277718 , -0.201774 , 0.939234}, {-0.277718 , 0.201774 , 0.939234}}
```

In[ ]:=
```
τ@HexPanel0
```

Out[ ]=
```
{{0.212158 , -0.652955 , 0.727076}, {0.555436 , -0.652955 , 0.514918},
 {0.792636 , -0.326477 , 0.514918}, {0.686557 , 1.66533 × 10^-16 , 0.727076},
 {0.343279 , 1.66533 × 10^-16 , 0.939234}, {0.106079 , -0.326477 , 0.939234}}
```

Actually HexPanel0 is left unchanged by 6-fold rotation so is a regular hexagon. However there are only 3 pentagons adjacent to a given hexagon on the soccer ball so there is only a three fold rotation of the soccer ball at the center of a hexagonal panel.

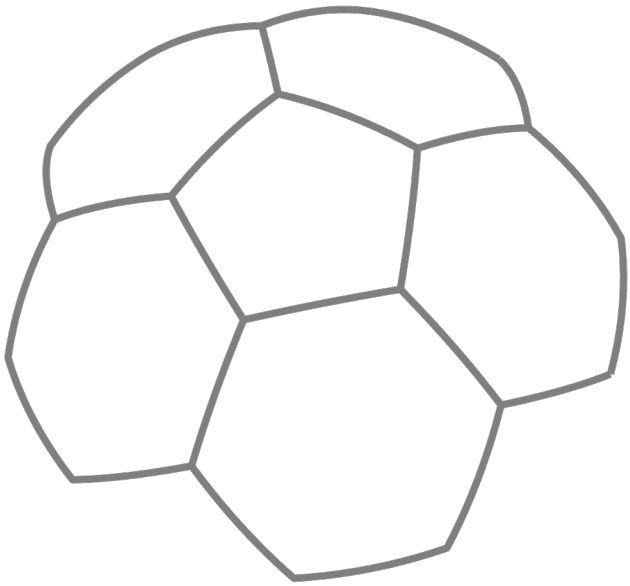
In[ ]:=
```
RotationTransform[2 Pi / 6, hexCent]@HexPanel0
```

Out[ ]=
```
{{0.106079 , -0.326477 , 0.939234}, {0.212158 , -0.652955 , 0.727076},
 {0.555436 , -0.652955 , 0.514918}, {0.792636 , -0.326477 , 0.514918},
 {0.686557 , 2.77556 × 10^-16 , 0.727076}, {0.343279 , 3.88578 × 10^-16 , 0.939234}}
```

This pentagon and hexagon have been chosen because, just looking at their outlines, we have the following picture.

In[ ]:=
```
Graphics3D[{Gray, Thickness[.01],
  Line[1.02 spOutline[HexPanel0]], Line[1.02 spOutline[σ@HexPanel0]],
  Line[1.02 spOutline[σ@*σ@HexPanel0]], Line[1.02 spOutline[σ@*σ@*σ@HexPanel0]],
  Line[1.02 spOutline[σ@*σ@*σ@*σ@HexPanel0]],
  Line[1.02 spOutline[PentPanel0]]}, Boxed → False]
```

Out[ ]=



The 5 regular hexagons exactly surround the regular pentagon. Remember these are all drawn on the sphere, this picture cannot be drawn on the plane (check angles)!

To draw the entire figure

```
In[ ]:= GIcosHexA = groupAssoc[GIcos, ⟨|1 → σ, 2 → τ|⟩, hexCent];

In[ ]:= Length[GIcosHexA]

Out[ ]= 20

In[ ]:= hexPanels = Table[TasTF[key, ⟨|1 → σ, 2 → τ|⟩]@HexPanel0, {key, Keys[GIcosHexA]}];

In[ ]:= GIcosPentA = groupAssoc[GIcos, ⟨|1 → σ, 2 → τ|⟩, {0, 0, 1}];

In[ ]:= Length[GIcosPentA]

Out[ ]= 12

In[ ]:= pentPanels = Table[TasTF[key, ⟨|1 → σ, 2 → τ|⟩]@PentPanel0, {key, Keys[GIcosPentA]}];

In[ ]:= Graphics3D[{Table[{Gray, Thickness[.009], EdgeForm[{LightOrange, Thickness[.03]}],
          Polygon[spPolygon[hexPanels⟦i⟧]], Line[1.02 spOutline[hexPanels⟦i⟧]]}, {i, 1, 20}],
        {EdgeForm[{Black, Thickness[.03]}], Table[Polygon[spPolygon[pentPanels⟦i⟧]], {i, 12}]}},
       Boxed → False, ImageSize → Small]

Out[ ]=
```

3.2.2 Variations on the traditional soccer ball

The traditional ball has a regular pentagon and hexagon. There is only one possibility of the ratio of the sides of the pentagon vs. the radius of the ball. However if we relax the condition that the hexagonal panel is regular to simply having 3-fold rotations about the centroids then we can produce an infinite family of balls. This can be seen on many imitation soccer balls.

So we pick a point **p1** on the unit sphere closer to {0,0,1} than the vertexes of the traditional pentagon.

```
In[ ]:= p1 = spLn[{0, 0, 1}, PentPanel0⟦1⟧⟦8⟧]

Out[ ]= {0.243421, 7.73268 × 10^-18, 0.969921}
```

We will have the pentagon

```
In[ ]:= smPent = RecurrenceTable[{p[i + 1] == σ@p[i], p[1] == p1}, p, {i, 5}]

Out[ ]= {{0.243421, 7.73268 × 10^-18, 0.969921},
         {0.0752213, 0.231508, 0.969921}, {-0.196932, 0.14308, 0.969921},
         {-0.196932, -0.14308, 0.969921}, {0.0752213, -0.231508, 0.969921}}
```

These small pentagons can then populate the sphere

```
In[ ]:= smPentPanels = Table[TasTF[key, <|1 → σ, 2 → τ|>]@smPent, {key, Keys[GIcosPentA]}];

In[ ]:= Show[StandardSphere, Graphics3D[
          {{EdgeForm[{Cyan, Thickness[.03]}], Table[Polygon[spPolygon[smPentPanels[[i]]], {i, 12}]}}],
        ImageSize → Small, Boxed → False]
```

Out[ ]=

I have drawn a hexagon with a 3-fold, but not 6 -fold rotation symmetry around its centroid.

```
In[ ]:= smHex = {smPentPanels[[1, 3]], smPentPanels[[6, 2]], smPentPanels[[6, 1]],
            smPentPanels[[2, 2]], smPentPanels[[2, 1]], smPentPanels[[1, 4]]};
```

This hexagon has centroid

```
In[ ]:= smHexCent = Normalize[Total[smHex]/6]
Out[ ]= {-0.187592, -0.57735, 0.794654}
```

which is just

```
In[ ]:= axisPoints3[[10]]
Out[ ]= {-0.187592, -0.57735, 0.794654}
```

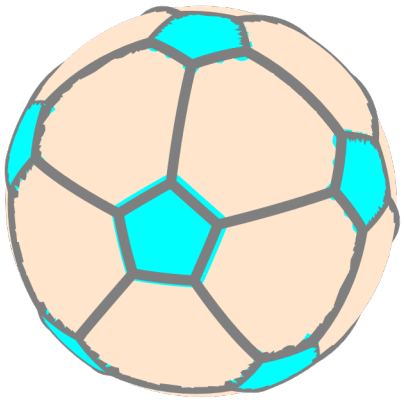
Then using our standard procedure we get panels

```
In[ ]:= GIcossmHexA = groupAssoc[GIcos, <|1 → σ, 2 → τ|>, smHexCent];

In[ ]:= smhexPanels = Table[TasTF[key, <|1 → σ, 2 → τ|>]@smHex, {key, Keys[GIcossmHexA]}];
```

giving the graphic

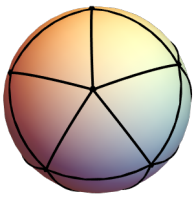
In[ ]:=
```
Show[StandardSphere , Graphics3D [{{EdgeForm [{Cyan , Thickness [.03]}],
    Table[Polygon[1.0 spPolygon [smPentPanels ⟦i⟧]], {i , 12}]},
  {Gray , Thickness [.015], Table[Line[1.04 spOutline [smhexPanels ⟦i⟧]], {i , 20}]},
  {EdgeForm [{Thickness [.03], LightOrange }],
    Table[Polygon[.99 spPolygon [smhexPanels ⟦i⟧]], {i , 20}]}}], Boxed → False]
```

Out[ ]=



If we continue shrinking the size of the pentagons we end up with

In[ ]:=



which is essentially an icosahedron .

On the other hand we could take a larger pentagon than the traditional one . We keep the center at {0,0,1} but put one vertex at

In[ ]:=
```
vlp1 = {0.4500000000000001` , 0.` , 0.8930285549745878` }
```

Out[ ]=
```
{0.45 , 0. , 0.893029}
```

In[ ]:=
```
vlPent = RecurrenceTable [{p[i + 1] == κ@p[i], p[1] == vlp1}, p, {i , 5}]
```

Out[ ]=
```
{{0.45 , 0. , 0.893029}, {-0.225 , 0.389711 , 0.893029}, {-0.225 , -0.389711 , 0.893029},
  {0.45 , 2.77556 × 10^-17 , 0.893029}, {-0.225 , 0.389711 , 0.893029}}
```

Using a similar construction to the above we get

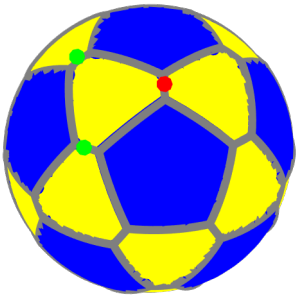
In[ ]:=

Continuing enlarging the pentagon we don't get a dodecahedron. Rather the hexagons will become triangles, with one vertex, red, at

In[ ]:=
```
vlp2 = axisPoints2[[1]]
```
Out[ ]=
```
{0.525731 , 0, 0.850651}
```

This pentagon will be

In[ ]:=



The triangle formed by the red and green points will replace the hexagons.

In[ ]:=
```
triPanel0 = {axisPoints2[[1]], axisPoints2[[2]], axisPoints2[[3]]}
```
Out[ ]=
```
{{0.525731 , 0, 0.850651}, {0.16246 , 0.5 , 0.850651}, {0.688191 , 0.5 , 0.525731}}
```

The pentagon will be

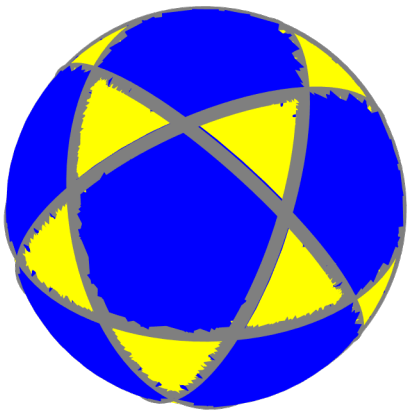
In[ ]:=
```
vlpent2 = RecurrenceTable [{q[i + 1] == σ @ q[i], q[1] == vlp2}, q , {i , 5}]
```
Out[ ]=
```
{{0.525731 , 0. , 0.850651}, {0.16246 , 0.5 , 0.850651}, {-0.425325 , 0.309017 , 0.850651},
 {-0.425325 , -0.309017 , 0.850651}, {0.16246 , -0.5 , 0.850651}}
```

```
In[ • ]:= centerTriPanel0 = Normalize[Total[triPanel0] / 6]
Out[ • ]= {0.491123 , 0.356822 , 0.794654}
```

We now proceed as before

```
In[ • ]:= vlPentPanels = Table[TasTF[key, ⟨| 1 → σ, 2 → τ |⟩]@vlpent2 , {key, Keys[GIcosPentA ]}];
In[ • ]:= triPanelA = groupAssoc [GIcos , ⟨| 1 → σ, 2 → τ |⟩ , centerTriPanel0 ];
In[ • ]:= triPanels = Table[TasTF[key, ⟨| 1 → σ, 2 → τ |⟩]@triPanel0 , {key, Keys[triPanelA ]}];
In[ • ]:= Show[StandardSphere , Graphics3D [
    {{EdgeForm [{Thickness [.03], Blue}], Table[Polygon[1.01 spPolygon [vlPentPanels ⟦i⟧]],
       {i, 12}]}, {EdgeForm [{Thickness [.03], Yellow}],
     Table[Polygon[spPolygon [triPanels ⟦i⟧]], {i, 20}], Gray, Thickness [.02],
     Table[Line[1.04 spOutline [triPanels ⟦i⟧]], {i, 20}]}}, Boxed → False]
```

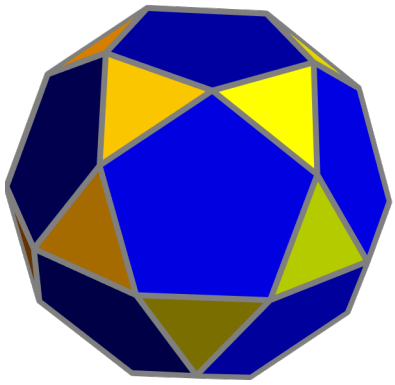
Out[•]=



The corresponding polyhedron is known as the icosidodecahedron.

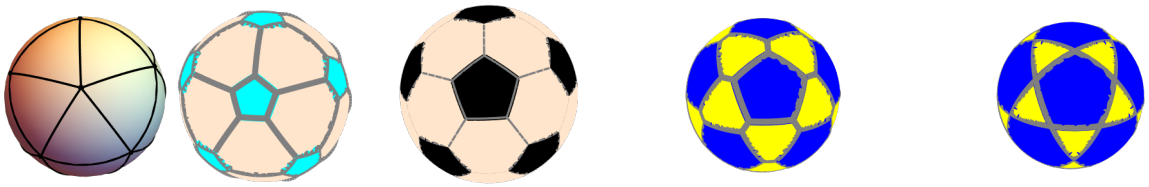
In[ ]:=
```
Graphics3D[
 {{Blue, EdgeForm[{Thickness[.015], Gray}], Table[Polygon[vlPentPanels⟦i⟧], {i, 12}]},
  {Yellow, EdgeForm[{Thickness[.015], Gray}],
   Table[Polygon[triPanels⟦i⟧], {i, 20}]}}, Boxed → False]
```

Out[ ]=

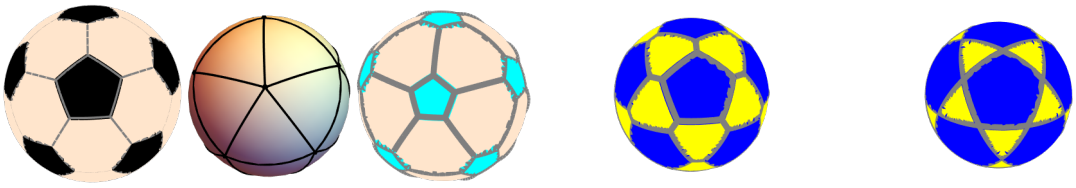


So we have a continuum of Archimedean polyhedra/paneling from the icosahedron to the icosidodeca - hedron which has the traditional soccer ball in the middle.

In[ ]:=



Out[ ]=



3.2.3 Another continuum

We give another continua of polyhedra directly from the icosahedron to the dodecahedron, or their ball panelings with GIcos as their transformation group. This will miss the soccer balls but clearly show the group action. But they will be in the class of prisms and now we will have 3 types of sides/panels.

We will define the initial panels by using the pentagons above starting with the traditional soccer ball

pentagon.

```
In[ ]:= PentPanel0
Out[ ]= {{0.343279 , 1.09048 × 10^-17 , 0.939234},
  {0.106079 , -0.326477 , 0.939234}, {-0.277718 , -0.201774 , 0.939234},
  {-0.277718 , 0.201774 , 0.939234}, {0.106079 , 0.326477 , 0.939234}}
```

We apply a rotation about {0,0,1}, the centroid, of angle $2\pi/10$

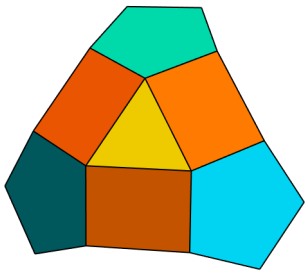
```
In[ ]:= υ = N[RotationTransform [Pi / 5, {0, 0, 1}]]
Out[ ]= TransformationFunction [(  0.809017  -0.587785  0.  |  0.
                                   0.587785   0.809017  0.  |  0.
                                   0.         0.        1.  |  0.
                                  ------------------------------
                                   0.         0.        0.  |  1.  )]
```

Our basic panel is then

```
In[ ]:= prismPanel0 = υ@PentPanel0
Out[ ]= {{0.277718 , 0.201774 , 0.939234},
  {0.277718 , -0.201774 , 0.939234}, {-0.106079 , -0.326477 , 0.939234},
  {-0.343279 , 0. , 0.939234}, {-0.106079 , 0.326477 , 0.939234}}
```

Applying τ to this panel we get more copies of this panel and we see the space between adjacent panels can be filled by 2 rectangles and an equilateral triangle.

```
In[ ]:=
```



We now follow our basic construction,

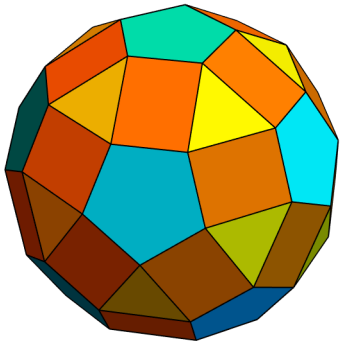
```
In[ ]:= prismPentA = groupAssoc [GIcos , ⟨| 1 → σ, 2 → τ |⟩ , {0, 0, 1}];
        prismPentPanels =
          Table[TasTF[key, ⟨| 1 → σ, 2 → τ |⟩]@prismPanel0 , {key, Keys[prismPentA ]}];

In[ ]:= prismRect0 = {{-0.697756276176401` , -0.25754309750239746` , 0.6684367823401948` },
          {-0.34327861303195667` , 0.` , 0.9392336204772782` },
          {-0.10607892523233564` , -0.3264773618288047` , 0.9392336204772789` },
          {-0.4605565883767806` , -0.5840204593312023` , 0.6684367823401951` }};

In[ ]:= prismRect0cent = Normalize [Total[prismRect0 ]]
Out[ ]= {-0.425325 , -0.309017 , 0.850651}
```

```
In[ ]:= prismRectA = groupAssoc[GIcos, <|1 → σ, 2 → τ|>, prismRect0cent];
        prismRectPanels =
          Table[TasTF[key, <|1 → σ, 2 → τ|>]@prismRect0, {key, Keys[prismRectA]}];

In[ ]:= prismTriPanel0 = {{-0.10607892523233564` , -0.3264773618288047` , 0.9392336204772789` },
          {0.02931949383620608` , -0.7431908471556119` , 0.6684367823401948` },
          {-0.4605565883767806` , -0.5840204593312023` , 0.6684367823401951` }};

In[ ]:= prismTriPanel0cent = Normalize[Total[prismTriPanel0]]
Out[ ]= {-0.187592, -0.57735, 0.794654}

In[ ]:= prismTriA = groupAssoc[GIcos, <|1 → σ, 2 → τ|>, prismTriPanel0cent];
        prismTriPanels =
          Table[TasTF[key, <|1 → σ, 2 → τ|>]@prismTriPanel0, {key, Keys[prismTriA]}];
```

Now we can draw our central prism

```
In[ ]:= Graphics3D[{{Cyan, Table[Polygon[prismPentPanels⟦i⟧], {i, 12}]},
          {Orange, Table[Polygon[prismRectPanels⟦i⟧], {i, 30}]},
          {Yellow, Table[Polygon[prismTriPanels⟦i⟧], {i, 20}]}}, Boxed → False, ImageSize → Small]
```

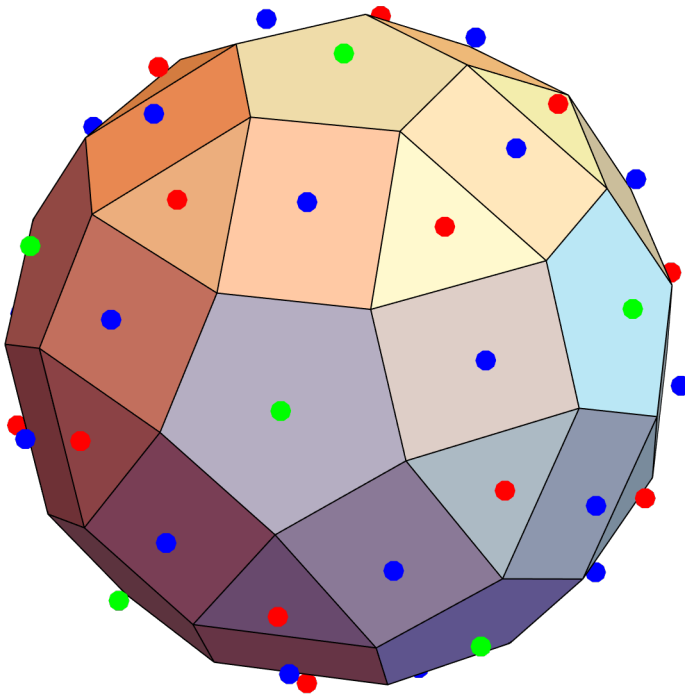
Out[ ]=



Notice we have 62 panels. In fact each panel centroid corresponds to one of our axis points.

In[ ]:=
```
Graphics3D[{{LightCyan, Table[Polygon[prismPentPanels[[i]]], {i, 12}]},
  {LightOrange, Table[Polygon[prismRectPanels[[i]]], {i, 30}]},
  {LightYellow, Table[Polygon[prismTriPanels[[i]]], {i, 20}]},
  {PointSize[.03], {Green, Point[axisPoints5]}, {Red, Point[axisPoints3]},
   {Blue, Point[axisPoints2]}}}, Boxed → False, ImageSize → Medium]
```

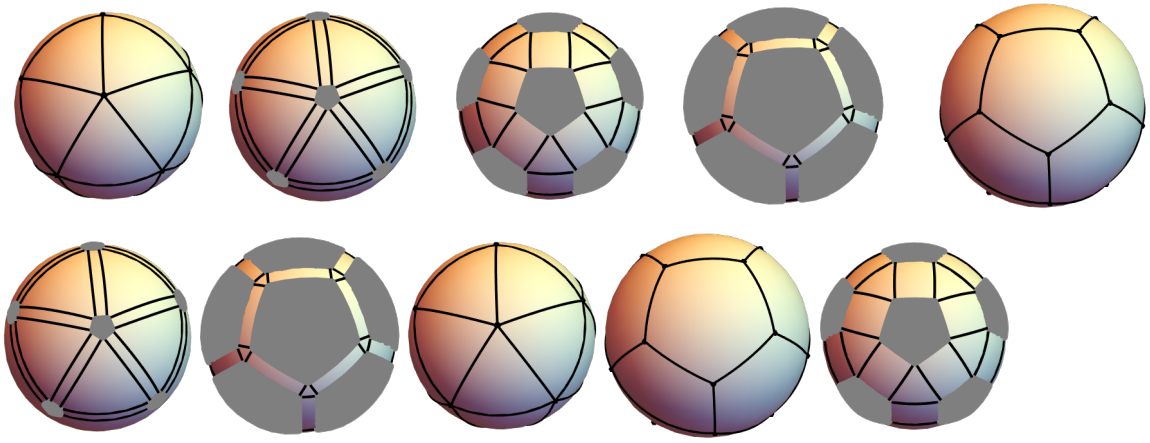
Out[ ]=



As above with the earlier continuum we get another parameterized by the side of a pentagon running from the icosahedron to the dodecahedron. The constructions are the same as before so are not given. The corresponding balls look like this.

In[ ]:=

Out[ ]=



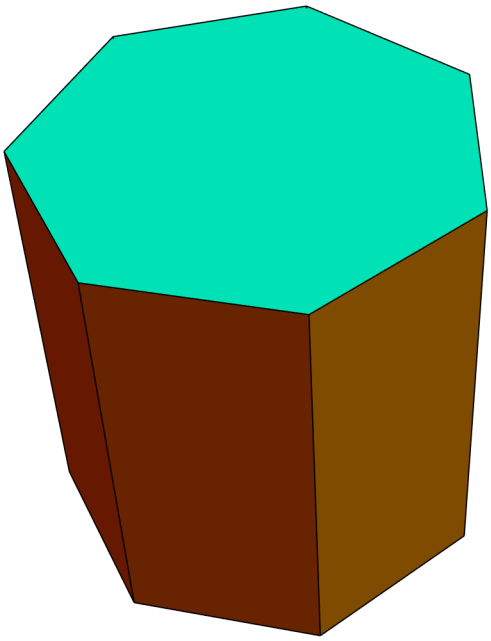
Note the smaller the pentagon the larger the triangles until the pentagon shrinks to a point. As the pentagon shrinks the rectangles shrink in width until they become a line segment. On the the other hand as the pentagons get larger the triangles shrink until they become a point and the rectangles shrink in length until they become a line segment so only the 12 pentagons of the dodecahedron

remain.

3.2.4 The polyhedral can

In addition to the platonic solids there is one other family of solids that have finite rotation groups. I call these cans, they are cylinders but instead of circular the vertical panels are rectangles and the top and bottom are regular polygons. Cylinders don't have finite, or even countable transformation groups but with a top and bottom specified they are compact polyhedra and have finite transformation groups. For example the 7 can, can7, with $-1 \le z \le 1$ looks like this

In[ ]:=



The transformation group is generated by a 7 fold rotation about the z-axis and a half turn about the x-axis.

In[ ]:=
```
θ = N[RotationTransform [2 Pi / 7, {0, 0, 1}]]
η = RotationTransform [Pi, {1, 0, 0}]
```

Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 0.62349 & -0.781831 & 0. & 0. \\ 0.781831 & 0.62349 & 0. & 0. \\ 0. & 0. & 1. & 0. \\ \hline 0. & 0. & 0. & 1. \end{array}\right)\right]$$

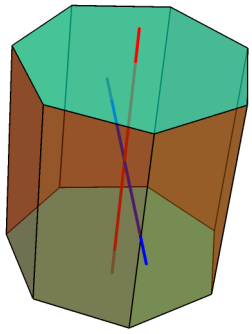
Out[ ]=
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

```
In[ ]:= D7 = finiteTransGroup [<| 1 → θ, 2 → η |> , {2.23 , 3.31 , 1.11}, 5]
```

```
» number of group elements calculated 14
```

```
Out[ ]= {{1}, {2}, {1, 1}, {1, 2}, {2, 1}, {2, 2}, {1, 1, 1}, {1, 1, 2},
  {2, 1, 1}, {2, 1, 2}, {1, 1, 1, 1}, {1, 1, 1, 2}, {2, 1, 1, 1}, {2, 1, 1, 2}}
```

```
In[ ]:=
```



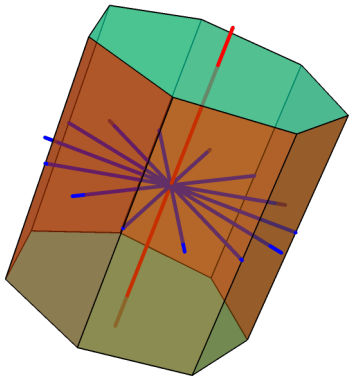
The order association is

```
In[ ]:= ordAss7 = <| Table[k → orderTF[TasTF[k, <| 1 → θ, 2 → η |>], {2.23 , 3.31 , 1.11}, 8], {k, D7}] |>
```

```
Out[ ]= <| {1} → 7 , {2} → 2 , {1, 1} → 7 , {1, 2} → 2 , {2, 1} → 2 ,
  {2, 2} → 1 , {1, 1, 1} → 7 , {1, 1, 2} → 2 , {2, 1, 1} → 2 , {2, 1, 2} → 7 ,
  {1, 1, 1, 1} → 7 , {1, 1, 1, 2} → 2 , {2, 1, 1, 1} → 2 , {2, 1, 1, 2} → 7 |>
```

So we have the 6 rotations of order 7 plus the identity rotation about the z-axis (red) and 7 half turns about horizontal axes in the xy plane from each vertical edge to the midpoint between vertical edges opposite. This is certainly a new group for us. These are shown

```
In[ ]:=
```



We have a similar situation for all integers greater than 2. These groups are called *dihedral* groups of order 2n. If n is odd they have n-1 rotations of order n and the identity rotations and n half-turns order 2. For even n there will be 2 rotations of order n and more than n rotations of order 2. For example if n=8

```
In[ ]:= θ2 = RotationTransform [2. Pi / 8, {0, 0, 1}];
```

```
In[ ]:= D8 = finiteTransGroup [⟨| 1 → θ2 , 2 → η |⟩ , {2.23 , 3.31 , 1.11}, 5]
```

» number of group elements calculated 16

```
Out[ ]= {{1}, {2}, {1, 1}, {1, 2}, {2, 1}, {2, 2}, {1, 1, 1}, {1, 1, 2}, {2, 1, 1}, {2, 1, 2},
  {1, 1, 1, 1}, {1, 1, 1, 2}, {2, 1, 1, 1}, {2, 1, 1, 2}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 2}}
```

```
In[ ]:= ⟨| Table[k → orderTF[TasTF[k, ⟨| 1 → θ2 , 2 → η |⟩], {2.23 , 3.31 , 1.11}, 8], {k, D8}] |⟩
```

```
Out[ ]= ⟨| {1} → 8 , {2} → 2 , {1, 1} → 4 , {1, 2} → 2 , {2, 1} → 2 , {2, 2} → 1 , {1, 1, 1} → 8 ,
  {1, 1, 2} → 2 , {2, 1, 1} → 2 , {2, 1, 2} → 8 , {1, 1, 1, 1} → 2 , {1, 1, 1, 2} → 2 ,
  {2, 1, 1, 1} → 2 , {2, 1, 1, 2} → 4 , {1, 1, 1, 1, 1} → 8 , {1, 1, 1, 1, 2} → 2 |⟩
```

One check on a group being a dihedral group is that it should be generated by two of its half turns other than using the vertical axis.

3.2.5 The Main Theorem on rotation groups in $\mathbb{R}^3$

Of course for any RotationTransform of finite order the set of powers of this transformation form a *cyclic* group. We can now list all finite groups of rotation transforms.

Theorem : Any finite group of rotation Transforms in 3 dimensions is either a cyclic group, a dihedral group of rotations of a polyhedral can, the group Gtet of of rotation transforms of the tetrahedron, the group GOct of rotations of the octahedron, or the group GIcos of the icosahedron.

Paul Yale gives a proof in his book . Note Gtet is not dihedral because it is a group of order 12 with only 3 half turns.

In particular, subgroups of these groups must be of one of these types. In particular the group of the icosahedron GIcos will have cyclic subgroups of order 2,3 and 5, but not 4. The elements of GIcos

```
In[ ]:= γ = RotationTransform [Pi ,
    {0.6881909602355869` , -0.5000000000000001` , 0.5257311121191335` }];
δ = RotationTransform [Pi , {0.42532540417601994` ,
     0.30901699437494734` , -0.8506508083520401` }];
```

of axis points half turns give the dihedral group D5

```
In[ ]:= D5 = finiteTransGroup [⟨| 1 → γ , 2 → δ |⟩ , {1.234 , 2.431 , 0.176}, 7]
```

» number of group elements calculated 10

```
Out[ ]= {{1}, {2}, {1, 1}, {1, 2}, {2, 1}, {1, 2, 1}, {2, 1, 2}, {1, 2, 1, 2}, {2, 1, 2, 1}, {1, 2, 1, 2, 1}}
```

```
In[ ]:= orderAssoc [D5, ⟨| 1 → γ , 2 → δ |⟩ , {1.233 , -2.134 , 3.1001}, 6]
```

```
Out[ ]= ⟨| {1} → 2 , {2} → 2 , {1, 1} → 1 , {1, 2} → 5 , {2, 1} → 5 , {1, 2, 1} → 2 ,
  {2, 1, 2} → 2 , {1, 2, 1, 2} → 5 , {2, 1, 2, 1} → 5 , {1, 2, 1, 2, 1} → 2 |⟩
```

On the other hand the 2-fold axis rotation

```
In[ ]:= κ = RotationTransform [Pi , {0.5257311121191336` , 0 , 0.85065080835204` }];
```

And three point axis rotation

```
In[ ]:= λ = RotationTransform [2 Pi / 3,
          {0.7946544722917661` , -0.5773502691896258` , 0.18759247408507973` }];
```

give

```
In[ ]:= H = finiteTransGroup [⟨| 1 → κ, 2 → λ |⟩ , {1.234 , 2.431 , 0.176}, 4]
```

» number of group elements calculated 12

```
Out[ ]= {{1}, {2}, {1, 1}, {1, 2}, {2, 1}, {2, 2}, {1, 2, 1},
         {1, 2, 2}, {2, 1, 2}, {2, 2, 1}, {2, 1, 2, 2}, {2, 2, 1, 2}}
```

```
In[ ]:= orderAssoc [H, ⟨| 1 → κ, 2 → λ |⟩ , {1.234 , 2.431 , 0.176}, 4]
```

```
Out[ ]= ⟨| {1} → 2, {2} → 3, {1, 1} → 1, {1, 2} → 3, {2, 1} → 3, {2, 2} → 3, {1, 2, 1} → 3,
           {1, 2, 2} → 3, {2, 1, 2} → 3, {2, 2, 1} → 3, {2, 1, 2, 2} → 2, {2, 2, 1, 2} → 2 |⟩
```

which matches Gtet.

3.2.6 One last soccer ball

In 2024 in honor of the Olympic Games Adidas produced a replica of its Olympic ball which was somewhat simplified from its match ball

In[ ]:=

Like the Nike balls, this ball has 12 pentagonal panels. But unlike the Nike balls this panel is not a regular pentagon, one side is half the length of the other 4 sides. So this pentagon does not have rotational symmetry. We do, however see two symmetries of this ball, the rotations κ and λ above where κ is the half turn about the top circular region and λ is the 3 fold rotation seen in the dark blue area above right. As we just showed, these generate a 12-order group isomorphic to the group Gtet. Starting with one given pentagon each of the others is given by exactly one Transformation Function in the group applied to the given pentagon.

The trick is to find one base panel. We start with a regular pentagon

In[ ]:=
```
PentPanel =
 {axisPoints3[[11]], axisPoints3[[14]], axisPoints3[[20]], axisPoints3[[16]], axisPoints3[[15]]}
```

Out[ ]=
```
{{0.491123 , -0.356822 , 0.794654}, {0.491123 , 0.356822 , 0.794654},
 {0.794654 , 0.57735 , 0.187592}, {0.982247 , 0, -0.187592}, {0.794654 , -0.57735 , 0.187592}}
```

In[ ]:=

In[ ]:=
```
bsideMid = spMid[axisPoints3[[11]], axisPoints3[[14]]]
q1 = spMid[axisPoints3[[11]], bsideMid]
q2 = spMid[bsideMid , axisPoints3[[14]]]
```

Out[ ]=
$$\{0.525731 , -5.94228 \times 10^{-17} , 0.850651\}$$

Out[ ]=
```
{0.517007 , -0.181422 , 0.836535}
```

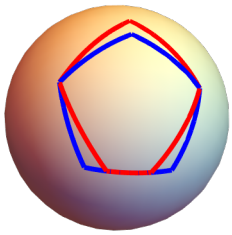
Out[ ]=
```
{0.517007 , 0.181422 , 0.836535}
```

In[ ]:=
```
q3 = λ@*λ@q1
```

Out[ ]=
$$\{0.931914 , -1.11022 \times 10^{-16} , -0.36268\}$$

In[ ]:=
```
Show[StandardSphere , Graphics3D[{{Blue , Thickness[.01], Line[spOutline[PentPanel]]},
    {Red, Thickness[.01], Line[spLn[q1, q2]], Line[spLn[q1, axisPoints3[[15]]],
     Line[spLn[q2, axisPoints3[[20]]], Line[spLn[axisPoints3[[20]], q3]],
     Line[spLn[axisPoints3[[15]], q3]]}], Boxed → False , ImageSize → Small]
```

Out[ ]=

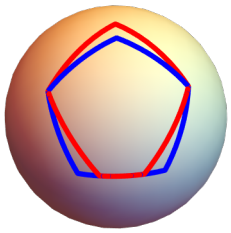


In[ ]:=
```
OP0 = {q1, axisPoints3[[15]], q3, axisPoints3[[20]], q2}
```

Out[ ]=
```
{{0.517007 , -0.181422 , 0.836535},
 {0.794654 , -0.57735 , 0.187592}, {0.931914 , -1.11022 × 10^-16 , -0.36268},
 {0.794654 , 0.57735 , 0.187592}, {0.517007 , 0.181422 , 0.836535}}
```

In[ ]:=
```
Show[StandardSphere , Graphics3D [{{Blue , Thickness [.015], Line[spOutline [PentPanel ]]},
    {Red , Thickness [.015], Line[spOutline [OP0]]}}], Boxed → False , ImageSize → Small]
```

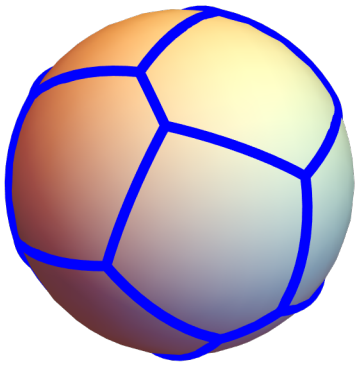
Out[ ]=



In[ ]:=
```
OPanels = Table[TasTF[key , ⟨| 1 → κ , 2 → λ |⟩ ]@OP0 , {key , H}];
```

In[ ]:=
```
Show[StandardSphere , Graphics3D [
    {Blue , Thickness [.015], Table[Line[spOutline [OPanels⟦i⟧]], {i , 12}]}], Boxed → False]
```

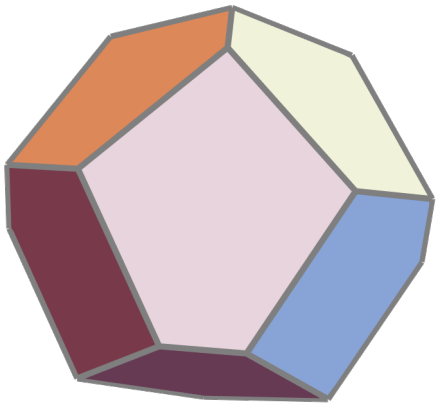
Out[ ]=



Here is the polyhedron version of the Olympic Replica Ball paneling.

In[ ]:=
```
Graphics3D[
 {EdgeForm[{Gray, Thickness[.01]}], Table[Polygon[OPanels[[i]]], {i, 12}]}, Boxed → False]
```

Out[ ]=



The relationship between the this polyhedron and the tetrahedron can be shown by

In[ ]:=

In[ ]:=
```
HA1 = <|{1} → {-0.18759247408507945` , 0.5773502691896261` , 0.7946544722917663` },
  {2} → {0.7946544722917659` , -0.5773502691896262` , 0.18759247408507915` },
  {2, 1} → {-0.7946544722917662` , -0.577350269189626` , -0.18759247408507942` },
  {1, 2, 1} → {0.18759247408508034` , 0.577350269189626` , -0.7946544722917661` }|>
```

Out[ ]=
```
<|{1} → {-0.187592 , 0.57735 , 0.794654}, {2} → {0.794654 , -0.57735 , 0.187592},
 {2, 1} → {-0.794654 , -0.57735 , -0.187592}, {1, 2, 1} → {0.187592 , 0.57735 , -0.794654}|>
```

In[ ]:=
```
Graphics3D[
 {{Red, Thickness[.015], Line[{HA1[{1}], HA1[{1, 2, 1}], HA1[{2, 1}], HA1[{1}], HA1[{2}]}],
   Line[{HA1[{2, 1}], HA1[{2}], HA1[{1, 2, 1}]}]}, {EdgeForm[{Gray, Thickness[.01]}],
   Opacity[.7], Table[Polygon[OPanels⟦i⟧], {i, 12}]}}, Boxed → False]
```

Out[ ]=