Tiling 3D Space

I want to use the unit cell from the FCC lattice as the basis for 3D tiling for a rogue-like game movement in 3D.

Lattice Vectors

The lattice vectors for the FCC lattice are typically written as:

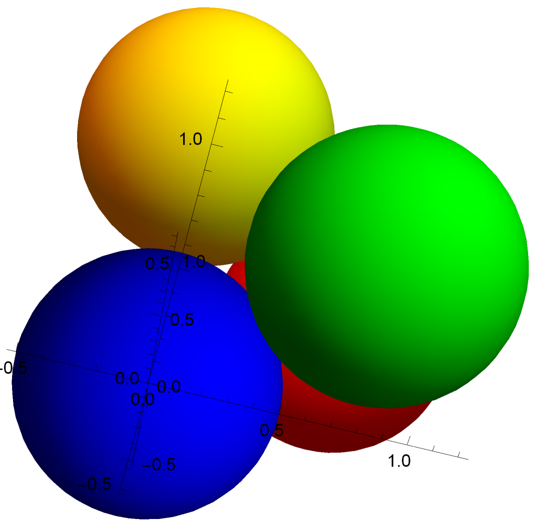
$$\mathtt{a1} = \left\{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right\};$$

$$\mathtt{a2} = \left\{\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}\right\};$$

$$\mathtt{a3} = \left\{0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\};$$

This leads to the classic FCC packing for spheres.

```
Graphics3D[{Blue, Sphere[{0, 0, 0}, 1/2],
  Red, Translate[Sphere[{0, 0, 0}, 1/2], a1],
  Green, Translate[Sphere[{0, 0, 0}, 1/2], a2],
  Yellow, Translate[Sphere[{0, 0, 0}, 1/2], a3]},
 AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False]
```



Rotated Lattic Vectors

I want to rotate these vectors so that one basis vector lies along the z axis and the other is in the xz plane. This will work well for the layout I want to use in the game.

The first step is to rotate the vectors so that one is along the z-axis. It turns out that rotating a2 is easy enough.

```
angle = ArcCos[Dot[a2, {0, 0, 1}]]
rotVec = Cross[a2, {0, 0, 1}]
rotMat = RotationMatrix[angle, rotVec] ;
rotMat // MatrixForm
a1p = RotationMatrix[0, {0, 0, 1}].rotMat.a1
a2p = RotationMatrix[0, {0, 0, 1}].rotMat.a2
a3p = RotationMatrix[0, {0, 0, 1}].rotMat.a3
Graphics3D[{Green, Arrow[{{0, 0, 0}, a1p}], Black,
  Arrow[{{0, 0, 0}, a2p}], Red, Arrow[{{0, 0, 0}, a3p}]}, AxesOrigin → {0, 0, 0},
 Axes → True, Boxed → False, AxesLabel → {x, y, z}, ViewPoint → {0, 0, ∞}]
```

$\frac{\pi}{4}$

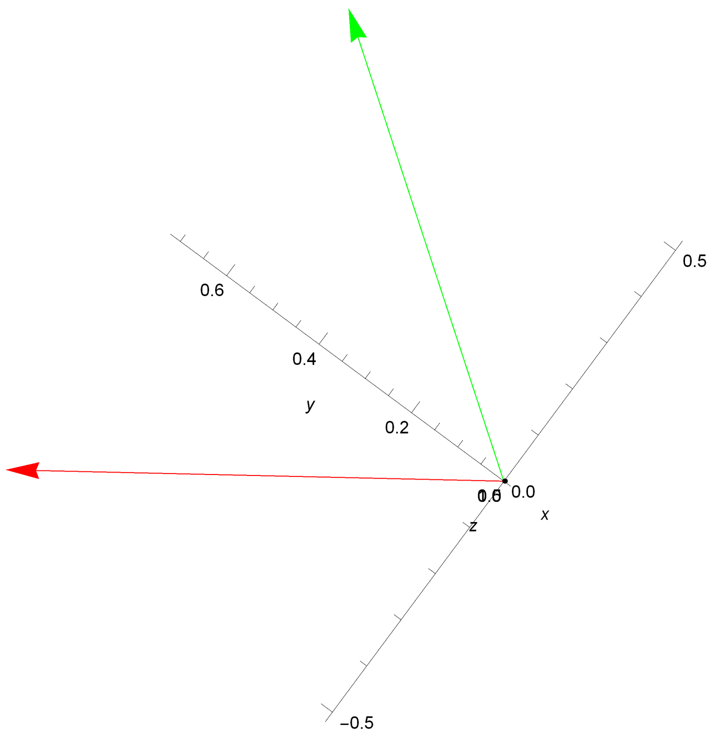
$\left\{0, -\frac{1}{\sqrt{2}}, 0\right\}$

$\begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$

$\left\{\frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{2}\right\}$

$\{0, 0, 1\}$

$\left\{-\frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{2}\right\}$



So, a $\frac{-\pi}{4}$ rotation about the y-axis did the job.

Now I want one of the other two vectors to lie in the XZ plane (i.e. its Y coordinate is zero). To do this, I will project the green vector (a1p) onto the xy plane, then use it to find the angle to rotate around the z-axis.

$$\text{a1proj} = \left\{\frac{1}{2}, \frac{1}{\sqrt{2}}, 0\right\} \Big/ \text{Norm}\left[\left\{\frac{1}{2}, \frac{1}{\sqrt{2}}, 0\right\}\right]$$

$$\left\{\frac{1}{\sqrt{3}}, \sqrt{\frac{2}{3}}, 0\right\}$$

```
angle2 = ArcCos[Abs[Dot[a1proj, {1, 0, 0}]]]
rotVec2 = {0, 0, -1}
rotMat2 = RotationMatrix[angle2, rotVec2] ;
rotMat2 // MatrixForm
Print["Basis Vector 1"]
a1p2 = FullSimplify[rotMat2.a1p]
Print["Basis Vector 2"]
a2p2 = FullSimplify[rotMat2.a2p]
Print["Basis Vector 3"]
a3p2 = FullSimplify[rotMat2.a3p]
Graphics3D[{Green, Arrow[{{0, 0, 0}, a1p2}],
   Black, Arrow[{{0, 0, 0}, a2p2}], Red, Arrow[{{0, 0, 0}, a3p2}]},
  AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False, AxesLabel → {x, y, z}]
```

$$\text{ArcCos}\left[\frac{1}{\sqrt{3}}\right]$$

```
{0, 0, -1}
```

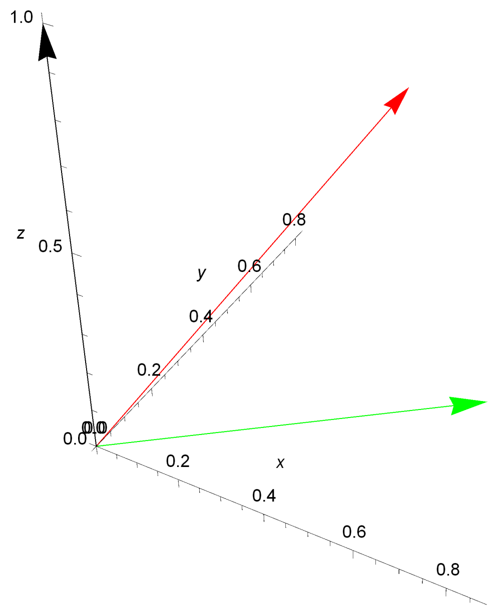
$$\begin{pmatrix} \frac{1}{\sqrt{3}} & \sqrt{\frac{2}{3}} & 0 \\ -\sqrt{\frac{2}{3}} & \frac{1}{\sqrt{3}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
Basis Vector 1
```

$$\left\{\frac{\sqrt{3}}{2}, 0, \frac{1}{2}\right\}$$

```
Basis Vector 2
{0, 0, 1}
Basis Vector 3
```

$$\left\{\frac{1}{2\sqrt{3}}, \sqrt{\frac{2}{3}}, \frac{1}{2}\right\}$$



So it turns out that I needed to rotate it by the ArcCos$\left[\sqrt{\frac{1}{3}}\right]$, which is an unusual angle, but it gives us what we want. The total rotation matrix is:

```
rotMat2.rotMat // MatrixForm
```

$$\begin{pmatrix} \frac{1}{\sqrt{6}} & \sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}$$

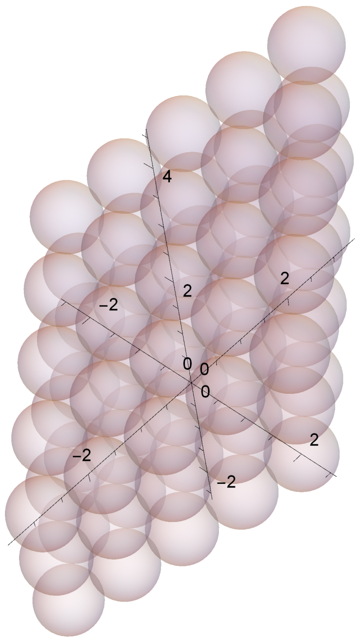
Just to have them, I'm converting the basis vectors into decimal form.

```
N[{a1p2, a2p2, a3p2}]
```

```
{{0.866025, 0., 0.5}, {0., 0., 1.}, {0.288675, 0.816497, 0.5}}
```

And we check the unit lattice.

```
Graphics3D[Flatten[
  {Opacity[0.3], Table[Translate[Sphere[{0, 0, 0}, 1/2], {u a1p + v a2p + w a3p}],
    {u, -2, 2}, {v, 0, 2}, {w, -2, 2}]}],
 AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False]
```

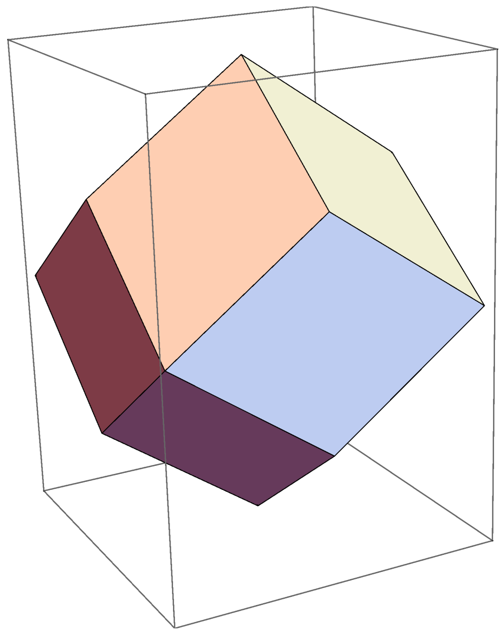


Tiling Polyhedra

I now want to tile space with the rhombic dodecahedron. The FCC Vornois tesellation is the Rhomic dodcahedron, so this should mesh perfectly with our new basis vectors.

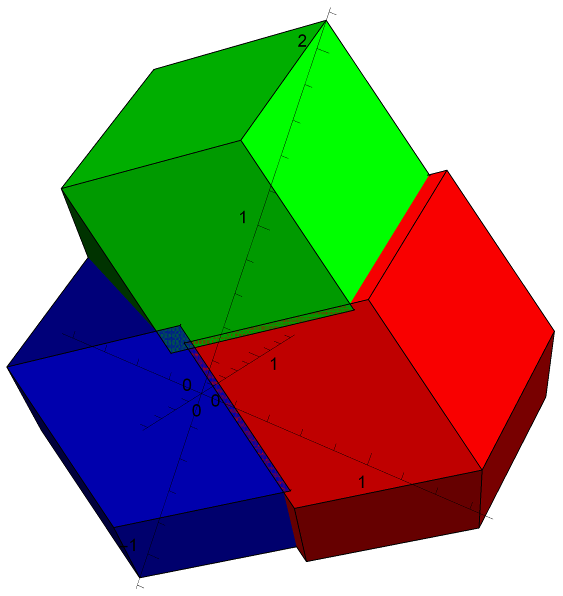
We first visualize this shape.

```
PolyhedronData["RhombicDodecahedron"]
```



We check the tesellation of this shape using our basis vectors.

```
Graphics3D[{Blue, PolyhedronData["RhombicDodecahedron", "Faces"],
  Red, Translate[PolyhedronData["RhombicDodecahedron", "Faces"], a1p2],
  Green, Translate[PolyhedronData["RhombicDodecahedron", "Faces"], a2p2],
  Yellow, Translate[PolyhedronData["RhombicDodecahedron", "Faces"], a3p2]},
 AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False]
```



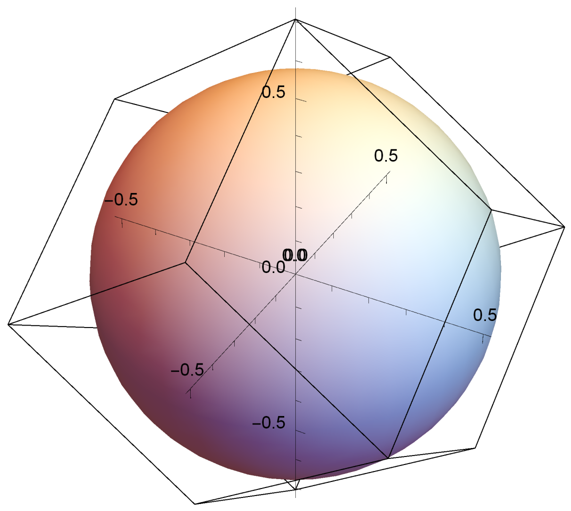
First step, it looks like the dodecahera are too large for a unit translation. Let's get the scale.

```
PolyhedronData["RhombicDodecahedron", "Insphere"]
```

Sphere$\left[\{0, 0, 0\}, \sqrt{\frac{2}{3}}\right]$

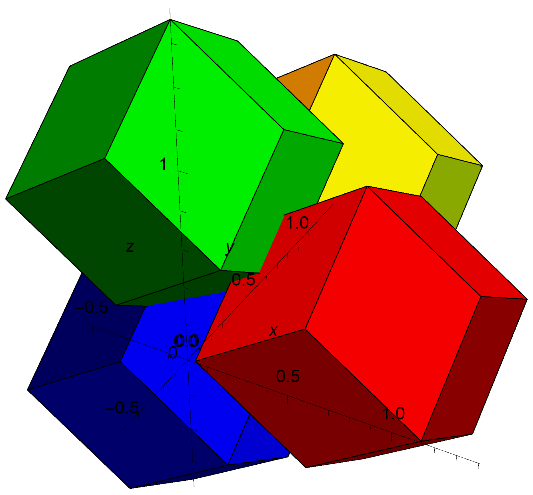
As suspected, the radius of the inscribe sphere is not $\frac{1}{2}$ (unit diameter). Fixing that:

```
Graphics3D[{Scale[PolyhedronData["RhombicDodecahedron", "Insphere"], 1/2 Sqrt[3/2]],
  Scale[PolyhedronData["RhombicDodecahedron", "Edges"], 1/2 Sqrt[3/2]]},
 AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False]
```



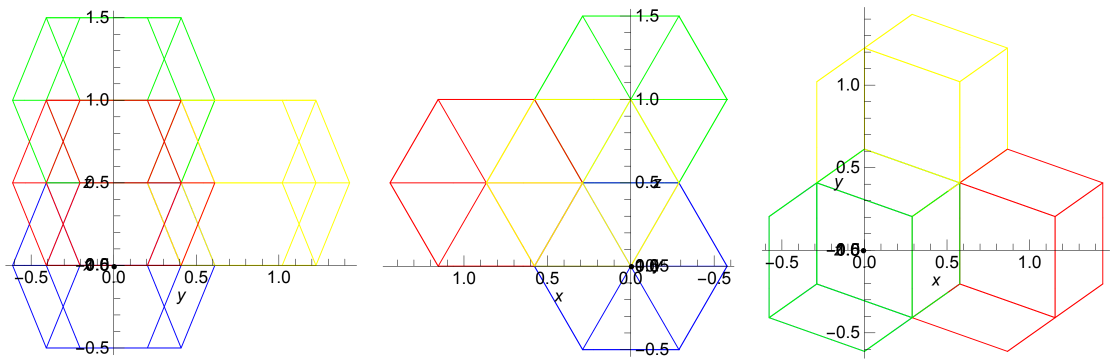
That looks better. Let’s try the tesellation again with the re-sized polyhedra.

```
Graphics3D[{Blue, Scale[PolyhedronData["RhombicDodecahedron", "Faces"], 1/2 Sqrt[3/2]],
  Red,
  Translate[Scale[PolyhedronData["RhombicDodecahedron", "Faces"], 1/2 Sqrt[3/2]], a1p2],
  Green, Translate[Scale[PolyhedronData["RhombicDodecahedron", "Faces"], 1/2 Sqrt[3/2]],
   a2p2],
  Yellow, Translate[Scale[PolyhedronData["RhombicDodecahedron", "Faces"], 1/2 Sqrt[3/2]],
   a3p2]}, AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False, AxesLabel → {x, y, z}]
```



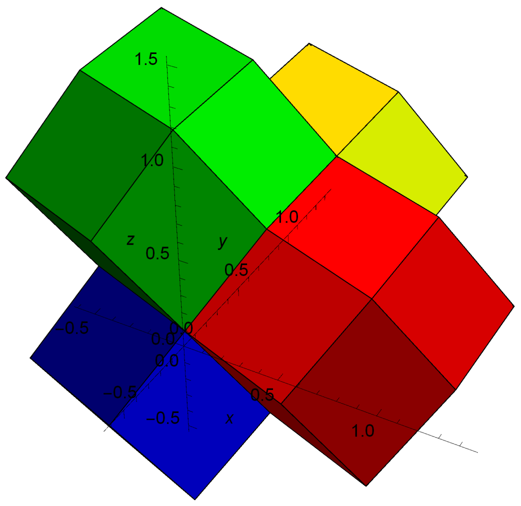
That fixed the scale problem. Now the rotation of the dodecahedron is off. I want the tiling to match faces. This requres two rotations. First, we rotate by $\frac{\pi}{2}$ around the y-axis. Then we have to do a second rotation that turns out to be a rotation of ArcCos$\left[\sqrt{\frac{2}{3}}\right]$ radians around the z-axis. We then look at the projections in all three directions to make sure that things are tiled correctly.

```
rotationAngle = π/2;
rotationVector = {0, 1, 0};
MakePoly[transVec_] := Translate[GeometricTransformation[
    GeometricTransformation[Scale[PolyhedronData["RhombicDodecahedron", "Edges"],
      1/2 Sqrt[3/2]], RotationMatrix[rotationAngle, rotationVector]],

    RotationMatrix[ArcCos[Sqrt[2/3]], {0, 0, 1}]], transVec];

GraphicsGrid[{{Graphics3D[{Blue, MakePoly[{0, 0, 0}],
      Red, MakePoly[a1p2],
      Green, MakePoly[a2p2],
      Yellow, MakePoly[a3p2]}, AxesOrigin → {0, 0, 0}, Axes → True,
    Boxed → False, AxesLabel → {x, y, z}, ViewPoint → {∞, 0, 0}],
   Graphics3D[{Blue, MakePoly[{0, 0, 0}],
      Red, MakePoly[a1p2],
      Green, MakePoly[a2p2],
      Yellow, MakePoly[a3p2]}, AxesOrigin → {0, 0, 0}, Axes → True,
    Boxed → False, AxesLabel → {x, y, z}, ViewPoint → {0, ∞, 0}],
   Graphics3D[{Blue, MakePoly[{0, 0, 0}],
      Red, MakePoly[a1p2],
      Green, MakePoly[a2p2],
      Yellow, MakePoly[a3p2]}, AxesOrigin → {0, 0, 0}, Axes → True, Boxed → False,
    AxesLabel → {x, y, z}, ViewPoint → {0, 0, ∞}]}}, ImageSize → Large]
```



And we check the tiling in 3D to make sure it looks good.

```
MakePolyFace[transVec_] :=
  Translate[GeometricTransformation[GeometricTransformation[
      Scale[PolyhedronData["RhombicDodecahedron", "Faces"], 1/2 Sqrt[3/2]],
      RotationMatrix[rotationAngle, rotationVector]],
    RotationMatrix[ArcCos[Sqrt[2/3]], {0, 0, 1}]], transVec];
Graphics3D[{Blue, MakePolyFace[{0, 0, 0}],
  Red, MakePolyFace[a1p2],
  Green, MakePolyFace[a2p2],
  Yellow, MakePolyFace[a3p2]}, AxesOrigin → {0, 0, 0},
 Axes → True, Boxed → False, AxesLabel → {x, y, z}]
```

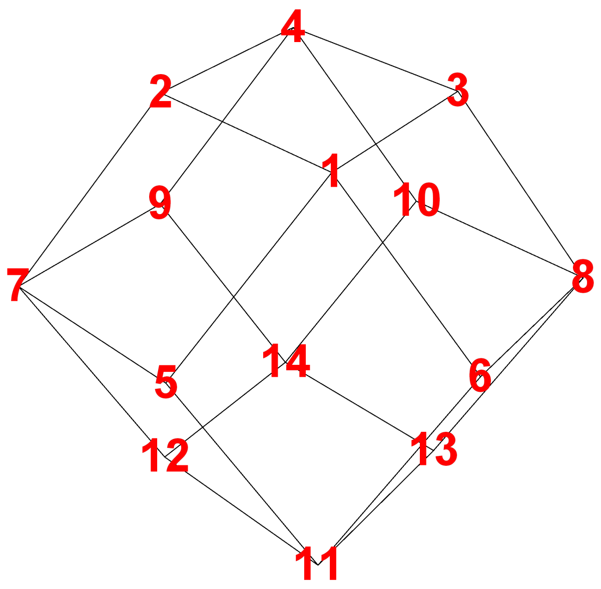


These two rotations now give us the coordinates for all of the vertices for the rhombic dodecahedron in our basis.

```
points = FullSimplify[Map[
    1/2 Sqrt[3/2] RotationMatrix[ArcCos[Sqrt[2/3]], {0, 0, 1}].RotationMatrix[π/2, {0, 1, 0}].# &,
    PolyhedronData["RhombicDodecahedron", "VertexCoordinates"]]];
Transpose@{PolyhedronData["RhombicDodecahedron", "VertexIndices"], points} //
 MatrixForm
```

$$\begin{pmatrix}
1 & \left\{\frac{1}{2\sqrt{3}}, -\frac{1}{\sqrt{6}}, \frac{1}{2}\right\} \\
2 & \left\{-\frac{1}{2\sqrt{3}}, -\frac{1}{2\sqrt{6}}, \frac{1}{2}\right\} \\
3 & \left\{\frac{1}{2\sqrt{3}}, \frac{1}{2\sqrt{6}}, \frac{1}{2}\right\} \\
4 & \left\{-\frac{1}{2\sqrt{3}}, \frac{1}{\sqrt{6}}, \frac{1}{2}\right\} \\
5 & \left\{0, -\frac{\sqrt{\frac{3}{2}}}{2}, 0\right\} \\
6 & \left\{\frac{1}{\sqrt{3}}, -\frac{1}{2\sqrt{6}}, 0\right\} \\
7 & \left\{-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{6}}, 0\right\} \\
8 & \left\{\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{6}}, 0\right\} \\
9 & \left\{-\frac{1}{\sqrt{3}}, \frac{1}{2\sqrt{6}}, 0\right\} \\
10 & \left\{0, \frac{\sqrt{\frac{3}{2}}}{2}, 0\right\} \\
11 & \left\{\frac{1}{2\sqrt{3}}, -\frac{1}{\sqrt{6}}, -\frac{1}{2}\right\} \\
12 & \left\{-\frac{1}{2\sqrt{3}}, -\frac{1}{2\sqrt{6}}, -\frac{1}{2}\right\} \\
13 & \left\{\frac{1}{2\sqrt{3}}, \frac{1}{2\sqrt{6}}, -\frac{1}{2}\right\} \\
14 & \left\{-\frac{1}{2\sqrt{3}}, \frac{1}{\sqrt{6}}, -\frac{1}{2}\right\}
\end{pmatrix}$$

```
Show[Graphics3D[GeometricTransformation[GeometricTransformation[
    Scale[PolyhedronData["RhombicDodecahedron", "Edges"], 1/2 Sqrt[3/2]],
    RotationMatrix[rotationAngle, rotationVector]],
   RotationMatrix[ArcCos[Sqrt[2/3]], {0, 0, 1}]], Boxed → False],
 Graphics3D[{MapThread[Text[Style[#1, Bold, Large, Red], #2] &,
    {PolyhedronData["RhombicDodecahedron", "VertexIndices"], points}]}]]
```



Finally, we get the numerical values of the points for translation to the video game.

```
Transpose@{PolyhedronData["RhombicDodecahedron", "VertexIndices"], N[points]} //
 MatrixForm
```

```
 1    {0.288675, -0.408248, 0.5}
 2   {-0.288675, -0.204124, 0.5}
 3     {0.288675, 0.204124, 0.5}
 4    {-0.288675, 0.408248, 0.5}
 5         {0., -0.612372, 0.}
 6     {0.57735, -0.204124, 0.}
 7    {-0.57735, -0.408248, 0.}
 8      {0.57735, 0.408248, 0.}
 9     {-0.57735, 0.204124, 0.}
10          {0., 0.612372, 0.}
11   {0.288675, -0.408248, -0.5}
12  {-0.288675, -0.204124, -0.5}
13    {0.288675, 0.204124, -0.5}
14   {-0.288675, 0.408248, -0.5}
```