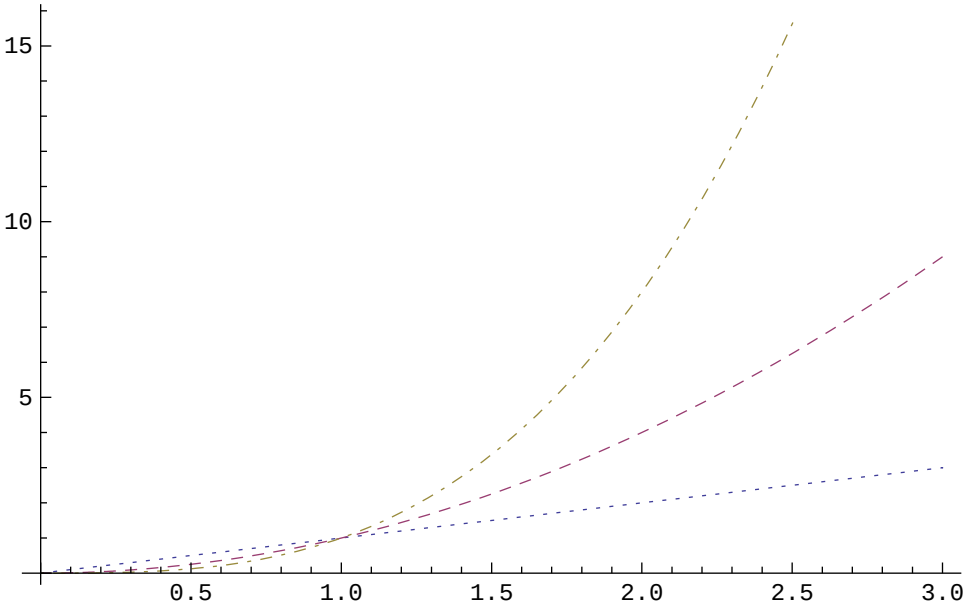
Mathematica als Visualisierungswerkzeug

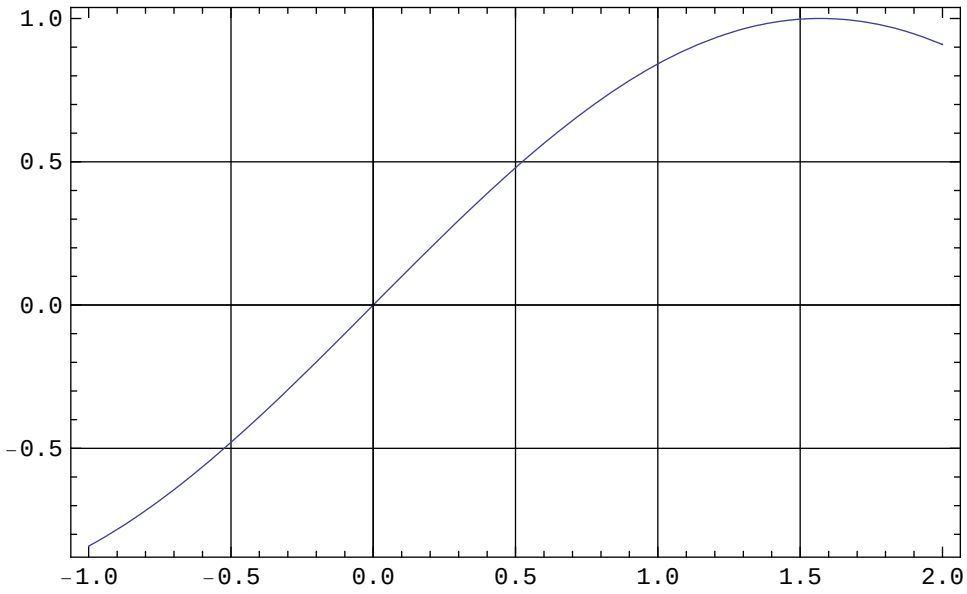
■ Beispiele

```
ClearAll["Global`*"]
```

```
g = Plot[{x, x^2, x^3}, {x, 0, 3}, PlotStyle → {Dotted, Dashed, DotDashed}]
```

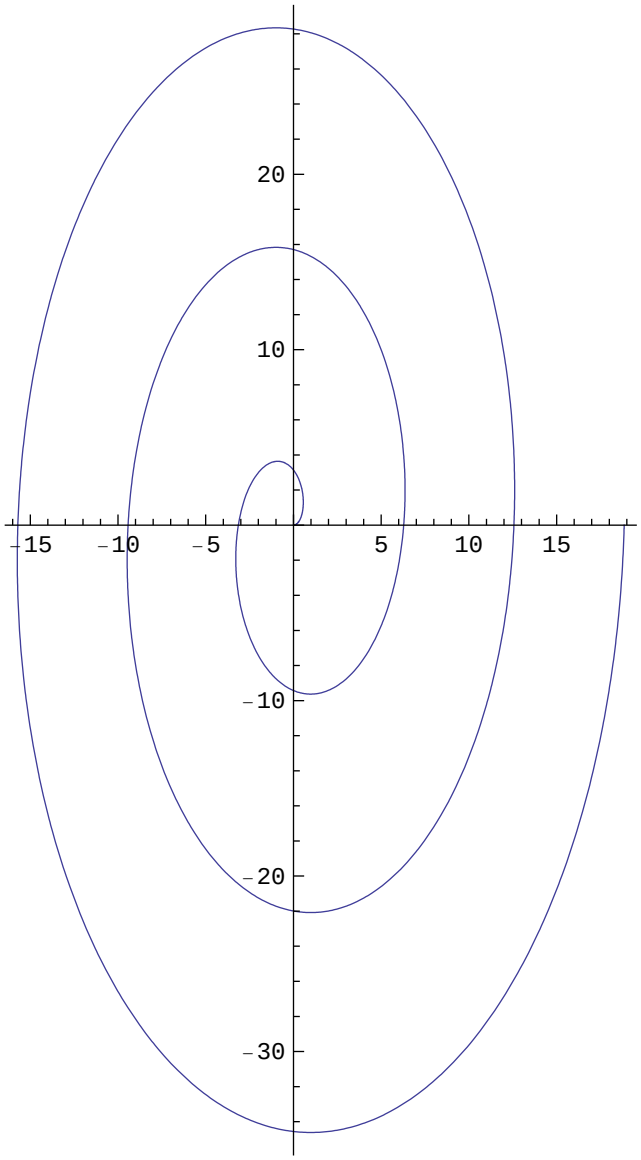



```
Plot[Sin[x], {x, -1, 2}, GridLines → Automatic, Frame → True]
```

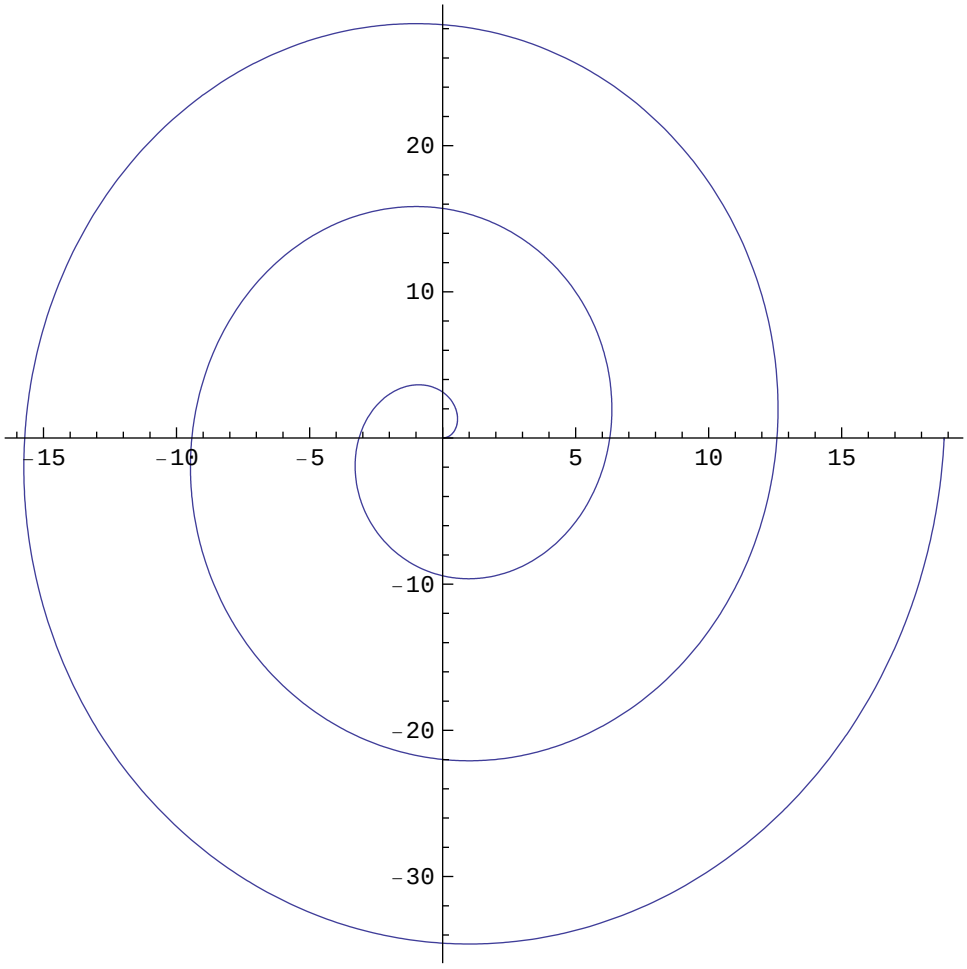


AspectRatio bezieht sich nicht auf die Koordinaten−Ticks, sondern auf die Größenverhältnisse der Box, in welcher das Bild dargestellt wird.

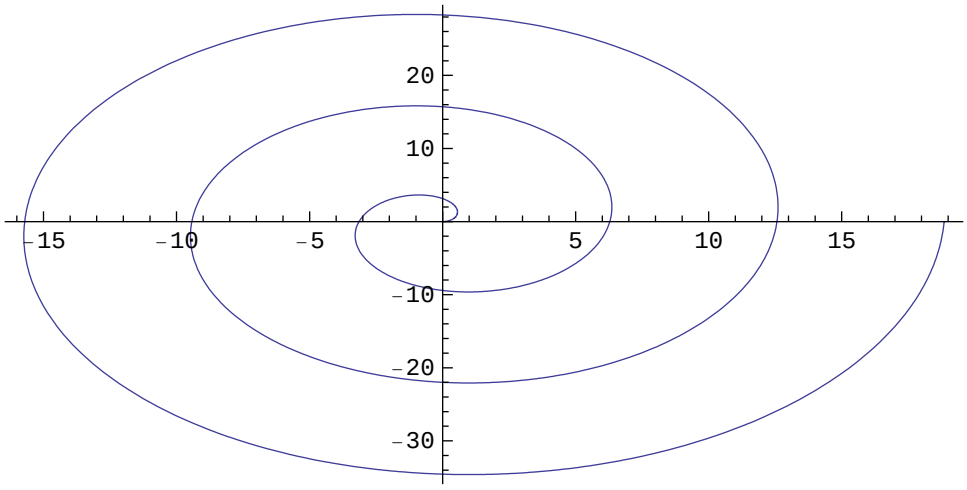
```
ParametricPlot[{t Cos[t], 2 t Sin[t]}, {t, 0, 6 π}]
```



```
ParametricPlot[{t Cos[t], 2 t Sin[t]}, {t, 0, 6 π}, AspectRatio → 1]
```



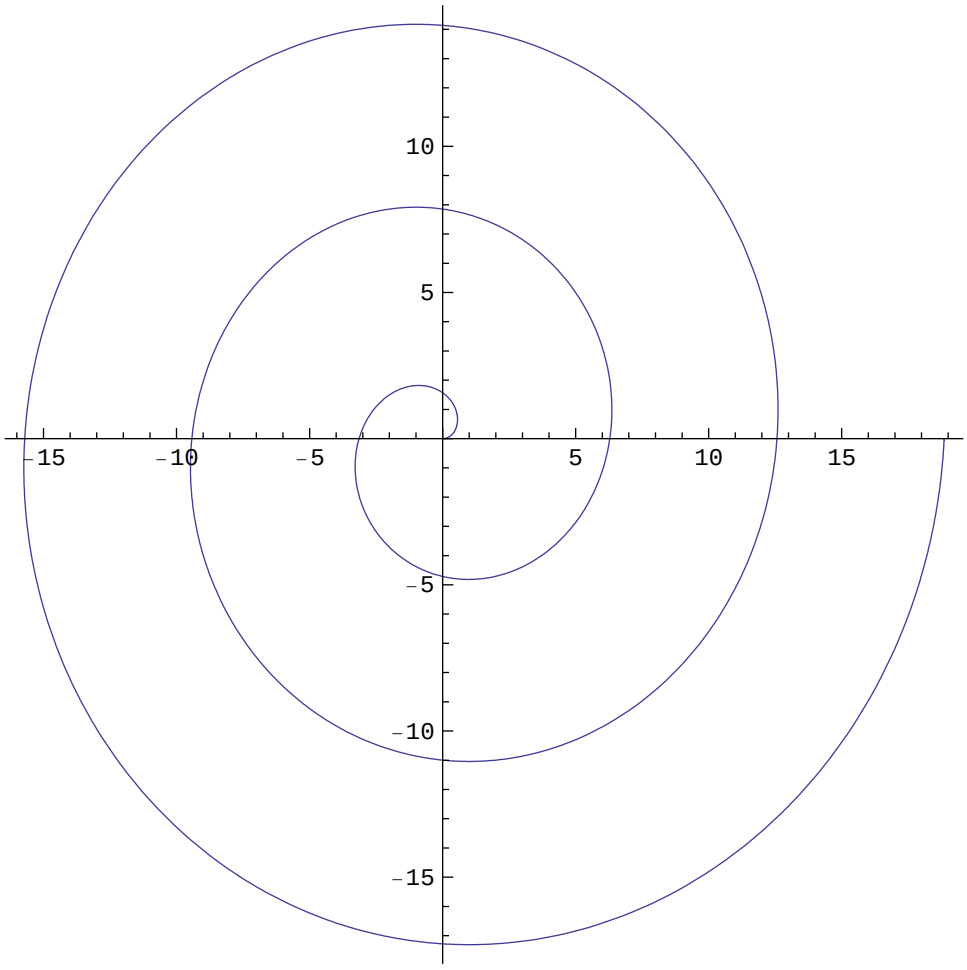

```
ParametricPlot[{t Cos[t], 2 t Sin[t]}, {t, 0, 6 π}, AspectRatio → 1 / 2]
```



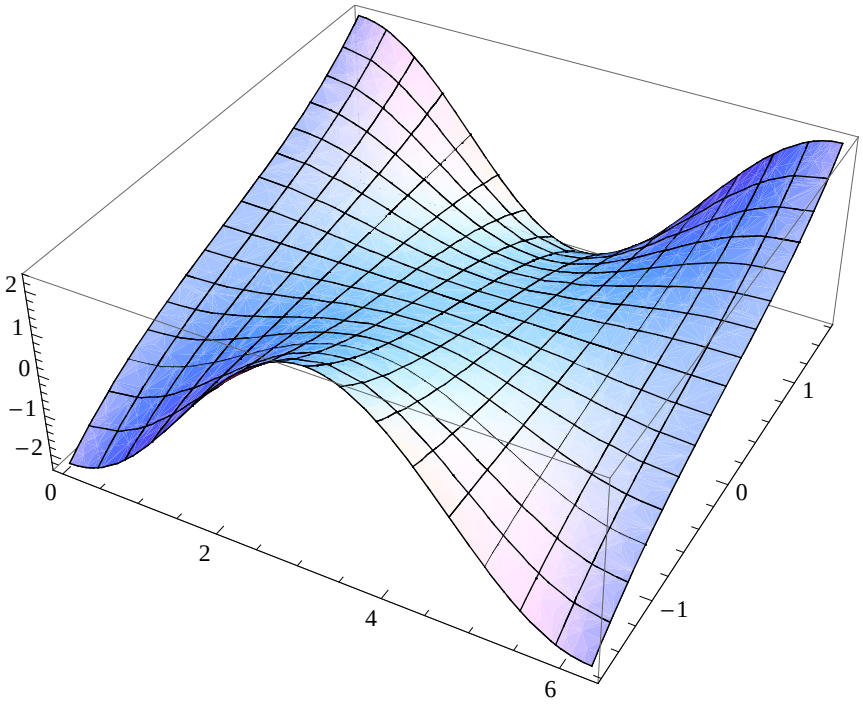
```
ParametricPlot[{t Cos[t], t Sin[t]}, {t, 0, 6 π}];
AbsoluteOptions[%, AspectRatio]
```

```
{AspectRatio → 0.91011}
```

```
ParametricPlot[{t Cos[t], t Sin[t]}, {t, 0, 6 π}, AspectRatio → 1]
```

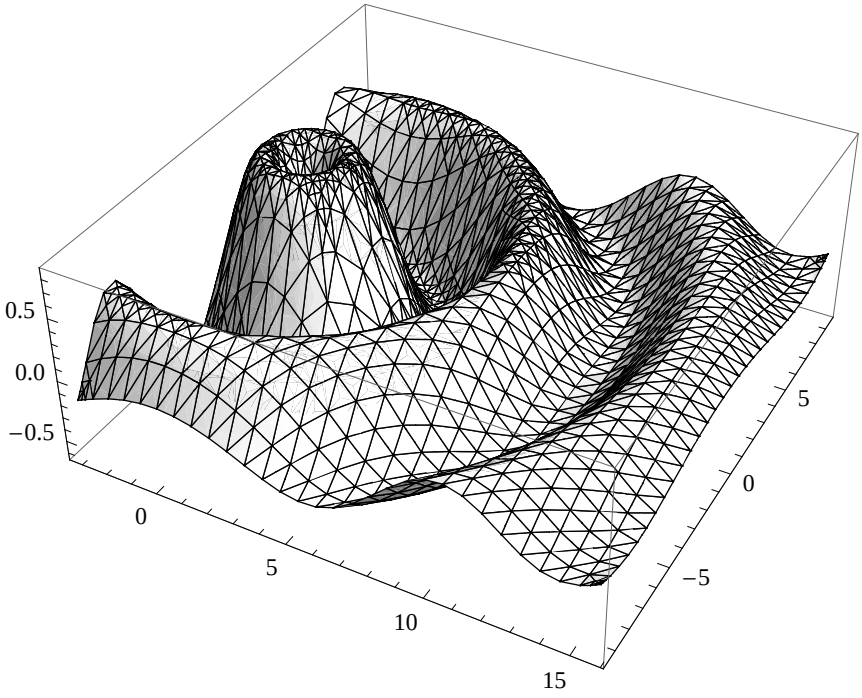


```
Plot3D[Im[Sin[x + I y]], {x, 0, 2 π}, {y, -π/2, π/2}]
```



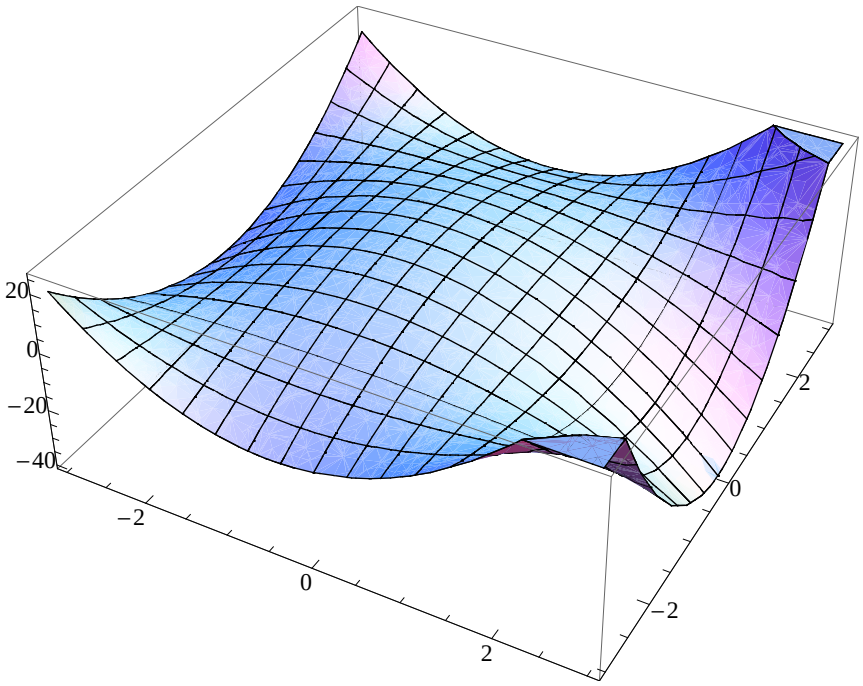
Mesh→ All zeigt jetzt das Netz an wie es der Triangulierung entspricht. Ansonsten wird ein separates Netz auf die Fläche gelegt, welches nichts mit der Traigulierung zu tun hat. Insbesondere hat der Wert **PlotPoints** dann keinen Einfluss auf die Maschengröße des Netzes.

```
p = Plot3D[Sin[√(x^2 + y^2)] Exp[-.1 √(x^2 + y^2)], {x, -π, 5 π}, {y, -3 π, 3 π},
  PlotPoints → 30, Mesh → All, Lighting → "Neutral"]
```

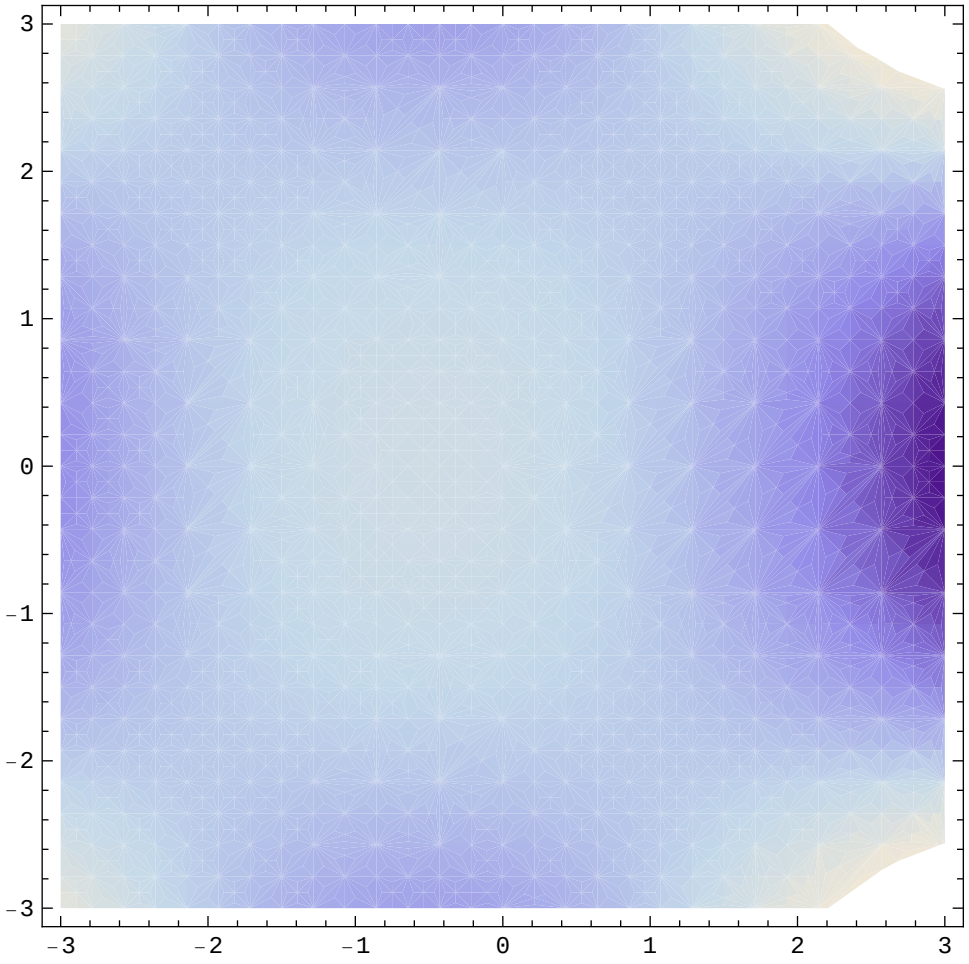


Mathematica unterstützt auch andere Arten der zweidimensionalen Darstellung höherdimensionaler Strukturen, indem einzelne Dimensionen durch Schattierungen oder Farben repräsentiert werden. **DensityPlot** und **ContourPlot** sind zwei solche Funktionen. **ContourPlot** produziert eine Darstellung durch Höhenlinien, wobei der Funktionswert einer Höhenlinie als Tooltip angezeigt wird.

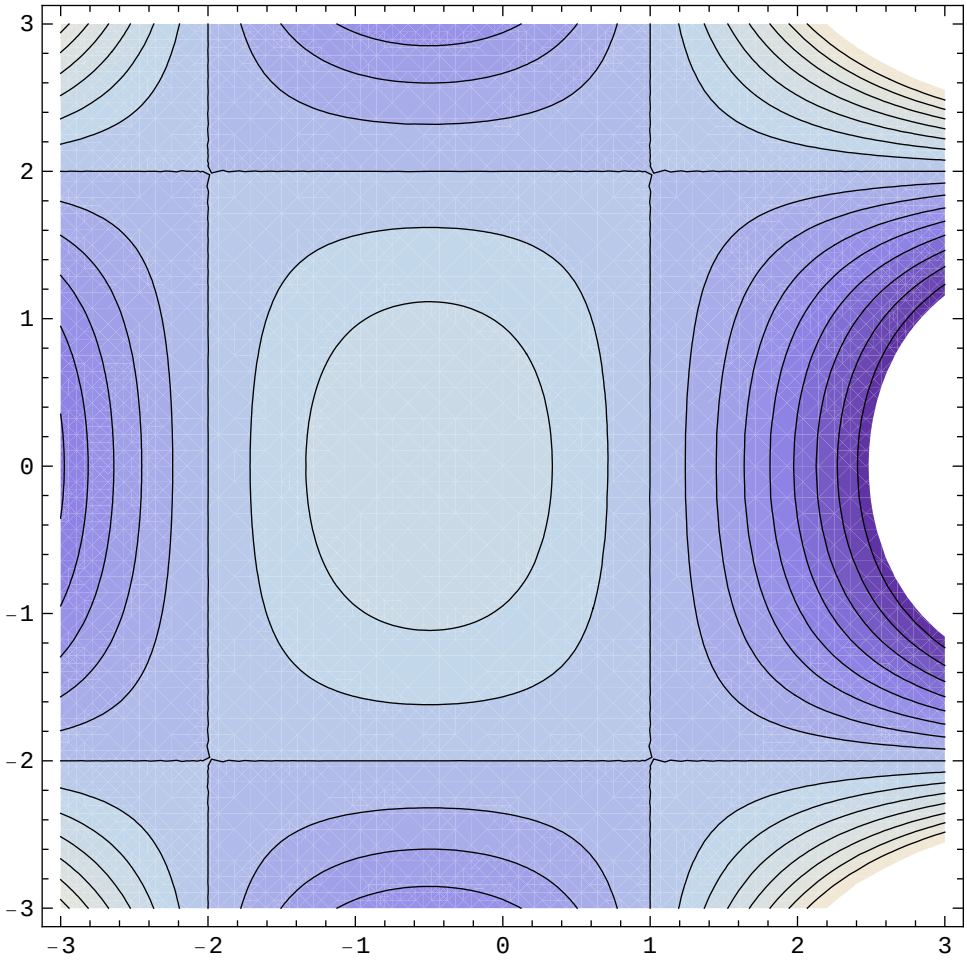
```
f = (x - 1) (x + 2) (y - 2) (y + 2);
Plot3D[f, {x, -3, 3}, {y, -3, 3}]
```



```
DensityPlot[f, {x, -3, 3}, {y, -3, 3}]
```

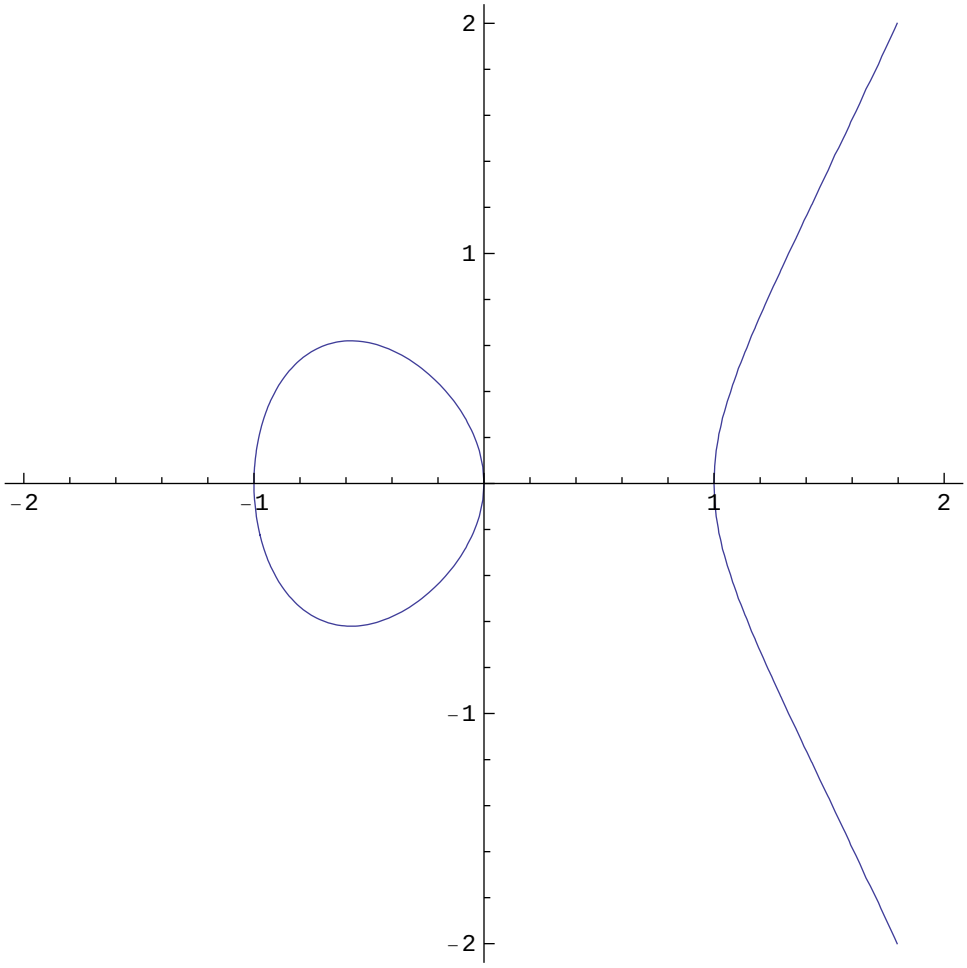


```
ContourPlot[f, {x, -3, 3}, {y, -3, 3}, Contours → 20]
```

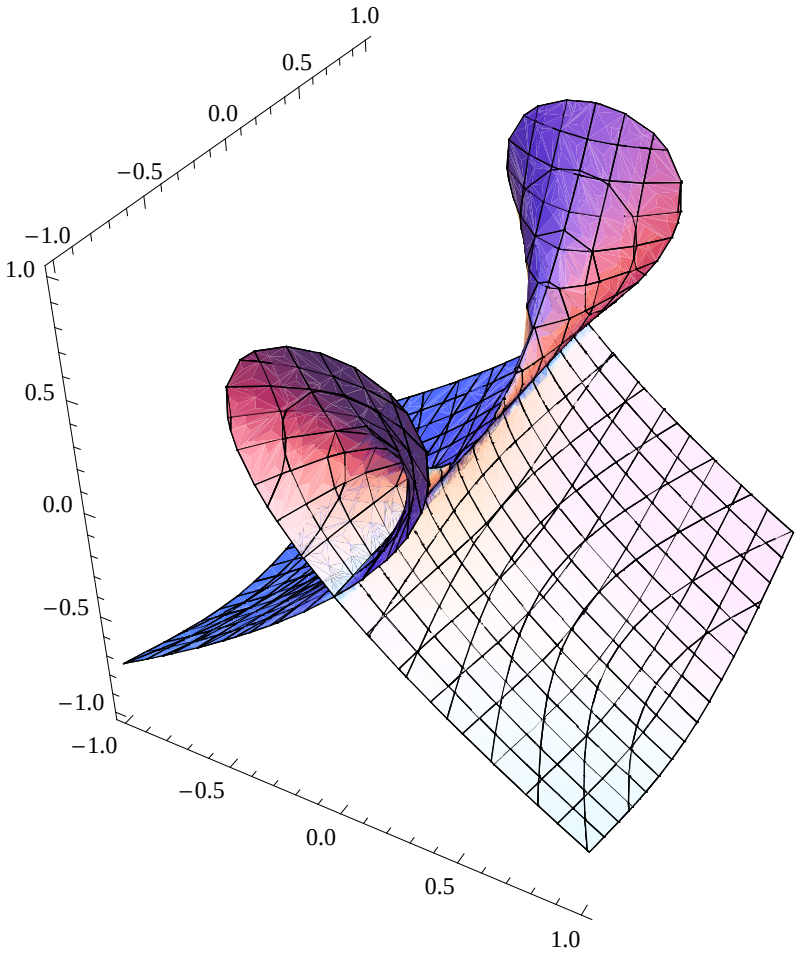


ContourPlot lässt sich auch verwenden, um einzelne Höhenlinien zu zeichnen, was äquivalent zur Herstellung einer impliziten Grafik ist.

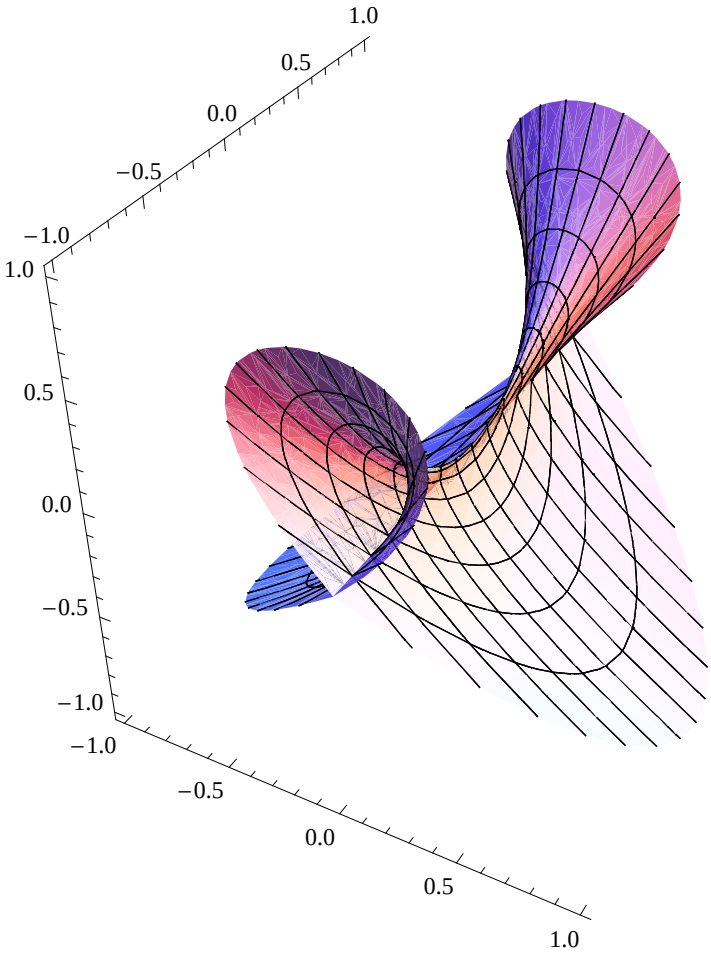
```
ContourPlot[y^2 == x^3 - x, {x, -2, 2}, {y, -2, 2}, Frame → False, Axes → True]
```



```
ContourPlot3D[x^2 - y^2 z^2 + z^3 == 0, {x, -1, 1}, {y, -1, 1}, {z, -1, 1}, Boxed → False]
```



```
ParametricPlot3D[{t (u^2 - t^2), u, u^2 - t^2}, {t, -1, 1}, {u, -1, 1}, Boxed → False]
```

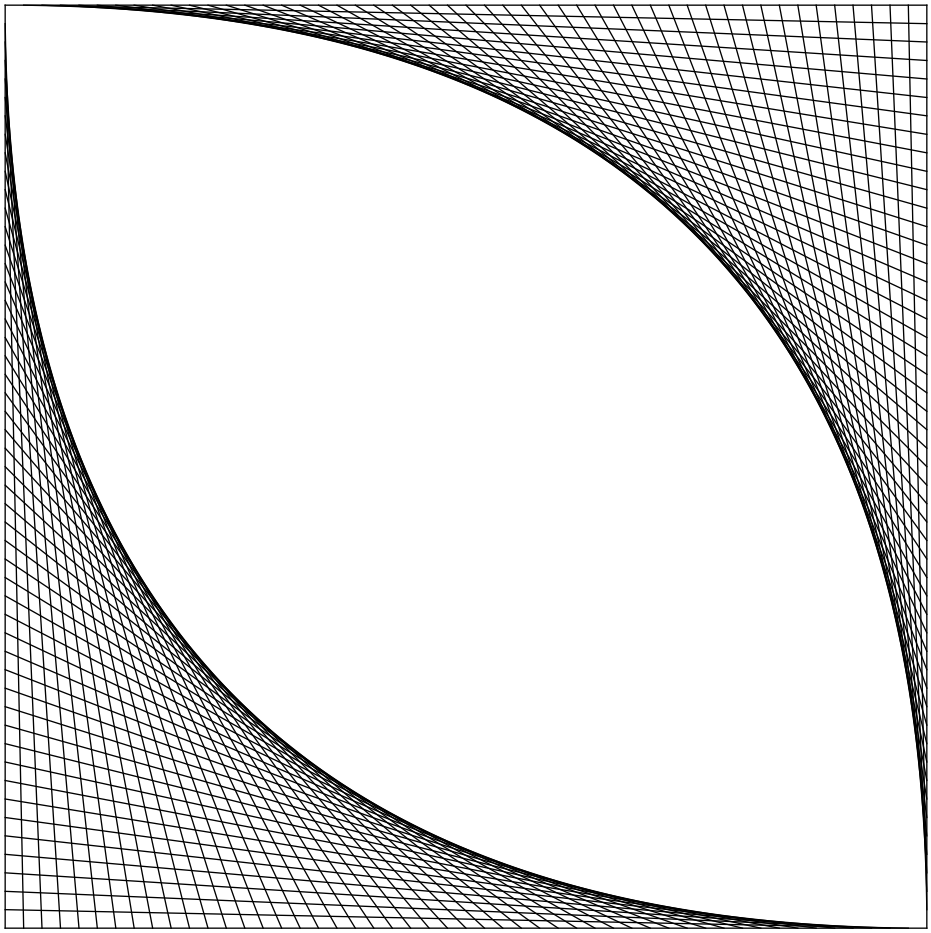


■ Zum Aufbau von Grafiken

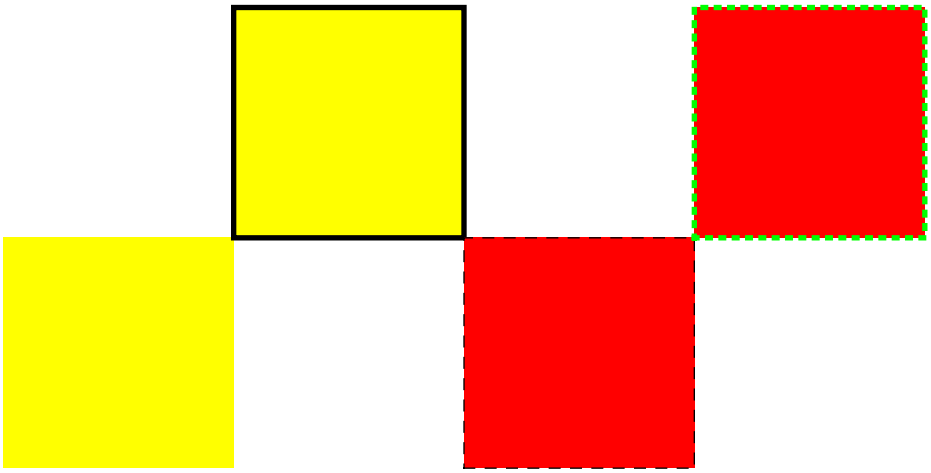
▪ 2D−Grafiken

Eine 2D−Grafik aus Linienelementen zusammenbauen.

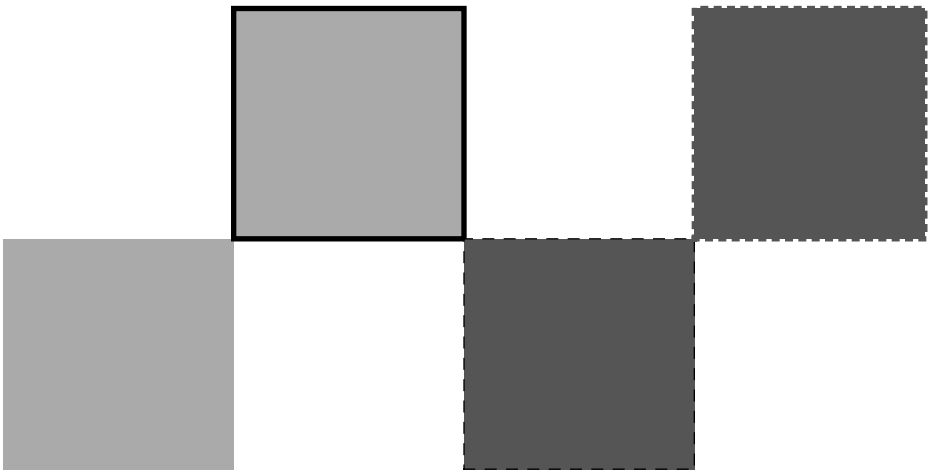
```
l1 = Table[Line[{{1, x}, {1 - x, 1}}], {x, 0, 1, 0.02}];
l2 = Table[Line[{{0, x}, {1 - x, 0}}], {x, 0, 1, 0.02}];
Graphics[{l1, l2}, AspectRatio → 1]
```



```
g = Graphics[{Yellow, Rectangle[{0, 0}],
   {EdgeForm[Thick], Rectangle[{1, 1}]}, Red,
   {EdgeForm[Dashed], Rectangle[{2, 0}]},
   {EdgeForm[{Thick, Dotted, Green}], Rectangle[{3, 1}]}
  }]
```



```
g /. RGBColor[a__] :> GrayLevel[Mean[{a}]]
```



■ Eine 3D−Grafik: Ein Viereck im Raum

Eine 3D−Grafik: Wie ein Viereck mit zufällig gewählten Eckpunkten im Raum dargestellt wird. Inklusive Punktbeschriftungen und Sphären um die Punkte herum.

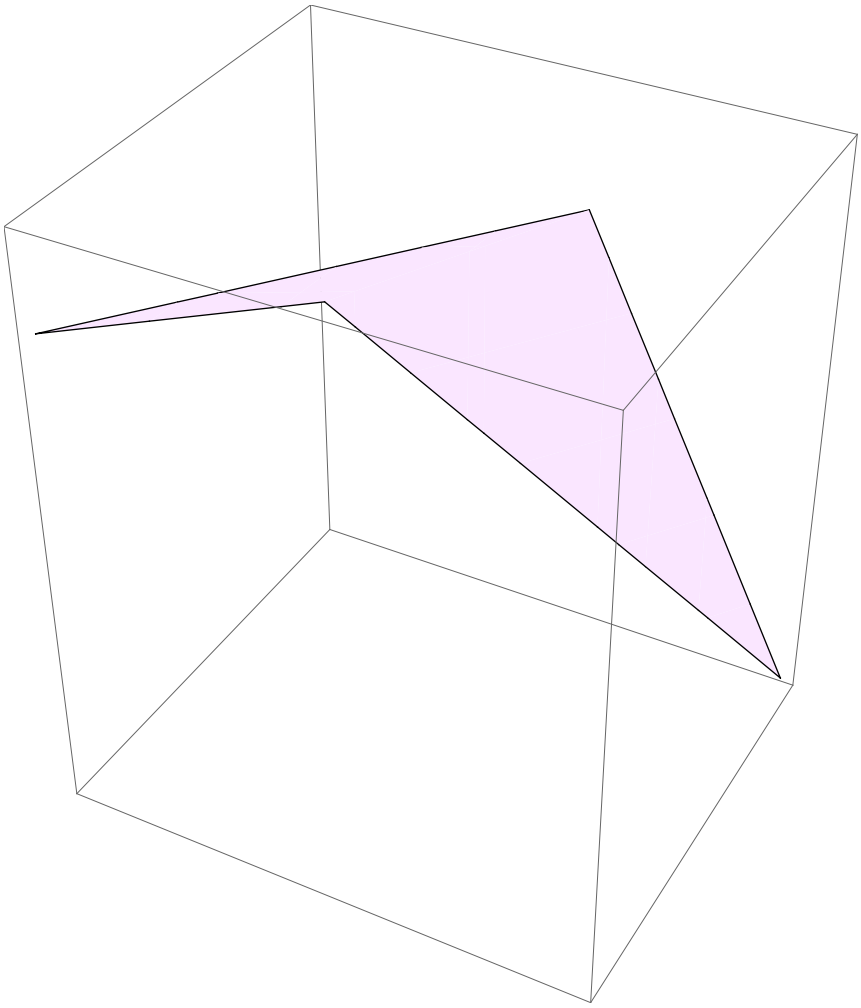
```
RP[n_] := Table[Random[], {n}]

p = Table[RP[3], {4}]
```

Und hier die Punkte, mit denen die aktuellen Grafiken erzeugt wurden.

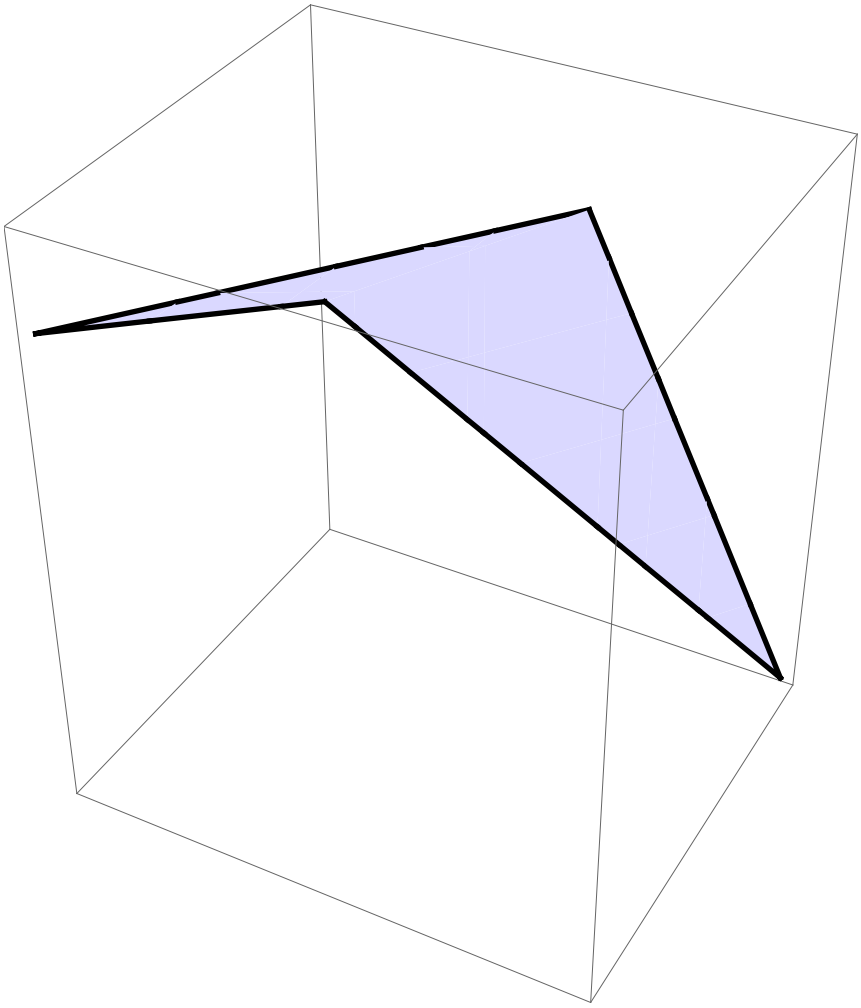
```
p = {{0.2780630066006887', 0.13354357331178807', 0.6857581819746135'},
   {0.5669237236908962', 0.5348852878846041', 0.09519123065898773'},
   {0.41961156086013174', 0.3325578741561961', 0.7100178257644869'},
   {0.029218381149627496', 0.06729916702760365', 0.616399264335642'}}
```

```
{{0.278063, 0.133544, 0.685758}, {0.566924, 0.534885, 0.0951912},
 {0.419612, 0.332558, 0.710018}, {0.0292184, 0.0672992, 0.616399}}
```

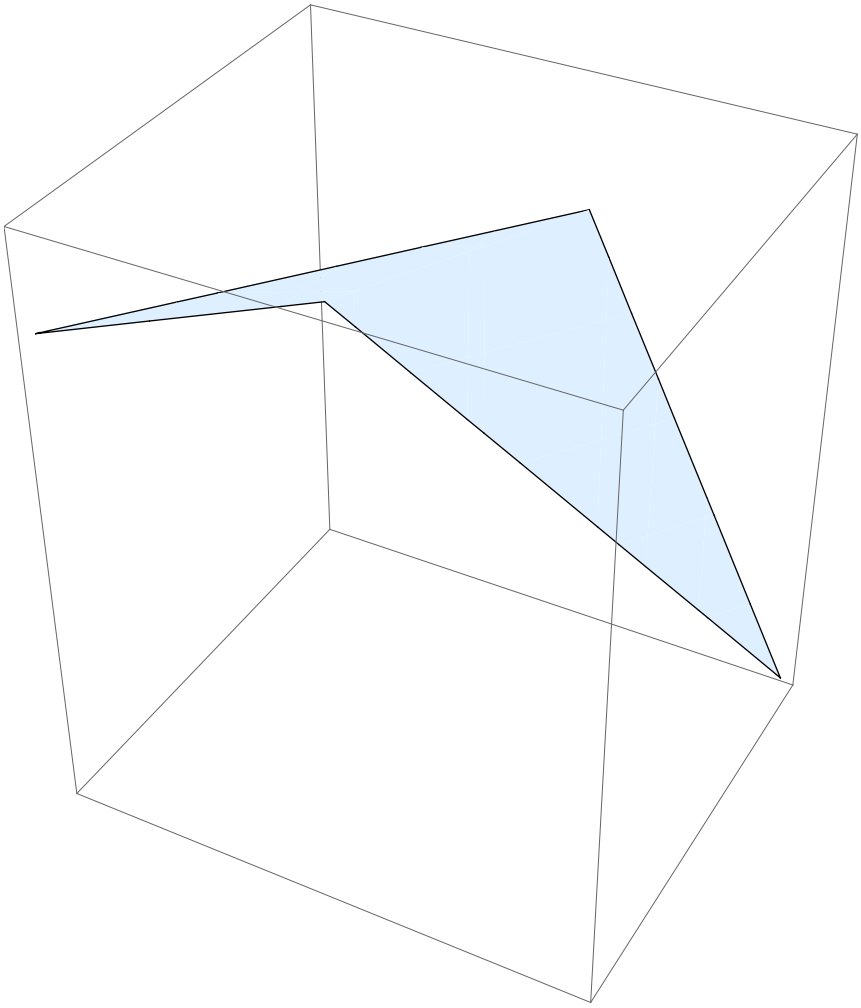
```
Graphics3D[Polygon[p]]
```



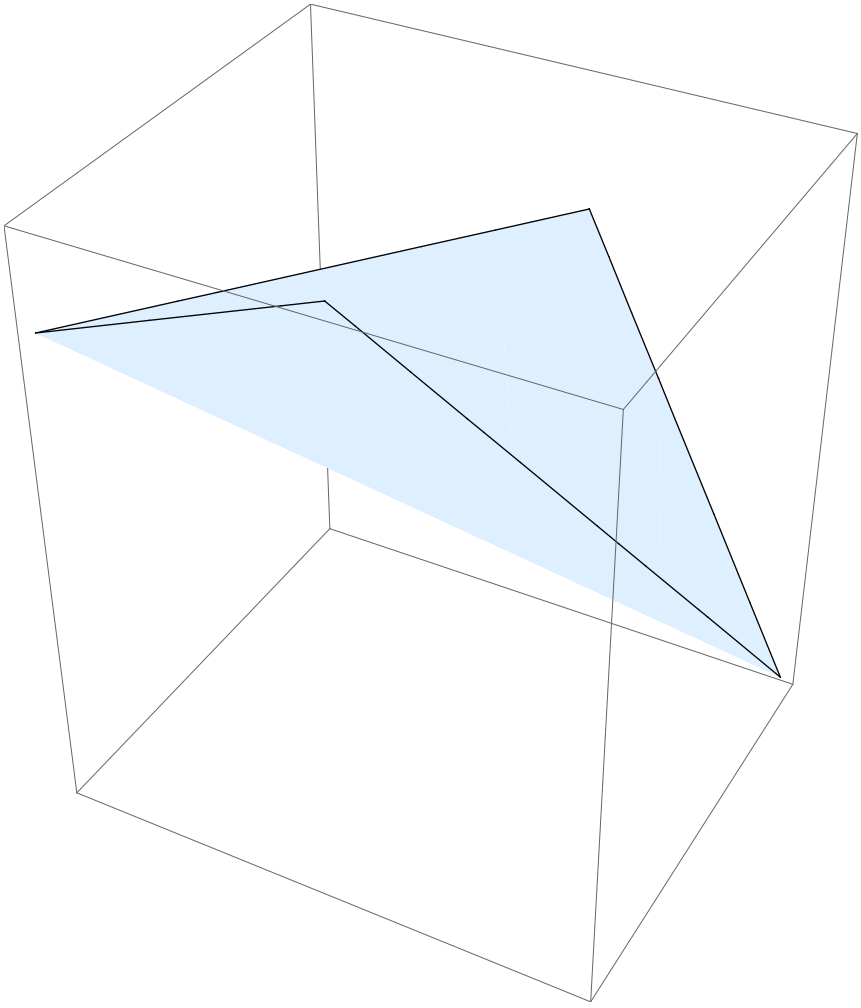
```
Graphics3D[{EdgeForm[Thick], FaceForm[LightBlue], Polygon[p]}]
```



```
Graphics3D[{FaceForm[LightBlue], Polygon[p]}, Lighting → {White}, PlotRange → All]
```



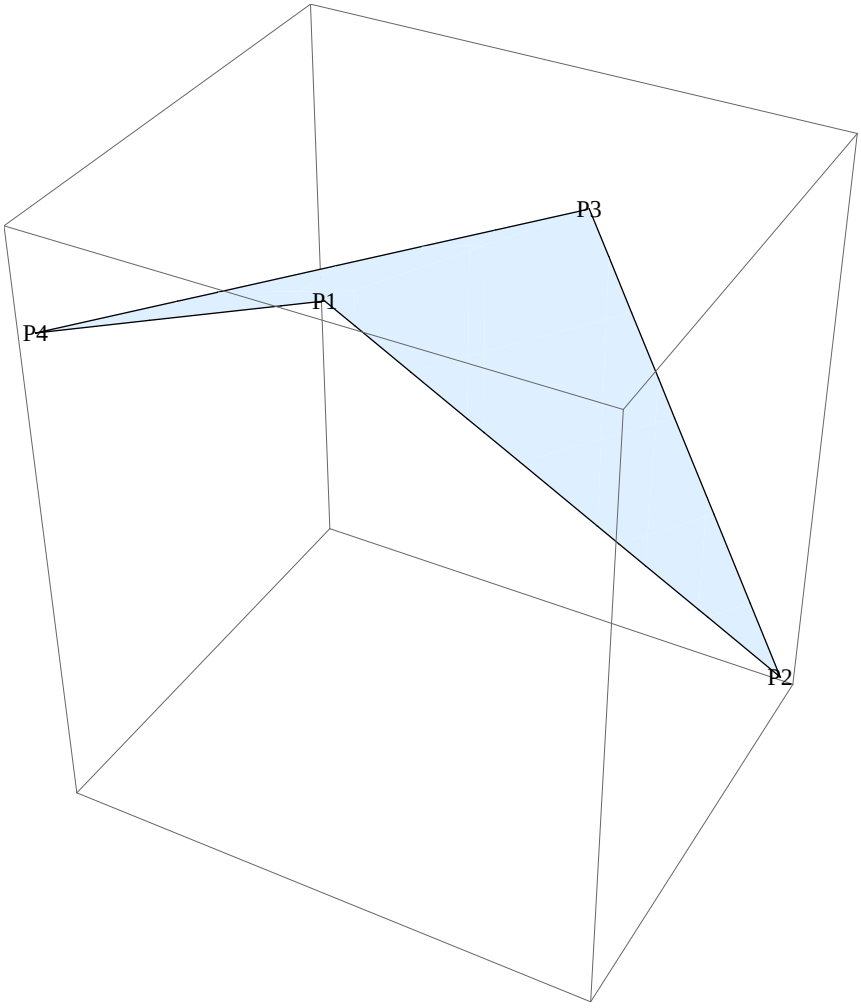
```
q = RotateLeft[p];
Graphics3D[{FaceForm[LightBlue], Polygon[q]}, Lighting → {White}, PlotRange → All]
```



```
txt = {P1, P2, P3, P4};
zeigeDasBild := Graphics3D[{Beschriftung, {FaceForm[LightBlue], Polygon[p]}},
  Lighting → {White}, PlotRange → All]
```

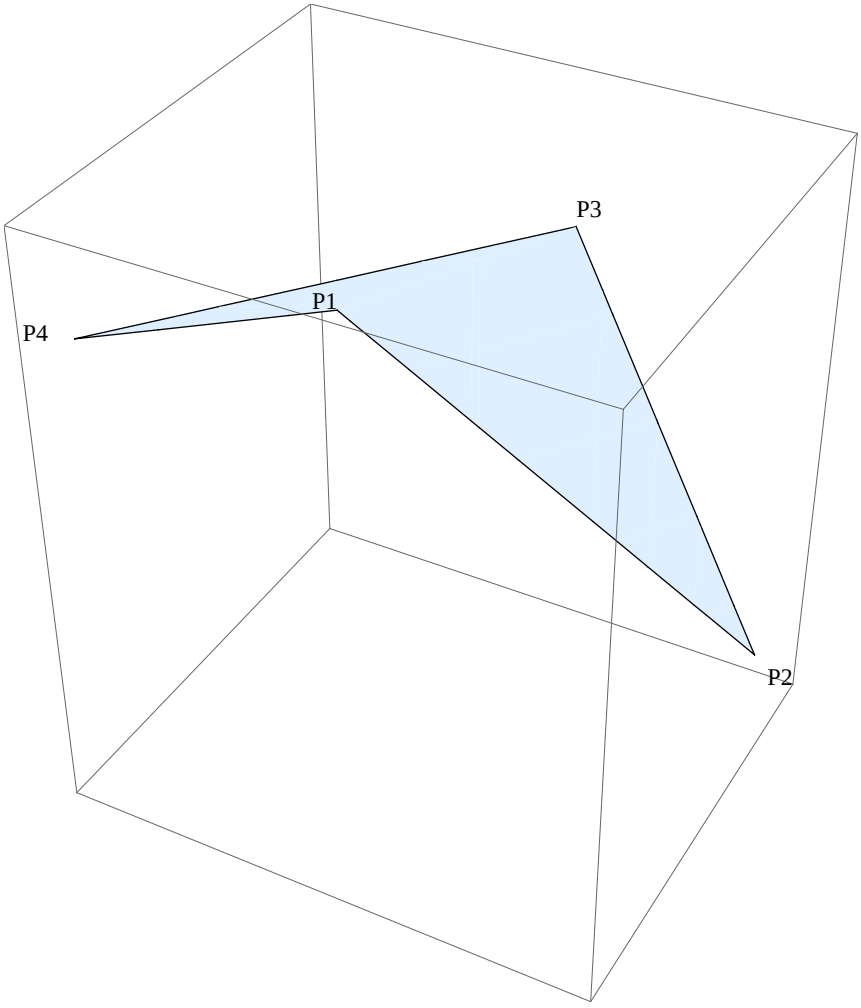
Beschriftung − Version 1

```
Beschriftung = Table[Text[txt[[i]], p[[i]]], {i, 1, 4}];
zeigeDasBild
```



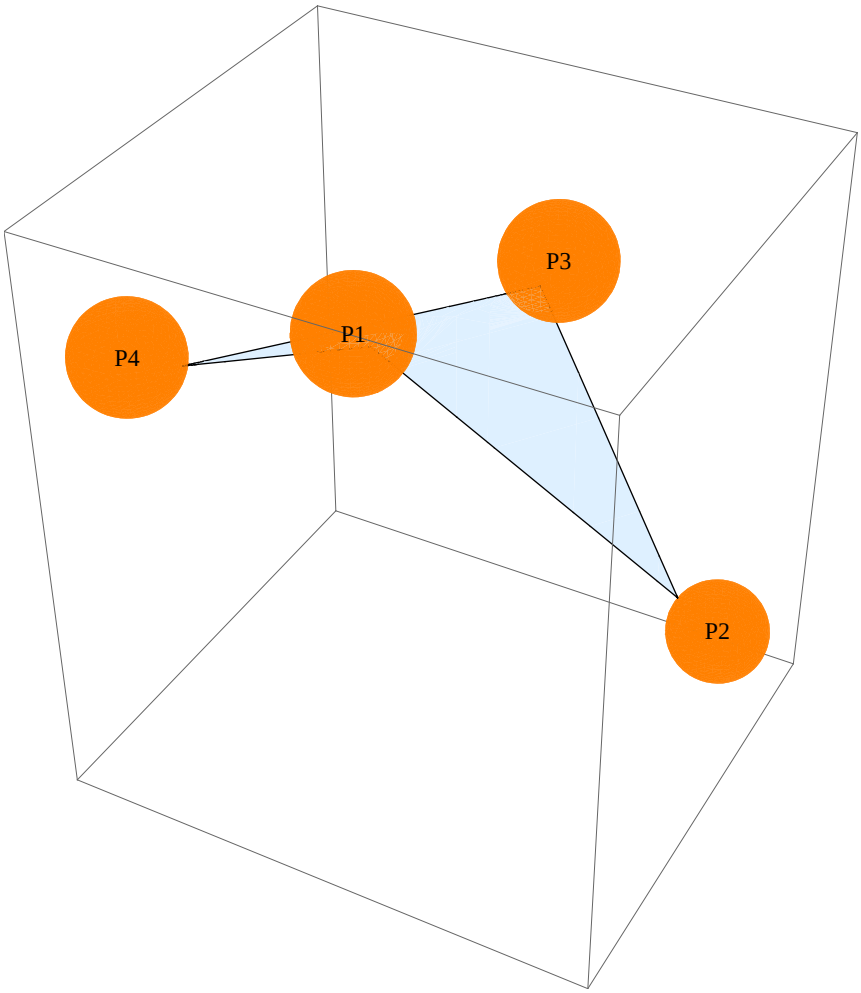
Beschriftung − Version 2

```
M = Mean[p];
Beschriftung = Table[Text[txt[[i]], 1.1 p[[i]] - 0.1 M], {i, 1, 4}];
zeigeDasBild
```



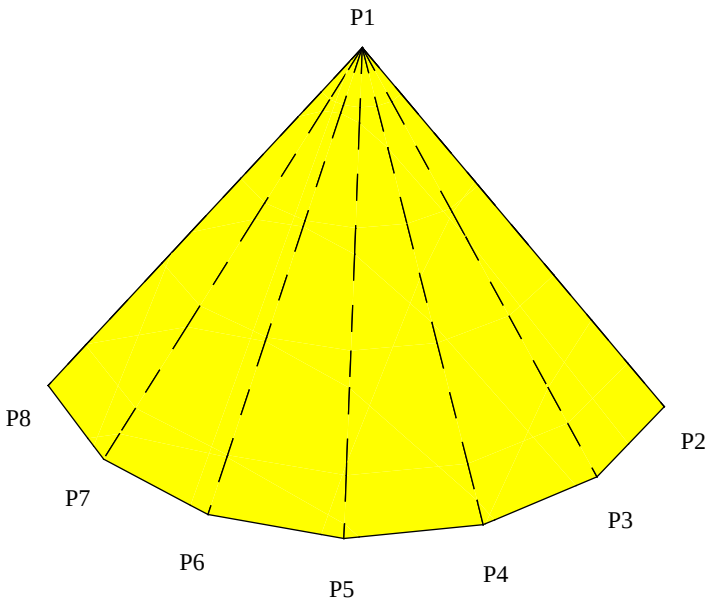
Beschriftung − Version 3

```
Beschriftung = Table[With[{pos = 1.2 p[[i]] - 0.2 M},
    {Text[txt[[i]], pos], {Orange, Sphere[pos, 0.08]}}], {i, 1, 4}];
zeigeDasBild
```



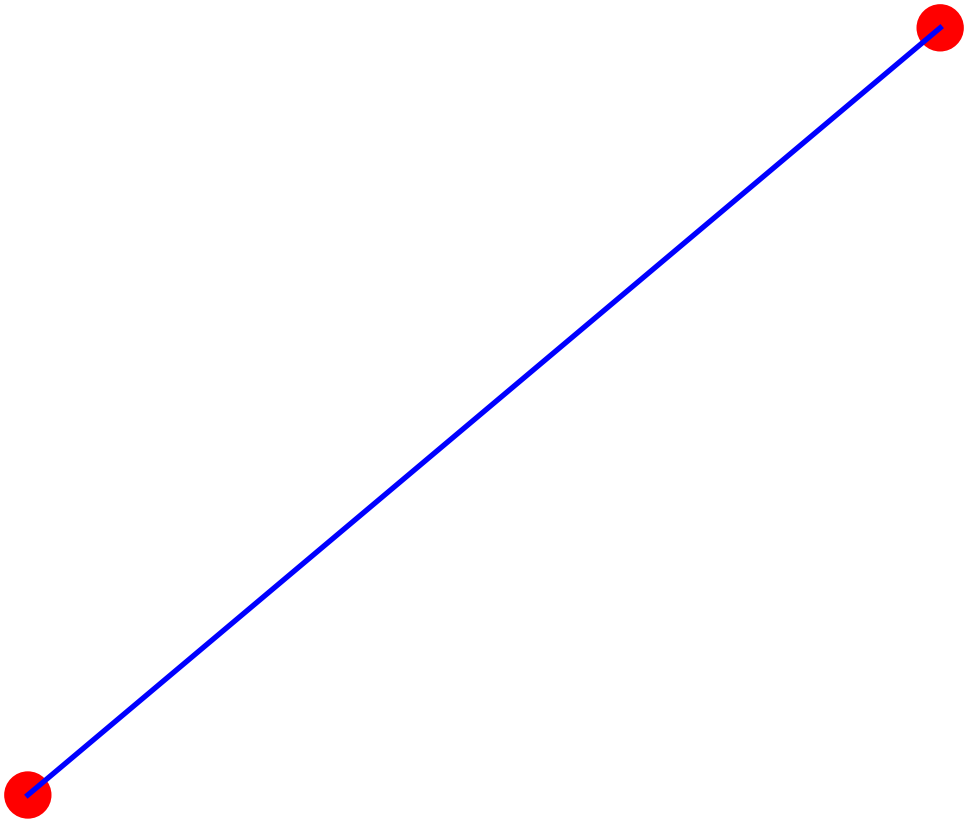
Und nun noch das letzte Bild aus dem Abschnitt.

```
p = Prepend[Table[{Sin[i], Cos[i], 0}, {i, 1.5, 4, .4}], {0, 0, 1}];
n = Length[p];
txt = Table["P" <> ToString[i], {i, 1, n}];
lines = Table[Line[{p[[1]], p[[i]]}], {i, 2, n}];
Beschriftung = Table[Text[txt[[i]], If[i > 1, 1.1 p[[i]] - 0.1 p[[1]], {0, 0, 1.1}]], {i, 1, n}];
Graphics3D[{Beschriftung, {Dashing[.03], lines}, {FaceForm[Yellow], Polygon[p]}},
 Lighting → {White}, PlotRange → All, Boxed → False]
```



Duale Polyeder

```
gc = GraphicsComplex[{{1.2, 3.5}, {5.6, 7.2}},
   {{RGBColor[1, 0, 0], Point[1], Point[2]}, Line[{1, 2}]}];
Graphics[{Blue, PointSize[.05], Thick, gc}]
```



```
gc // Normal
```

```
{{RGBColor[1, 0, 0], {Point[{1.2, 3.5}]}, {Point[{5.6, 7.2}]}},
 Line[{{1.2, 3.5}, {5.6, 7.2}}]}
```

Mathematica enthält mit der Version 6 eine Datenbank über viele reguläre und halbreguläre Körper.

```
PolyhedronData[]
```

```
{{Antiprism, 4}, {Antiprism, 5}, {Antiprism, 6}, {Antiprism, 7},
 {Antiprism, 8}, {Antiprism, 9}, {Antiprism, 10}, AugmentedDodecahedron,
 AugmentedHexagonalPrism, AugmentedPentagonalPrism, AugmentedSphenocorona,
 AugmentedTriangularPrism, AugmentedTridiminishedIcosahedron, AugmentedTruncatedCube,
 AugmentedTruncatedDodecahedron, AugmentedTruncatedTetrahedron,
 BiaugmentedPentagonalPrism, BiaugmentedTriangularPrism, BiaugmentedTruncatedCube,
 BigyrateDiminishedRhombicosidodecahedron, Bilunabirotunda, CsaszarPolyhedron,
 Cube, Cuboctahedron, DeltoidalHexecontahedron, DeltoidalIcositetrahedron,
 DiminishedRhombicosidodecahedron, {Dipyramid, 3}, {Dipyramid, 5},
 DisdyakisDodecahedron, DisdyakisTriacontahedron, Disphenocingulum, Dodecahedron,
 DuerersSolid, ElongatedPentagonalCupola, ElongatedPentagonalDipyramid,
 ElongatedPentagonalGyrobicupola, ElongatedPentagonalGyrobirotunda,
 ElongatedPentagonalGyrocupolarotunda, ElongatedPentagonalOrthobicupola,
 ElongatedPentagonalOrthobirotunda, ElongatedPentagonalOrthocupolarotunda,
 ElongatedPentagonalPyramid, ElongatedPentagonalRotunda, ElongatedSquareCupola,
 ElongatedSquareDipyramid, ElongatedSquareGyrobicupola, ElongatedSquarePyramid,
 ElongatedTriangularCupola, ElongatedTriangularDipyramid,
 ElongatedTriangularGyrobicupola, ElongatedTriangularOrthobicupola,
 ElongatedTriangularPyramid, EschersSolid, GreatDodecahedron, GreatIcosahedron,
 GreatRhombicosidodecahedron, GreatRhombicuboctahedron, GreatStellatedDodecahedron,
 GyrateBidiminishedRhombicosidodecahedron, GyrateRhombicosidodecahedron,
 Gyrobifastigium, GyroelongatedPentagonalBicupola, GyroelongatedPentagonalBirotunda,
 GyroelongatedPentagonalCupola, GyroelongatedPentagonalCupolarotunda,
 GyroelongatedPentagonalPyramid, GyroelongatedPentagonalRotunda,
 GyroelongatedSquareBicupola, GyroelongatedSquareCupola, GyroelongatedSquareDipyramid,
 GyroelongatedSquarePyramid, GyroelongatedTriangularBicupola,
 GyroelongatedTriangularCupola, Hebesphenomegacorona, Icosahedron, Icosidodecahedron,
 JessensOrthogonalIcosahedron, MathematicaPolyhedron, MetabiaugmentedDodecahedron,
 MetabiaugmentedHexagonalPrism, MetabiaugmentedTruncatedDodecahedron,
 MetabidiminishedIcosahedron, MetabidiminishedRhombicosidodecahedron,
 MetabigyrateRhombicosidodecahedron, MetagyrateDiminishedRhombicosidodecahedron,
 Octahedron, ParabiaugmentedDodecahedron, ParabiaugmentedHexagonalPrism,
 ParabiaugmentedTruncatedDodecahedron, ParabidiminishedRhombicosidodecahedron,
 ParabigyrateRhombicosidodecahedron, ParagyrateDiminishedRhombicosidodecahedron,
 PentagonalCupola, PentagonalGyrobicupola, PentagonalGyrocupolarotunda,
 PentagonalHexecontahedron, PentagonalIcositetrahedron, PentagonalOrthobicupola,
 PentagonalOrthobirotunda, PentagonalOrthocupolarotunda, PentagonalRotunda,
 PentakisDodecahedron, {Prism, 3}, {Prism, 5}, {Prism, 6}, {Prism, 7},
 {Prism, 8}, {Prism, 9}, {Prism, 10}, {Pyramid, 4}, {Pyramid, 5},
 RhombicDodecahedron, RhombicTriacontahedron, SmallRhombicosidodecahedron,
 SmallRhombicuboctahedron, SmallStellatedDodecahedron, SmallTriakisOctahedron,
 SnubCube, SnubDisphenoid, SnubDodecahedron, SnubSquareAntiprism, Sphenocorona,
 Sphenomegacorona, SquareCupola, SquareGyrobicupola, SquareOrthobicupola,
 SzilassiPolyhedron, Tetrahedron, TetrakisHexahedron, TriakisIcosahedron,
 TriakisTetrahedron, TriangularCupola, TriangularHebesphenorotunda,
 TriangularOrthobicupola, TriaugmentedDodecahedron, TriaugmentedHexagonalPrism,
 TriaugmentedTriangularPrism, TriaugmentedTruncatedDodecahedron,
 TridiminishedIcosahedron, TridiminishedRhombicosidodecahedron,
 TrigyrateRhombicosidodecahedron, TruncatedCube, TruncatedDodecahedron,
 TruncatedIcosahedron, TruncatedOctahedron, TruncatedTetrahedron}
```

Darunter auch über die fünf Platonischen Körper.

```
platonischeKoerper = PolyhedronData["Platonic"]
```

```
{Cube, Dodecahedron, Icosahedron, Octahedron, Tetrahedron}
```

Die Liste der Schlüsselworte (keys) der etwa zu einem Würfel gespeicherten Informationen:

```
PolyhedronData["Cube", "Properties"]
```

```
{AdjacentFaceIndices, AlternateNames, AlternateStandardNames, Centroid,
 Circumcenter, Circumradius, Circumsphere, Classes, Compound, DefaultOrientation,
 DihedralAngles, DihedralAngleRules, Dual, EdgeCount, EdgeIndices, EdgeLengths,
 Edges, FaceCount, FaceCountRules, FaceIndices, Faces, GeneralizedDiameter, Image,
 InertiaTensor, InertiaTriangle, Information, Incenter, Inradius, Insphere,
 Midcenter, Midradius, Midsphere, Name, NetEdgeIndices, NetEdges, NetFaceIndices,
 NetFaces, NetImage, NetCoordinates, NetCount, NotationRules, Note, Orientations,
 RegionFunction, Scale, SchlaefliSymbol, SkeletonRules, SkeletonCoordinates,
 SkeletonImage, SkeletonGraph, StandardName, SurfaceArea, SymmetryGroupString,
 VertexCoordinates, VertexCount, VertexIndices, Volume, WythoffSymbol}
```

```
PolyhedronData["Cube", "Volume"]
```

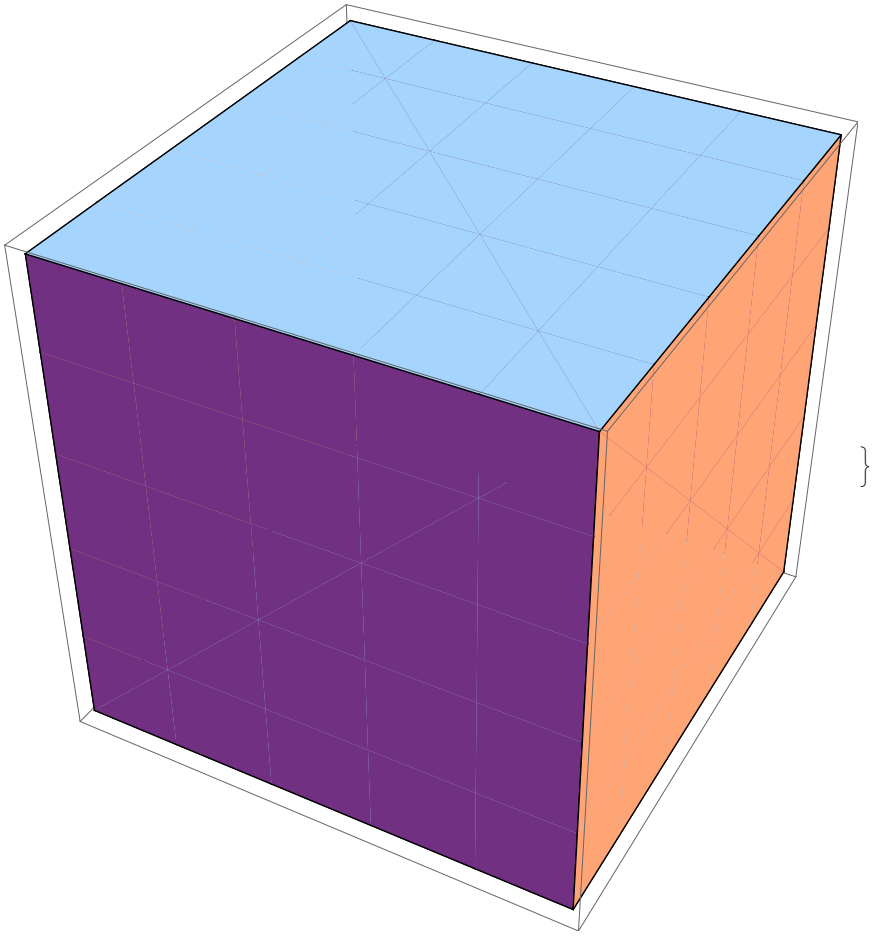
```
1
```

Und hier die Liste der key/value−Paare

```
{#, PolyhedronData["Cube", #]} & /@ PolyhedronData["Cube", "Properties"]
```

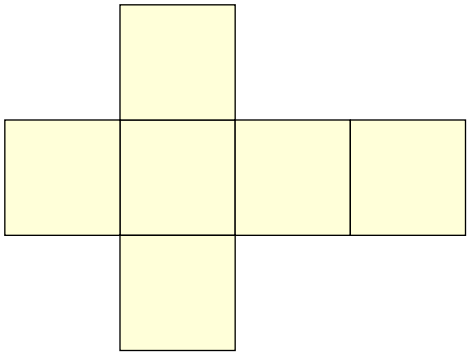
```
{{AdjacentFaceIndices, {{4, 6}, {4, 5}, {5, 6}, {1, 4},
    {1, 6}, {3, 4}, {3, 5}, {1, 3}, {2, 6}, {2, 5}, {1, 2}, {2, 3}}},
 {AlternateNames, {hexahedron, regular cuboid, unit cube, square prism}},
 {AlternateStandardNames, {Hexahedron, {Hypercube, 3}, {Prism, 4}}},
 {Centroid, {0, 0, 0}}, {Circumcenter, {0, 0, 0}},
 {Circumradius, Sqrt[3]/2}, {Circumsphere, Sphere[{0, 0, 0}, Sqrt[3]/2]},
 {Classes, {Convex, Cuboid, Equilateral, Hypercube, Orthotope, Platonic, Prism,
   RectangularParallelepiped, Rhombohedron, Rigid, SpaceFilling, Uniform, Zonohedron}},
 {Compound, Missing[NotAvailable]}, {DefaultOrientation, C4},
 {DihedralAngles, {Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2, Pi/2}},
 {DihedralAngleRules, {{4, 6} → Pi/2, {4, 5} → Pi/2, {5, 6} → Pi/2, {1, 4} → Pi/2, {1, 6} → Pi/2,
    {3, 4} → Pi/2, {3, 5} → Pi/2, {1, 3} → Pi/2, {2, 6} → Pi/2, {2, 5} → Pi/2, {1, 2} → Pi/2, {2, 3} → Pi/2}},
 {Dual, Octahedron}, {EdgeCount, 12}, {EdgeIndices, {{1, 2}, {1, 3}, {1, 5}, {2, 4},
   {2, 6}, {3, 4}, {3, 7}, {4, 8}, {5, 6}, {5, 7}, {6, 8}, {7, 8}}}, {EdgeLengths, {1}},
 {Edges, GraphicsComplex[{{-1/2, -1/2, -1/2}, {-1/2, -1/2, 1/2}, {-1/2, 1/2, -1/2}, {-1/2, 1/2, 1/2},
    {1/2, -1/2, -1/2}, {1/2, -1/2, 1/2}, {1/2, 1/2, -1/2}, {1/2, 1/2, 1/2}}, Line[{{1, 2}, {1, 3},
    {1, 5}, {2, 4}, {2, 6}, {3, 4}, {3, 7}, {4, 8}, {5, 6}, {5, 7}, {6, 8}, {7, 8}}]]},
 {FaceCount, 6}, {FaceCountRules, {4 → 6}}, {FaceIndices,
  {{8, 4, 2, 6}, {8, 6, 5, 7}, {8, 7, 3, 4}, {4, 3, 1, 2}, {1, 3, 7, 5}, {2, 1, 5, 6}}},
 {Faces, GraphicsComplex[{{-1/2, -1/2, -1/2}, {-1/2, -1/2, 1/2}, {-1/2, 1/2, -1/2}, {-1/2, 1/2, 1/2},
    {1/2, -1/2, -1/2}, {1/2, -1/2, 1/2}, {1/2, 1/2, -1/2}, {1/2, 1/2, 1/2}}, Polygon[
```

```
    {{8, 4, 2, 6}, {8, 6, 5, 7}, {8, 7, 3, 4}, {4, 3, 1, 2}, {1, 3, 7, 5}, {2, 1, 5, 6}}]],
 {GeneralizedDiameter, √3}, {Image,
```



```
 },
 {InertiaTensor, {{1/6, 0, 0}, {0, 1/6, 0}, {0, 0, 1/6}}},
 {InertiaTriangle, {{1/6, 0, 0}, {1/6, 0}, {1/6}}},
 {Information, http://mathworld.wolfram.com/Cube.html},
 {Incenter, {0, 0, 0}}, {Inradius, 1/2}, {Insphere, Sphere[{0, 0, 0}, 1/2]},
 {Midcenter, {0, 0, 0}}, {Midradius, 1/√2},
 {Midsphere, Sphere[{0, 0, 0}, 1/√2]}, {Name, cube}, {NetEdgeIndices,
  {{1, 2}, {1, 4}, {2, 5}, {3, 4}, {3, 7}, {4, 5}, {4, 8}, {5, 6}, {5, 9}, {6, 10},
   {7, 8}, {8, 9}, {8, 11}, {9, 10}, {9, 12}, {11, 12}, {11, 13}, {12, 14}, {13, 14}}},
 {NetEdges, GraphicsComplex[{{0, 1}, {0, 2}, {1, 0}, {1, 1}, {1, 2}, {1, 3},
    {2, 0}, {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {4, 1}, {4, 2}},
   Line[{{1, 2}, {1, 4}, {2, 5}, {3, 4}, {3, 7}, {4, 5}, {4, 8}, {5, 6}, {5, 9}, {6, 10},
     {7, 8}, {8, 9}, {8, 11}, {9, 10}, {9, 12}, {11, 12}, {11, 13}, {12, 14}, {13, 14}}]]},
 {NetFaceIndices, {{5, 4, 8, 9}, {6, 5, 9, 10}, {2, 1, 4, 5},
   {4, 3, 7, 8}, {9, 8, 11, 12}, {12, 11, 13, 14}}},
 {NetFaces, GraphicsComplex[{{0, 1}, {0, 2}, {1, 0}, {1, 1}, {1, 2}, {1, 3}, {2, 0},
```

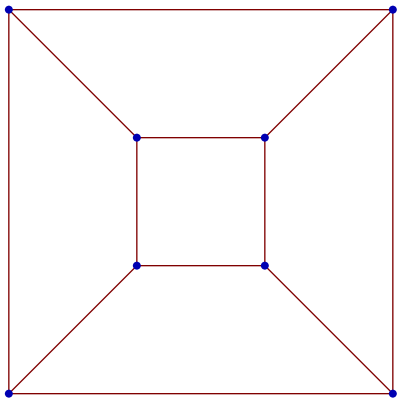
```
    {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {4, 1}, {4, 2}}, Polygon[{{5, 4, 8, 9},
     {6, 5, 9, 10}, {2, 1, 4, 5}, {4, 3, 7, 8}, {9, 8, 11, 12}, {12, 11, 13, 14}}]]},
```

{NetImage,  },

```
{NetCoordinates, {{0, 1}, {0, 2}, {1, 0}, {1, 1}, {1, 2}, {1, 3},
  {2, 0}, {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {4, 1}, {4, 2}}},
{NetCount, 11}, {NotationRules, {Uniform → 6, WenningerModel → 3,
  SchlaefliSymbol → {4, 3}, WythoffSymbol → {{3}, {2, 4}}}},
{Note, Missing[NotApplicable]}, {Orientations, {C2, C3, C4}},
```

$$\left\{\text{RegionFunction}, 2\,\#3 \le 1 \&\& 2\,\#1 \le 1 \&\& 2\,\#2 \le 1 \&\& \#1 \ge -\frac{1}{2} \&\& \#3 \ge -\frac{1}{2} \&\& \#2 \ge -\frac{1}{2} \&\right\},$$

```
{Scale, Missing[NotApplicable]},
{SchlaefliSymbol, {4, 3}}, {SkeletonRules,
 {1 → 2, 1 → 3, 1 → 5, 2 → 4, 2 → 6, 3 → 4, 3 → 7, 4 → 8, 5 → 6, 5 → 7, 6 → 8, 7 → 8}},
{SkeletonCoordinates, {{-0.333, -0.333}, {-1., -1.}, {-0.333, 0.333},
  {-1., 1.}, {0.333, -0.333}, {1., -1.}, {0.333, 0.333}, {1., 1.}}},
```

{SkeletonImage,  },

```
{SkeletonGraph, CubicalGraph}, {StandardName, Cube},
{SurfaceArea, 6}, {SymmetryGroupString, Oh},
```

$$\left\{\text{VertexCoordinates}, \left\{\left\{-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}\right\}, \left\{-\frac{1}{2}, -\frac{1}{2}, \frac{1}{2}\right\}, \left\{-\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}\right\},\right.\right.$$

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

```
{VertexCount, 8}, {VertexIndices, {1, 2, 3, 4, 5, 6, 7, 8}},
{Volume, 1}, {WythoffSymbol, {{3}, {2, 4}}}}
```

Würfel und Oktaeder sind zueinander dual: Verbindet man die Seitenmitten eines Würfels, so ergibt sich ein Oktaeder und umgekehrt.
Wir wollen dies in einem 3D−Bild veranschaulichen. Ausgangspunkt sind die "Faces"−Darstellungen der beiden Körper aus der Datenbank.

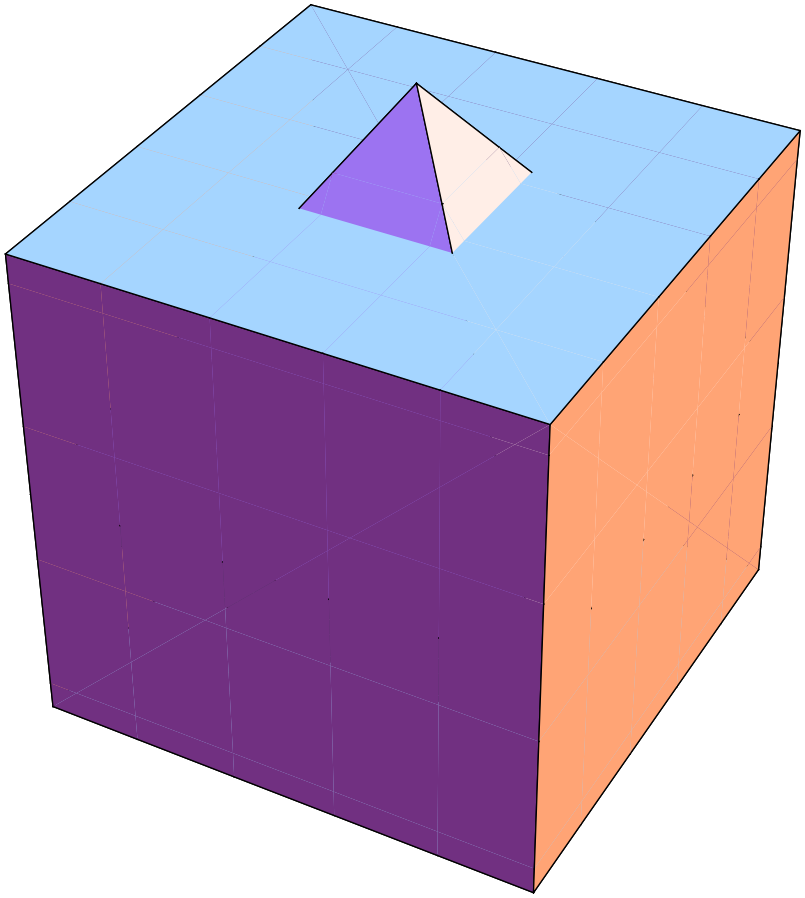
```
cube = PolyhedronData["Cube", "Faces"]
octahedron = PolyhedronData["Octahedron", "Faces"]
```

```
GraphicsComplex[{{-1/2, -1/2, -1/2}, {-1/2, -1/2, 1/2}, {-1/2, 1/2, -1/2},
  {-1/2, 1/2, 1/2}, {1/2, -1/2, -1/2}, {1/2, -1/2, 1/2}, {1/2, 1/2, -1/2}, {1/2, 1/2, 1/2}}, Polygon[
  {{8, 4, 2, 6}, {8, 6, 5, 7}, {8, 7, 3, 4}, {4, 3, 1, 2}, {1, 3, 7, 5}, {2, 1, 5, 6}}]]
```

```
GraphicsComplex[
 {{-1/2, -1/2, 0}, {-1/2, 1/2, 0}, {0, 0, -1/Sqrt[2]}, {0, 0, 1/Sqrt[2]}, {1/2, -1/2, 0}, {1/2, 1/2, 0}}, Polygon[
  {{4, 5, 6}, {4, 6, 2}, {4, 2, 1}, {4, 1, 5}, {5, 1, 3}, {5, 3, 6}, {3, 1, 2}, {6, 3, 2}}]]
```

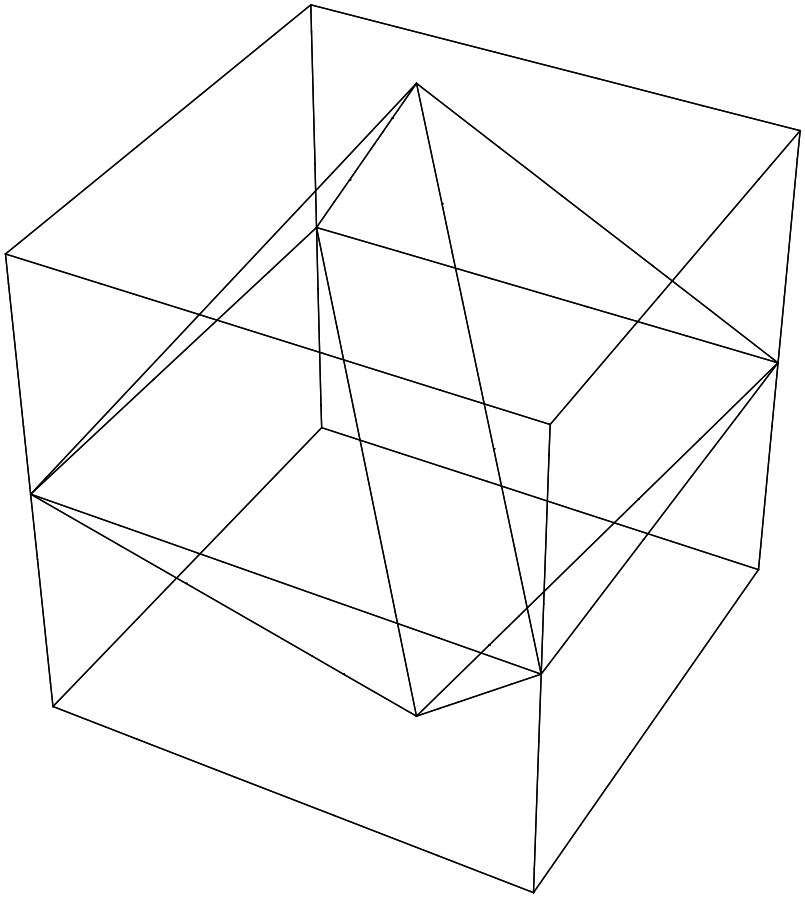
Beide sind **GraphicsComplex**−Konstrukte, aus denen sich sofort ein erstes Bild erzeugen lässt.

```
g = Graphics3D[{cube, octahedron}, Boxed → False]
```



Das Oktaeder ragt aus dem Würfel heraus, muss also noch skaliert werden. Komisch, dass es nur an zwei Seiten aus dem Würfel herausragt.
Ein Drahtgitterbild bringt den Grund ans Licht: Das Oktaeder liegt "falsch herum" im Würfel und muss auch noch gedreht werden.

```
Graphics3D[{FaceForm[], cube, octahedron}, Boxed → False]
```



cube und **octahedron** sind **GraphicsComplex**−Primitive, auf denen sich geometrische Transformationen leicht ausführen lassen. Das erste Argument eines **GraphicsComplex**−Primitivs ist eine Liste von Punkt−Koordinaten, das zweite eine Beschreibung der Linien− und Flächenelemente des Objekts relativ zu diesen Punkten. Wir müssen also nur die Koordinaten der Punkte mit der entsprechenden Transformationsmatrix multiplizieren und daraus das neue Grafikprimitiv nach denselben Anweisungen aufbauen.
Hier ist schon mal eine Funktionsdefinition, die das erledigt:

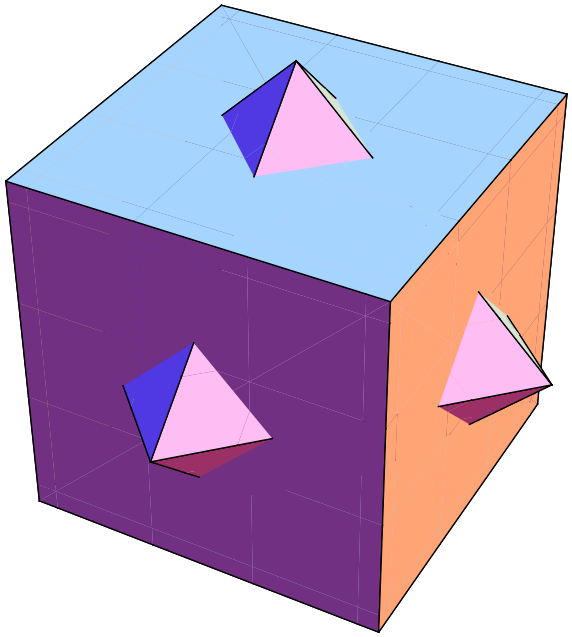
```
gcTransform[r_, g_GraphicsComplex] := GraphicsComplex[(r.# &) /@ g[[1]], g[[2]]]
```

Die erforderliche Transformation des Oktaeders können wir aus zwei Schritten zusammensetzen. Zunächst die Drehung in z−Richtung um 90°.

```
(r = RotationMatrix[π / 4, {0, 0, 1}]) // MatrixForm
```

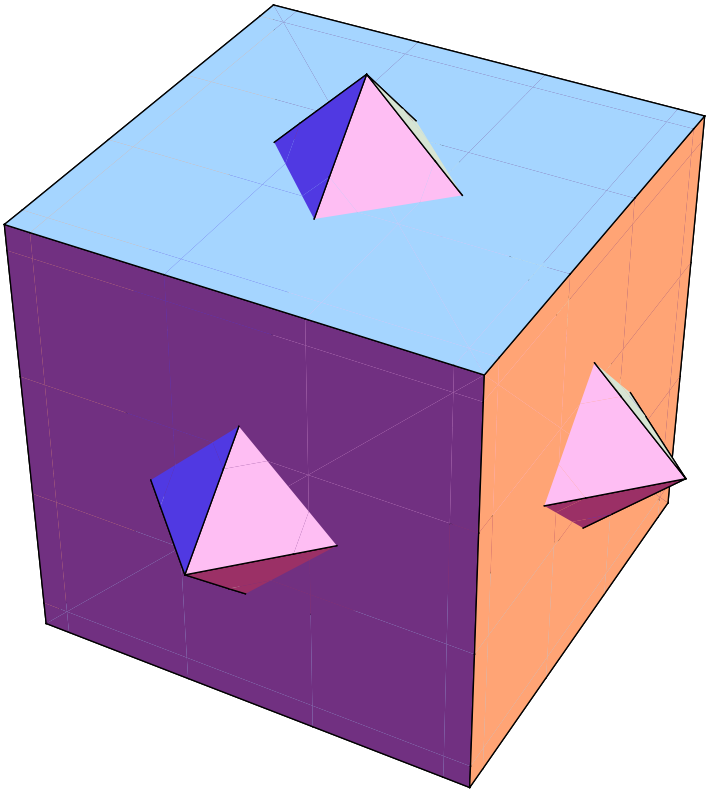
$$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
p = Graphics3D[{cube, gcTransform[r, octahedron]}, Boxed → False]
```



Rund um die Grafik ist noch extrem viel Platz. Das hat damit zu tun, dass die 3D−Begrenzungsbox durch die Mitten des Oktaeders bestimmt wird, also um den Faktor $\sqrt{2}$, wie wir gleich sehen werden, größer ist als der Würfel. Diese Begrenzungsbox muss ihrerseits standardmäßig vollständig ins 2D−Bildfenster passen. Mit der Option **ViewAngle** können Sie die Größe der Szene beeinflussen.

```
Show[p, ViewAngle -> 20 °]
```



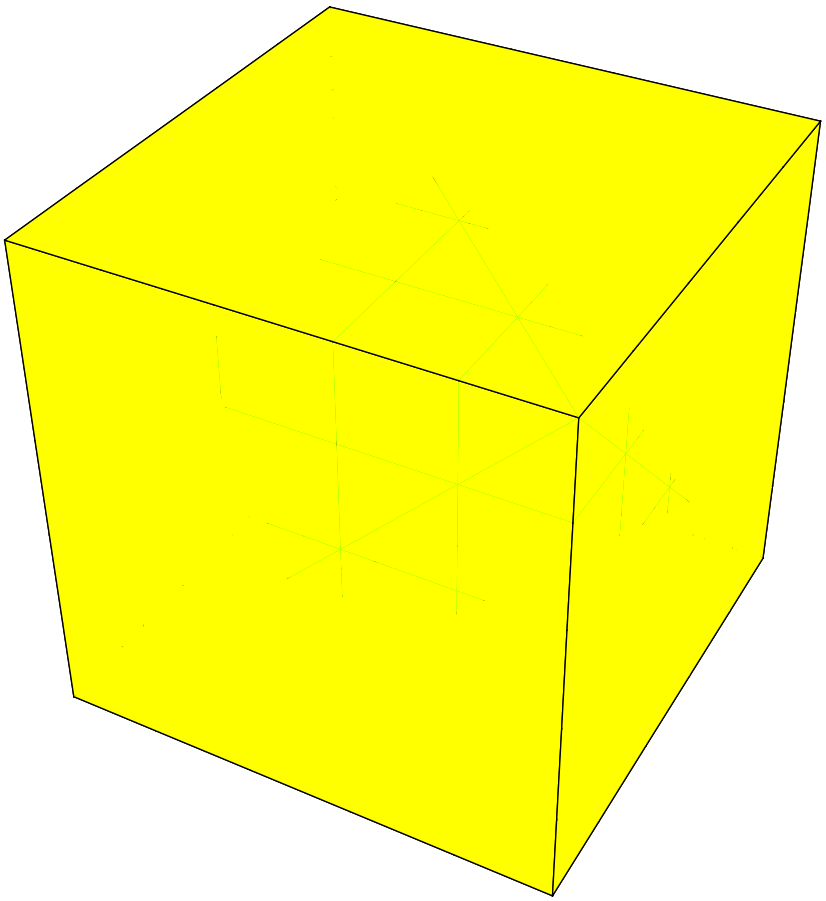
Nun schaut das Oktaeder an allen Seiten auf gleiche Weise heraus und muss nur noch skaliert werden, so dass Inkugelradius des Würfels und Umkugelradius des Oktaeders übereinstimmen.
Die erforderlichen Informationen können wir wieder der Datenbank entnehmen. Sie sind für alle fünf platonischen Körper in der folgenden Tabelle zusammengestellt.

```
radii = {#, PolyhedronData[#, "Inradius"], PolyhedronData[#, "Circumradius"]} & /@
   platonischeKoerper;
Text@Grid[Prepend[radii, {"Körper", "Inkugelradius", "Umkugelradius"}],
  Dividers → {{{True}}, {True, True, {False}, True}},
  Spacings → {1, 2}]
```

Körper	Inkugelradius	Umkugelradius
Cube	$\frac{1}{2}$	$\frac{\sqrt{3}}{2}$
Dodecahedron	$\frac{1}{20}\sqrt{250+110\sqrt{5}}$	$\frac{1}{4}\left(\sqrt{3}+\sqrt{15}\right)$
Icosahedron	$\frac{1}{12}\left(3\sqrt{3}+\sqrt{15}\right)$	$\frac{1}{4}\sqrt{10+2\sqrt{5}}$
Octahedron	$\frac{1}{\sqrt{6}}$	$\frac{1}{\sqrt{2}}$
Tetrahedron	$\frac{1}{2\sqrt{6}}$	$\frac{\sqrt{\frac{3}{2}}}{2}$

Das Oktaeder ist also um den Faktor $\sqrt{2}$ zu verkleinern.
Die Szene ist mit weißem Licht ausgeleuchtet, um die Eigenfarben der beiden Körper unverfälscht zur Geltung zu bringen. Der Würfel ist außerdem leicht durchsichtig, um auch einen Blick ins Innere zu ermöglichen.

```
g = Graphics3D[{{Yellow, Opacity[.3], cube}, {Green, gcTransform[r/Sqrt[2], octahedron]}},
  PlotRange → All, Boxed → False, Lighting → {White}]
```



```
rt = RotationTransform[π / 4, {0, 0, 1}]
st = ScalingTransform[Table[1/√2, {3}]]
ct = Composition[st, rt]
transformedOctahedron = GeometricTransformation[octahedron, ct]
```

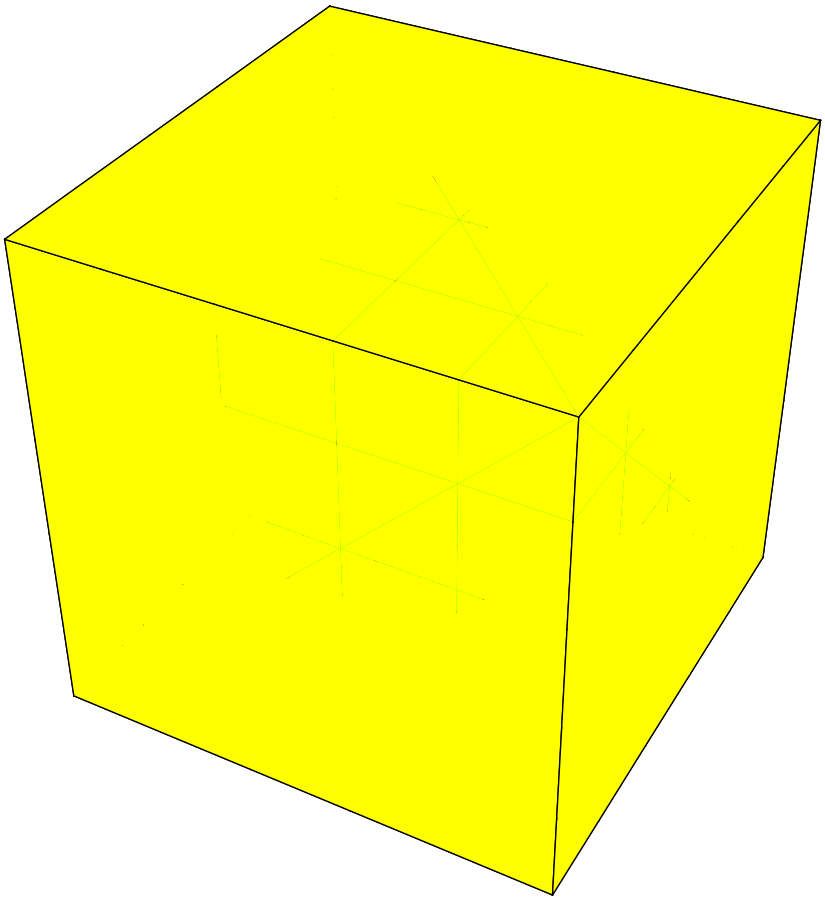
$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

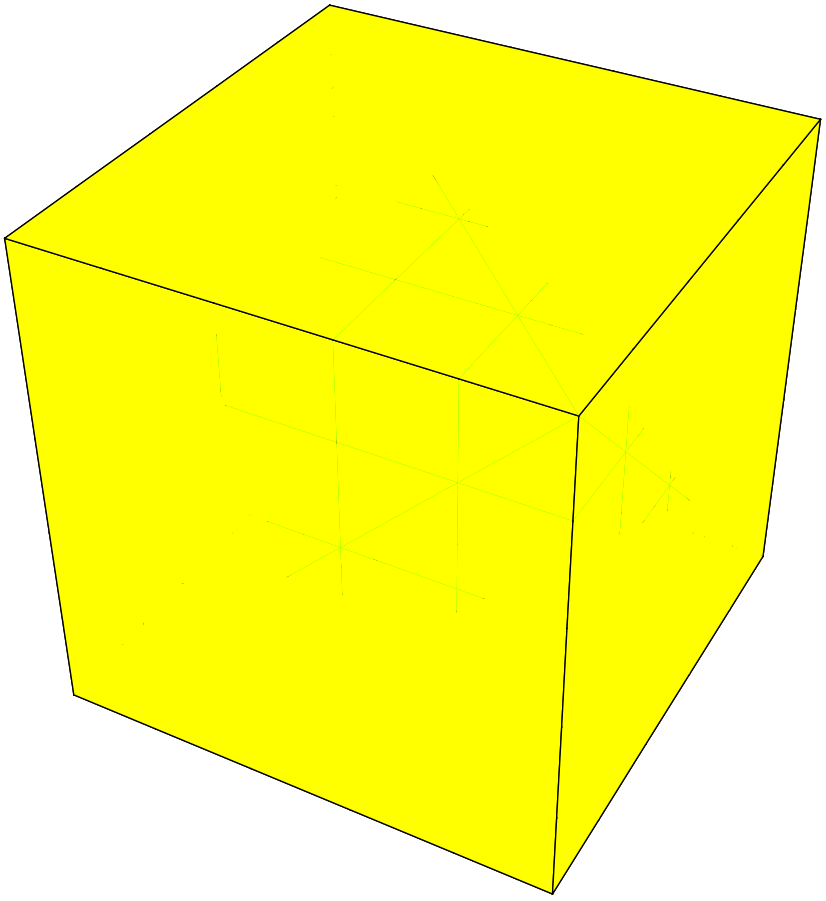
```
GeometricTransformation[GraphicsComplex[{{-1/2, -1/2, 0}, {-1/2, 1/2, 0}, {0, 0, -1/√2}, {0, 0, 1/√2}, {1/2, -1/2, 0}, {1/2, 1/2, 0}}, Polygon[
  {{4, 5, 6}, {4, 6, 2}, {4, 2, 1}, {4, 1, 5}, {5, 1, 3}, {5, 3, 6}, {3, 1, 2}, {6, 3, 2}}]],
 {{{1/2, -1/2, 0}, {1/2, 1/2, 0}, {0, 0, 1/√2}}, {0, 0, 0}}]
```

```
Graphics3D[{{Yellow, Opacity[.3], cube}, {Green, transformedOctahedron}},
 PlotRange → All, Boxed → False, Lighting → {White}]
```



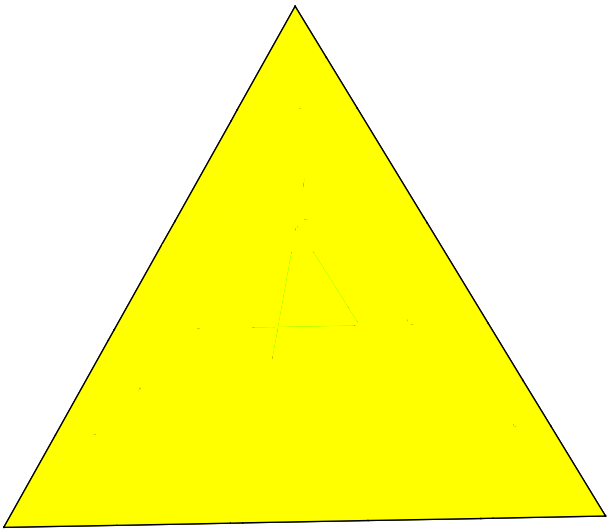
Und nun packen wir noch den Würfel in das Oktaeder.

```
Graphics3D[{{Yellow, Opacity[.3], cube}, {Green, Opacity[.5],
    gcTransform[r/Sqrt[2], octahedron], {Red, gcTransform[1/3 IdentityMatrix[3], cube]}}},
 PlotRange → All, Boxed → False, Lighting → {White}]
```



Das Tetraeder ist zu sich selbst dual.

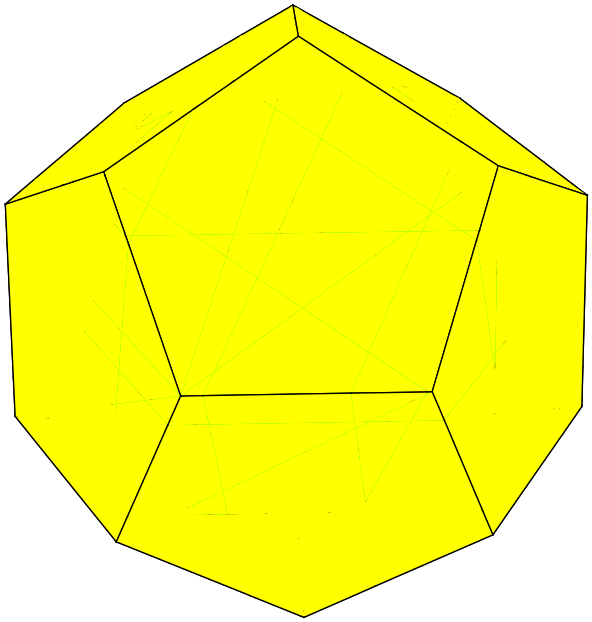
```
tetrahedron = PolyhedronData["Tetrahedron", "Faces"];
Graphics3D[{{Yellow, Opacity[.3], tetrahedron},
  {Green, Opacity[.5], gcTransform[-1/3 IdentityMatrix[3], tetrahedron]},
  {Red, gcTransform[1/9 IdentityMatrix[3], tetrahedron]}},
 PlotRange → All, Boxed → False, Lighting → {White}]
```



Dasselbe nun auch noch für Dodekaeder und Ikosaeder.

```
dodecahedron = PolyhedronData["Dodecahedron", "Faces"];
icosahedron = PolyhedronData["Icosahedron", "Faces"];
```

```
α = radii⟦3, 3⟧ / radii⟦2, 2⟧;
β = radii⟦2, 3⟧ / radii⟦3, 2⟧;
r = RotationMatrix[2 π / 5, {0, 1, 0}];
Graphics3D[{{Yellow, Opacity[.3], dodecahedron},
  {Green, Opacity[.5], gcTransform[r / α, icosahedron]},
  {Red, gcTransform[IdentityMatrix[3] / (α β), dodecahedron]}},
 PlotRange → All, Boxed → False, Lighting → {White}]
```



■ Dynamische Grafiken und Animationen

```
ClearAll["Global`*"]
```

Animate als automatisches Abspielen, **Manipulate** als interaktive Variation.

```
Manipulate[Factor[x^n - 1], {n, 2, 10, 1}]
```

```
Animate[Plot[Sin[x + a], {x, 0, 10}], {a, 0, 5}]
```

Das **Manipulate**−Objekt kann über Schieberegler oder ein Kontrollpanel gesteuert werden. Das Kontrollpanel wird durch Knopfdruck geöffnet. In einer Dialogbox finden sich Knöpfe, über welche die Szene automatisch oder im Schrittmodus abgespielt werden kann.

```
Manipulate[Plot[Sin[x (1 + a x)], {x, 0, 6}], {a, 0, 2}]
```

Animation einer 3D−Grafik − wir fliegen um das Objekt herum, indem wir den **ViewPoint** ändern.
Hier zunächst ein Blick auf das ruhende Objekt.

```
Plot3D[Sin[x y], {x, -π, π}, {y, -π, π}, PlotPoints → 5, PlotRange → 3, Axes → None]
```

Die Animation funktioniert nun, allein das Gitternetz hat ein Eigenleben.

```
Manipulate[Plot3D[Sin[x y], {x, -π, π}, {y, -π, π}, PlotPoints → 5, PlotRange -> 3,
  ViewPoint → {3 Sin[α], 3 Cos[α], 2} , ViewAngle -> 25 °, Axes → None], {α, 0., 2 π, π/12}]
```

Man kann die Daten auch erst separat erzeugen und dann mit **ListAnimate** animieren.

```
data = Table[Plot3D[Sin[x y], {x, -π, π}, {y, -π, π}, PlotPoints → 5, PlotRange → 3,
    ViewPoint → {3 Sin[α], 3 Cos[α], 2}, ViewAngle -> 25 °, Axes -> None], {α, 0, 2 π, π/12}];
ListAnimate[data, .2]
Manipulate[data[[i]], {i, 1, Length[data], 1}]
```