Optical component Library

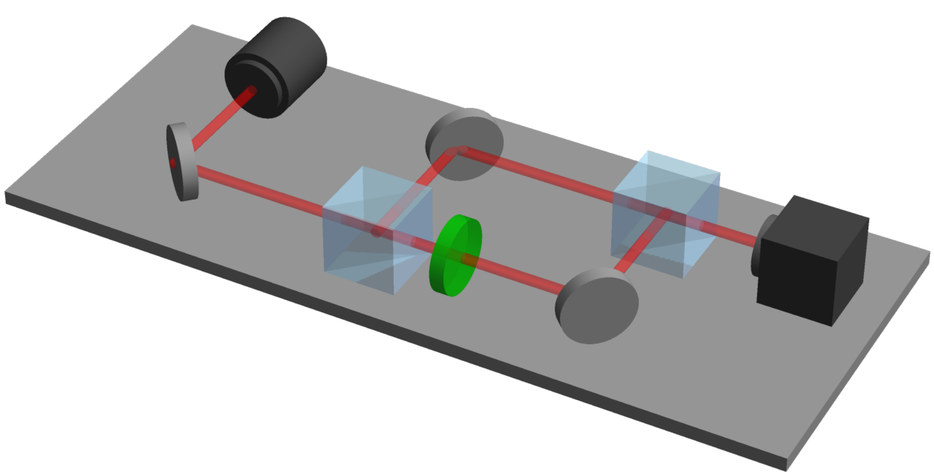
This library is a free, open collection of designs for sketching diagrams related to laser optics. Each element is defined as a function that can placed in the working space

Mach-Zehnder Interferometer

In[185]:=

```
Beam = Graphics3D[{{CapForm["Square"], Opacity[0.3], Red,
     Tube[{{1, 2, 0.5}, {1, 0, 0.5}, {7, 0, 0.5}, {7, 2, 0.5}}, 0.07]}, {CapForm["Square"],
     Opacity[0.3], Red, Tube[{{4, 0, 0.5}, {4, 2, 0.5}, {9, 2, 0.5}}, 0.07]}}];
Show[
 OpticalTable[{11, 4.5}],
 LaserCyl[{1, 3, 0}, -π / 2],
 Beam,
 M[{1, 0, 0}, π / 4],
 BS[{4, 0, 0}, π / 2],
 QW[{5, 0, 0}],
 M[{7, 0, 0}, -π / 4],
 BS[{7, 2, 0}, π / 2],
 M[{4, 2.2, 0}, -π / 4],
 CCD[{9, 2, 0}, π],
 PlotRange → All, ImageSize → 500]
```

Out[186]=

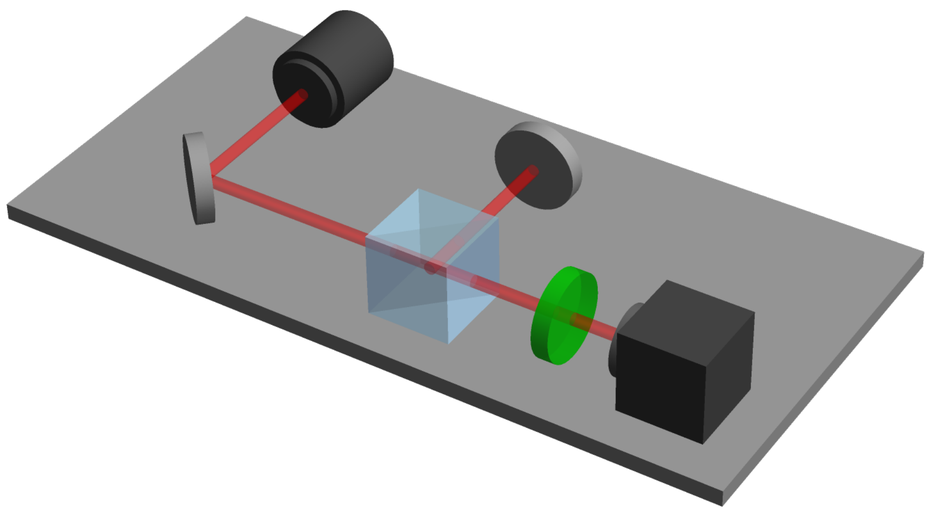


Michelson Interferometer

In[187]:=

```
Beam = Graphics3D[{{CapForm["Square"],
     Opacity[0.3], Red, Tube[{{1, 2, 0.5}, {1, 0, 0.5}, {7, 0, 0.5}}, 0.07]},
    {CapForm["Square"], Opacity[0.3], Red, Tube[{{4, 0, 0.5}, {4, 2, 0.5}}, 0.07]}}];
Show[
 OpticalTable[{9, 4.5}],
 LaserCyl[{1, 3, 0}, -π / 2],
 Beam,
 M[{0.9, -0.1, 0}, π / 4],
 BS[{4, 0, 0}, π / 2],
 QW[{5.5, 0, 0}],
 M[{7, 0, 0}, -π / 2],
 M[{4, 2.2, 0}, -π / 2],
 CCD[{7, 0, 0}, π], ImageSize → 500]
```

Out[188]=

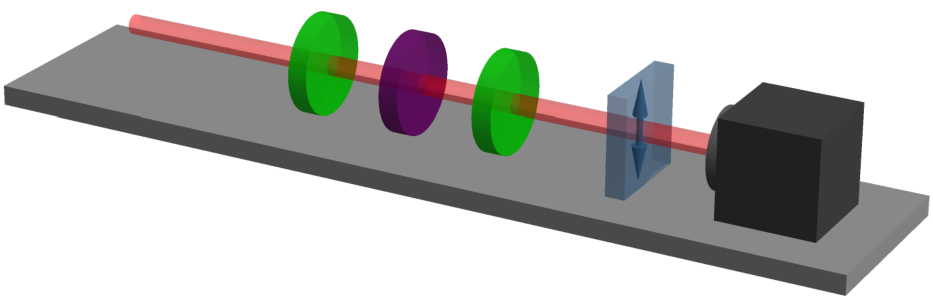


Retardation measurement

In[189]:=

```
Beam = Graphics3D[{{CapForm["Square"], Opacity[0.3],
     Red, Tube[{{0, 0, 0.5}, {6, 0, 0.5}, {7, 0, 0.5}}, 0.1]}}];
Show[
 OpticalTable[{9, 2.}],
 Beam,
 QW[{2, 0, 0}],
 HW[{3, 0, 0}],
 QW[{4, 0, 0}],
 Pol[{5.5, 0, 0}, π / 2, π / 2],
 CCD[{7, 0, 0}, π], ImageSize → 500]
```

Out[190]=



Functions

In[2]:=

```
(*Establishes the directory where the notebook is located as the default one*)
SetDirectory@NotebookDirectory[];
```

Color definition

In[3]:=

```
NiceBlue = RGBColor[2 / 5, 178 / 255, 1];

KnottyBlue = RGBColor[1 / 255, 89 / 255, 185 / 255];

MexGreen = RGBColor[0, 104 / 255, 71 / 255];

Pantone2459 = RGBColor[1 / 255, 181 / 255, 174 / 255];

Pantone218 = RGBColor[206 / 255, 102 / 255, 161 / 255];

Pantone199 = RGBColor[227 / 255, 56 / 255, 109 / 255];

Pantone149 = RGBColor[243 / 255, 194 / 255, 66 / 255];

PantoneProceBlue = RGBColor[63 / 255, 143 / 255, 205 / 255];

Pantone7664 = RGBColor[104 / 255, 48 / 255, 120 / 255];
```

BasicTools

In[12]:=

```
(*Cartesian Axes*)
AxisDe = Graphics3D[{(*X-Direction*)
     {Opacity[0.6], Blue, Arrowheads[0.1], Arrow[Tube[{{0, 0, 0}, {2, 0, 0}}, 0.1]]},
                     (*Z-direction*)
     {Opacity[0.6], Red, Arrowheads[0.1], Arrow[Tube[{{0, 0, 0}, {0, 0, 2}}, 0.1]]},
                     (*Y-direction*)
     {Opacity[0.6], Green, Arrowheads[0.1], Arrow[Tube[{{0, 0, 0}, {0, 2, 0}}, 0.1]]}},
   ViewPoint → {0.8, 1.5, 0.4}, Boxed → False, Lighting → "Neutral"];
```

Basic Elements

In[13]:=
```
(*Basic Plates*)
Options[PLATE] = {"Position" → {0, 0, 0.5}, "Radius" → 0.5,
   "Rotation" → {0, 0}, "Length" → 0.2, "Color" → Gray, "Opacity" → 1};
PLATE[OptionsPattern[]] := Module[{Pos, Len, Col, Op, Rot, Rad},
   {Pos, Len, Col, Op, Rot, Rad} =
    OptionValue[{"Position", "Length", "Color", "Opacity", "Rotation", "Radius"}];
   Graphics3D[{Opacity[Op], EdgeForm[], Col, Cylinder[
      {Pos, Len * {Cos[Rot〚2〛] Sin[Rot〚1〛], Sin[Rot〚1〛] Sin[Rot〚2〛], Cos[Rot〚1〛]} + Pos},
      Rad]}, Lighting → "Neutral", Boxed → False]];

(*Basic Rectangular Element: The entries should be *)
Options[PRISM] = {"Position" → {0, 0, 0}, "Dimensions" → {1, 1, 1},
   "Color" → Gray, "Opacity" → 1, "Rotation" → {π / 2, 0}};
PRISM[OptionsPattern[]] := Module[{Pos, Dim, Col, Op, Rot, T},
  {Pos, Dim, Col, Op, Rot} =
   OptionValue[{"Position", "Dimensions", "Color", "Opacity", "Rotation"}];
  Show[Graphics3D[{{Opacity[Op], EdgeForm[], Col,
      T = (Dim〚1〛 {Cos[Rot〚2〛], Sin[Rot〚2〛], 0} +
          Dim〚2〛 {-Sin[Rot〚2〛], Cos[Rot〚2〛], 0}) / 2;
      Parallelepiped[Pos - T, {Dim〚3〛 {0, 0, 1},
        Dim〚1〛 {Cos[Rot〚2〛], Sin[Rot〚2〛], 0}, Dim〚2〛 {-Sin[Rot〚2〛], Cos[Rot〚2〛], 0}}]
     }}, Lighting → "Neutral", Boxed → False]]]
```

Basic Beam operations

In[17]:=
```
Graphics3D[{Opacity[0.3], Darker@Red, EdgeForm[],
  Cone[{{0, -1.0, 0}, {0, -2, 0}}, 0.07]}], Graphics3D[
 {Opacity[0.3], Darker@Red, EdgeForm[], Cone[{{0, -3, 0}, {0, -1.8, 0}}, 0.07]}]
```

Basic Optical Elements

```
(*Optical Table*)
OpticalTable[A_, Thick_ : -.2] := Graphics3D[{EdgeForm[], Gray,
   Parallelepiped[{-1, -1, 0}, {Thick {0, 0, 1}, A〚1〛 {1, 0, 0}, A〚2〛 {0, 1, 0}}]},
  Lighting → "Neutral", Boxed → False]

(*Mirrors *) (*Still working on the proper alignment*)
M[Position_, Degree_ : 0] :=
 PLATE["Position" → Position + {0, 0, 0.5}, "Rotation" → {π / 2, Degree}]

(*Tubular Laser*)
```

```
LaserCyl[Position_, Degree_ : 0] :=
 Show[PLATE["Position" → Position + {0, 0, 0.5}, "Length" → 1,
   "Rotation" → {π / 2, Degree}, "Color" → Darker@Darker@Gray] ,
  PLATE["Position" → Position + {Cos[Degree], Sin[Degree], 0.5}, "Length" → 0.1,
   "Radius" → 0.4, "Rotation" → {π / 2, Degree}, "Color" → Darker@Darker@Gray]]

(*BoxLaser*)
LaserSquare[Position_, Degree_ : 0] := Show[PRISM["Position" → Position,
   "Rotation" → {π / 2, Degree}, "Color" → Darker[Lighter@Cyan, 0.8] ],
  PLATE["Position" → Position + 0.5 {Cos[Degree], Sin[Degree], 1}, "Color" →
    Darker@Darker@Cyan, "Radius" → 0.4, "Length" → 0.15, "Rotation" → {π / 2, Degree}]]

(*Beam Splitter*)
BS[Position_, Degree_ : 0] := Module[{a, b, c, d},
  a = (RotationMatrix[Degree, {0, 0, 1}].#) & /@
    {{0.5, -0.5, 1}, {0.5, -0.5, 0}, {-0.5, 0.5, 0}, {-0.5, 0.5, 1}};
  b = (# + Position) & /@ a;
  c = Graphics3D[{EdgeForm[], Gray, Opacity[0.3], Polygon[b]}];
  d = PRISM["Position" → Position, "Opacity" → 0.4,
    "Rotation" → {π / 2, Degree}, "Color" → Lighter@NiceBlue ];
  Return[Show[c, d]]]

(*Camera*)
CCD[Position_, Degree_ : 0] := Show[PRISM["Position" → Position,
   "Rotation" → {π / 2, Degree}, "Color" → Darker@Darker@Gray ],
  PLATE["Position" → Position + 0.5 {Cos[Degree], Sin[Degree], 1}, "Color" → Darker@Gray,
   "Radius" → 0.4, "Length" → 0.15, "Rotation" → {π / 2, Degree}]]

(*Lens*)
Lens[Position_, α_ : 0] := ParametricPlot3D[
  Position + {0, 0, 0.5} + {0.1 Cos[α] Cos[θ] Cos[ϕ] - 0.5 Cos[ϕ] Sin[α] Sin[θ],
    0.1 Cos[θ] Cos[ϕ] Sin[α] + 0.5 Cos[α] Cos[ϕ] Sin[θ], 0.5 Sin[ϕ]},
  {θ, 0, π}, {ϕ, 0, 2 π}, MaxRecursion → 1, Mesh → None, Boxed → False,
  PlotStyle → {Lighter@NiceBlue, Opacity[0.5]}, PlotPoints → 50, Lighting → "Neutral"]

(*QWP*)
QW[Position_, Degree_ : 0] := PLATE["Position" → Position + {0, 0, .5},
   "Rotation" → {π / 2, Degree}, "Color" → Darker@Green, "Opacity" → 0.7];

(*HWP*)
HW[Position_, Degree_ : 0] := PLATE["Position" → Position + {0, 0, .5},
   "Rotation" → {π / 2, Degree}, "Color" → Darker@Purple, "Opacity" → 0.7];

(*Iris*)
Iris[Position_, Degree_ : 0] :=
  Graphics3D[{Darker[Cyan, 0.8], EdgeForm[], Annulus3D[{Position + {0, 0, 0.5},
      Position + {-0.1 Sin[Degree], +0.1 Cos[Degree], 0.5}}, {0.05, 0.3}]}];
```

```
(*Phase Plate*)
PP[Position_, Degree_ : 0, Color_ : Orange] := PLATE["Position" → Position + {0, 0, .5},
   "Rotation" → {π / 2, Degree}, "Color" → Darker@Color, "Opacity" → 0.7];

(*Polarizer*)
Pol[Position_, α_ : 0, Degree_ : 0] := Module[{A, B, C},
  A =
   With[{ra = .03, arhd = .25, arrowtip = Graphics3D[{EdgeForm[], Darker@Darker@NiceBlue,
         Cone[{{0, 0, 0}, {0.15, 0, 0}}, 0.03]}]}, Graphics3D[
     {Darker@Darker@NiceBlue, Arrowheads[{{-arhd, 0, arrowtip}, {arhd, 1, arrowtip}}],
      Arrow[Tube[{0.15 {-Cos[α] Cos[Degree], -Cos[α] Sin[Degree], -Sin[α]} +
           {0, 0, 0.5} + Position, Position +
           0.15 {Cos[α] Cos[Degree], Cos[α] Sin[Degree], Sin[α]} + {0, 0, 0.5}}, ra]]}]];
  B = PRISM["Position" → Position, "Opacity" → 0.4, "Rotation" → {π / 2, Degree},
    "Color" → Darker@Darker@NiceBlue, "Dimensions" → {1, 0.2, 1}];
  Show[B, A, PlotRange → All, Boxed → None]
 ]

F1[Position_, F1_] := Graphics3D[
  {Opacity[0.3], Darker@Red, EdgeForm[], Cone[{Position, Position + {F1, 0, 0}}, 0.07]}]
Focusing[Position_, F1_, F1F2_] := Graphics3D[
  {{Opacity[0.3], Darker@Red, EdgeForm[], Cone[{Position, Position + {F1, 0, 0}}, 0.07]},
   {Opacity[0.3], Darker@Red, EdgeForm[],
    Cone[{Position + {F1F2, 0, 0}, Position + {F1, 0, 0}}, 0.07]}}]
```

OptoMechanics

In[30]:=
```
XStage[Position_] := Show[
   PRISM["Position" → Position + {0, 0, -0.3},
    "Dimensions" → {4, 2.5, 0.1}, "Color" → Gray],
   PRISM["Position" → Position + {0, 0, -0.1},
    "Dimensions" → {3.5, 0.7, 0.05}, "Color" → Darker@Gray],
   PRISM["Position" → Position + {0, 0, -0.05},
    "Dimensions" → {3.5, 0.5, 0.05}, "Color" → Darker@Gray],
   Graphics3D[{Lighter@Blue, Arrowheads[{-0.03, 0.03}],
     Arrow[Tube[{Position + {-1.5, 0.8, 0.3}, Position + {1.5, 0.8, 0.3}}, 0.03]]}]];

YStage[Position_] := Show[
   PRISM["Position" → Position + {0, 0, -0.3},
    "Dimensions" → {2.5, 4, 0.1}, "Color" → Gray],
   PRISM["Position" → Position + {0, 0, -0.1},
    "Dimensions" → {0.7, 3.5, 0.05}, "Color" → Darker@Gray],
   PRISM["Position" → Position + {0, 0, -0.05}, "Dimensions" → {0.5, 3.5, 0.05},
    "Color" → Darker@Gray], Graphics3D[{Lighter@Blue, Arrowheads[{-0.03, 0.03}],
     Arrow[Tube[{Position + {0.8, -1.5, 0.3}, Position + {0.8, 1.5, 0.3}}, 0.03]]}]];

ZStage[Position_] := Show[
   PRISM["Position" → Position + {0, 0, -0.3},
    "Dimensions" → {2.5, 4, 0.1}, "Color" → Gray],
   PRISM["Position" → Position + {0, 0, -0.1},
    "Dimensions" → {1.7, 1.7, 0.05}, "Color" → Darker@Gray],
   PRISM["Position" → Position + {0, 0, -0.05}, "Dimensions" → {1.5, 1.5, 0.05},
    "Color" → Darker@Gray], Graphics3D[{Lighter@Blue, Arrowheads[{-1, 1}],
     Arrow[Tube[{Position + {0.8, 1, 0}, Position + {0.8, 1, 1}}, 0.03]]}]];

RotStage[Position_] := Graphics3D[{{Darker@Gray, EdgeForm[],
     Cylinder[{Position + {0, 0, 0.25}, Position + {0, 0, 0.15}}, 0.4]},
    {Gray, EdgeForm[], Cylinder[{Position + {0, 0, 0.}, Position + {0, 0, 0.15}}, 0.6]}}];
```

Annulus3D from the Repository

In[34]:=
```
ClearAll[Annulus3D]
```

In[35]:=
```
Annulus3D::invpt = "Invalid endpoint specification: ``.";
Annulus3D::invrad = "Invalid radius specification: ``.";
Annulus3D::invang = "Invalid angle specification: ``.";
Annulus3D::zeroh = "Endpoints coincide.";
Annulus3D::inout = "The inner radius `` is larger than the outer radius ``.";
Annulus3D::degdir = "The normal vector `` must
    have non-zero magnitude. Default setting {0,0,1} used instead.";
```

In[41]:=
```
preprocessPoint[pt_] := If[TrueQ@Element[pt, Vectors[3, Reals]], N@pt,
   ResourceFunction["ResourceFunctionMessage"][Annulus3D::invpt, pt];
   $Failed];
preprocessRadius[rad_] := If[TrueQ@Element[rad, PositiveReals], N@rad,
   ResourceFunction["ResourceFunctionMessage"][Annulus3D::invrad, rad];
   $Failed];
preprocessAngles[θ1_, θ2_] := Module[{anginit = θ1, dang = θ2 - θ1},
   If[AnyTrue[{θ1, θ2}, NotElement[#, Reals] &],
    ResourceFunction["ResourceFunctionMessage"][Annulus3D::invang, {θ1, θ2}];
    Return[{$Failed, $Failed}]];
   If[dang > 2 π || dang < -2 π, dang = 2 π];
   If[dang < 0, dang += 2 π];
   {anginit, dang}
  ];
```

In[44]:=
```
annulus2D[rIn_, rOut_, k_, w_, coords_, z_, closed_ : False] :=
 BSplineSurface[Map[TranslationTransform[{0, 0, z}], {rIn coords, rOut coords}, {2}],
  SplineDegree → {1, 2}, SplineKnots → {{0, 0, 1, 1}, k},
  SplineWeights → {w, w}, SplineClosed → {False, closed}]
roundSurface[coords_, k_, w_, closed_ : False] :=
 BSplineSurface[coords, SplineDegree → {1, 2}, SplineKnots → {{0, 0, 1, 1}, k},
  SplineWeights → {w, w}, SplineClosed → {False, closed}]
```

In[46]:=
```
getSpecs[dang_] := Module[{k, w, coords, segment, rest},
  coords = {{1, 0, 0}, {1, 1, 0}, {0, 1, 0}, {-1, 1, 0},
    {-1, 0, 0}, {-1, -1, 0}, {0, -1, 0}, {1, -1, 0}, {1, 0, 0}};

  {segment, rest} = QuotientRemainder[dang, π / 2];
  If[segment ≥ 4, {segment, rest} = {3, π / 2}];
  coords = Join[coords⟦ ;; 2 segment + 1⟧, RotationTransform[segment π / 2, {0, 0, 1}] /@
     {{1, Tan[rest / 2], 0}, {Cos[rest], Sin[rest], 0}}];

  w = {1, 1/√2, 1, 1/√2, 1, 1/√2, 1, 1/√2, 1};
  w = Flatten[{w⟦ ;; 2 segment + 1⟧, Cos[rest / 2], 1}];
  k =
   Flatten[{0, 0, 0, Transpose[{Range[segment + 1], Range[segment + 1]}], segment + 1}];

  {k, w, coords}
 ]
```

In[47]:=
```
Annulus3D[] := Annulus3D[{{0, 0, -1}, {0, 0, 1}}, {1 / 2, 1}, {0, 2 π}]
Annulus3D[r_?NumericQ] := Annulus3D[{{0, 0, -1}, {0, 0, 1}}, {r / 2, r}, {0, 2 π}]
Annulus3D[r : {_?NumericQ, _?NumericQ}] := Annulus3D[{{0, 0, -1}, {0, 0, 1}}, r, {0, 2 π}]
Annulus3D[pts : {{__?NumericQ}, {__?NumericQ}}] := Annulus3D[pts, {1 / 2, 1}, {0, 2 π}]
Annulus3D[pts : {{__?NumericQ}, {__?NumericQ}}, r_?NumericQ] :=
 Annulus3D[pts, {r / 2, r}, {0, 2 π}]
Annulus3D[pts : {{__?NumericQ}, {__?NumericQ}}, r : {_?NumericQ, _?NumericQ}] :=
 Annulus3D[pts, r, {0, 2 π}]
```

In[53]:=
```
Annulus3D[{pt1i : {__?NumericQ}, pt2i : {__?NumericQ}},
   {rIni_?NumericQ, rOuti_?NumericQ}, {ang1i_?NumericQ, ang2i_?NumericQ}] :=
  Module[{pt1 = pt1i, pt2 = pt2i, rIn = rIni, rOut = rOuti, anginit, ann, baseL, baseH,
    circumI, circumO, faces, coords , height, k, w, segment, rest, dang, lastcc, l1, l2},

   (*preprocessing*)
   {pt1, pt2} = preprocessPoint /@ {pt1, pt2};
   {rIn, rOut} = preprocessRadius /@ {rIn, rOut};
   {anginit, dang} = preprocessAngles[ang1i, ang2i];
   If[! FreeQ[{pt1, pt2, rIn, rOut, anginit, dang}, $Failed], Return[{}]];

   height = Norm[pt2 - pt1];
   If[height == 0, ResourceFunction["ResourceFunctionMessage"][Annulus3D::zeroh];
    Return[{}, Module]];
   If[rOut < rIn,
    ResourceFunction["ResourceFunctionMessage"][Annulus3D::inout, rIn, rOut]];

   (*weights, knots, coordinates*)
   {k, w, coords} = getSpecs[dang];
   lastcc = Last[coords];
   {l1, l2} = coords[[-1, ;; 2]];

   (*net of annulus*)
   (*note: Reverse used to ensure proper normal orientation of all the faces*)
   baseL = annulus2D[rIn, rOut, k, Reverse@w, Reverse@coords, 0, dang == 2 π];
   baseH = annulus2D[rIn, rOut, k, w, coords, height, dang == 2 π];

   circumI = roundSurface[
     {rIn coords, TranslationTransform[{0, 0, height}][rIn coords]}, k, w, dang == 2 π];
   circumO = roundSurface[{rOut Reverse@coords, TranslationTransform[{0, 0, height}][
       rOut Reverse@coords]}, k, Reverse@w, dang == 2 π];

   faces = If[dang < 2 π, {
      Polygon[{{rIn, 0, 0}, {rOut, 0, 0}, {rOut, 0, height}, {rIn, 0, height}}],
      Polygon[{{rIn l1, rIn l2, height},
        {rOut l1, rOut l2, height}, {rOut l1, rOut l2, 0}, {rIn l1, rIn l2, 0}}]
     }];

   (*annulus*)
   ann = {baseL, baseH, circumI, circumO};
   If[dang < 2 π, ann = Join[ann, faces]];
   MapAt[TranslationTransform[pt1]@*RotationTransform[{{0, 0, 1}, pt2 - pt1}]@*
     RotationTransform[anginit, {0, 0, 1}], ann, {All, 1, All}]
];
```

In[54]:=
```
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ}] :=
 Annulus3D[pt, {0, 0, 1}, {1 / 2, 1}, {0, 2 π}]
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ}, r_?NumericQ] :=
 Annulus3D[pt, {0, 0, 1}, {r / 2, r}, {0, 2 π}]
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ}, r : {_?NumericQ, _?NumericQ}] :=
 Annulus3D[pt, {0, 0, 1}, r, {0, 2 π}]
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ},
  norm : {_?NumericQ, _?NumericQ, _?NumericQ}] :=
 Annulus3D[pt, norm, {1 / 2, 1}, {0, 2 π}]
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ},
  norm : {_?NumericQ, _?NumericQ, _?NumericQ}, r_?NumericQ] :=
 Annulus3D[pt, norm, {r / 2, r}, {0, 2 π}]
Annulus3D[pt : {_?NumericQ, _?NumericQ, _?NumericQ},
  norm : {_?NumericQ, _?NumericQ, _?NumericQ},
  r : {_?NumericQ, _?NumericQ}] := Annulus3D[pt, norm, r, {0, 2 π}]
```

In[60]:=
```
Annulus3D[pti : {_?NumericQ, _?NumericQ, _?NumericQ},
   normali : {_?NumericQ, _?NumericQ, _?NumericQ},
   {rIni_?NumericQ, rOuti_?NumericQ}, {ang1i_?NumericQ, ang2i_?NumericQ}] := Module[
   {pt = pti, normal = normali, rIn = rIni, rOut = rOuti, anginit, coords , k, w, dang, ann},

   (*preprocessing*)
   {pt, normal} = preprocessPoint /@ {pt, normal};
   {rIn, rOut} = preprocessRadius /@ {rIn, rOut};
   {anginit, dang} = preprocessAngles[ang1i, ang2i];
   If[! FreeQ[{pt, normal, rIn, rOut, anginit, dang}, $Failed], Return[{}, Module]];
   If[rOut < rIn,
    ResourceFunction["ResourceFunctionMessage"][Annulus3D::inout, rIn, rOut]];
   If[normal == {0., 0., 0.},
    ResourceFunction["ResourceFunctionMessage"][Annulus3D::degdir, normal];
    normal = {0, 0, 1},
    normal = Normalize[normal];
   ];

   (*weights, knots, coordinates*)
   {k, w, coords} = getSpecs[dang];

   (*annulus*)
   Switch[normal,
    {0., 0., 1.},
    Null,
    {0., 0., -1.},
    {coords, w} = Reverse /@ {coords, w},
    _,
    coords = RotationTransform[{{0, 0, 1}, normal}][coords]
   ];
   ann = annulus2D[rIn, rOut, k, w, coords, 0, dang == 2 π];
   MapAt[
    TranslationTransform[pt] @* RotationTransform[anginit, {0, 0, 1}], ann, {1, All}]
];
```