DeltaRobot geometry

Willy Svenningsson 2010-09-23
willy.svenningsson@f ager .st

Engine part

We start with some basics.
An engine iwill be located at origo with it's turning axes along the x-axes.
On the engine axis there is a rod attatched so it can sweep in the y-z plane.
We will later on deal with coordination the rod angle of three such engines in order to be a delta robot

```
theMotorAxesLength = 1.5;
theMotorAxesDiameter = 0.2;
theMotorWidthAndHeigth = theMotorAxesDiameter * 3.0;
theMotorOffset = .4;
theRodWidth = theMotorOffset / 2;
theRodHeight = 2 * theRodWidth;
theRodLength = theRodHeight * 5;
theRodOffset = theMotorAxesDiameter * 1.3;

theMotor = Cuboid[{-theMotorAxesLength,
     -theMotorWidthAndHeigth / 2, -theMotorWidthAndHeigth / 2},
    {-theMotorOffset, theMotorWidthAndHeigth / 2, theMotorWidthAndHeigth / 2}];
theMotorAxes = Cylinder[{{-theMotorAxesLength, 0, 0}, {.2, 0, 0}},
    theMotorAxesDiameter / 2];
theRod = Cuboid[{-theRodWidth / 2, -theRodOffset, -theRodHeight / 2},
    {theRodWidth / 2, theRodLength - theRodOffset, theRodHeight / 2}];
theTurnRod = Rotate[theRod, 30 Degree, {1, 0, 0}];
theRodSweepCircle = Line@Table[{0, (theRodLength - theRodOffset) Cos[alpha],
      (theRodLength - theRodOffset) Sin[alpha]}, {alpha, 0, 2 Pi, Pi / 16}];
```

We assign an anchor point (APE) at the end of the engine rod

```
theApeCoords =
  {0, theRodLength - theRodOffset, 0}.RotationMatrix[-30 Degree, {1, 0, 0}];
theAPE = {
    (*PointSize[.14],
    Point[theApeCoords],*)
    Sphere[theApeCoords, theRodWidth],
    Text["APE", theApeCoords, {-2, 2}]};
```

Composeone bearing

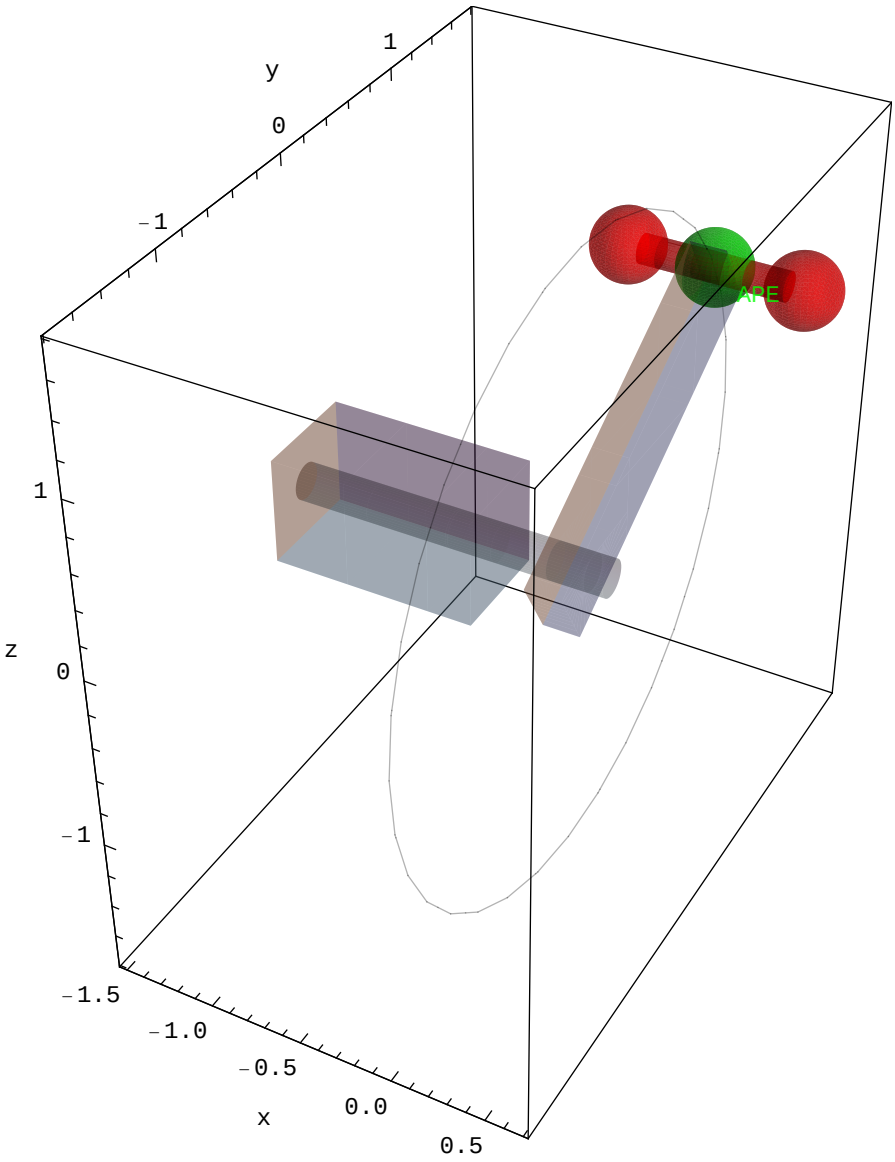
```
theRadillLength = 4 * theRodWidth;
theRadillRadi = theRodHeight / 5;
theRadillBearing = Cylinder[
    {{-theRadillLength / 2, 0, 0}, {theRadillLength / 2, 0, 0}}, theRadillRadi];

theSphericalBearing = Sphere[{0, 0, 0}, theRodWidth];

oneBearing = {theRadillBearing,
    Translate[theSphericalBearing, {-(theRadillLength / 2 + theRadillRadi), 0, 0}],
    Translate[theSphericalBearing, {+(theRadillLength / 2 + theRadillRadi), 0, 0}]
   };

bearingZero = Translate[oneBearing, theApeCoords];
```

```
Graphics3D[{Opacity[0.3]
  , theMotor
  , theMotorAxes
  , theTurnRod
  , theRodSweepCircle
  , Opacity[0.8]
  , Green
  , theAPE
  , Red
  , bearingZero
 }
 , Axes -> True, AxesLabel -> {"x", "y", "z"}]
```



Movable tool

The operated tool of this robot has desired position of {x,y,z}.
In this point we construct a triangle with equal sides.
Each corner is connected to the end of the above rod.
The very innovation in the delta robot is that this connection is built by two fixed length rods
which are kept in parallell using one radil bearing and two spherical bearings at each end.
But this double rod have no impact on the mathematical solution actually!

We look into this tool fixtur

```
theTriangleRadi = 1.5;
theTriangleCorner1 = {0, -theTriangleRadi, 0};
theTriangleCorner1Y = theTriangleCorner1[[2]];
rot1 = RotationMatrix[2 Pi / 3, {0, 0, 1}];
rot2 = RotationMatrix[2 Pi / 3 * 2, {0, 0, 1}];
theTringleCorners =
  {theTriangleCorner1, theTriangleCorner1.rot1, theTriangleCorner1.rot2};
```

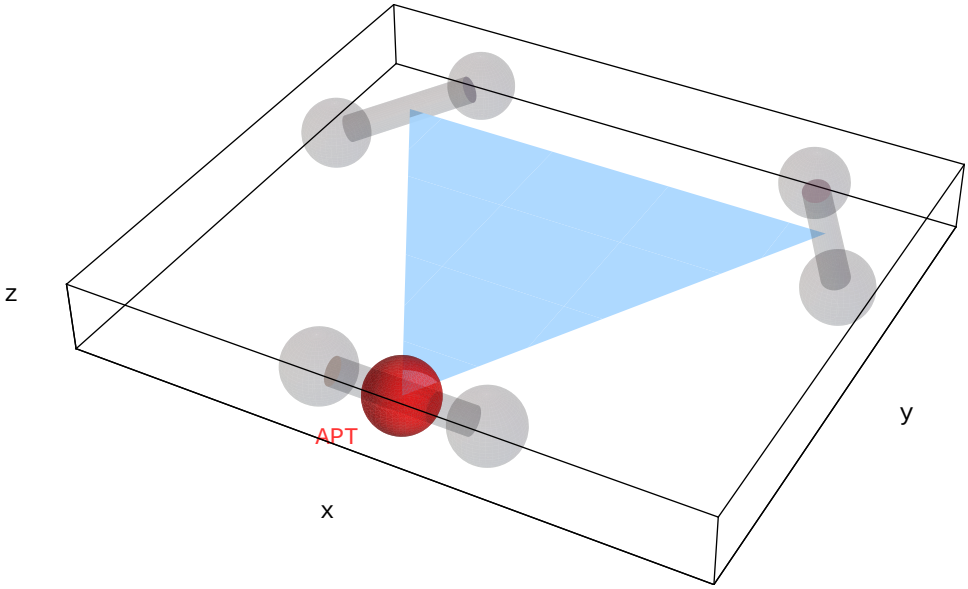
Tripple the tool bearing

```
bearing1 = Translate[oneBearing, theTriangleCorner1];
bearing23 =
  {Rotate[bearing1, 2 Pi / 3, {0, 0, 1}], Rotate[bearing1, 2 * 2 Pi / 3, {0, 0, 1}]};
```

Note the anchorpoint in the tool (APT) for the rod to the counterpart anchorpoint (APE) on the engine swing rod.

```
theAPT = {
   PointSize[.15],
   Sphere[theTriangleCorner1, theRodWidth],
   Text["APT", theTriangleCorner1, {3, 3}]
  };
```

```
Graphics3D[{Opacity[0.9]
   , Polygon@theTringleCorners
   , Opacity[0.2]
   , bearing1
   , bearing23
   , Opacity[0.8]
   , Red
   , theAPT
  }
  , Axes -> True, Ticks -> None, PlotRange -> Automatic, AxesLabel -> {"x", "y", "z"}]
```



Geometry

■ Functions

```
Clear@reduceDimension;
reduceDimension[direction2_, {x_, y_, z_}] := {x, y, z}[[direction2]]
reduceDimension[direction2_, l__] :=
 Map[(#[[direction2]]) &, l, {2}] /; Head[l] == Polygon
reduceDimension[direction2_, l__] :=
 {Opacity[1.0], Line@Table[l[[1]][[direction2]] + {l[[2]] Cos[t], l[[2]] Sin[t]},
     {t, 0, 2 Pi, Pi / 16}]} /; Head[l] == Sphere
reduceDimension[direction2_, l__] := Map[(reduceDimension[direction2, #]) &, l]
```

```
turnAPoint[thePoint_, inPoint_, vridAxel_, alfa_] :=
 Module[{vin, vut},
  vin = thePoint - inPoint;
  vut = inPoint + vin Cos[alfa] + Cross[vridAxel, vin] Sin[alfa];
  vut
 ]
```

```
PointStamp[p_, name_] := PointStamp[p, name, {-1, -1}];
PointStamp[p_, name_, dir_] := {Point[p], Text[name, p, dir]};
```

Solution

We have tool hub at desired 'tTool' {x,y,z} expressed in a coordinatsystem common to all three engines.
The hub is a equal sided (equilateral) triangle parallell to to x-y plane and with circum-circle radi of 'toolRadi'.
At each corner of the triangle the lower rod of length 'lowRodLength' is attached through a universal joint at 'jointTriangleLowerRod'.
The upper rod of length 'uppRodLength' is stiff attached to a rotating engine axis at 'engineAxisJoint'.
One axis has the direction 'engineAxisDirection' {xAxisJoint, yAxisJoint, 0} and the other two are rotated +- 120 Degrees.

The goal is find the angle 'rotAngle' for the rotating engine axis. For all three engines.

A calculation is worked through by first move the desired point so that the engine AxisJoint is at at {0,0,0}
and the direction is along the x-axis {1,0,0}

First illustration only

```
tTool := {x, y, z};
toolRadi := .5;
jointTriangleLowerRod := tTool + {0, -toolRadi, 0};
engineAxisJoint := {0, 0, 0};
engineAxisDirection := {1, 0, 0};
lowRodLength := .8;
uppRodLength := .9;
```

```
subsXYZ = {x -> .5, y -> 1.2, z -> -.9};
```

Drawing for tool

```
grTool = {Point[tTool], PointStamp[tTool, "tTool"]};
grjointTriangleLowerRod = {Point[jointTriangleLowerRod],
   PointStamp[jointTriangleLowerRod, "jointTriangleLowerRod", {1, 1}]};
grTriangle = Line[{jointTriangleLowerRod,
    turnAPoint[jointTriangleLowerRod, tTool, {0, 0, 1}, 120 Degree],
    turnAPoint[jointTriangleLowerRod, tTool, {0, 0, 1}, -120 Degree],
    jointTriangleLowerRod}];
```

Drawing for engine

```
grEngineAxisJoint = {Point[engineAxisJoint],
   PointStamp[engineAxisJoint, "engineAxisJoint", {1, 1}]};
upRodEnd = engineAxisJoint + {0, uppRodLength, 0};
grUppRod = {Point[upRodEnd], PointStamp[upRodEnd, "upRodEnd"],
   Line[{engineAxisJoint, upRodEnd}]};
```

Drawing for lower rod

```
grLowerRod =
  {Red, PointSize[.04], Point[jointTriangleLowerRod], Point[upRodEnd],
   Line[{jointTriangleLowerRod, upRodEnd}], Black};
```

Sweeping areas. The upper rod will sweep a circle in the y-z-plane and the lower will span a sphere at 'grjointTriangleLowerRod' with radi 'lowRodLength'.

This sphere will intesect with the y-z-plane at {0,isY,isZ} the forming a circle in the plane.
At the intesection of this circles we find two points (on the upper rod) (or one or none!).
The outermost is choosen as the point where the end of upper rod should be turned t o!
In this first drawing that of 'upRodEnd' turn is not shown! Look at the next section for that action.

```
grUpperRodSweep =
  Line[Table[engineAxisJoint + {0, upRodEnd[[2]] * Cos[v], upRodEnd[[2]] * Sin[v]},
    {v, 0, 2 Pi, Pi / 12}]];
grLowerRodSweep = {Opacity[.1],
   Sphere[jointTriangleLowerRod, lowRodLength], Opacity[1.0]};
```

```
grIntersectionSphereAndYZ =
  Module[{r, x, y, z},
   r = Sqrt[lowRodLength^2 - jointTriangleLowerRod[[1]]^2];
   {x, y, z} = jointTriangleLowerRod * {0, 1, 1};
   Line@Table[{0, y + r Cos@t, z + r Sin@t}, {t, 0, 2 Pi, Pi / 16}]
  ];
```

We have two these two circles in the y-z/plane. The upper rod at {0,0,0} and radi 'uppRodLength' and
the circle from the sphere intersecting the plane at center {0, jointTriangleLower - Rod(y), z}
and radi $\sqrt{\text{lowRodLength}^2 - (x)^2}$.

Here is one set of equations using some set of all knowledge
(here the distance from center of the two circles to the intersection points) .

```
equations = {
    isY^2 + isZ^2 == uppRodLength^2,
    (jointTriangleLowerRod[[2]] - isY)^2 + (z - isZ)^2 == (Sqrt[lowRodLength^2 - x^2])^2
   };
variables = {isY, isZ};
sol = Solve[equations, variables] /. subsXYZ // FullSimplify
```

```
{{isZ → -0.293942, isY → 0.850646}, {isZ → -0.896827, isY → 0.0755083}}
```

```
{{y1, z1}, {y2, z2}} = variables /. sol;
```

Now we can illustrate where the intersection points actually are

```
grInterSectionPoints = {Point[{0, y1, z1}], Point[{0, y2, z2}]};
```

```
glgr = {grTool
     , grjointTriangleLowerRod
     , grTriangle
     , grEngineAxisJoint
     , grUppRod
     , Arrow[{{0, 0, 0}, -engineAxisDirection}]
     , grLowerRod
     , Blue
     , grUpperRodSweep
     , grLowerRodSweep
     , grIntersectionSphereAndYZ
     , grInterSectionPoints
    } /. subsXYZ;
```

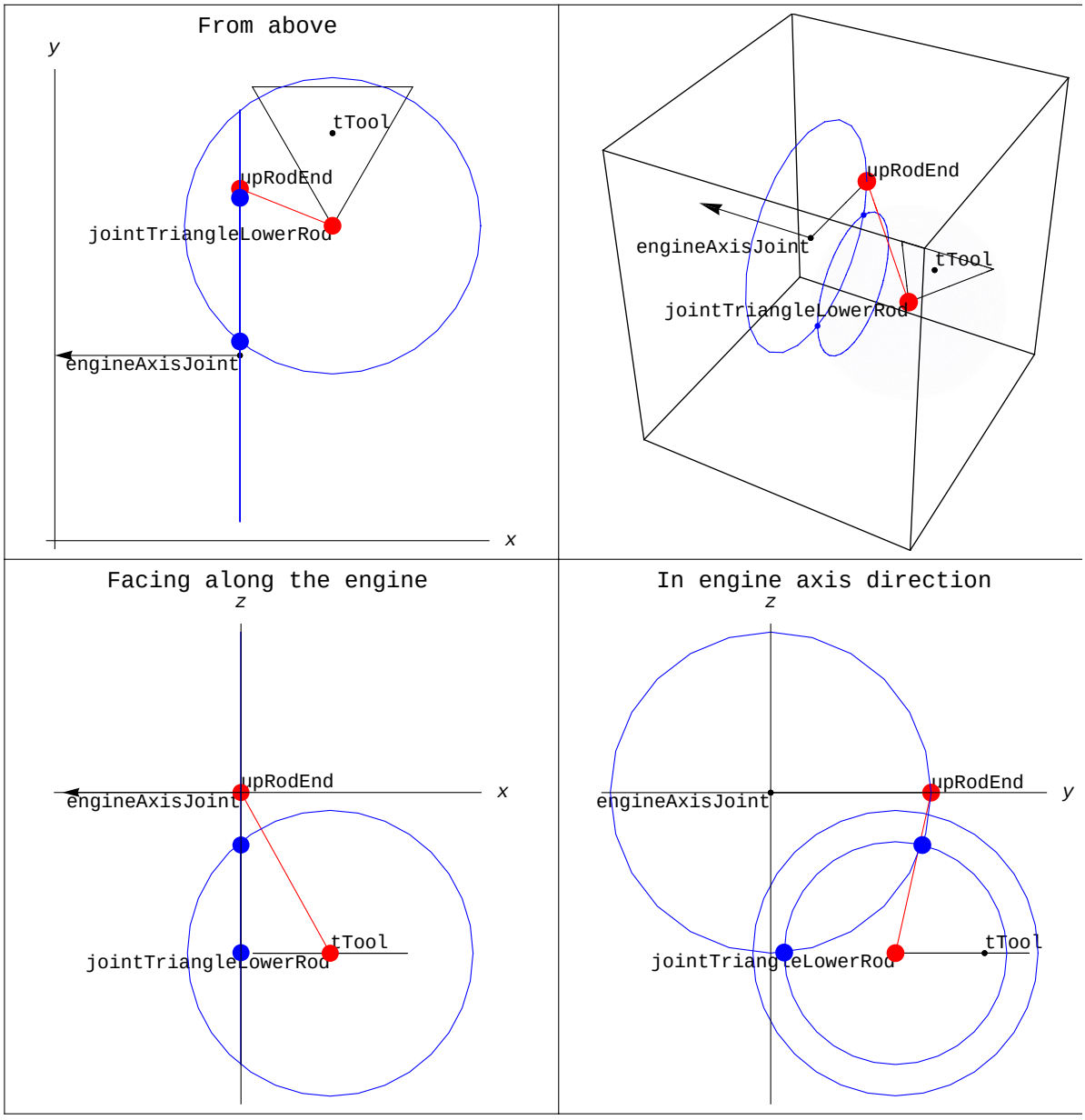
```
grz = Show@Graphics[{reduceDimension[{1, 2}, Flatten@glgr]},
     Axes -> True, Ticks -> None, AxesOrigin -> {-1, -1},
     AxesLabel → {x, y}, PlotLabel → "From above"];

gry = Show@Graphics[{reduceDimension[{1, 3}, Flatten@glgr]},
     Axes -> True, Ticks -> None, AxesOrigin -> {0, 0},
     AxesLabel → {x, z}, PlotLabel → "Facing along the engine"];

grx = Show@Graphics[{reduceDimension[{2, 3}, Flatten@glgr]},
     Axes -> True, Ticks -> None, AxesOrigin -> {0, 0},
     AxesLabel → {y, z}, PlotLabel → "In engine axis direction"];

grXYZ = Show@Graphics3D[{Opacity[1], glgr}];
```

```
GraphicsArray[{{grz, grXYZ}, {gry, grx}}, Frame → All, GraphicsSpacing -> 0.05]
```



■ and formulas

```
interSect2CircZero[r0_, {x1_, y1_}, r1_] :=
 Module[{d, a, h, p2x, p2y, x31x, y31y, x32x, y32y},
  d = Sqrt[x1^2 + y1^2];
  If[d > r0 + r1, Return[{d, "Separated", {0, 0}, {0, 0}}]];
  If[d < Abs[r0 - r1], Return[{d, "Contained", {0, 0}, {0, 0}}]];
  If[d == 0 && r0 == r1, Return[{d, "Same", {0, 0}, {0, 0}}]];
  a = (r0^2 - r1^2 + d^2) / (2 d);
  {p2x, p2y} = a * {x1, y1} / d;
  h = Sqrt[r0^2 - a^2];
  {x31x, y31y} = {p2x + h * y1 / d, p2y - h * x1 / d};
  {x32x, y32y} = {p2x - h * y1 / d, p2y + h * x1 / d};
  {{x31x, y31y}, {x32x, y32y}}
 ]
```

Tool is at {x,y,z}

```
tTool = {x, y, z};
x = .85; y = 1.0; z = -1.7; radiTool = .5;
jobRadi = 0; lenUpperRod = .9; lenLowerRod = 1.8;
```

```
subsFormulas = {x -> .85, y -> 1.0, z -> -1.7,
   radiTool -> .5, jobRadi -> 0, lenUpperRod -> .9, lenLowerRod -> 1.8}
```

```
{x → 0.85, y → 1., z → -1.7, radiTool → 0.5,
 jobRadi → 0, lenUpperRod → 0.9, lenLowerRod → 1.8}
```

Radi for tool join is radiTool

Join is at p1

```
{p1X, p1Y, p1Z} = tTool + {0, -radiTool, 0};
p1 = {p1X, p1Y, p1Z};
```

The engine is at {'jobRadi',0,0}. Move all so engine axis is at {0,0,0}

```
transform = TranslationTransform[{-jobRadi, 0, 0}];
{p2X, p2Y, p2Z} = transform[{p1X, p1Y, p1Z}];
p2 = {p2X, p2Y, p2Z};
```

P3 is p2 but in the y-z plane

```
{p3X, p3Y, p3Z} = p2;
p3X = 0;
p3 = {p3X, p3Y, p3Z};
```

Find the intersections of the two circles. Above were one approach with runtime solution of an equatios system.
Now use a tailor made function

```
radiSnittCircle = Sqrt[lenLowerRod^2 - p2X^2];
radiUpperRod = lenUpperRod;
```

P11and P12 is a solution. Choose the outermost

```
(*{{py11,pz11},{py12,pz12}}=
  interSect2CircZero[radiUpperRod,{p3Y,p3Z},radiSnittCircle];*)

{{py11, pz11}, {py12, pz12}} =
  solZeroFunction[radiUpperRod, p3Y, p3Z, radiSnittCircle];

(*solOneFunction[radiUpperRod,p3Y,p3Z,radiSnittCircle];*)
(*solZeroBlock/.{r1->3}//ReleaseHold*)
p11 = {0, py11, pz11};
p12 = {0, py12, pz12};
{pEndx, pEndy, pEndz} = p12;
pEnd = {pEndx, pEndy, pEndz};
```

...and we have the two angles. Make a suitabale solution

```
alpha = -ArcTan[pEndy, pEndz];
```

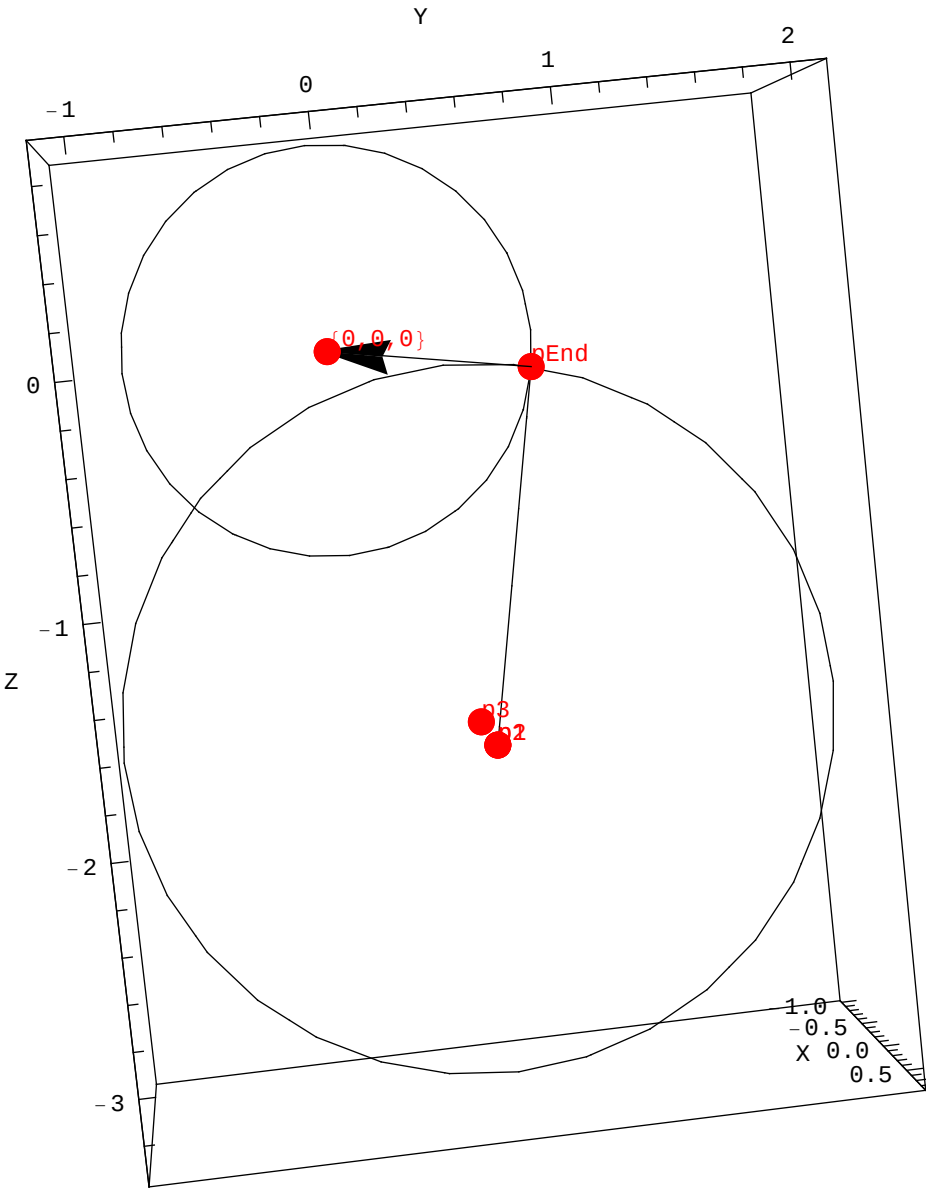
DONE. We have the required angle which the upper rod should bu turned on the engine axis

Now we got calculated value for the engine axis

```
grCircCentrum =
  Line@Table[{0, radiUpperRod Cos@t, radiUpperRod Sin@t}, {t, 0, 2 Pi, Pi / 16}];
grSnittCircle = Line@Table[{0, p3Y, p3Z} +
     {0, radiSnittCircle Cos@t, radiSnittCircle Sin@t}, {t, 0, 2 Pi, Pi / 16}];
```

```
grAll = {
   Thickness[Small],
   Red, PointSize[0.03],
   Point[p1], PointStamp[p2, "p1"],
   Point[p2], PointStamp[p2, "p2"],
   Point[p3], PointStamp[p3, "p3"],
   Point[pEnd], PointStamp[pEnd, "pEnd"],
   Point[{0, 0, 0}], PointStamp[{0, 0, 0}, "{0,0,0}"],
   Black,
   Line[{p2, pEnd}], Line[{{0, 0, 0}, pEnd}],
   {Thickness[Medium], Arrowheads[Large],
    Arrow[{{0, 0, 0}, {-1, 0, 0}}], Thickness[Small]},
   grCircCentrum,
   grSnittCircle
  };
```

```
Show@Graphics3D[
    {
     grAll
    },
    Axes -> True, AxesLabel -> {"X", "Y", "Z"},
    Ticks -> Automatic, PlotRange -> Automatic
   ] /. subsFormulas
```




```
subsFormulas
```

{x → 0.85, y → 1., z → −1.7, 0.5 → 0.5, 0 → 0, 0.9 → 0.9, 1.8 → 1.8}

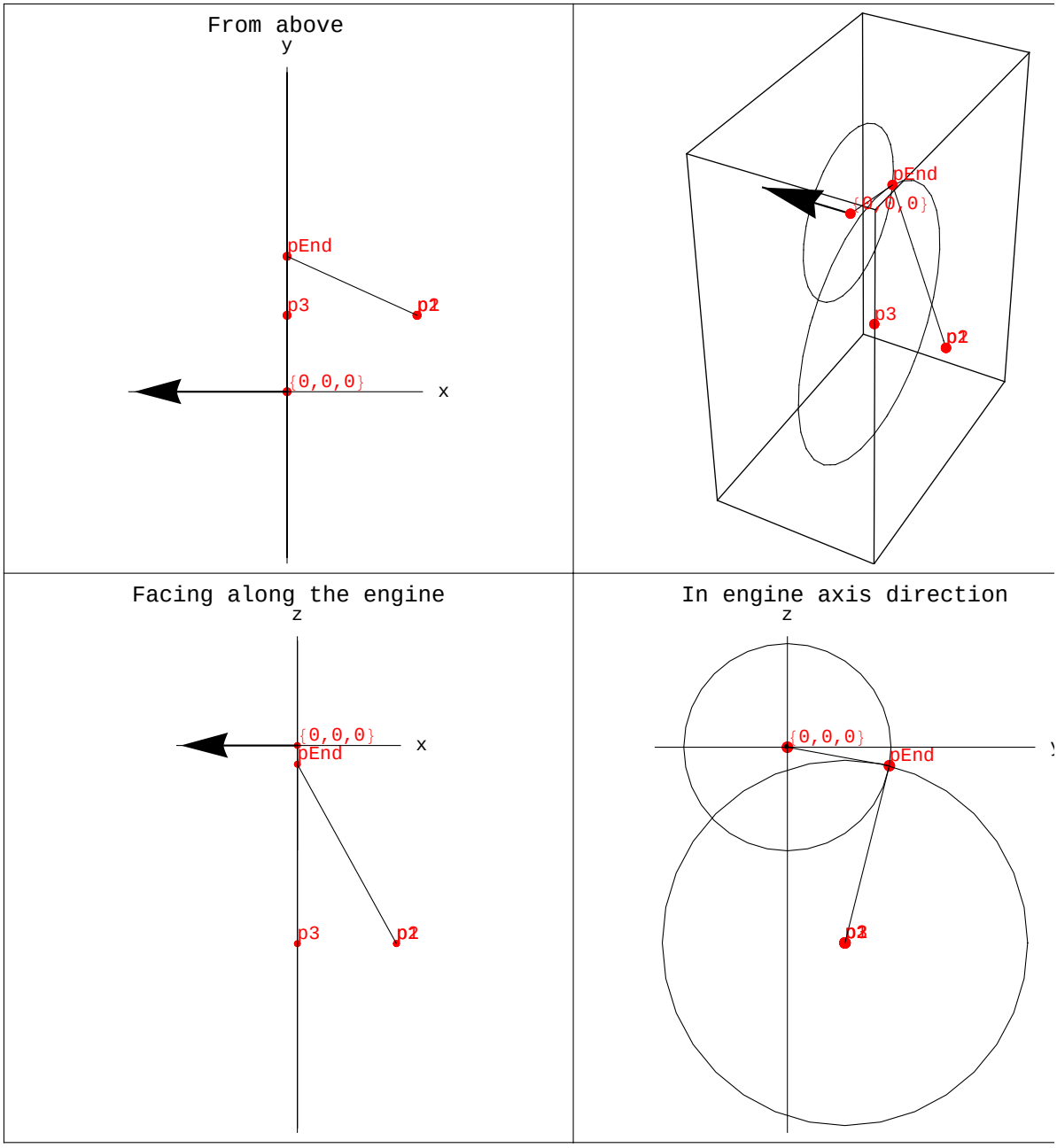
```
grz = Show@Graphics[{reduceDimension[{1, 2}, Flatten@grAll]},
      Axes -> True, Ticks -> None, AxesOrigin -> {0, 0},
      AxesLabel → {"x", "y"}, PlotLabel → "From above"]; /. subsFormulas

gry = Show@Graphics[{reduceDimension[{1, 3}, Flatten@grAll]},
      Axes -> True, Ticks -> None, AxesOrigin -> {0, 0}, AxesLabel → {"x", "z"},
      PlotLabel → "Facing along the engine"]; /. subsFormulas

grx = Show@Graphics[{reduceDimension[{2, 3}, Flatten@grAll]},
      Axes -> True, Ticks -> None, AxesOrigin -> {0, 0}, AxesLabel → {"y", "z"},
      PlotLabel → "In engine axis direction"]; /. subsFormulas

grXYZ = Show@Graphics3D[{Opacity[1], grAll}]; /. subsFormulas
```

```
GraphicsArray[{{grz, grXYZ}, {gry, grx}},
  Frame → All, GraphicsSpacing -> 0.05] /. subsFormulas
```



■ Optimize

```
(* EDIT FOR PROPER PATH *)
Needs["Optimize`", "/Users/willy/Matte/Optimize-1.m"];
```

The used function for intersection is a reduction from

- **http : // local.wasp.uwa.edu.au/~pbourke/geometry/2 circle/**

We have
1. a^2 + h^2 == r0^2
2. b^2 + h^2 == r1^2
3. d == a + b
From 2. we get
4. h^2 == r1^2-b^2
Substitute h^2 in 1
5. a^2 + r1^2-b^2 == r0^2
Replacing b in 5 with d - a from 3 gives
a^2 + r1^2-(d-a)^2 == r0^2
Expand this and you get
6. 2 a d-d^2-r0^2+r1^2==0
and then
Calculations:

a = (d^2+r0^2-r1^2)/(2 d) (from 6.)

h = Sqrt[r0^2-a^2] (from 1.)

P2 = P0 + a (P1 - P0) / d
x3 = x2 +- h (y1 - y0) / d
y3 = y2 -+ h (x1 - x0) / d

```
interSect2Circ[{x0_, y0_}, r0_, {x1_, y1_}, r1_] :=
 Module[{d, a, h, p2x, p2y, x31x, y31y, x32x, y32y},
  d = Sqrt[(x1 - x0)^2 + (y1 - y0)^2];
  If[d > r0 + r1, Return[{d, "Separated", {0, 0}, {0, 0}}]];
  If[d < Abs[r0 - r1], Return[{d, "Contained", {0, 0}, {0, 0}}]];
  If[d == 0 && r0 == r1, Return[{d, "Same", {0, 0}, {0, 0}}]];
  a = (r0^2 - r1^2 + d^2) / (2 d);
  {p2x, p2y} = {x0, y0} + a ({x1, y1} - {x0, y0}) / d;
  h = Sqrt[r0^2 - a^2];
  {x31x, y31y} = {p2x + h (y1 - y0) / d, p2y - h (x1 - x0) / d};
  {x32x, y32y} = {p2x - h (y1 - y0) / d, p2y + h (x1 - x0) / d};
  {d, {x31x, y31y}, {x32x, y32y}}
 ]
```

We reduce the x0,y0 as we always calucate with one of the circles in origo.

Lets try to optimez the evaluation of interSect2CircZero.
First evaluate with symbols

```
solZero = interSect2CircZero[r1, {y, z}, r2]
```

$$\left\{\left\{\frac{y\left(r1^2-r2^2+y^2+z^2\right)}{2\left(y^2+z^2\right)}+\frac{z\sqrt{r1^2-\frac{\left(r1^2-r2^2+y^2+z^2\right)^2}{4\left(y^2+z^2\right)}}}{\sqrt{y^2+z^2}},\right.\right.$$
$$\left.\frac{z\left(r1^2-r2^2+y^2+z^2\right)}{2\left(y^2+z^2\right)}-\frac{y\sqrt{r1^2-\frac{\left(r1^2-r2^2+y^2+z^2\right)^2}{4\left(y^2+z^2\right)}}}{\sqrt{y^2+z^2}}\right\},$$
$$\left\{\frac{y\left(r1^2-r2^2+y^2+z^2\right)}{2\left(y^2+z^2\right)}-\frac{z\sqrt{r1^2-\frac{\left(r1^2-r2^2+y^2+z^2\right)^2}{4\left(y^2+z^2\right)}}}{\sqrt{y^2+z^2}},\right.$$
$$\left.\left.\frac{z\left(r1^2-r2^2+y^2+z^2\right)}{2\left(y^2+z^2\right)}+\frac{y\sqrt{r1^2-\frac{\left(r1^2-r2^2+y^2+z^2\right)^2}{4\left(y^2+z^2\right)}}}{\sqrt{y^2+z^2}}\right\}\right\}$$

```
solZeroFunction = Compile[{r1, y, z, r2}, Optimize[solZero]]
```

```
CompiledFunction[{r1, y, z, r2},
 Block[{O$592, O$593, O$594, O$595, O$596, O$597, O$598, O$599, O$600,
   O$601, O$602, O$603, O$604, O$605, O$606}, O$592 = r1^2; O$593 = r2^2;
  O$594 = y^2; O$595 = z^2; O$596 = -O$593; O$597 = O$594 + O$595; O$598 = 1/O$597;
  O$599 = 1/Sqrt[O$597]; O$600 = O$592 - O$593 + O$597; O$601 = y O$598 O$600/2;
  O$602 = z O$598 O$600/2; O$603 = O$600^2; O$604 = -O$598 O$603/4; O$605 = O$592 + O$604;
  O$606 = Sqrt[O$605]; {{O$601 + O$599 O$606 z, O$602 - O$599 O$606 y},
   {O$601 - O$599 O$606 z, O$602 + O$599 O$606 y}}], -CompiledCode-]
```

- **The solution in the first part**

```
Clear[r0, r1, y, z, isY, isZ]
```

```
equations = {
    isY^2 + isZ^2 == r0^2,
    (y - isY)^2 + (z - isZ)^2 == r1^2
   };
variables = {isY, isZ};
solOne = Solve[equations, variables] // FullSimplify;
```

```
{{yi1, zi1}, {yi2, zi2}} = variables /. solOne
```

$$\left\{\left\{\frac{r0^2\,y^2+y^2\left(-r1^2+y^2+z^2\right)-z\sqrt{-y^2\left(r0^4+\left(-r1^2+y^2+z^2\right)^2-2\,r0^2\left(r1^2+y^2+z^2\right)\right)}}{2\,y\left(y^2+z^2\right)},\right.\right.$$
$$\left.\frac{r0^2\,z-r1^2\,z+y^2\,z+z^3+\sqrt{-y^2\left(r0^4+\left(-r1^2+y^2+z^2\right)^2-2\,r0^2\left(r1^2+y^2+z^2\right)\right)}}{2\left(y^2+z^2\right)}\right\},$$
$$\left\{\frac{r0^2\,y^2+y^2\left(-r1^2+y^2+z^2\right)+z\sqrt{-y^2\left(r0^4+\left(-r1^2+y^2+z^2\right)^2-2\,r0^2\left(r1^2+y^2+z^2\right)\right)}}{2\,y\left(y^2+z^2\right)},\right.$$
$$\left.\left.\frac{r0^2\,z-r1^2\,z+y^2\,z+z^3-\sqrt{-y^2\left(r0^4+\left(-r1^2+y^2+z^2\right)^2-2\,r0^2\left(r1^2+y^2+z^2\right)\right)}}{2\left(y^2+z^2\right)}\right\}\right\}$$

```
solOneFunction = Compile[{r0, y, z, r1}, Optimize[{{yi1, zi1}, {yi2, zi2}}]]
```

```
CompiledFunction[{r0, y, z, r1},
 Block[{O$592, O$593, O$594, O$595, O$596, O$597, O$598, O$599, O$600, O$601, O$602,
   O$603, O$604, O$605, O$606, O$607, O$608, O$609, O$610, O$611, O$612, O$613},
  O$592 = r0^2; O$593 = r0^4; O$594 = r1^2; O$595 = 1/y; O$596 = y^2; O$597 = z^2;
  O$598 = z^3; O$599 = -O$594; O$600 = O$592 O$596; O$601 = O$592 z; O$602 = -O$594 z;
  O$603 = O$596 z; O$604 = O$596 + O$597; O$605 = O$594 + O$596 + O$597;
  O$606 = 1/O$604; O$607 = -O$594 + O$604; O$608 = -2 O$592 O$605; O$609 = O$596 O$607;
  O$610 = O$607^2; O$611 = O$593 + O$610 + O$608; O$612 = -O$596 O$611;
  O$613 = Sqrt[O$612]; {{1/2 O$595 O$606 (O$600 + O$609 - O$613 z), 1/2 O$606
     (O$598 + O$601 + O$602 + O$603 + O$613)}, {1/2 O$595 O$606 (O$600 + O$609 + O$613 z),
    1/2 O$606 (O$598 + O$601 + O$602 + O$603 - O$613)}}], -CompiledCode-]
```

■ Comparasions

```
solZeroFunction[radiUpperRod, p3Y, p3Z, radiSnittCircle] // Timing
```

```
{0.000046, {{-0.657399, -0.614676}, {0.885504, -0.160881}}}
```

```
solOneFunction[radiUpperRod, p3Y, p3Z, radiSnittCircle] // Timing
```

```
{0.000059, {{0.885504, -0.160881}, {-0.657399, -0.614676}}}
```

```
Cost[solZeroFunction]
```

```
4 Blank + Block + CompiledFunction + Function + 56 List + 7 Plus + 8 Power + 15 Set + 9 Times
```

```
Cost[solOneFunction]
```

```
4 Blank + Block + CompiledFunction + Function + 71 List + 8 Plus + 10 Power + 22 Set + 16 Times
```