In [1]:

```
%reload_ext autoreload
%autoreload 2
from openscad3 import *
```

Basic of Drawing and 3D modeling with library openscad3

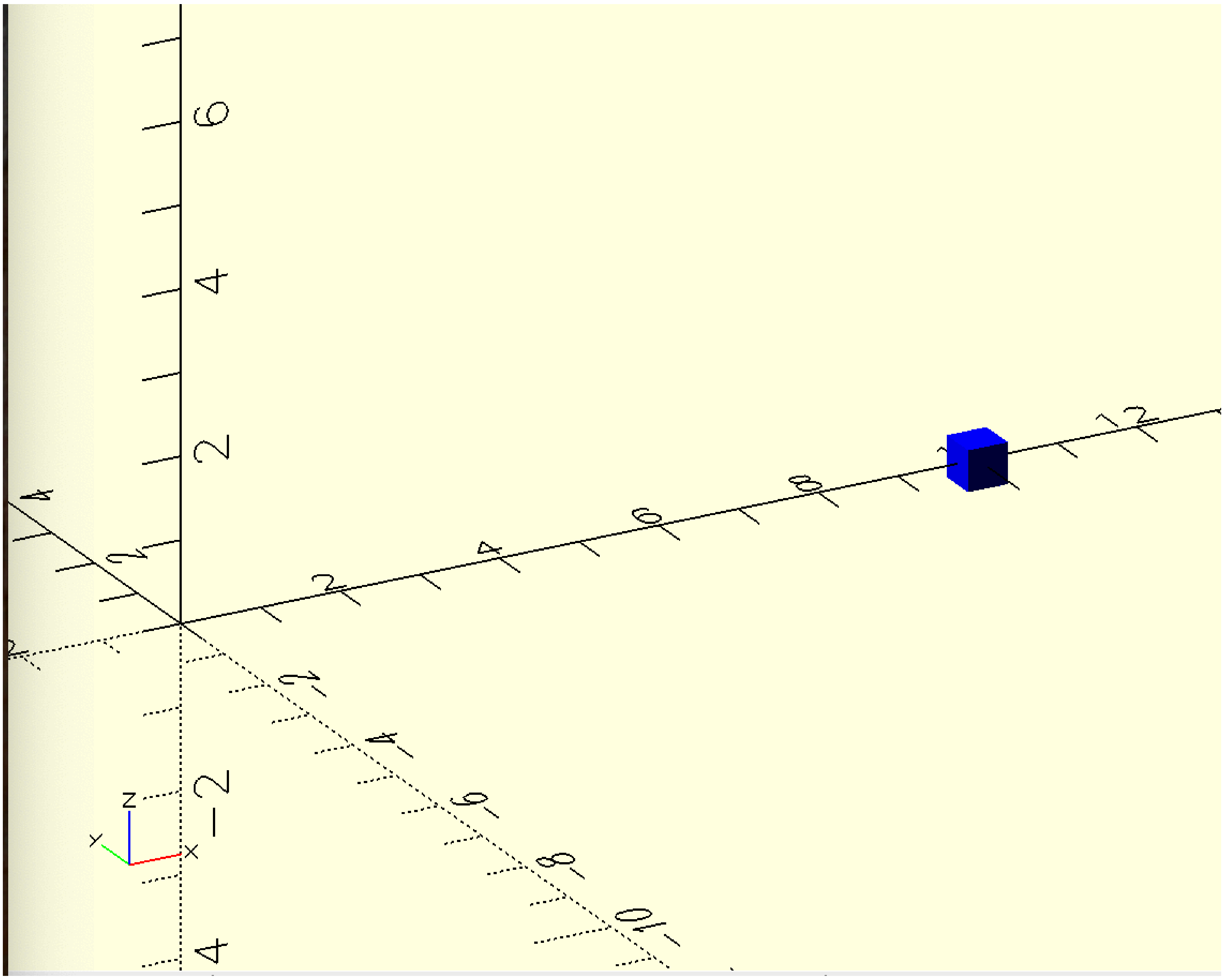
Basic elements are:

- point: defined by 2d or 3d coordinates
- line: defined by 2 points (2d or 3d coordinates)
- polyline: defined by more than 2 points (2d or 3d coordinates)
- surface: defined by arrangement of 2 or more lines or polylines where there is no volume
- solid: defined by arrangement of 2 or more polylines with ends closed and has volume
- plane: defined by a normal vector
- extrude along path: defined by extruding a 2d section along a 3d path
- Sculpting along path: defined by sculpting a 2d section along a 2d path
- Rotate objects: Objects can be rotated along a defined axis
- translate objects: objects can be translated by a defined vector from their relative positions
- wrapping a polyline/ surface/ solids around a path
- Intersections: between line to line, polyline to polyline/ line (2d or 3d) or between surface to surface
- offset: offsetting a section outward or inward
- bspline curves: Can be open and closed loop
- bezier curves
- interpolation curves
- convex hull
- concave hull
- projection of a surface on to another surface
- projecting a line on a surface
- fillets in 2d
- fillets in 3d (few approaches)

Points

In [2]:

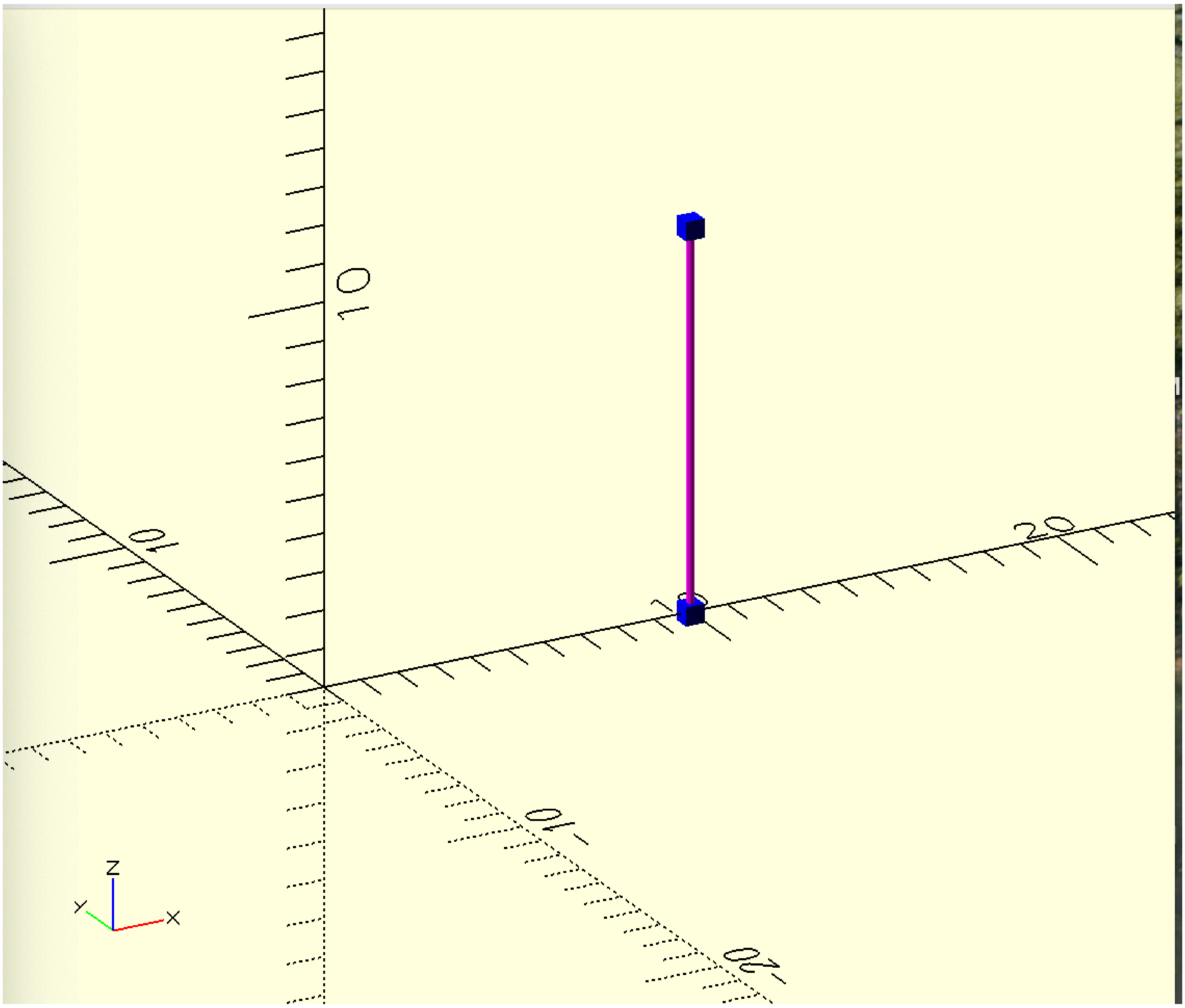
```
p0=[10,0,0]
fileopen(f'''
// pay attention to the points module here. Points are shown as cube
// In this example cube of size 0.5 is showing the location of the p0
color("blue") points({[p0]},.5);
''')
```



Lines

In [3]:

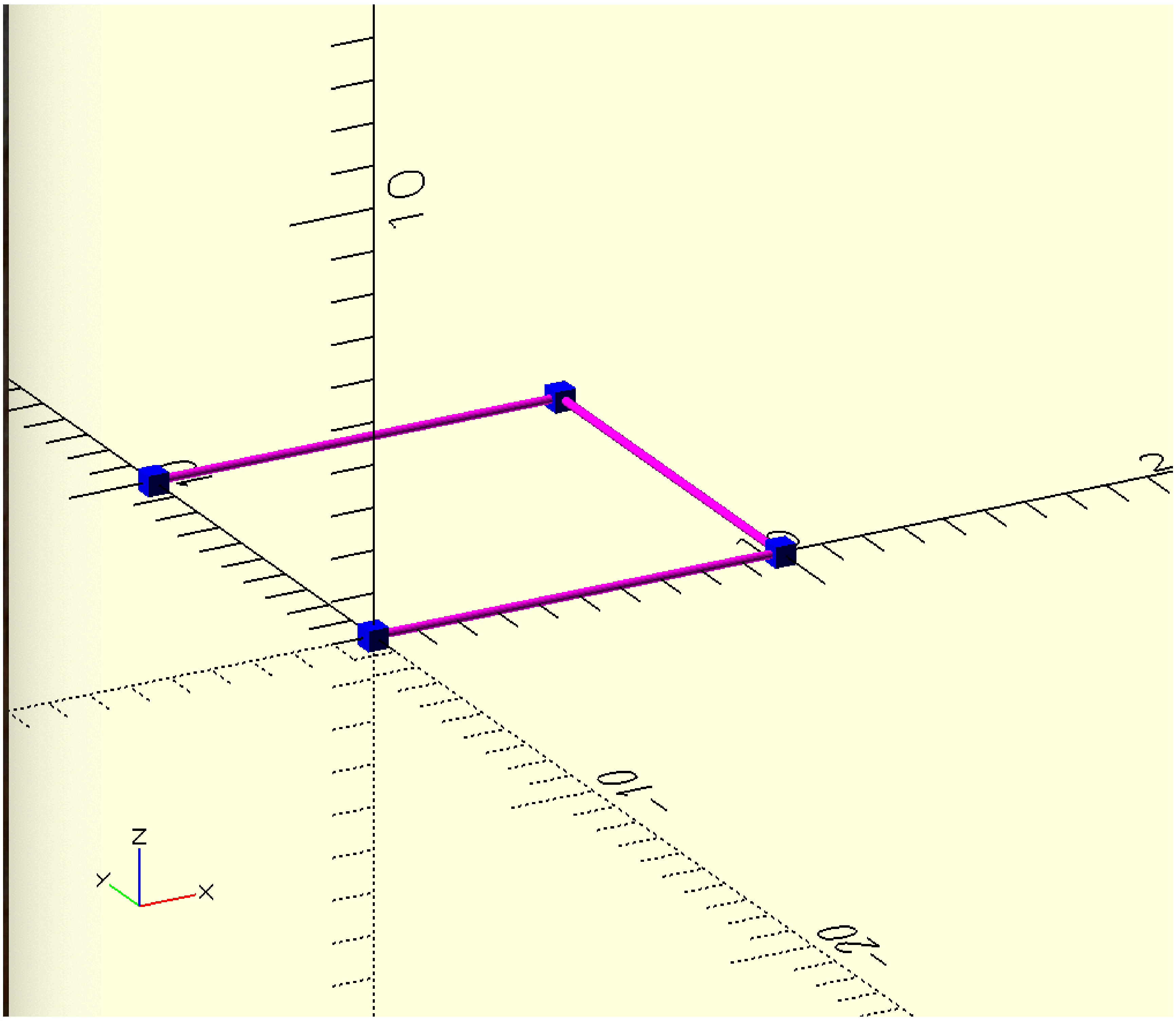
```
l1=[[10,0,0],[10,0,10]]
fileopen(f'''
color("blue") points({l1},.5);

// p_line3d module is used for showing lines or polylines
// in this example line "l1" of diameter 0.2 mm is shown

color("magenta") p_line3d({l1},.2);
''')
```



Polylines

In [4]:

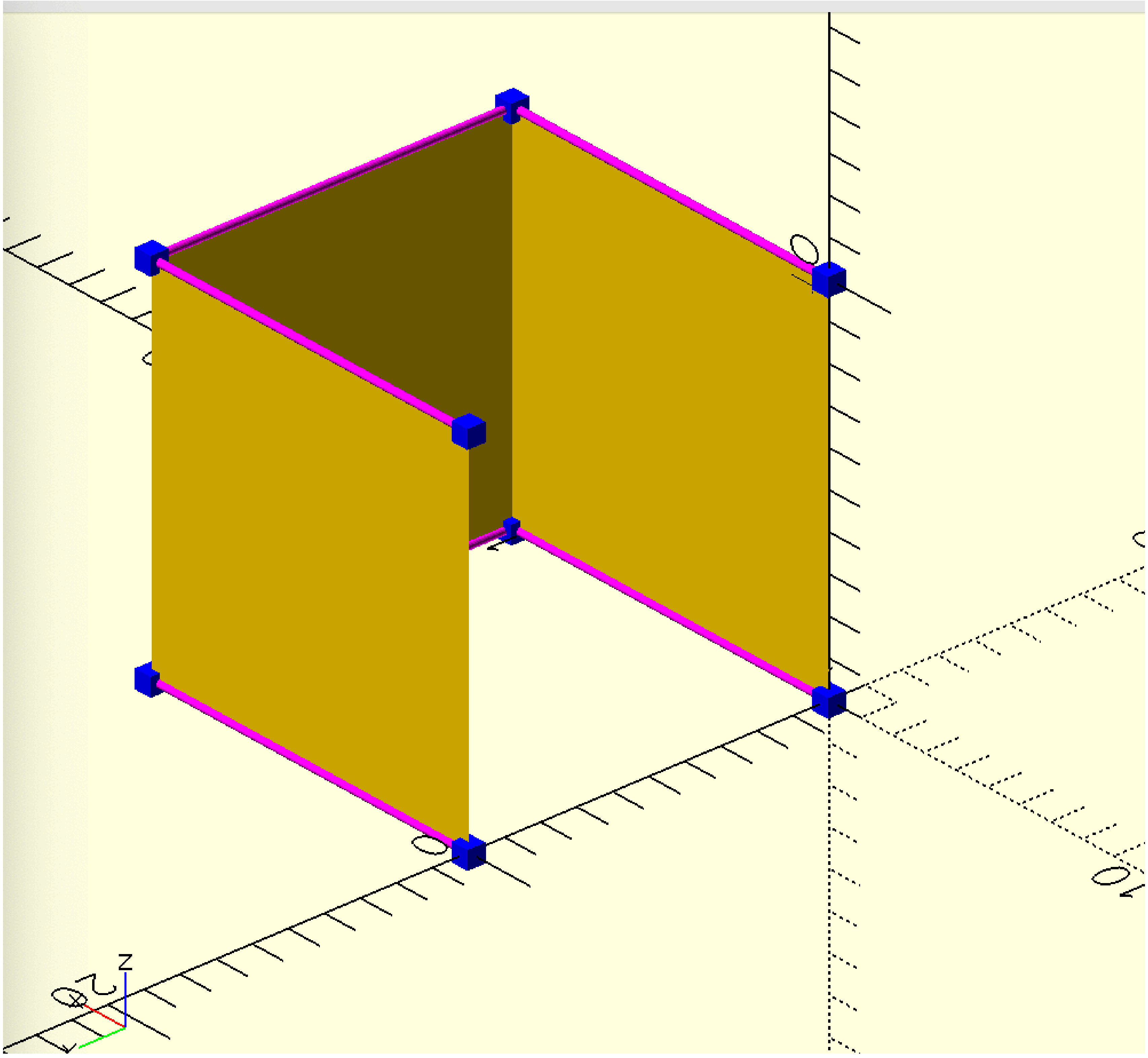
```
l2=cr2dt([[0,0],[10,0],[0,10],[-10,0]])
fileopen(f'''
color("blue") points({l2},.5);
color("magenta") p_line3d({l2},.2);

''')
```



Surface

In [5]:

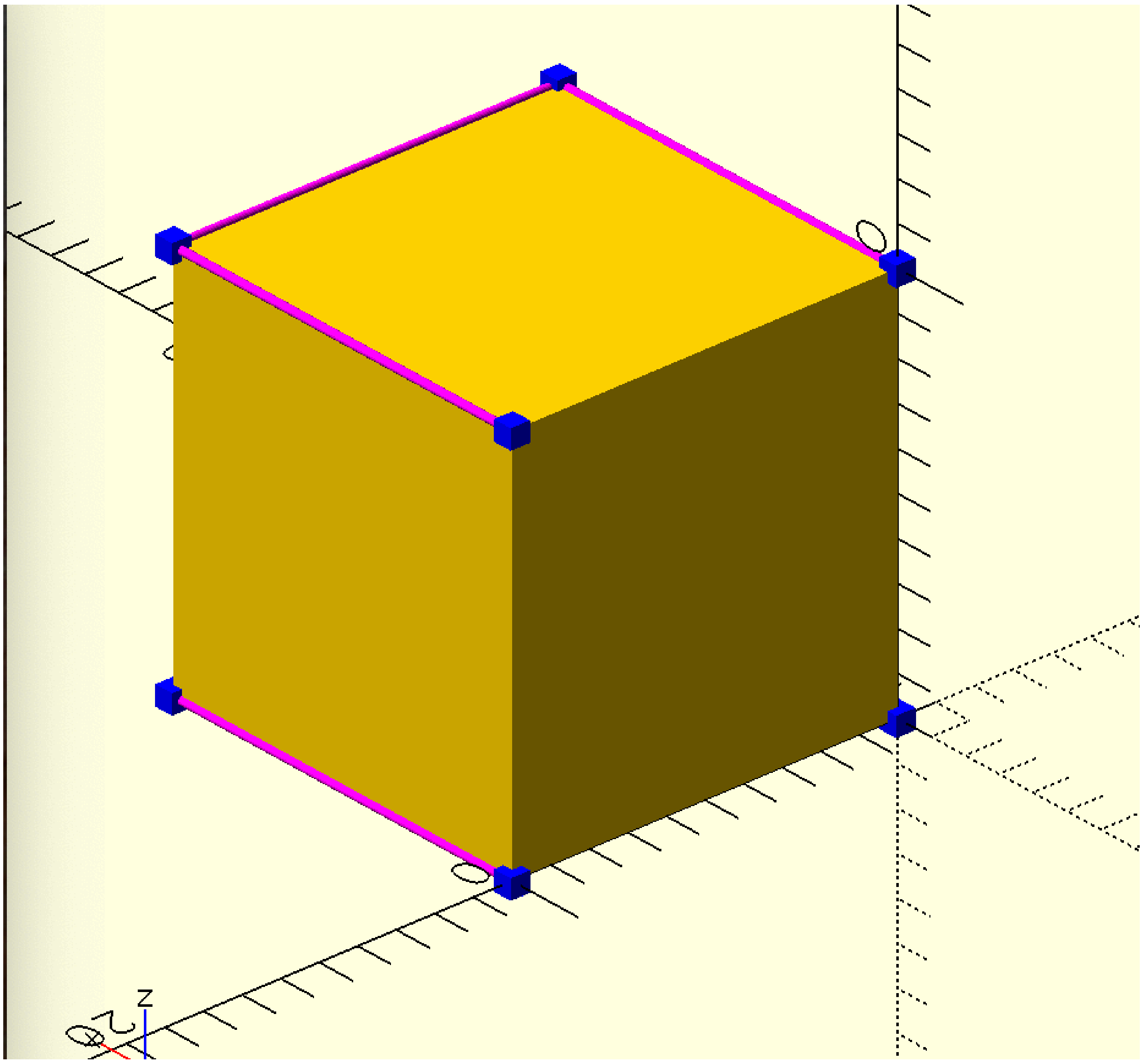
```
l2=cr2dt([[0,0],[10,0],[0,10],[-10,0]])
s1=linear_extrude(l2,10)
fileopen(f'''
color("blue") for(p={s1}) points(p,.5);
color("magenta")for(p={s1}) p_line3d(p,.2);

// pay attention to the swp_surf module here
// swp_surf shows the surface covered by the polylines and is very important
// to understand as intersections are calculated based on intersecting surfaces

{swp_surf(s1)}

''')
```



Solid

In [6]:

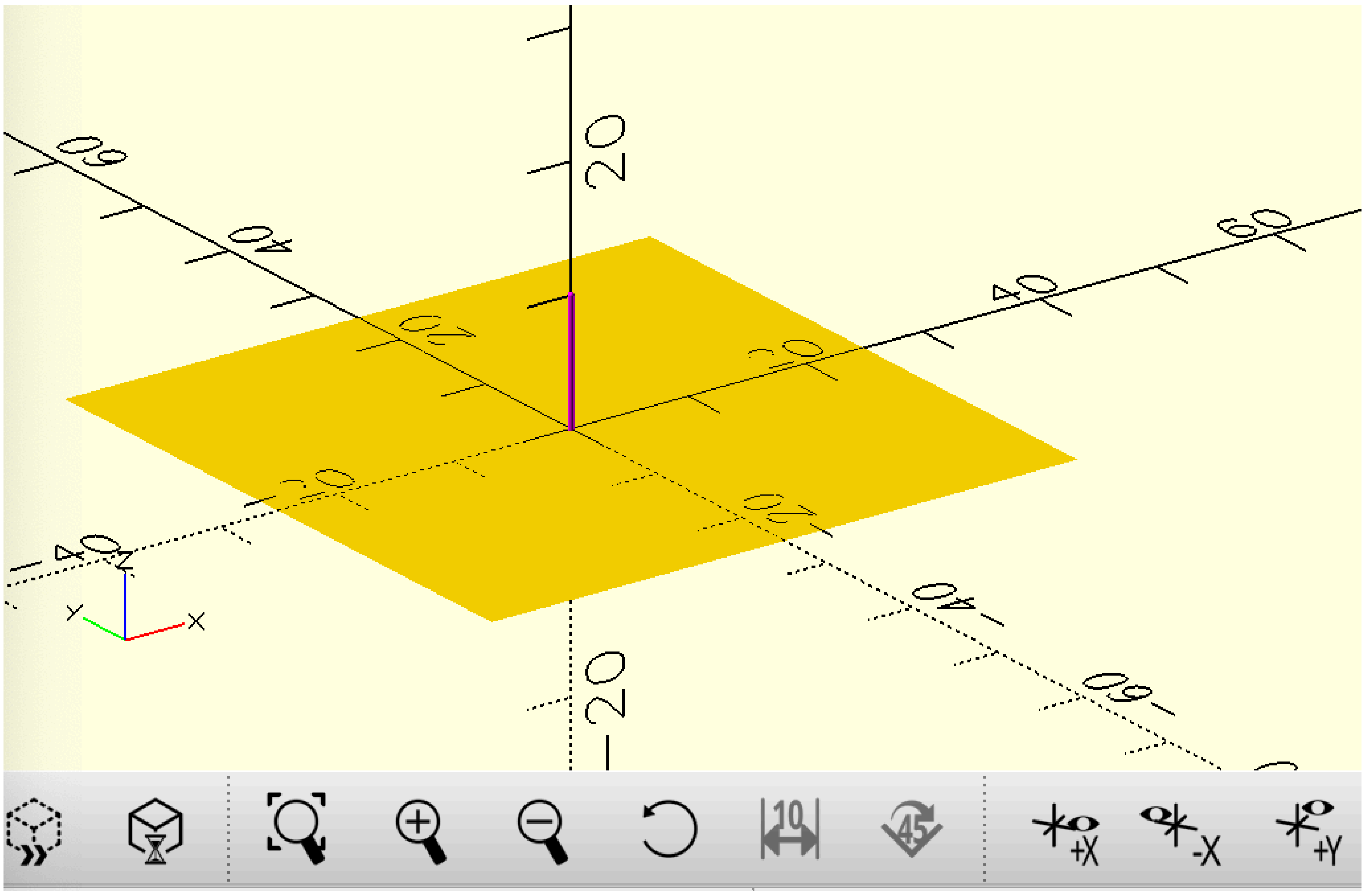
```
l2=cr2dt([[0,0],[10,0],[0,10],[-10,0]])
s1=linear_extrude(l2,10)
fileopen(f'''
color("blue") for(p={s1}) points(p,.5);
color("magenta")for(p={s1}) p_line3d(p,.2);
{swp(s1)}

''')
```



Planes

In [7]:

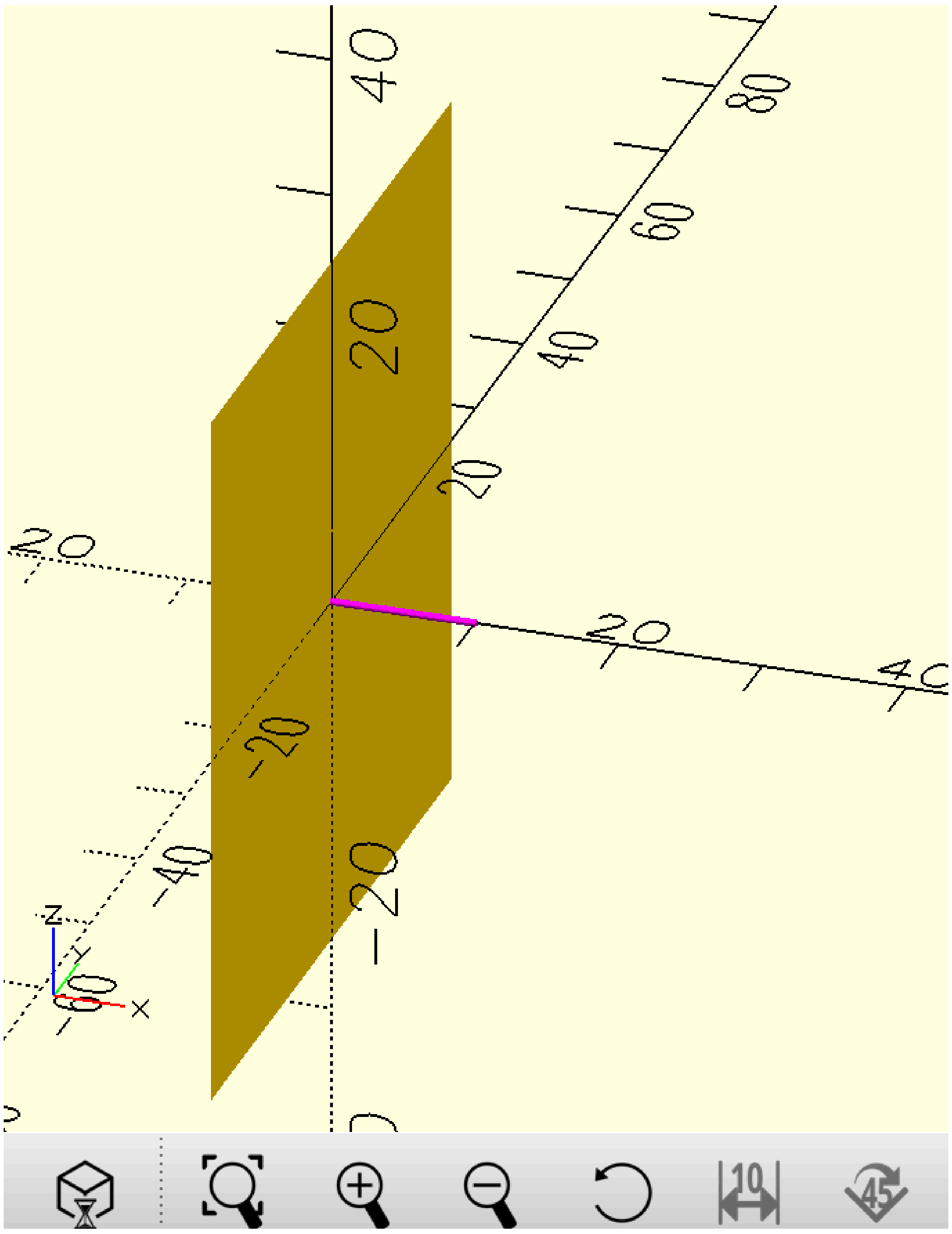
```
n1=[0,0,1]
l1=[[0,0,0],[0,0,10]]
# x-y plane
pl1=plane(n1,size=[50,50], intercept=[0,0,0])
fileopen(f'''
color("magenta") p_line3d({l1},.5);
{swp_surf(pl1)}
''')
```



In [8]:

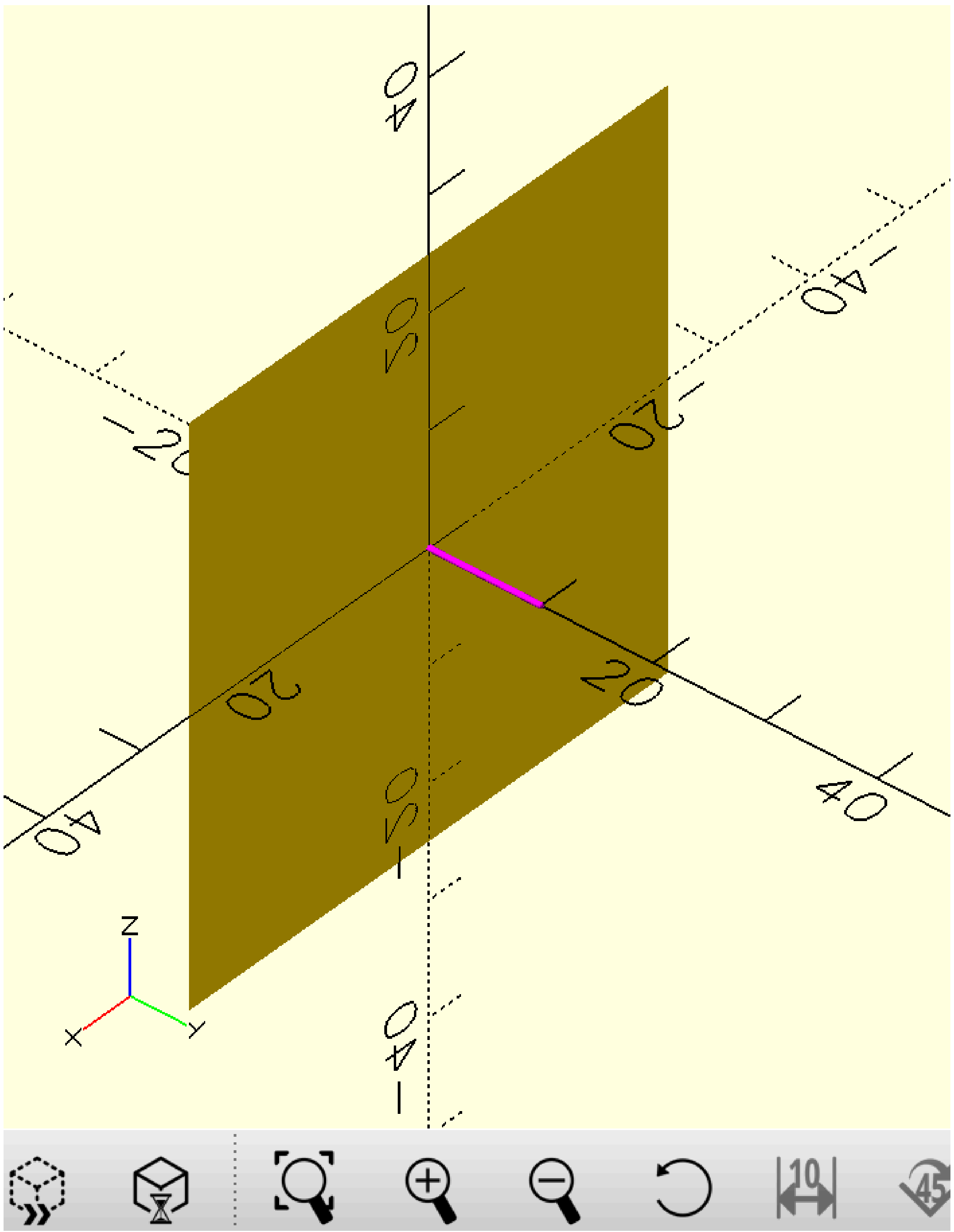
```
n1=[1,0,0]
l1=[[0,0,0],[10,0,0]]
# y-z plane
```

```
pl1=plane(n1,size=[50,50], intercept=[0,0,0])
fileopen(f'''
color("magenta") p_line3d({l1},.5);
{swp_surf(pl1)}
''')
```



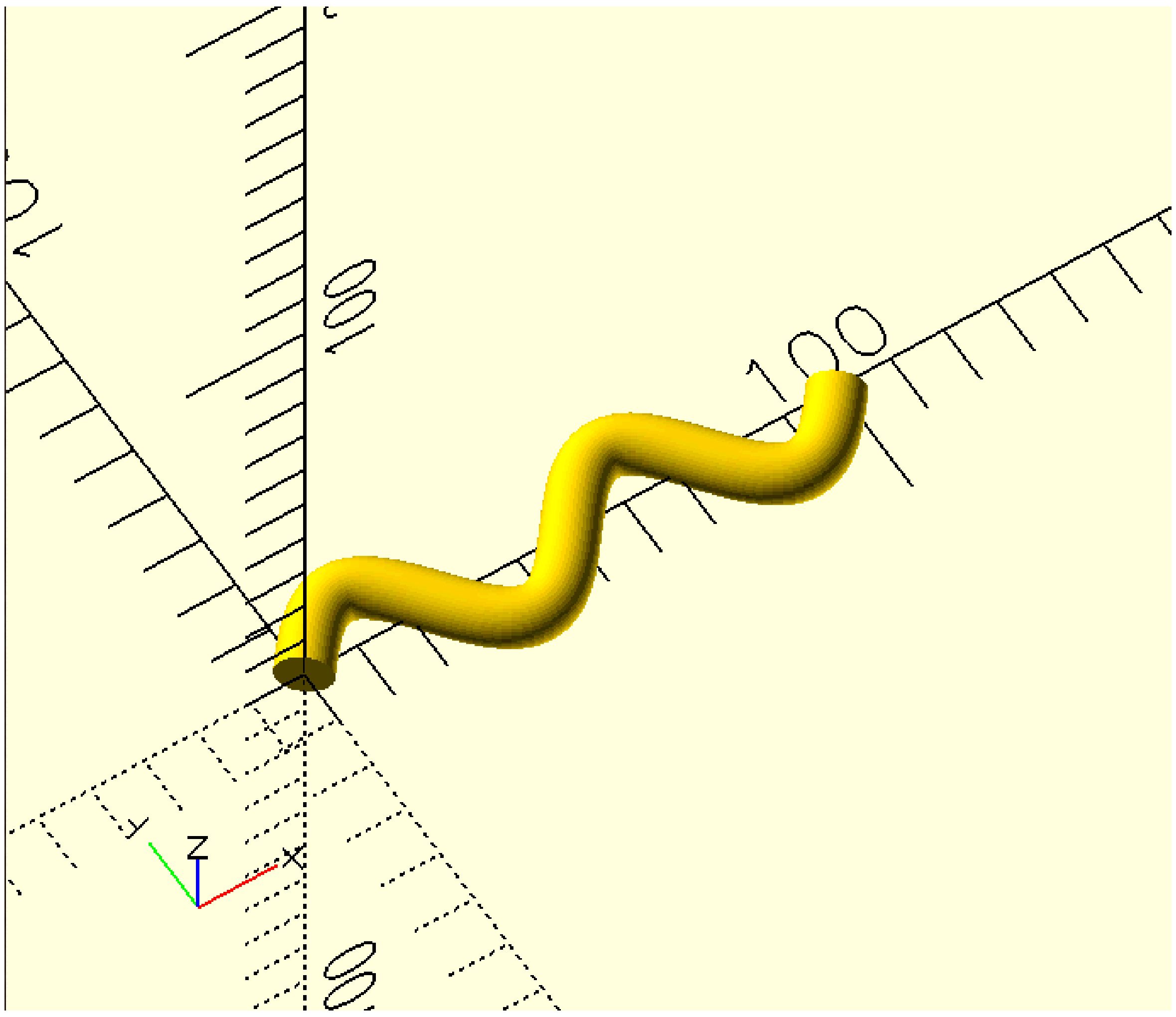
In [9]:

```
n1=[0,1,0]
l1=[[0,0,0],[0,10,0]]
# x-z plane
pl1=plane(n1,size=[50,50], intercept=[0,0,0])
fileopen(f'''
color("magenta") p_line3d({l1},.5);
{swp_surf(pl1)}
''')
```

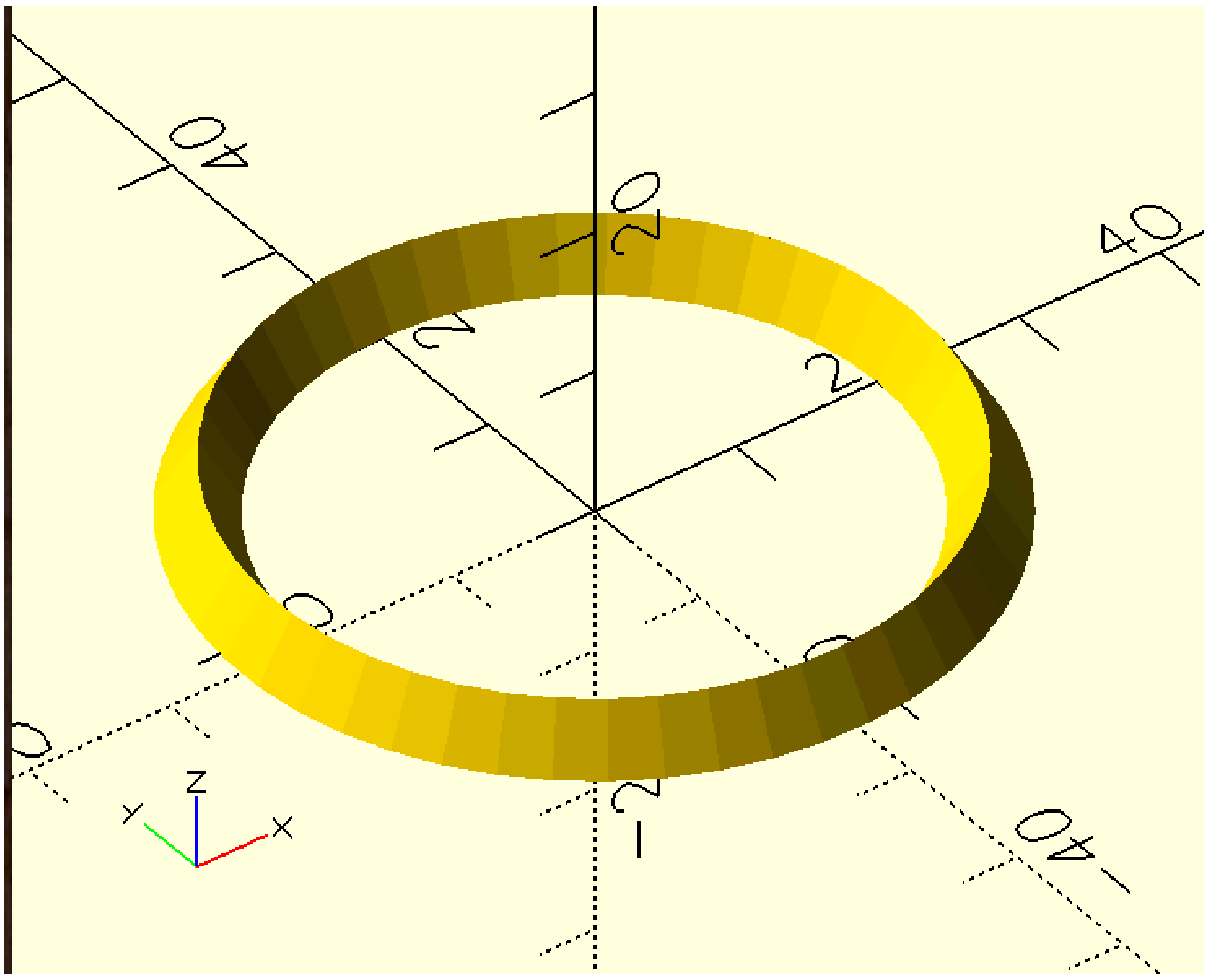


Extrude along path

In [10]:
```
# circular section extruded along open path
sec=circle(5)
path=c23(sinewave(l=100,n=2,a=10,p=100))
sol=path_extrude_open(sec,path)
fileopen(f'''
{swp(sol)}
''')
```



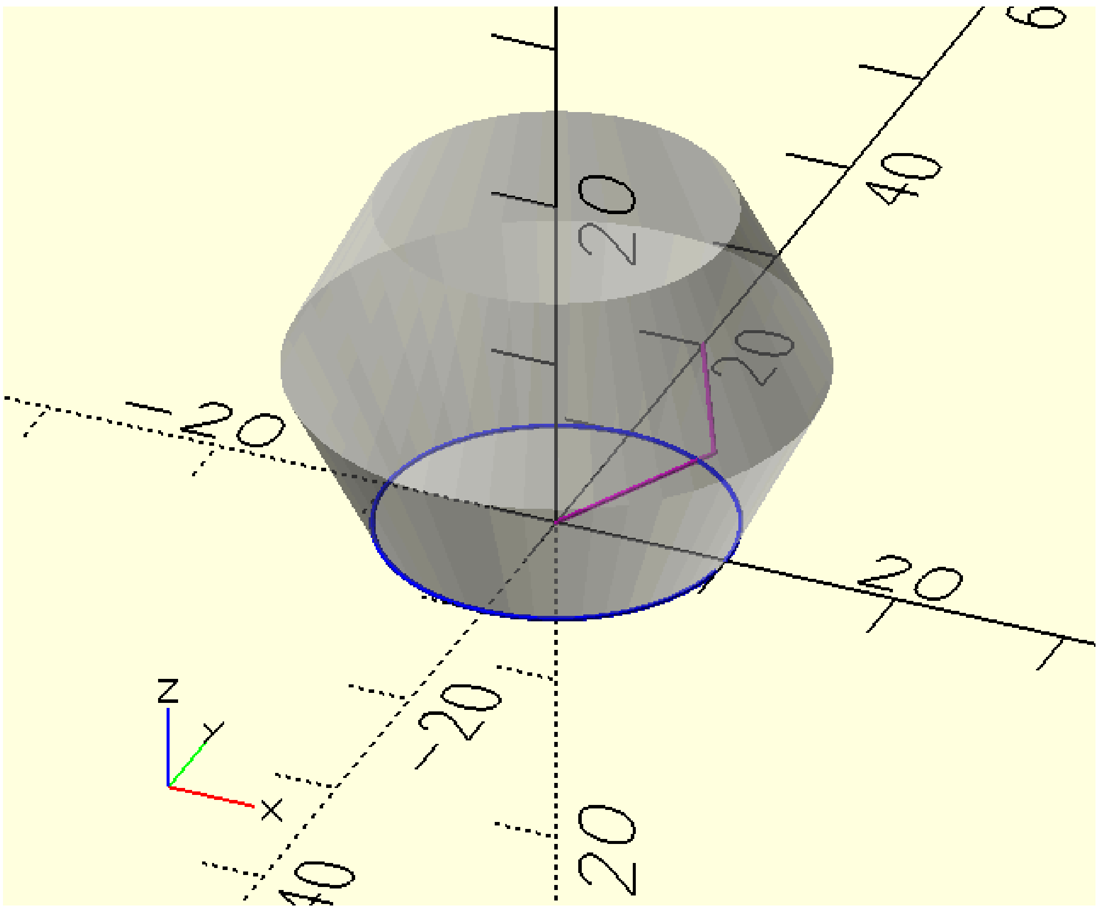
In [11]:
```
# triangular section extruded along closed path
sec=[[0,0],[5,0],[2.5,4]]
path=c23(circle(20))
sol=path_extrude_closed(sec,path)
fileopen(f'''
// pay attention to the swp_c module here
// swp_c is to be used where the loop is closing like the way here
{swp_c(sol)}
''')
```



Sculpting along path

In [12]:

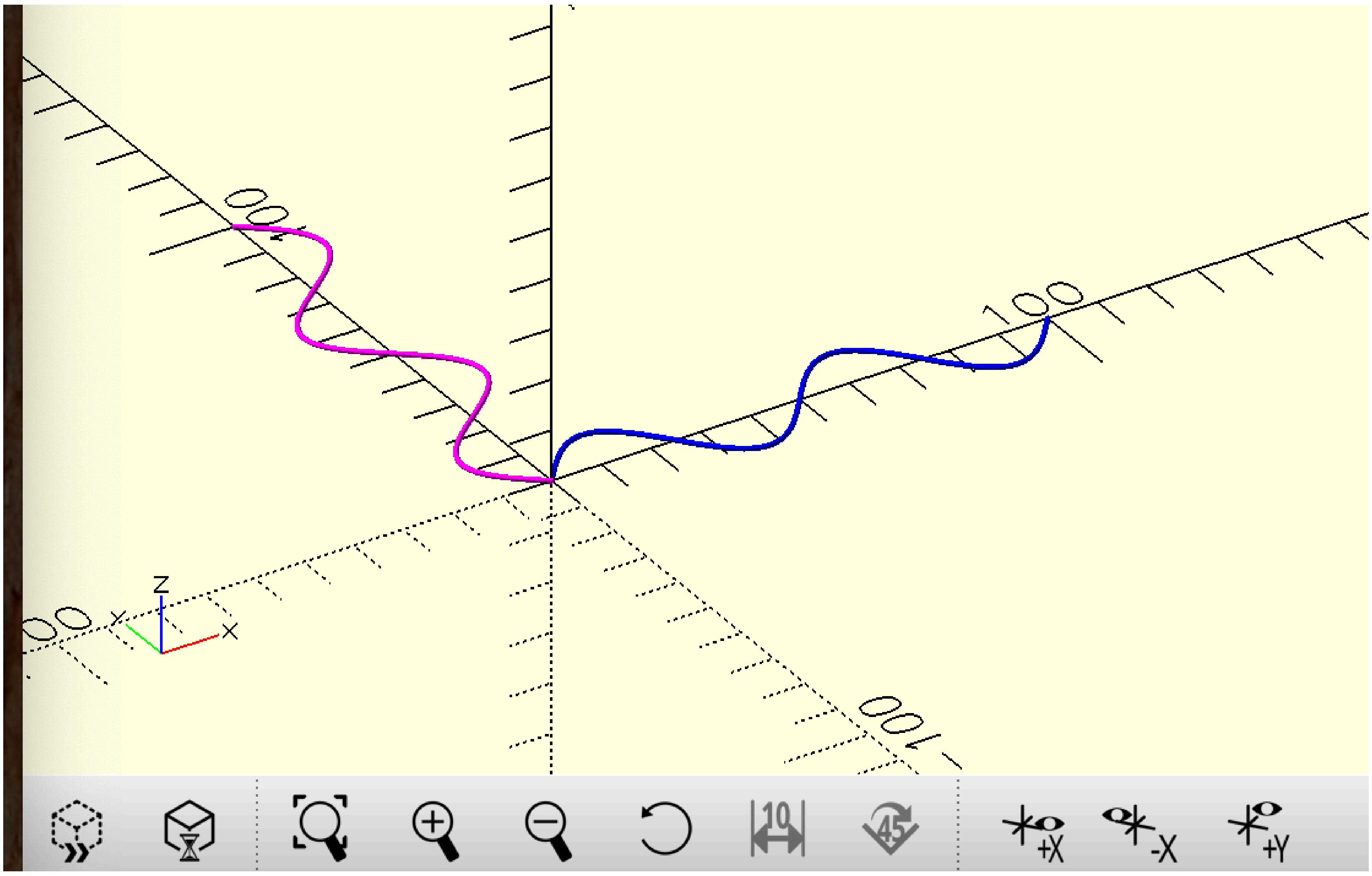
```
sec=circle(10)
path=[[0,0],[5,10],[0,20]] # x-coordinates work as offset and y-coordinates work as z-translate of sec
sol=prism(sec,path)
fileopen(f'''
color("blue") p_line3d({sec},.3);
color("magenta") p_line3d({path},.3);
%{swp(sol)}

''')
```



Rotation : Right hand thumb-rule (if thumb is pointed in the direction of axis, fingers curled in the direction of rotation)

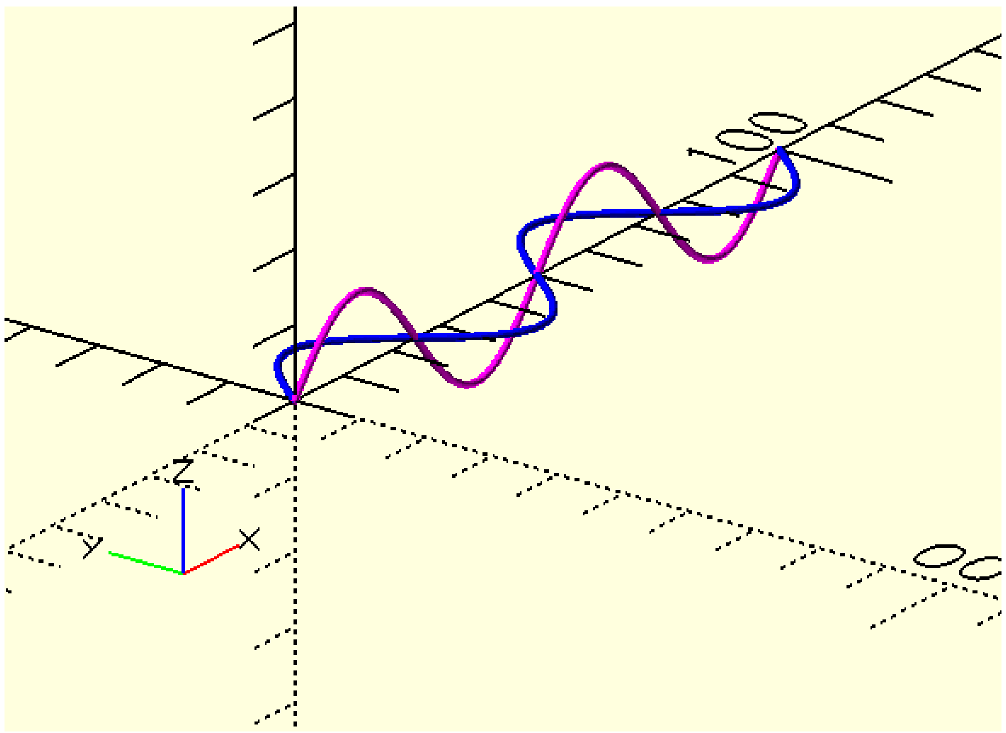
In [13]:

```
l1=sinewave(100,2,10,100)
l2=rot('z90',l1) # l1 rotated by 90 deg along z-axis
fileopen(f'''
// original line 'l1'
color("blue") p_line3d({l1},1);
//rotated line
color("magenta") p_line3d({l2},1);

''')
```



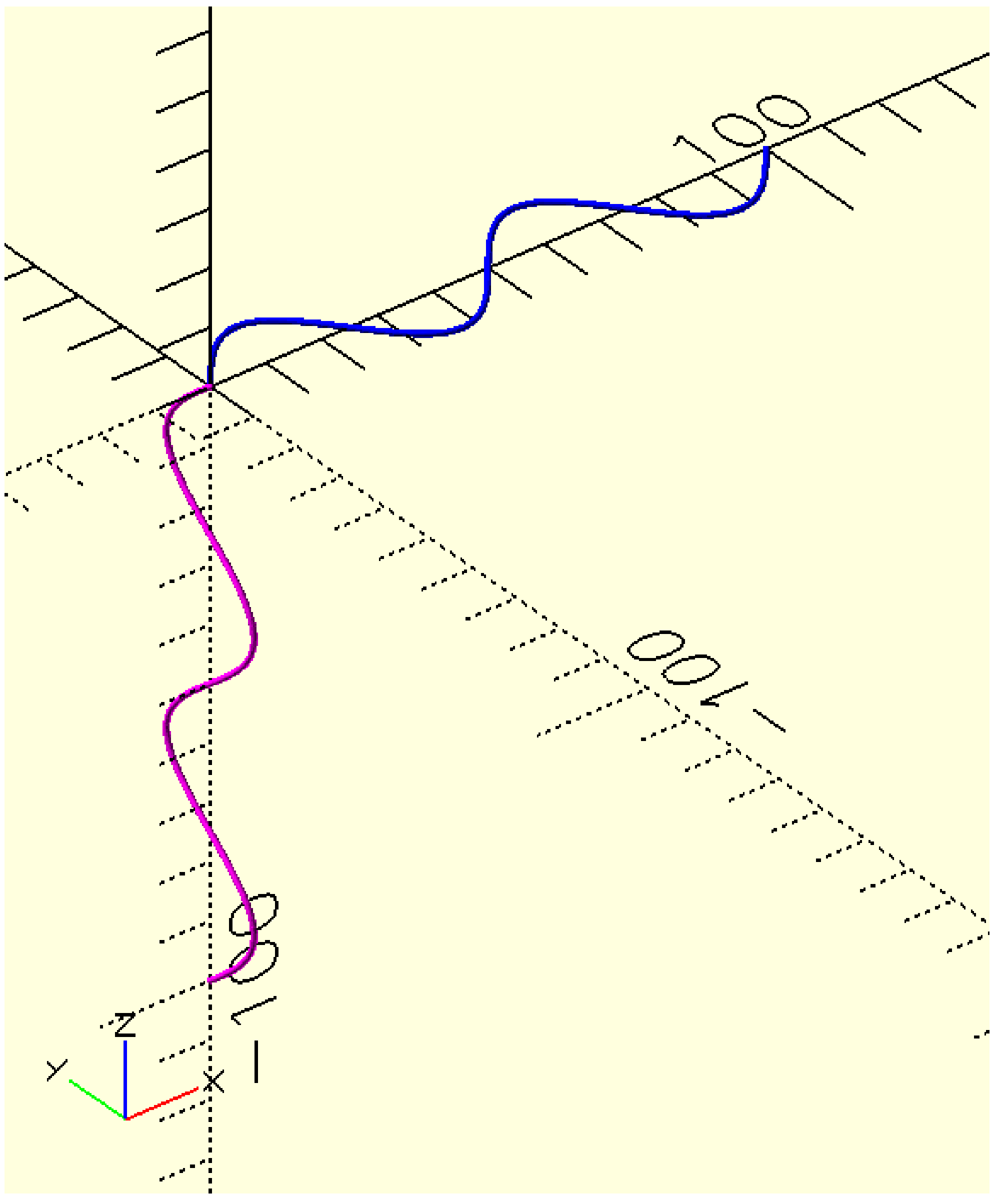
In [14]:

```
l1=sinewave(100,2,10,100)
l2=rot('x90',l1) # l1 rotated by 90 deg along x-axis
fileopen(f'''
// original line 'l1'
color("blue") p_line3d({l1},1);
//rotated line
color("magenta") p_line3d({l2},1);

''')
```



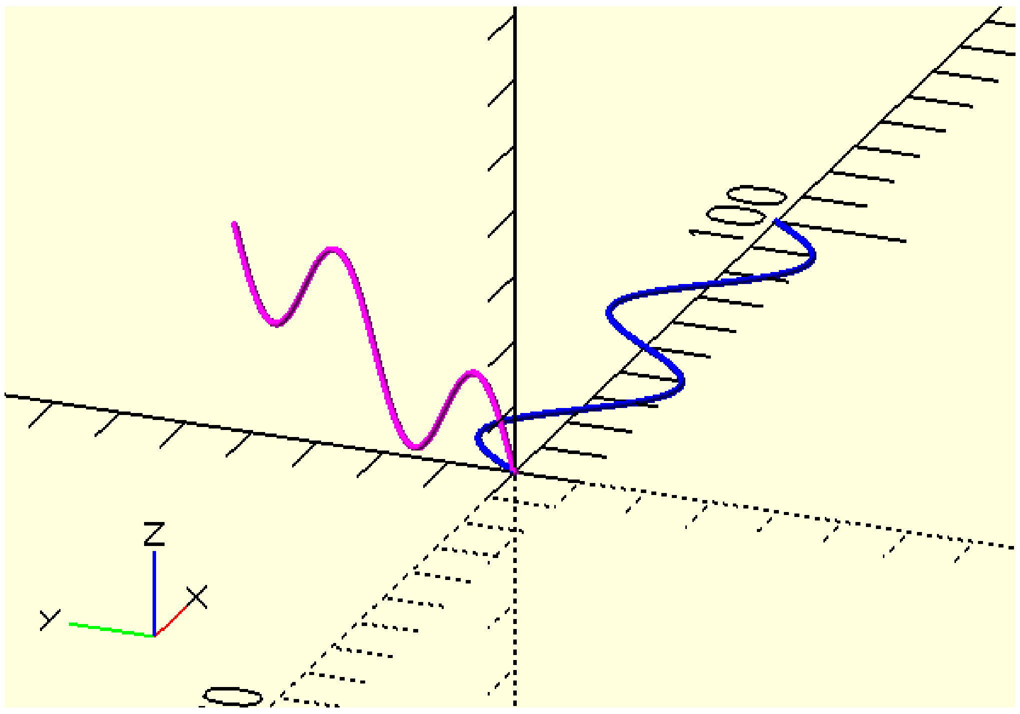
In [15]:

```
l1=sinewave(100,2,10,100)
l2=rot('y90',l1) # l1 rotated by 90 deg along y-axis
fileopen(f'''
// original line 'l1'
color("blue") p_line3d({l1},1);
//rotated line
color("magenta") p_line3d({l2},1);

''')
```



In [16]:

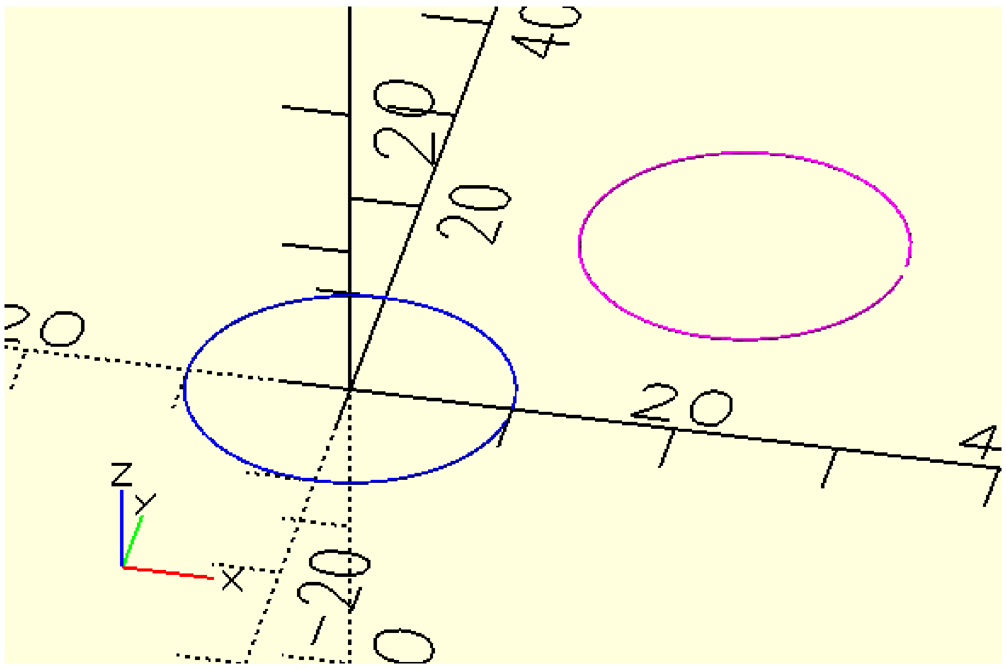
```
l1=sinewave(100,2,10,100)
l2=rot('x90z45',l1) # multiple rotation of l1 (rotated by 90 deg along x-axis
# and then 45 deg along z-axis
fileopen(f'''
// original line 'l1'
color("blue") p_line3d({l1},1);
//rotated line
color("magenta") p_line3d({l2},1);

''')
```



Translate: are of 2d and 3d type

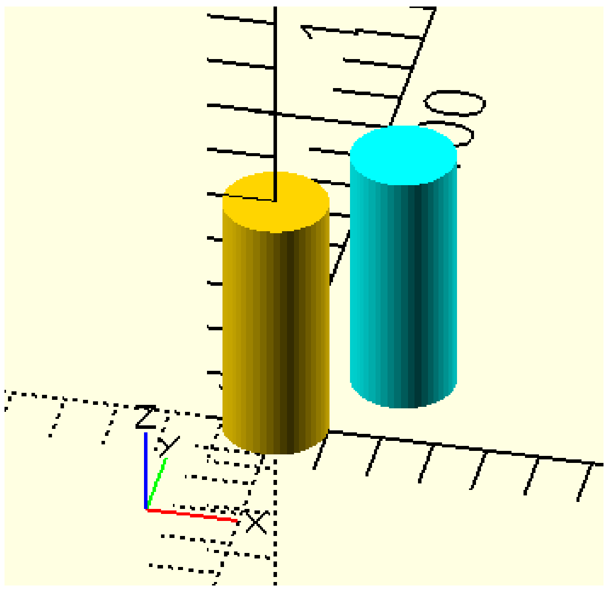
In [17]:

```
# example of translate in 2 d coordinates

c1=circle(10)
c2=translate_2d([20,20],c1)
fileopen(f'''
// original circle
color("blue") p_line3d({c1},.2);

// translated circle
color("magenta") p_line3d({c2},.2);

''')
```



In [18]:

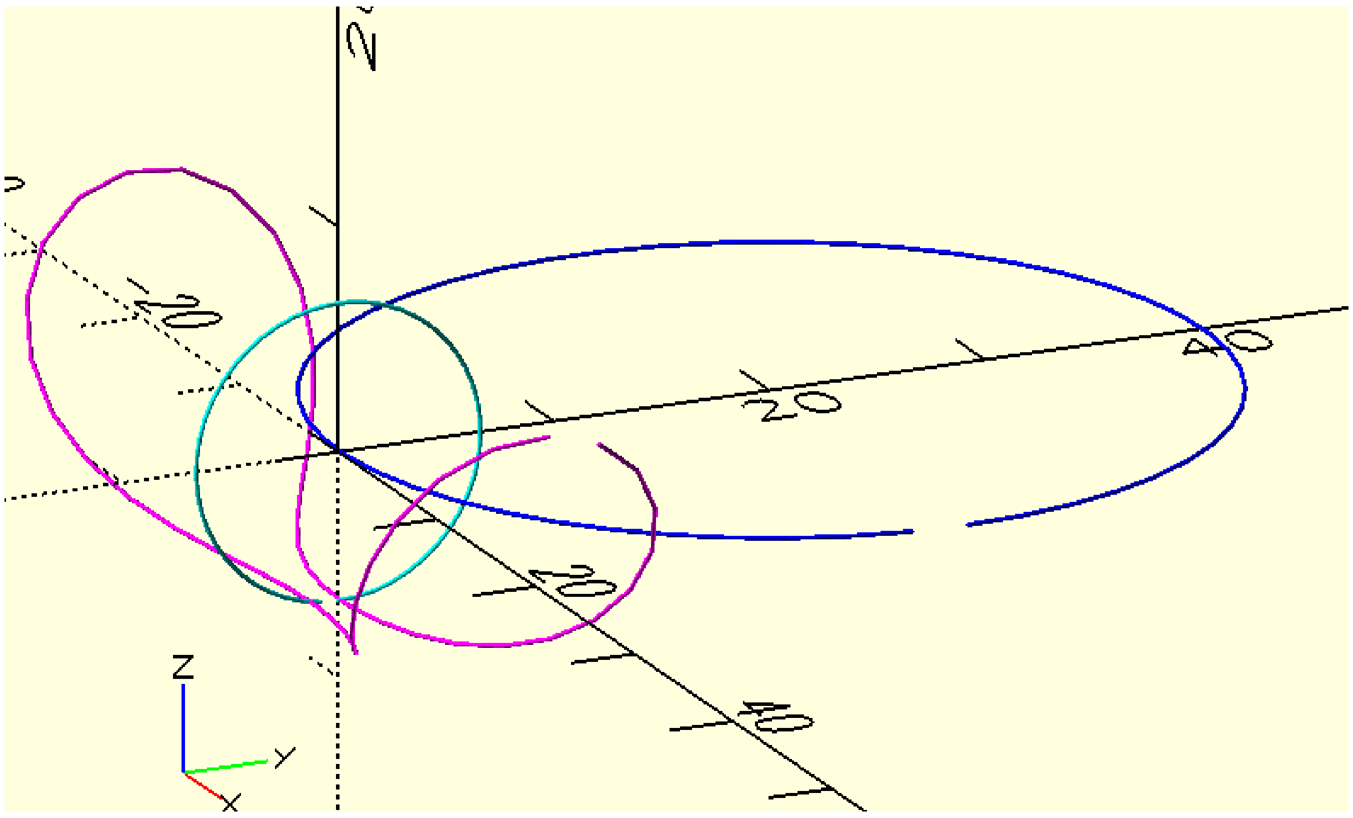
```
# example of translate in 3d coordinate
c1=linear_extrude(circle(10),50)
c2=translate([20,20,0],c1)
fileopen(f'''
// original cyclinder
{swp(c1)}
// translated cylinder by vector [20,20,0]
color("cyan"){swp(c2)}

''')
```



wrap around a section over a path

In [19]:

```
c1=translate([0,20.1,0],circle(20))
path=rot('y90',circle(40.2/(2*pi)+.2))
c2=wrap_around(c1,path)

fileopen(f'''
color("blue") p_line3d({c1},.2);
color("cyan") p_line3d({path},.2);
color("magenta") p_line3d({c2},.2);

''')
```

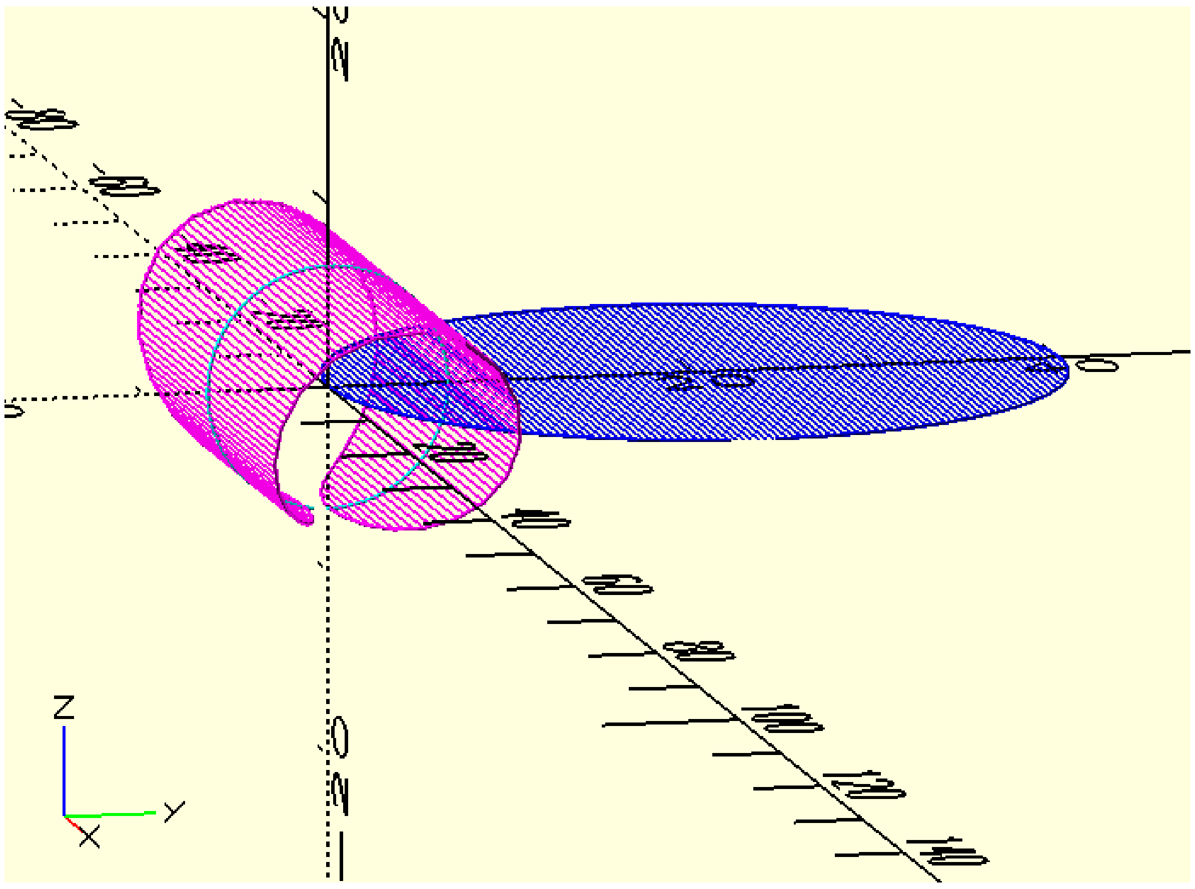


wrap around a surface over a path

In [20]:

```
c1=translate_2d([0,20.1],circle(20))
s1=h_lines_sec(c1,100)
path=rot('y90',circle(40.2/(2*pi)+.2))
c2=wrap_around(c1,path)
s2=[wrap_around(p,path) for p in s1]
```

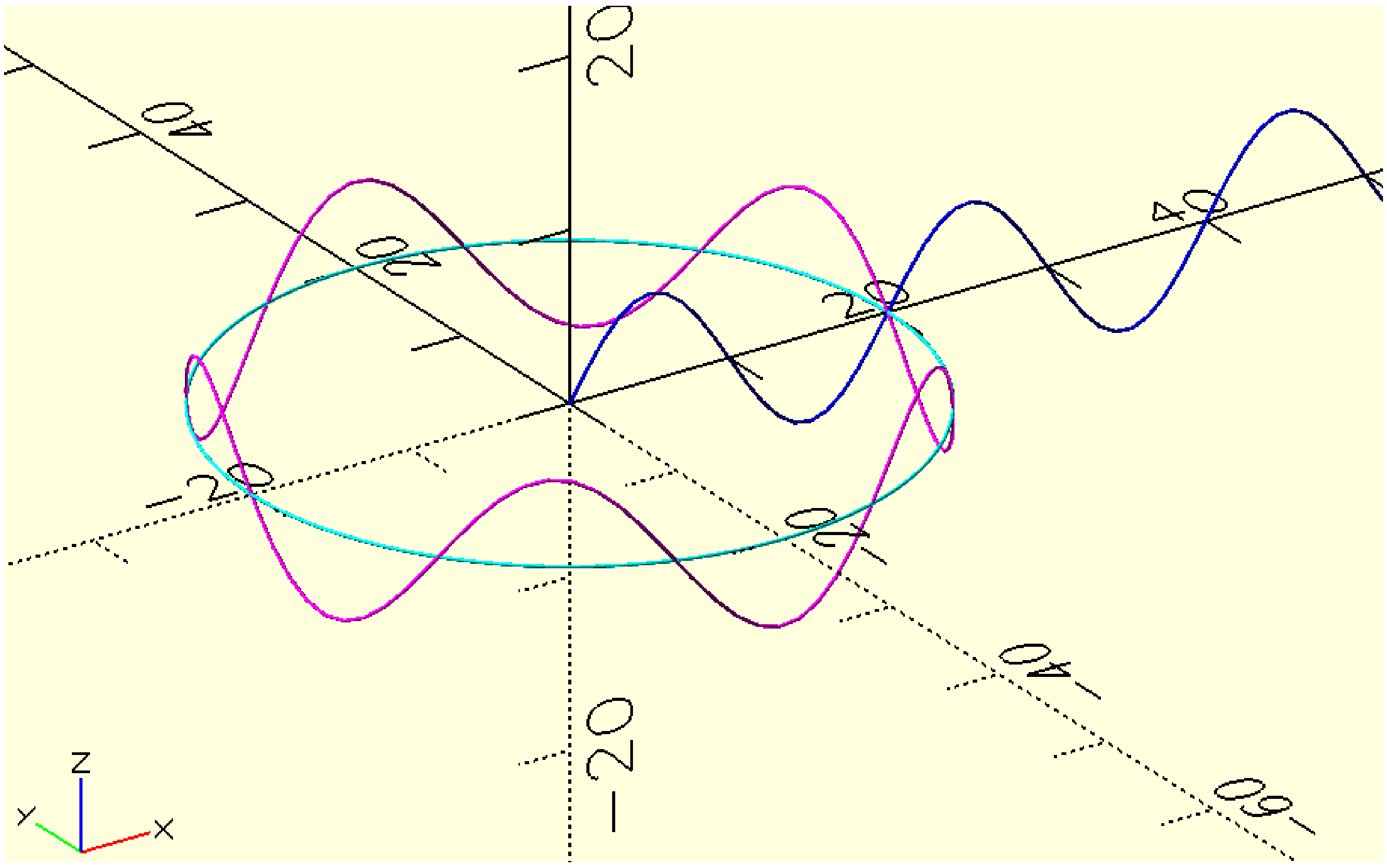
```
fileopen(f'''
color("blue") p_line3d({c1},.2);
color("cyan") p_line3d({path},.2);
color("magenta") p_line3d({c2},.2);

color("blue") for(p={s1}) p_line3d(p,.1,1);
color("magenta") for(p={s2}) p_line3d(p,.1,1);
''')
```



other methods of wrapping a polyline/ solid around a path

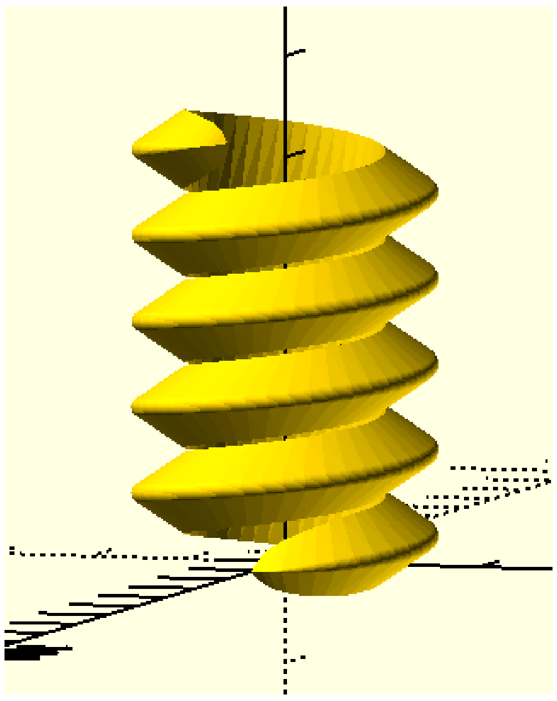
In [21]:

```
c1=rot('x90',sinewave(100,5,5,100))
path=c23(arc(20,0,360,s=99))
c2=extrude_wave2path(c1,path)

fileopen(f'''
color("blue") p_line3d({c1},.2);
color("cyan") p_line3d({path},.2);
color("magenta") p_line3dc({c2},.2);

''')
```



In [22]:

```
c1=rot2d(-90,cr2dt([[-4,0],[8,0],[-4,6,1]],10))
path=m_points1_o(cr2dt([[-2,0],[2,0.5,2],[0,50,2],[-2,0.5]],10),200,.01)
sol=prism(c1,path)
path1=helix(10,8.5,5,10)
path1=path2path1(path,path1)
# extruding sol to path1
sol1=sol2path(sol,path1)
fileopen(f'''
{swp(sol1)}
//color("blue") p_line3d({path},.5,1);
''')
```

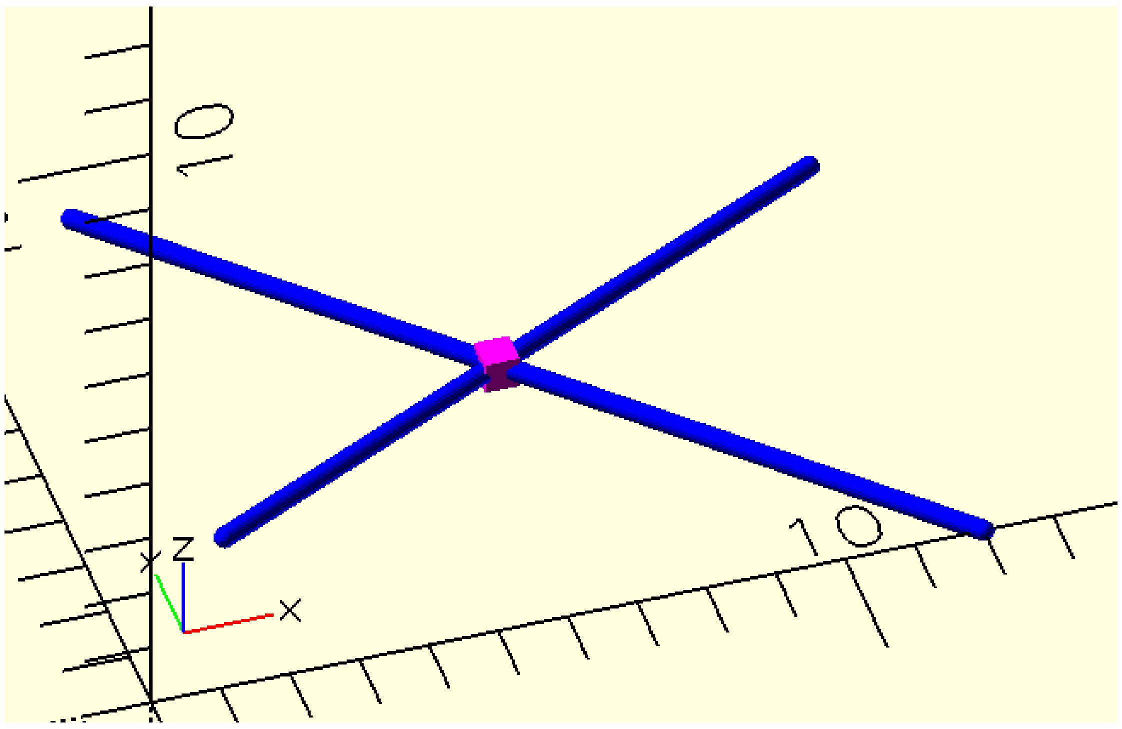


Intersections

intersection between line to line (2d)

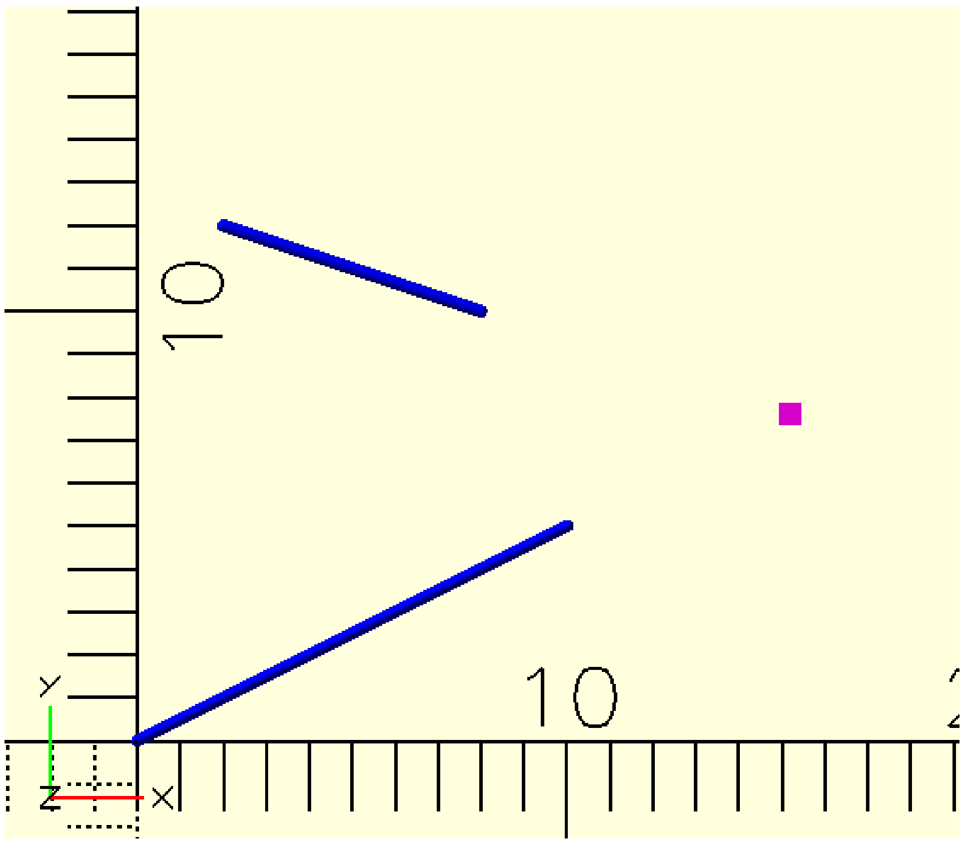
In [23]:

```
l1=point_vector([2,3],[10,5])
l2=point_vector([2,10],[10,-10])
p0=s_int1([l1,l2])[0]
fileopen(f'''
color("blue") for(p={[l1,l2]}) p_line3d(p,.3);
color("magenta") points({[p0]},.5);
''')
```



In [24]:

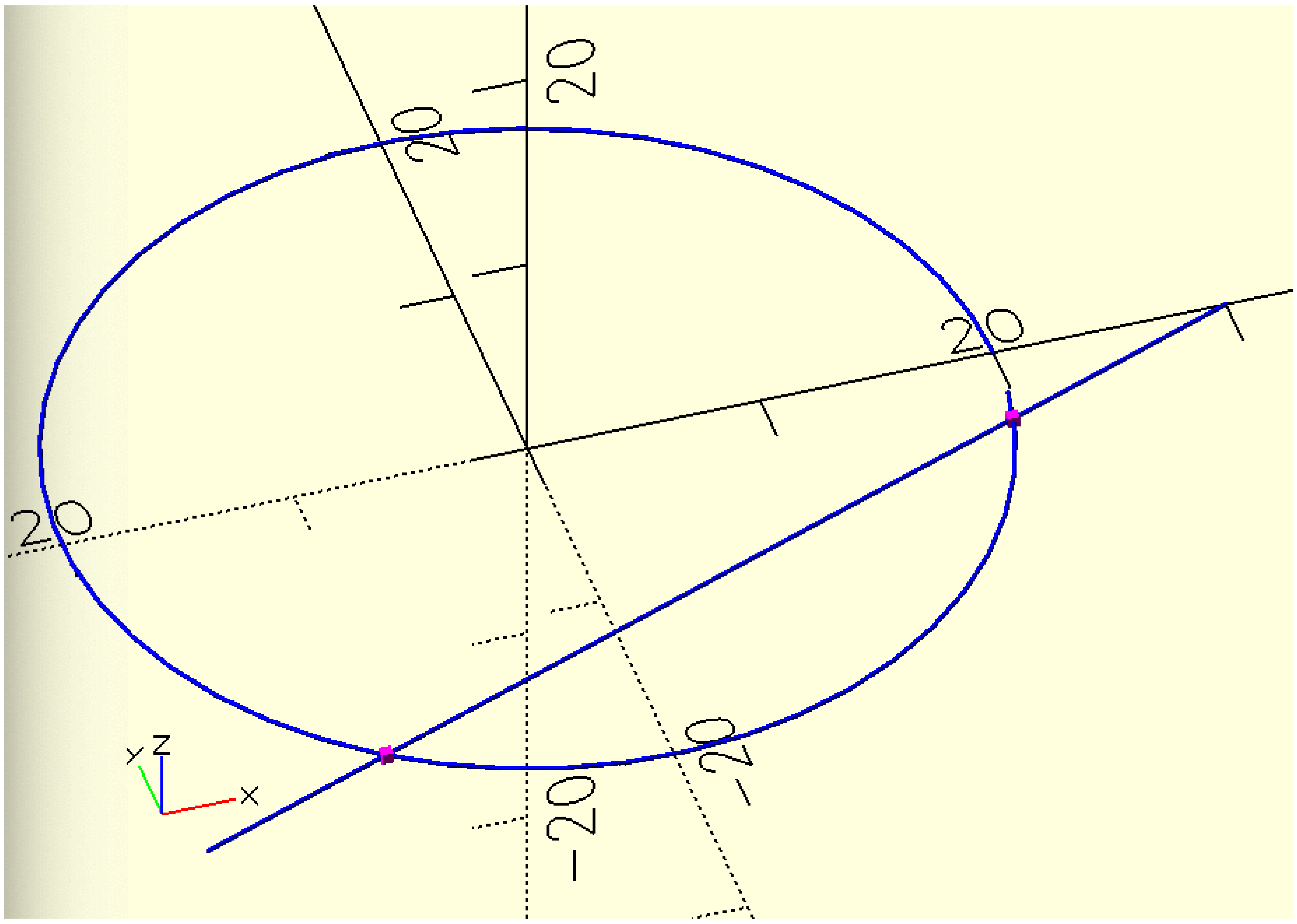
```
# intersection point between 2 lines even if they are not directly intersecting
l1=[[0,0],[10,5]]
l2=[[2,12],[8,10]]
p0=i_p2d(l1,l2)
fileopen(f'''
color("blue") for(p={[l1,l2]}) p_line3d(p,.3);
color("magenta") points({[p0]},.5);

''')
```



intersection between a polyline and line

In [25]:

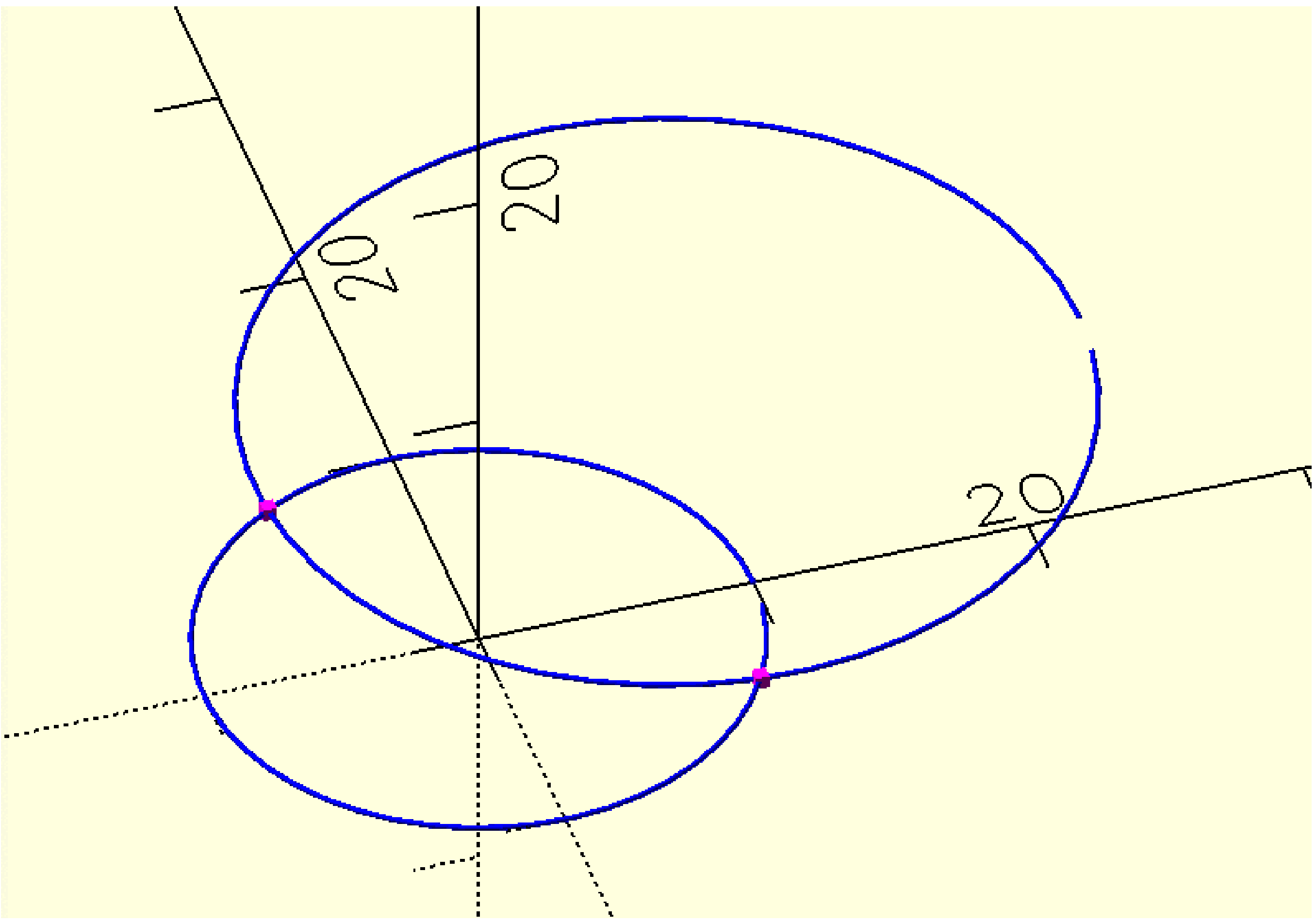
```
c1=circle(20)
l1=point_vector([-20,-20],[50,20])
p0=s_int1([l1]+seg(c1))
fileopen(f'''
color("blue") for(p={[l1,c1]}) p_line3d(p,.2);
color("magenta") points({p0},.5);

''')
```



intersection between 2 polylines

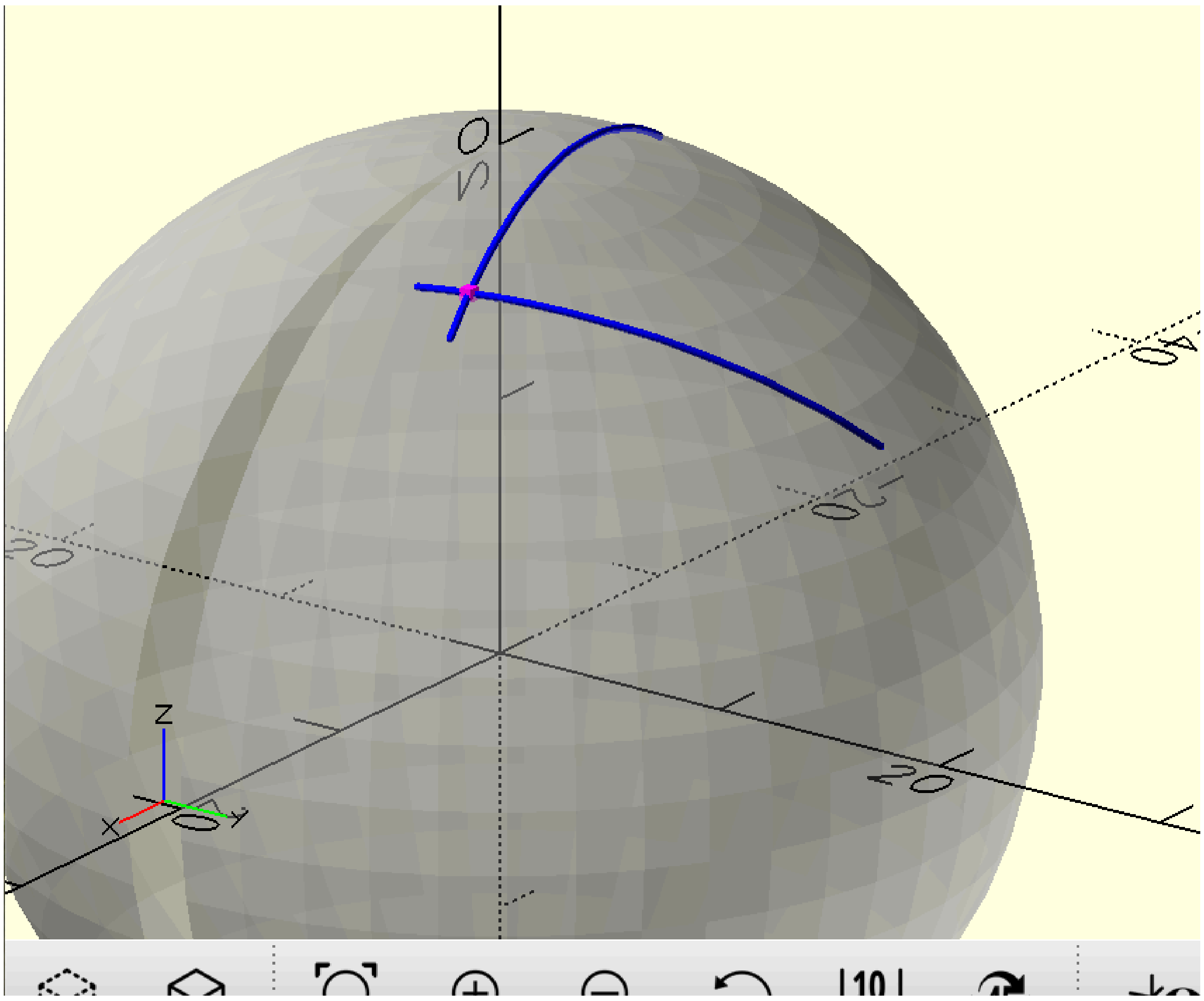
In [26]:

```
c1=circle(10)
c2=circle(15,[10,10])
p0=s_int1(seg(c1)+seg(c2))
fileopen(f'''
color("blue") for(p={[c1,c2]}) p_line3d(p,.2);
color("magenta")points({p0},.5);
''')
```



In [27]:

```
# intersection between 2 polylines in 3d space
s1=sphere(20)
l1=c23(homogenise([[-10,0],[10,5]],1))
l1=plos(s1,l1,[0,0,1])
l2=c23(homogenise([[0,-15,0],[-7,5,0]],1))
l2=plos(s1,l2,[1,2,2])
p0=s_int1_3d(seg(l1)+seg(l2))[0]
fileopen(f'''
%{swp_surf(s1)}
color("blue") for(p={[l1,l2]}) p_line3d(p,.3);
color("magenta") points({[p0]},.5);
''')
```

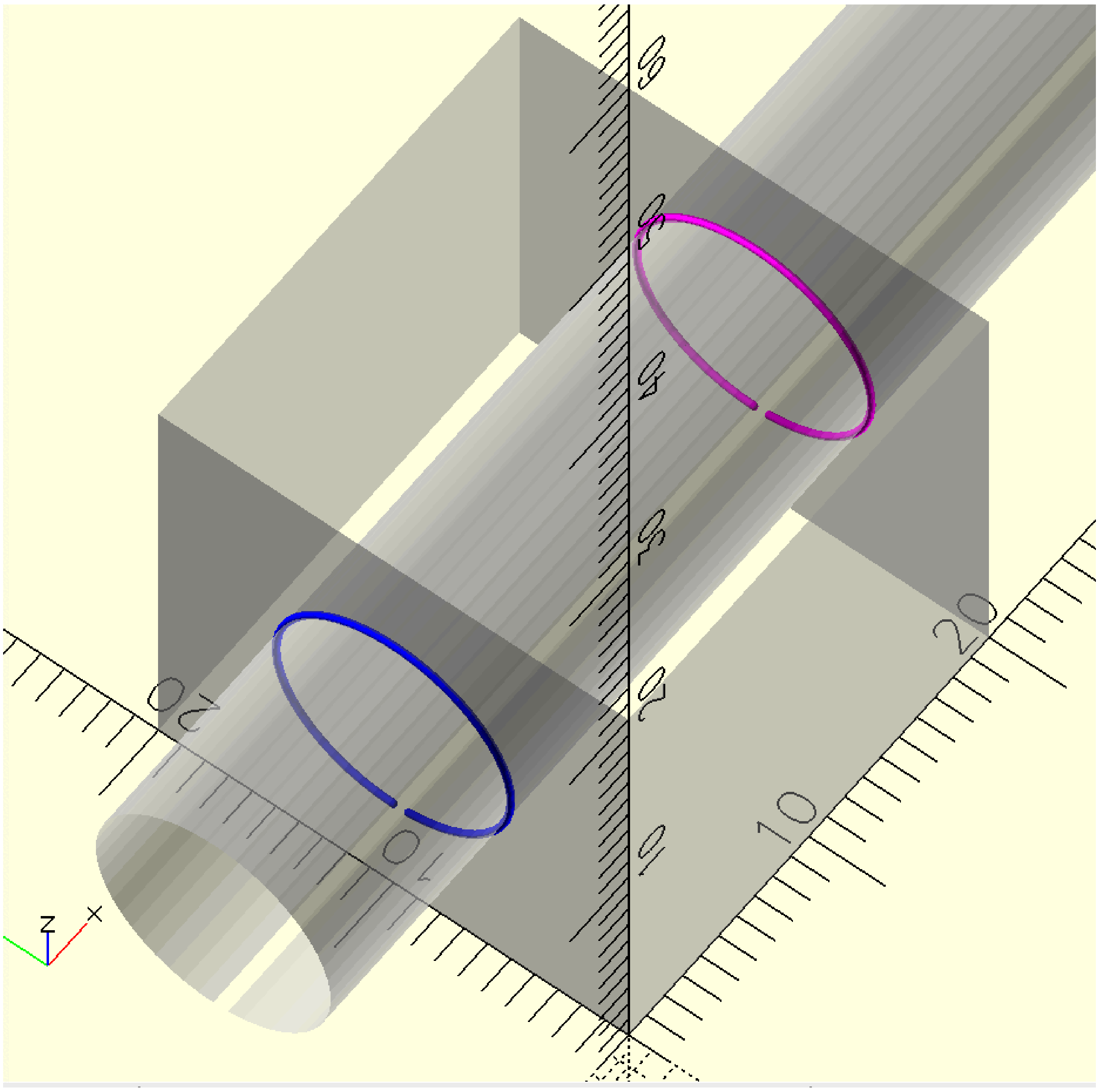


intersection between 2 surfaces

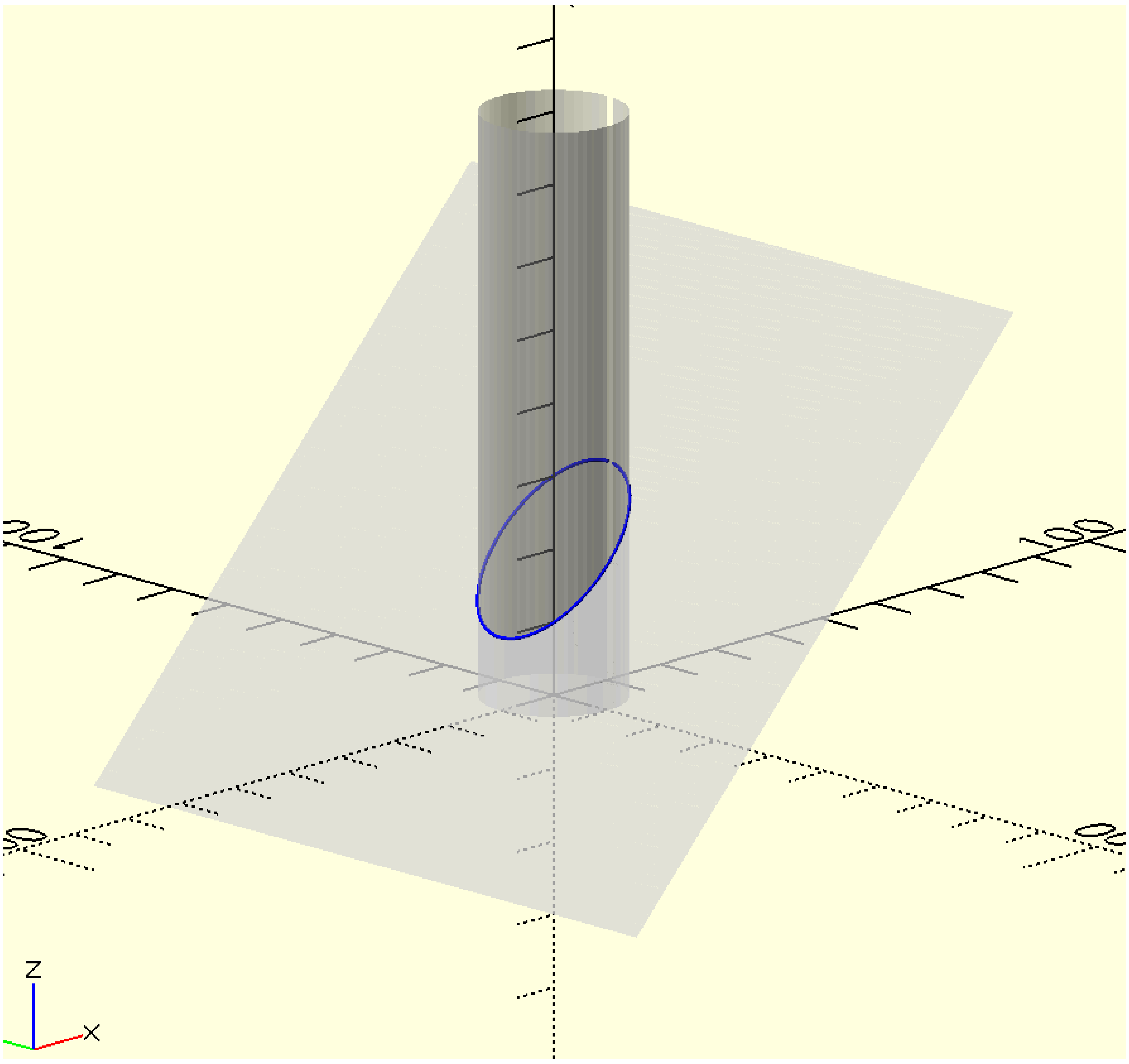
In [28]:

```
s1=linear_extrude(square(20),20)
s2=translate([-10,10,10],rot('y90',linear_extrude(circle(5),50)))
l1=ip_sol2sol(s1,s2,n=-1)
l2=ip_sol2sol(s1,s2,n=0)

fileopen(f'''
%{swp_c(s1)}
%{swp_surf(s2)}
color("blue") p_line3d({l1},.3);
color("magenta") p_line3d({l2},.3);
''')
# Note: To debug issues related to intersection:
# There are 2 surfaces surface1 (s1 in this case) and surface2(s2 in this case)
# surface 1 is intersected by surface 2
# So surface1 should be rendered with module "swp_c"
# surface2 should be rendered with module "swp_surf"
```

```
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3374: RuntimeWarning: divide by zero encountered in divide
  t=einsum('kl,ijkl->ijk',cross(p01,p02),la[:,:,None]-p0)/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))+.00000)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3375: RuntimeWarning: divide by zero encountered in divide
  u=einsum('ijkl,ijkl->ijk',cross(p02[None,None,:,:],(-lab)[:,:,None,:]),(la[:,:,None,:]-p0[None,None,:,:]))/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))
+.00000)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3376: RuntimeWarning: divide by zero encountered in divide
  v=einsum('ijkl,ijkl->ijk',cross((-lab)[:,:,None,:],p01[None,None,:,:]),(la[:,:,None,:]-p0[None,None,:,:]))/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))
+.00000)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3377: RuntimeWarning: invalid value encountered in add
  condition=(t>=0)&(t<=1)&(u>=0)&(u<=1)&(v>=0)&(v<=1)&(u+v<1)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3379: RuntimeWarning: invalid value encountered in multiply
  a=(la[:,None,:,None,:]+lab[:,None,:,None,:]*t[:,None,:,:,None])
```



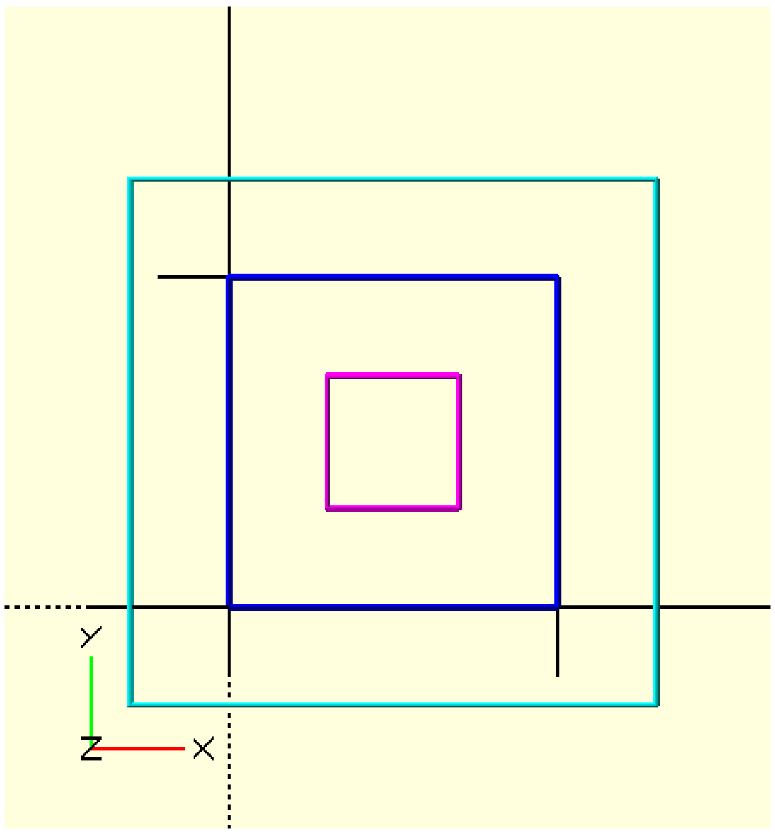
In [29]:
```
pl1=plane([-1,0,1],[100,100],[0,0,20])
c1=cylinder(r=10,h=80)
p0=ip_sol2sol(pl1,c1)
fileopen(f'''
%{swp_c(pl1)}
%{swp_surf(c1)}
color("blue") p_line3d({p0},.5);
''')
```



offset

In [30]:

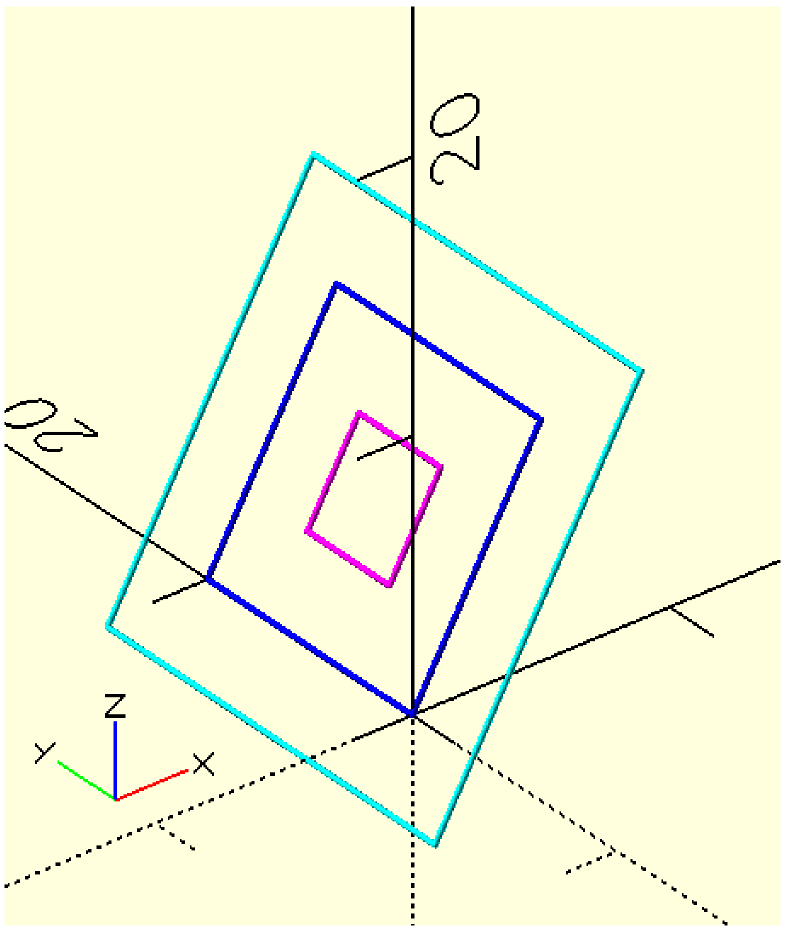
```
sec=square(10)
sec1=offset(sec,-3)
sec2=offset(sec,3)
fileopen(f'''
//original square
color("blue") p_line3dc({sec},.2);
// offset inwards by 3mm
color("magenta") p_line3dc({sec1},.2);
// offset outwards 3mm
color("cyan") p_line3dc({sec2},.2);

''')
```



offset_3d

In [31]:

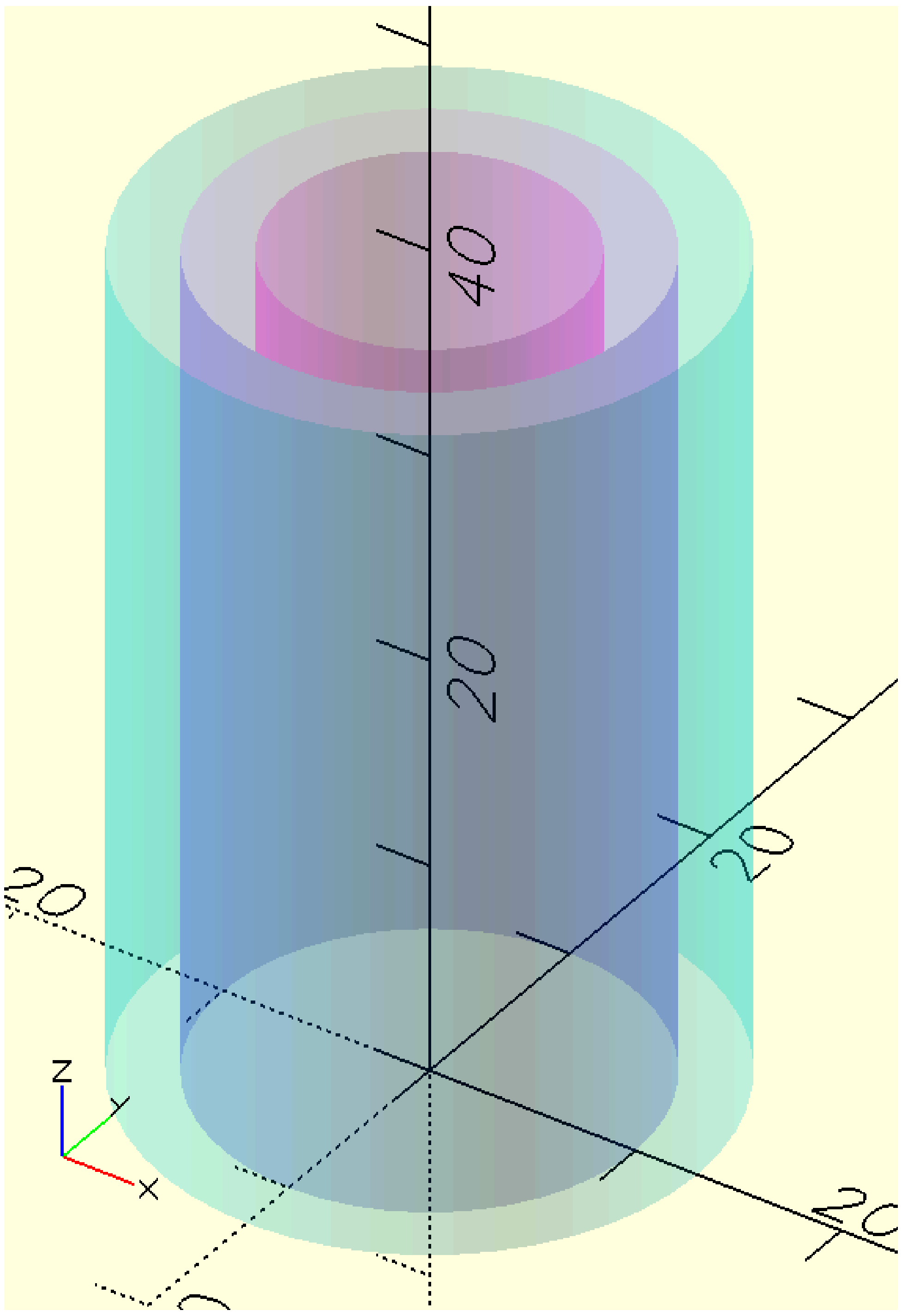
```
sec=rot('y-60',square(10))
sec1=offset_3d(sec,-3)
sec2=offset_3d(sec,3)

fileopen(f'''
//original square
color("blue") p_line3dc({sec},.2);
// offset inwards by 3mm
color("magenta") p_line3dc({sec1},.2);
// offset outwards 3mm
color("cyan") p_line3dc({sec2},.2);

''')
```



offset solids

In [32]:

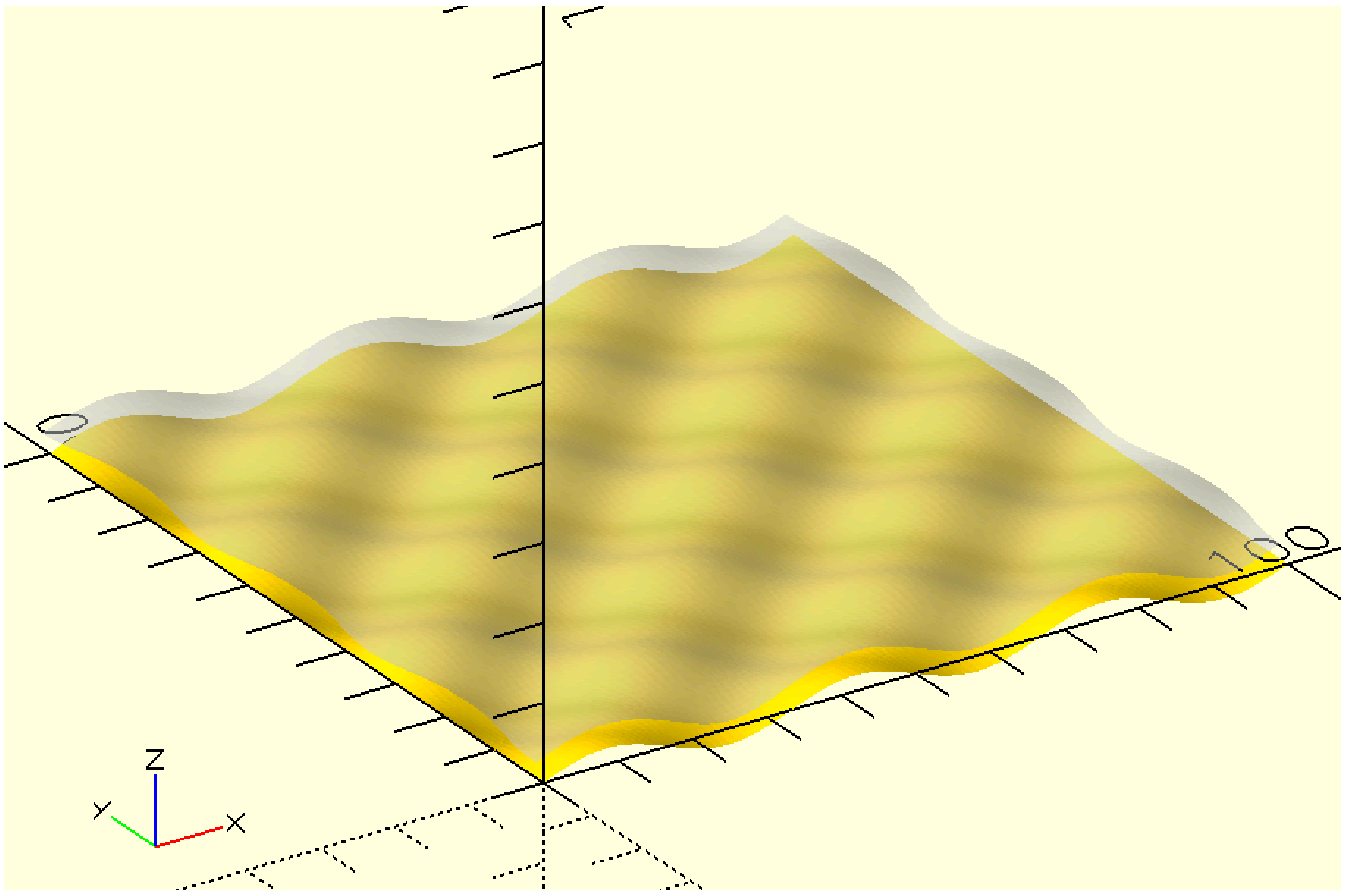
```
c1=cylinder(r=10,h=40)
c2=offset_sol(c1,-3)
c3=offset_sol(c1,3)
fileopen(f'''
//original cylinder
color("blue",.2) swp_c({c1});
// offset inwards by 3mm
color("magenta",.2) swp_c({c2});
// offset outwards 3mm
color("cyan",.2) swp_c({c3});

''')
```



offset surfaces

In [33]:

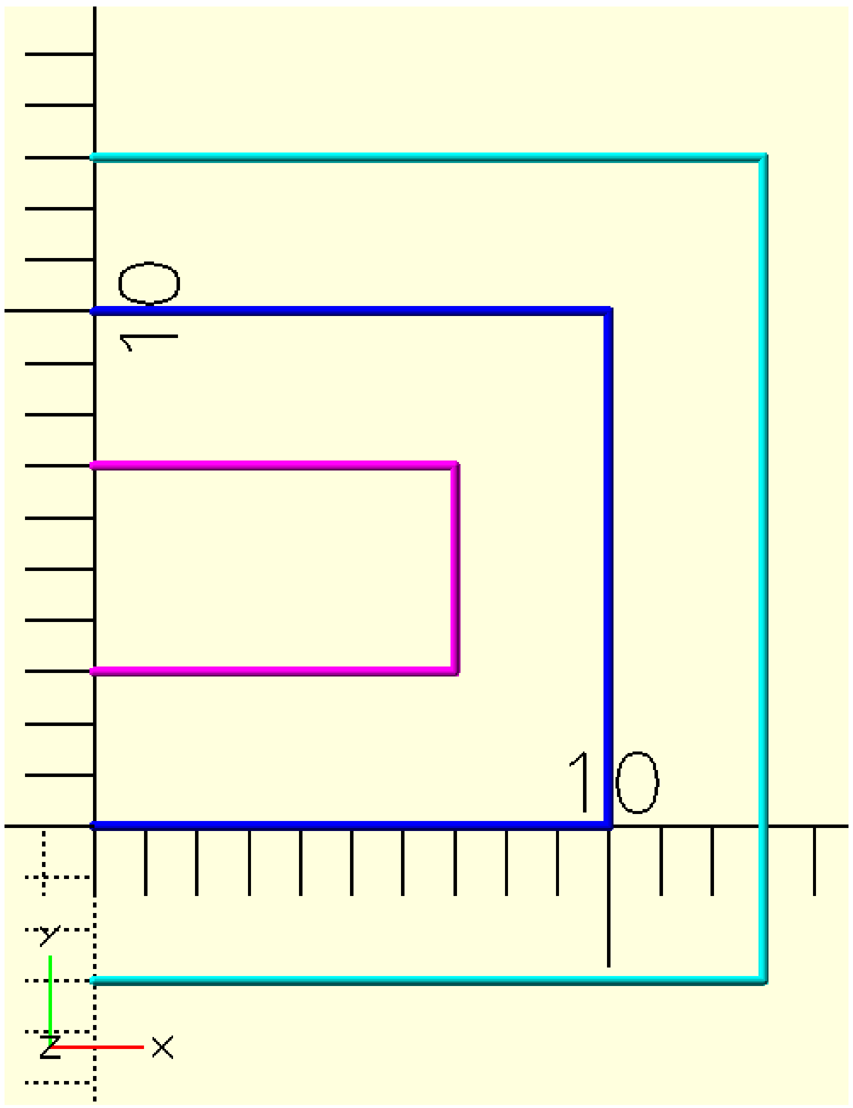
```
w1=rot('x90',sinewave(100,3,2,100))
w2=rot('x90z90',cosinewave(100,3,2,100))
s1=surface_from_2_waves(w1,w2,2)
s2=surface_offset(s1,3)
fileopen(f'''

//original surface
{swp_surf(s1)}
// offset surface
%{swp_surf(s2)}
''')
```



offset path or polylines

In [34]:

```
s1=square(10)
s2=path_offset(s1,-3)
s3=path_offset(s1,3)

fileopen(f'''
//original polyline
color("blue") p_line3d({s1},.2);
// offset inwards by 3mm
color("magenta") p_line3d({s2},.2);
// offset outwards 3mm
color("cyan") p_line3d({s3},.2);

''')
```

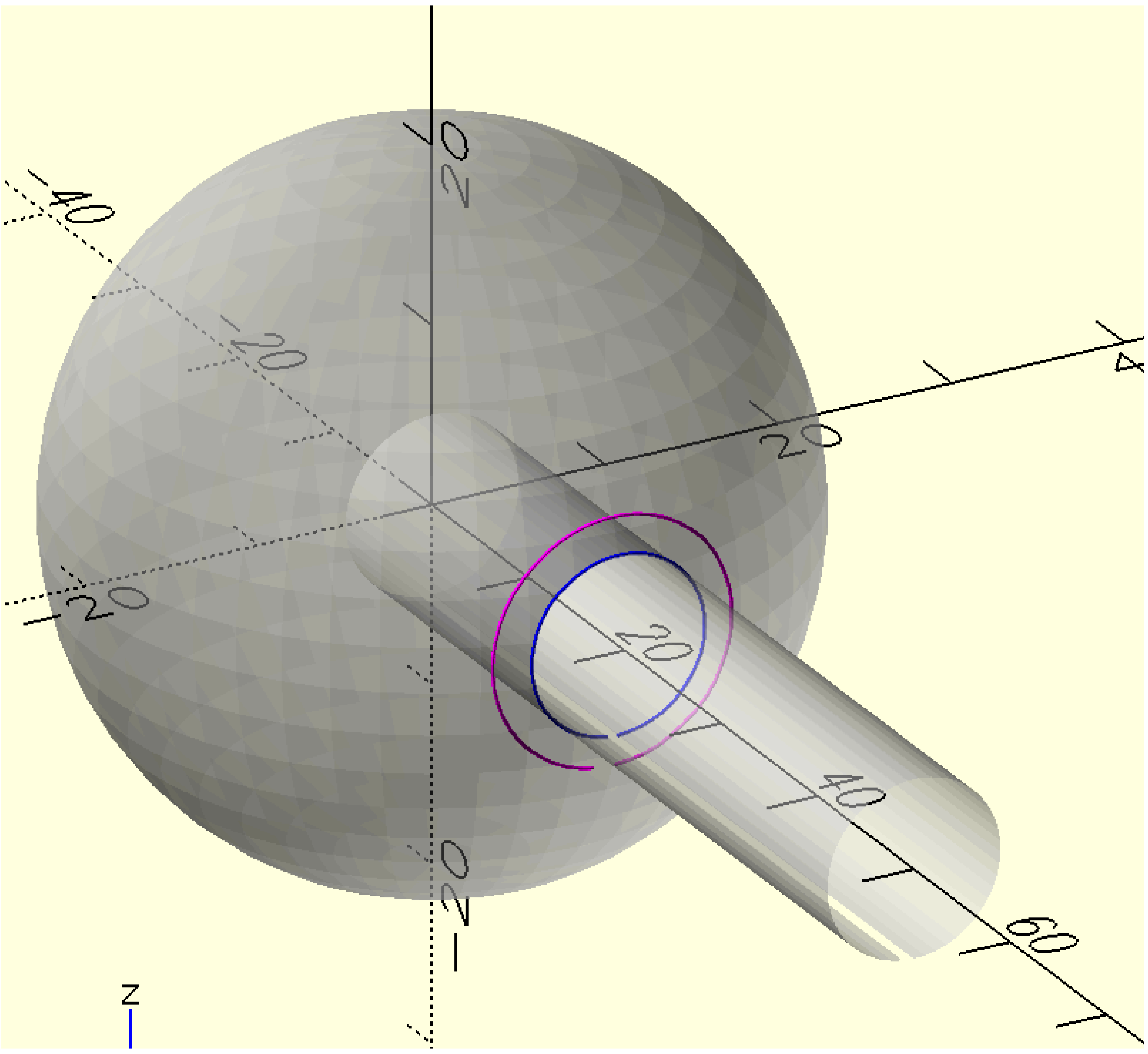


offset a polyline on surface

In [35]:

```
s1=sphere(20)
s2=rot('y90',cylinder(r=5,h=50))
l1=ip_sol2sol(s1,s2)
l2=o_3d(l1,s1,-2)

fileopen(f'''
%{swp_c(s1)}
%{swp_surf(s2)}
// original intersection line
color("blue") p_line3d({l1},.2);

// offset line on sphere
color("magenta") p_line3d({l2},.2);
''')
```

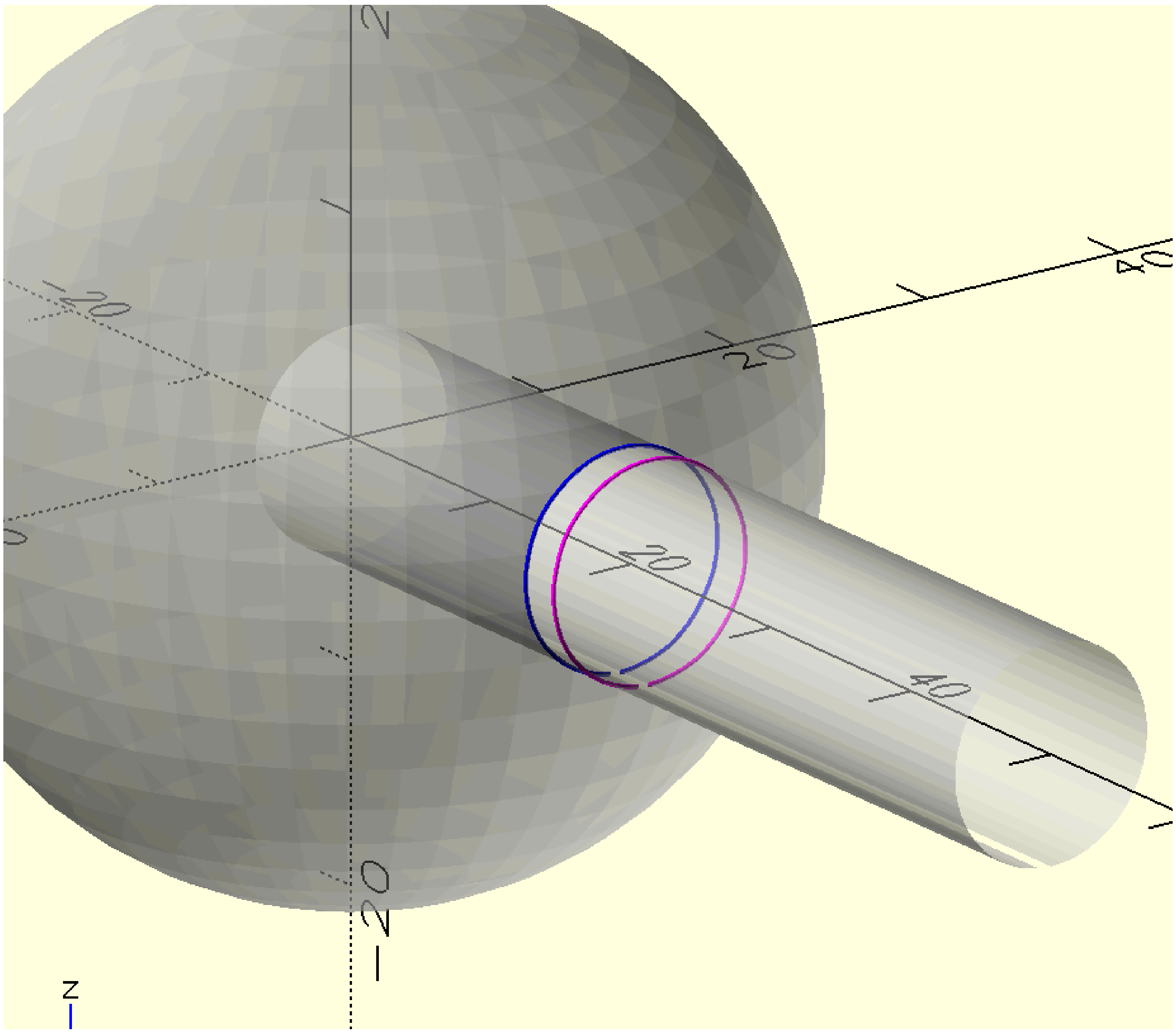
```
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3374: RuntimeWarning: invalid value encountered in divide
  t=einsum('kl,ijkl->ijk',cross(p01,p02),la[:,:,None]-p0)/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))+.00000)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3375: RuntimeWarning: invalid value encountered in divide
  u=einsum('ijkl,ijkl->ijk',cross(p02[None,None,:,:],(-lab)[:,:,None,:]),(la[:,:,None,:]-p0[None,None,:,:]))/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))
+.00000)
/Users/sanjeevprabhakar/python_openscad/openscad3.py:3376: RuntimeWarning: invalid value encountered in divide
  v=einsum('ijkl,ijkl->ijk',cross((-lab)[:,:,None,:],p01[None,None,:,:]),(la[:,:,None,:]-p0[None,None,:,:]))/(einsum('ijl,kl->ijk',(-lab),cross(p01,p02))
+.00000)
```



move the intersection line on the intersecting surface

In [36]:

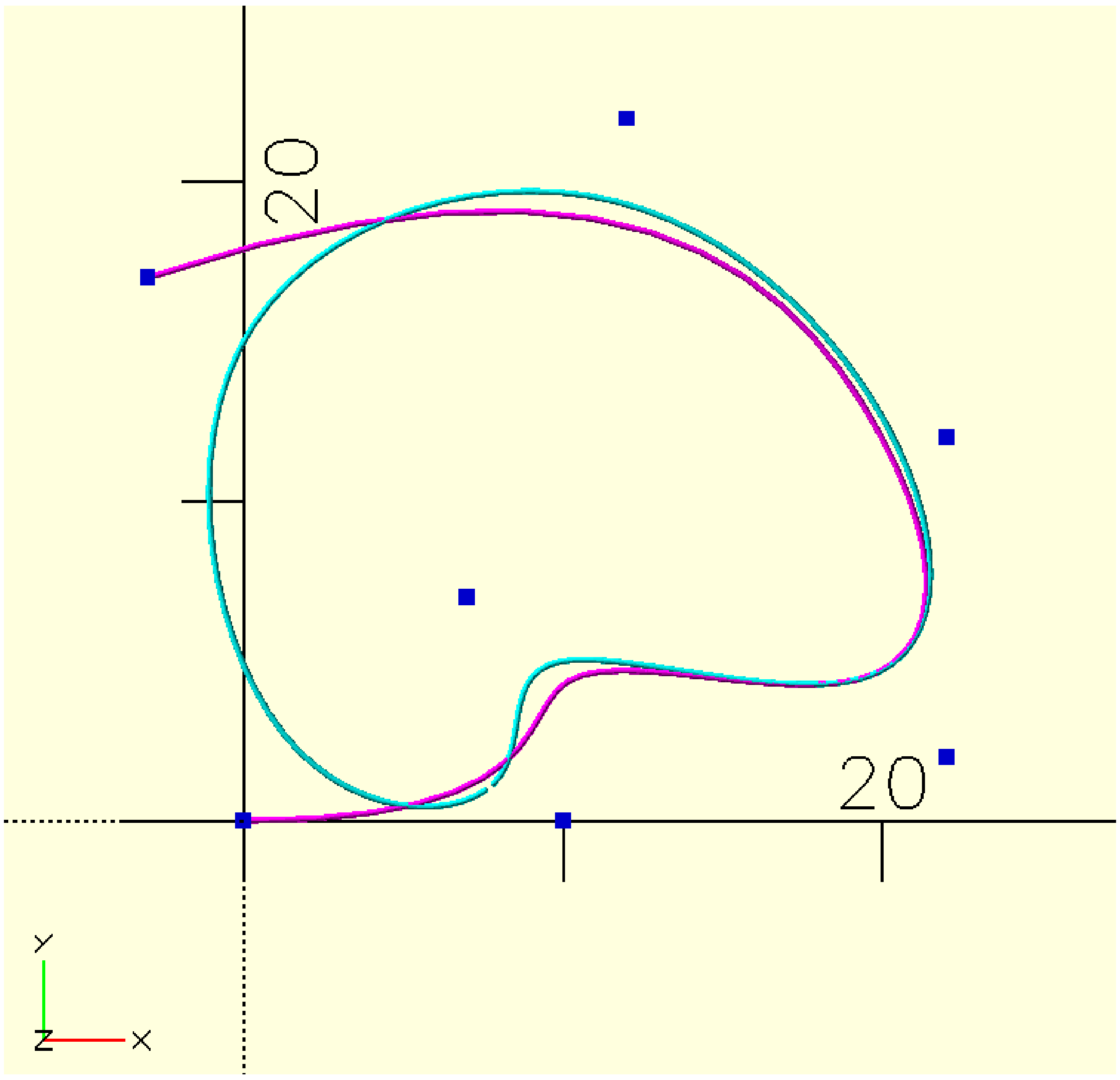
```
s1=sphere(20)
s2=rot('y90',cylinder(r=5,h=50))
l1=ip_sol2sol(s1,s2)
l2=i_p_p(s2,l1,2)

fileopen(f'''
%{swp_c(s1)}
%{swp_surf(s2)}

color("blue") p_line3d({l1},.2);
color("magenta") p_line3d({l2},.2);
''')
```



bspline curves

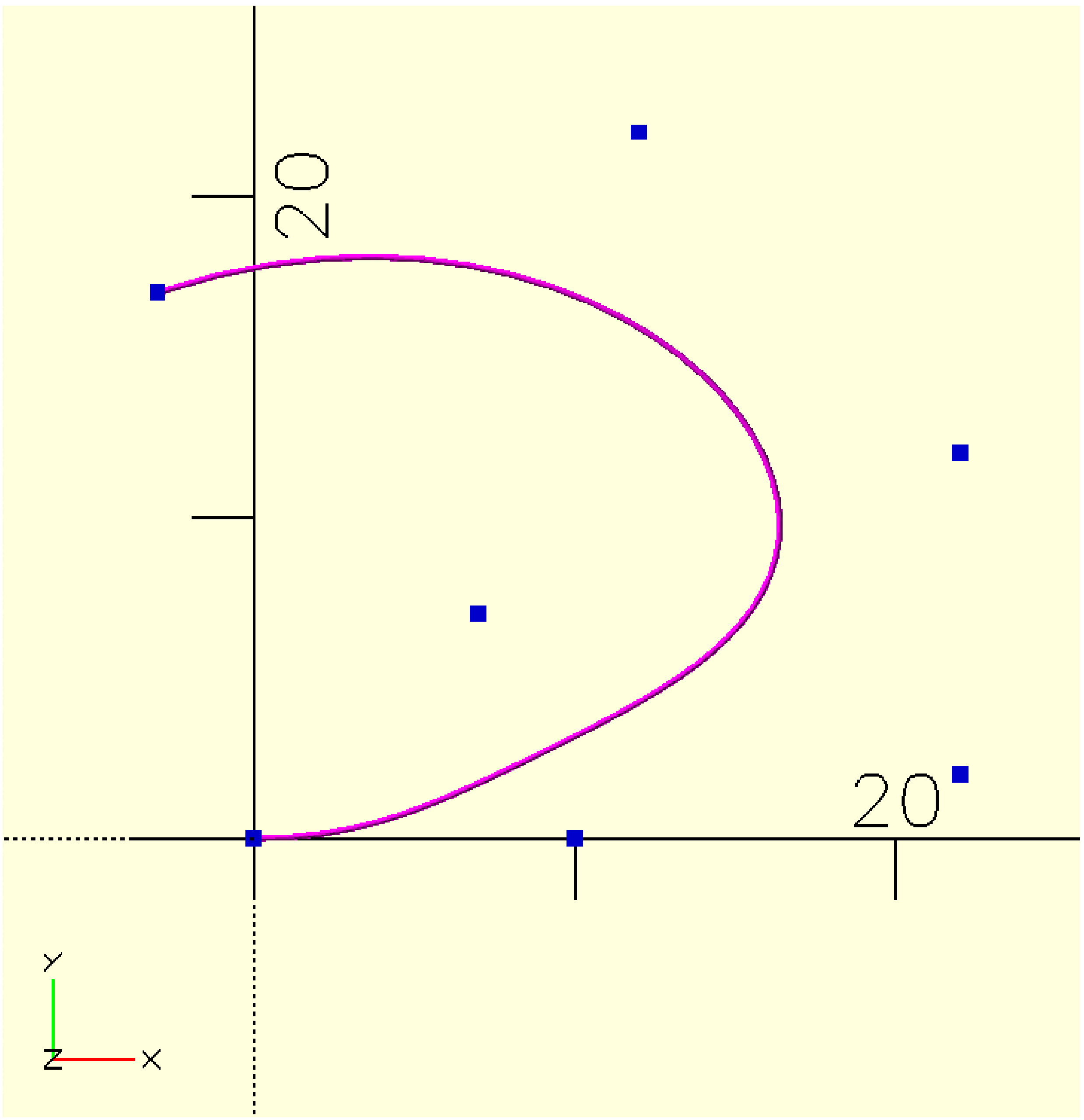
In [37]:

```
l1=cr2dt([[0,0],[10,0],[-3,7],[15,-5],[0,10],[-10,10],[-15,-5]])
l2=bspline_open(l1,3,50)
l3=bspline_closed(l1,3,100)
fileopen(f'''
color("blue") points({l1},.5);
color("magenta") p_line3d({l2},.2);
color("cyan") p_line3d({l3},.2);
''')
```



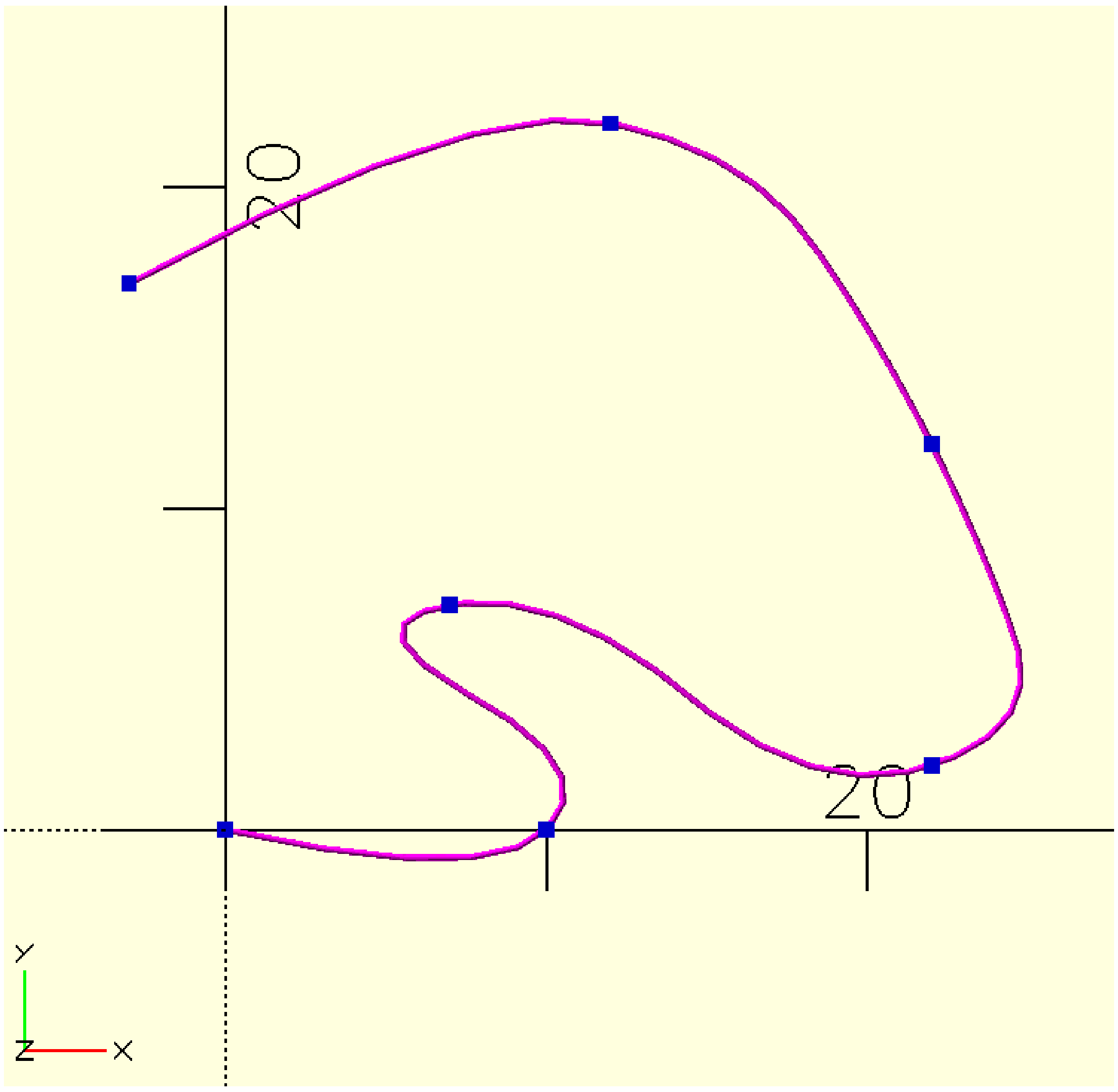
bezier curves

In [38]:
```
l1=cr2dt([[0,0],[10,0],[-3,7],[15,-5],[0,10],[-10,10],[-15,-5]])
l2=bezier(l1,50)

fileopen(f'''
color("blue") points({l1},.5);
color("magenta") p_line3d({l2},.2);

''')
```



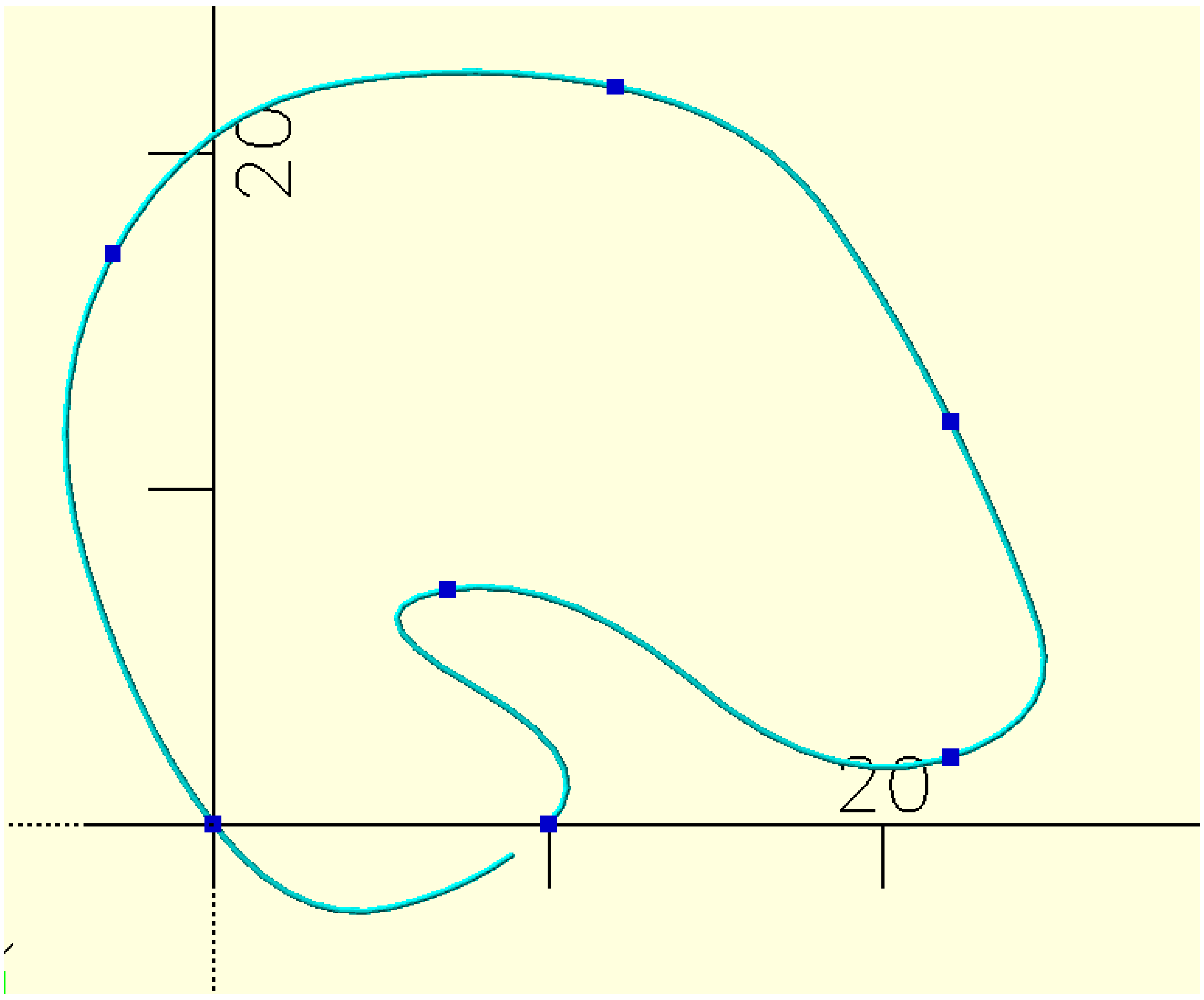
interpolation curves

In [39]:
```
# interpolation through bsplines open loop
l1=cr2dt([[0,0],[10,0],[-3,7],[15,-5],[0,10],[-10,10],[-15,-5]])
l2=interpolation_bspline_open(l1,50)
# l3=interpolation_bspline_closed(l1,50)
fileopen(f'''
color("blue") points({l1},.5);
color("magenta") p_line3d({l2},.2);
//color("cyan") p_line3d({l3},.2);

''')
```



In [40]:

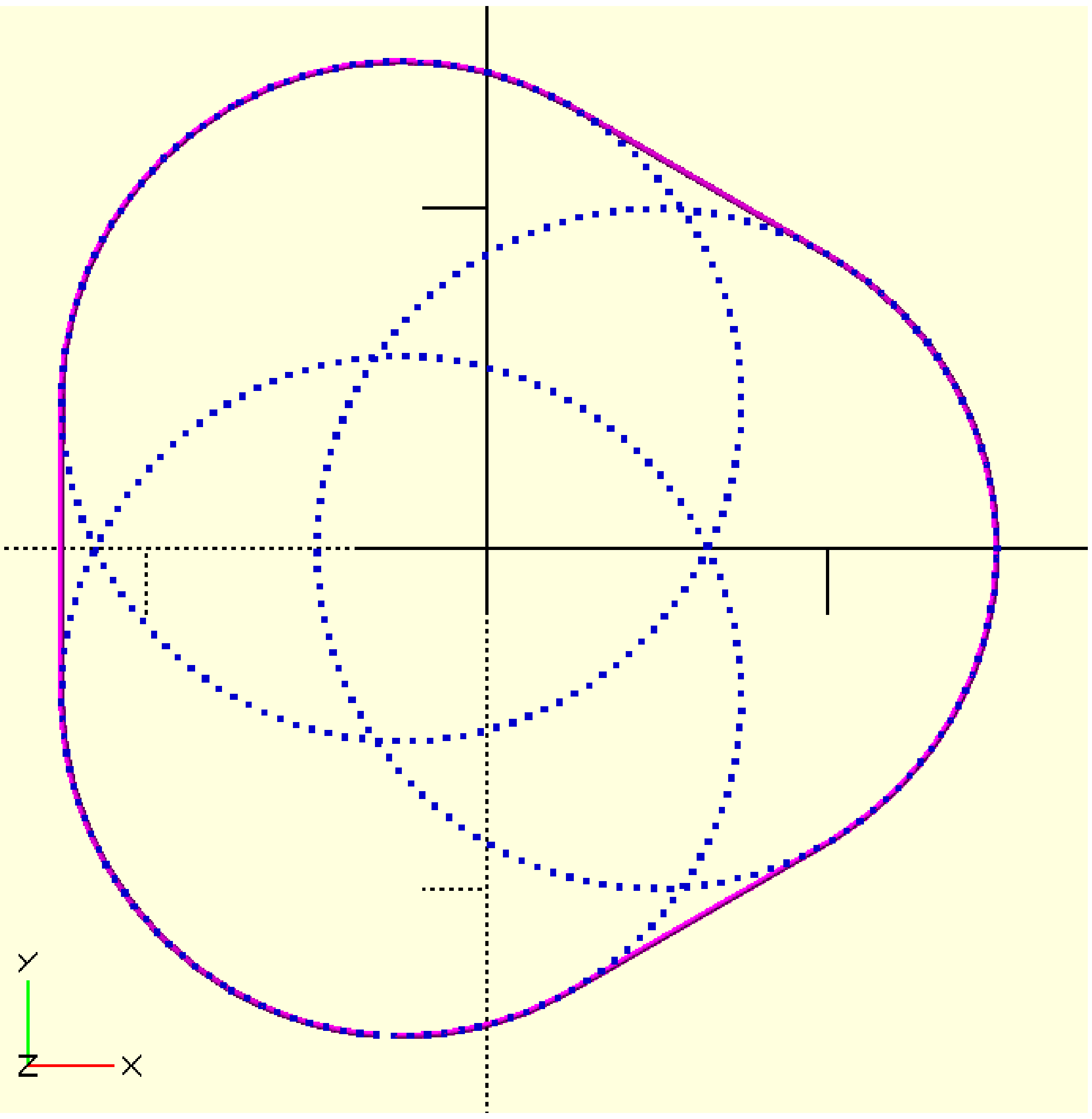
```
# interpolation through bsplines closed loop
l1=cr2dt([[0,0],[10,0],[-3,7],[15,-5],[0,10],[-10,10],[-15,-5]])
# l2=interpolation_bspline_open(l1,50)
l3=interpolation_bspline_closed(l1,100)
fileopen(f'''
color("blue") points({l1},.5);
//color("magenta") p_line3d({l2},.2);
color("cyan") p_line3d({l3},.2);

''')
```



Convex hull

In [41]:

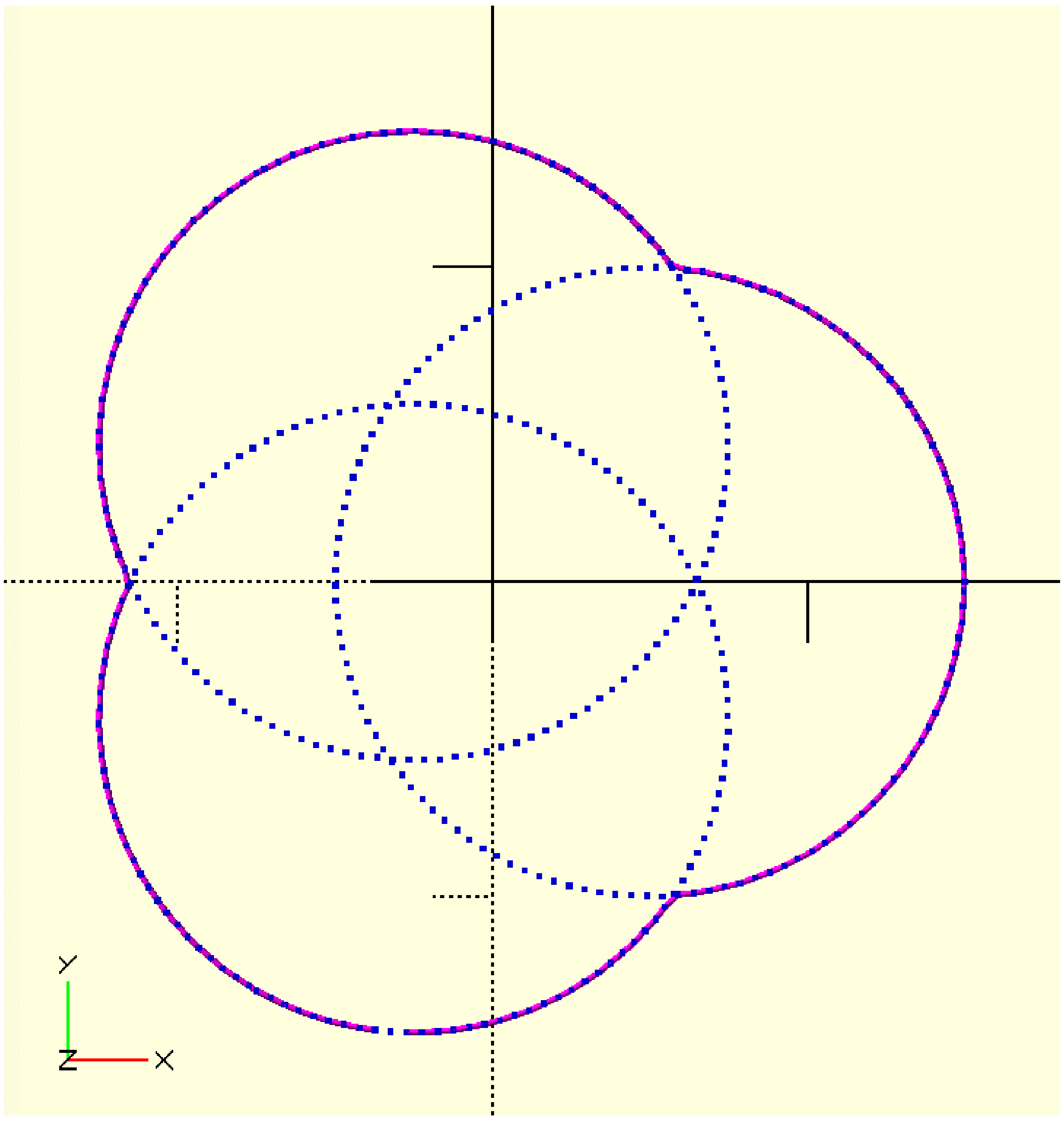
```
c1=circle(10,[5,0])
cx=[ rot2d(i,c1) for i in [0,120,240]]
cx=homogenise(cx,.5,1)
cy=convex_hull(cx)
fileopen(f'''
color("blue") points({cx},.2);
color("magenta") p_line3d({cy},.2);
''')
```



concave hull

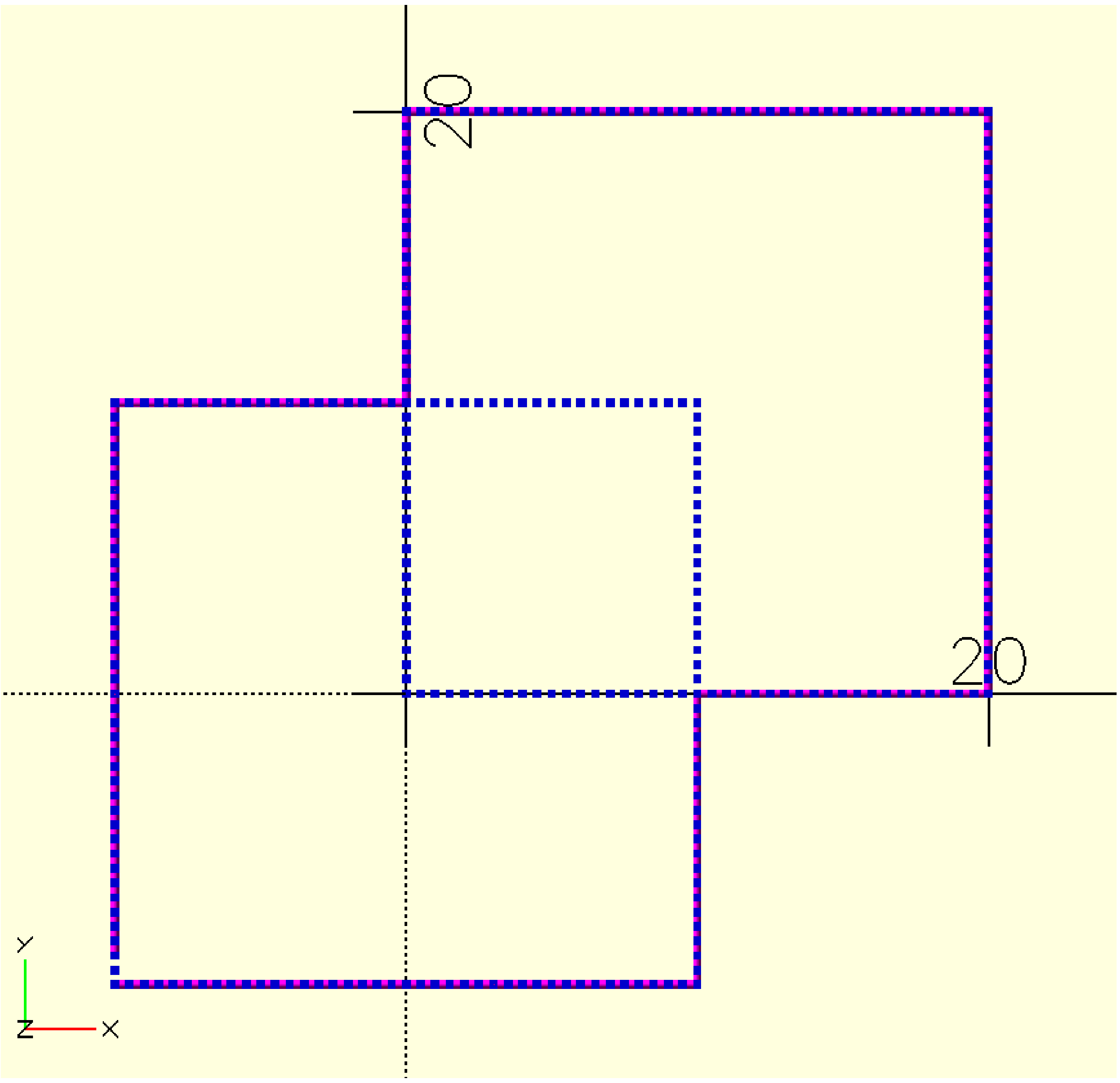
In [42]:

```
# Draw a circle with radius 10 and centered at [5,0]
c1=circle(10,[5,0])

# Create 3 copies of the circle 'c1' rotated at 0, 120 and 240 deg from origin
cx=[ rot2d(i,c1) for i in [0,120,240]]

# homogenise the 3 copies of circles created above, so that distance between
# each subsequent point of circle is 0.5 mm apart and these 3 circles are all
# closed loop sections individually as well
cx=homogenise(cx,pitch=.5,closed_loop=1)

# calculate the concave hull for these points
cy=concave_hull(cx)

fileopen(f'''
color("blue") points({cx},.2);
color("magenta") p_line3d({cy},.2);
polygon({cy});
''')
```



In [43]:

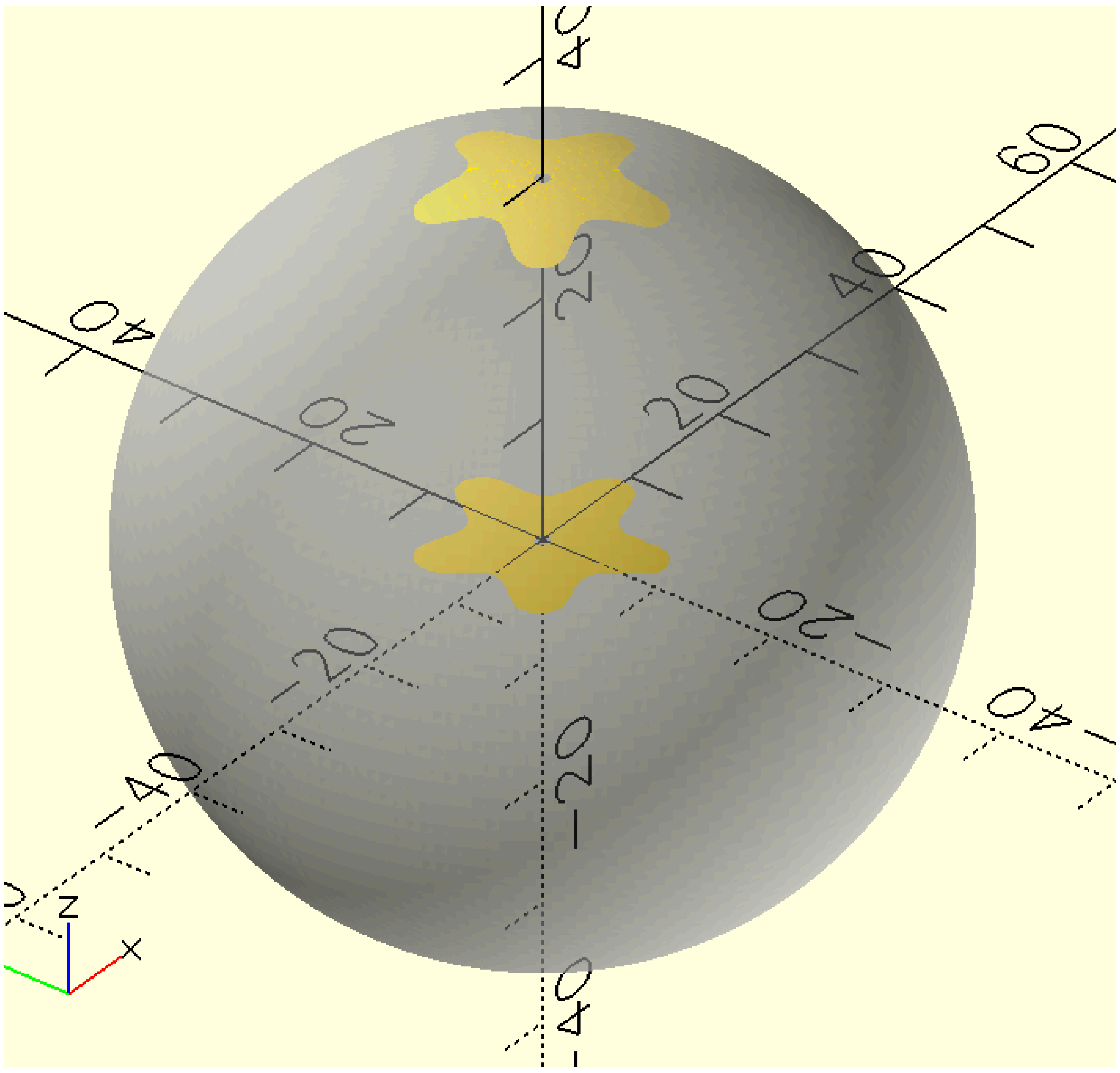
```
s1=square(20)
s2=square(20,center=True)

sx=homogenise([s1,s2],pitch=.5,closed_loop=1)
sy=concave_hull(sx)

fileopen(f'''
color("blue") points({sx},.3);
color("magenta") p_line3d({sy},.3);
//polygon({sy});
''')
```



projection of surface on to another surface

In [44]:

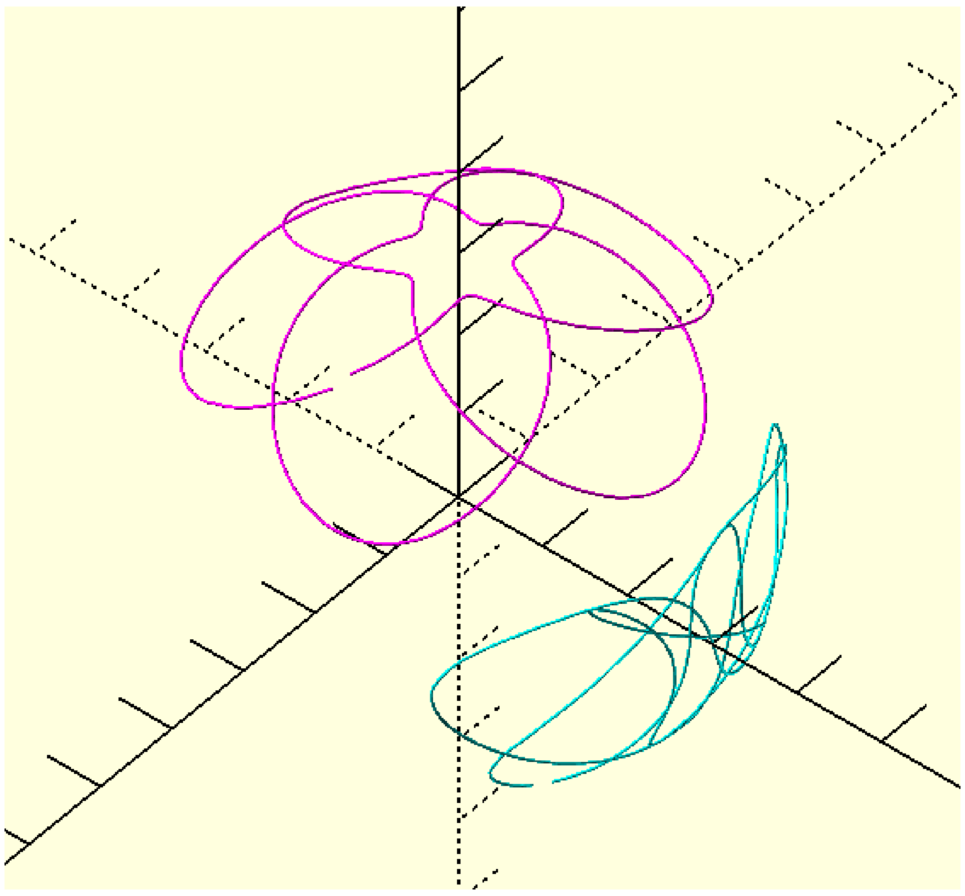
```
s1=sphere(30,s=200)
c1=circle(15,s=6)
c2=rot2d(360/5/2,circle(5,s=6))
s2=a_(c23(concatenate(cpo([c1,c2]))))+[0,0,2]
s2=cr2d(s2,10)
s3=c23([s2,offset(s2,-2.5),offset(s2,-4),offset(s2,-5)])
s3=bspline_surface(s3,3,3,100,10,[1,0])
s4=psos(c_(s1),s3,[0,0,1])
fileopen(f'''
//color("blue") for(p={s4})p_line3dc(p,.03);
%{swp(s1)}
{swp_c(s3)}
{swp_c(s4)}
''')
```



projecting a line on a surface

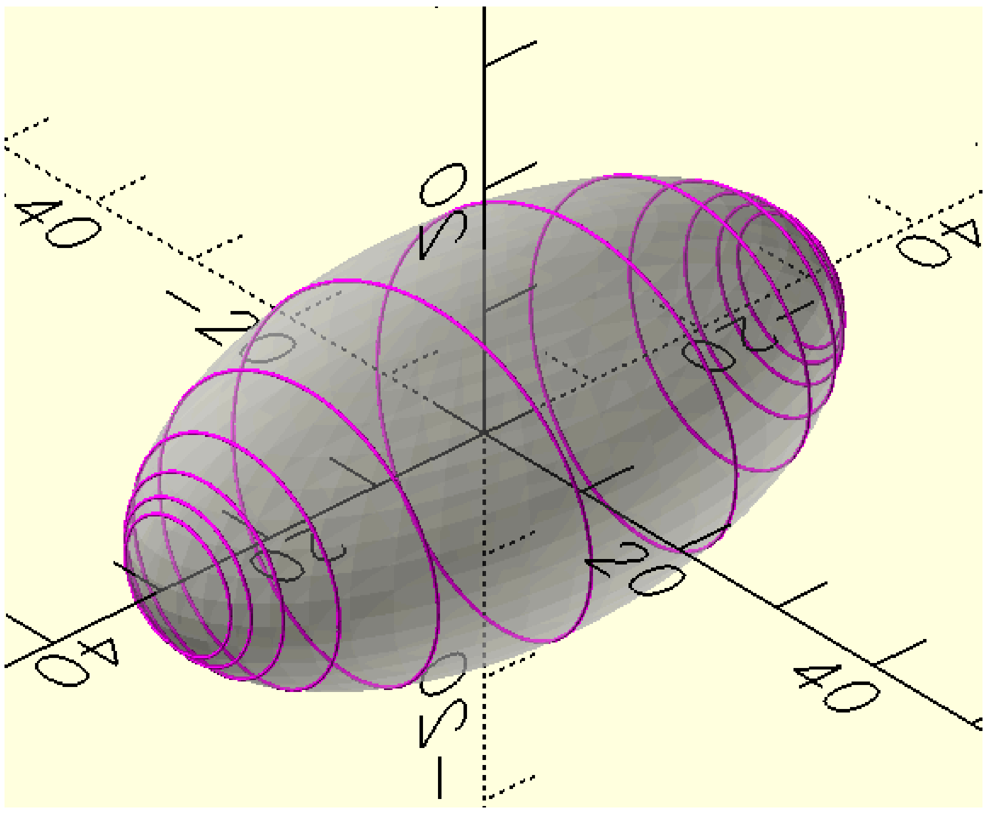
In [45]:

```
l1=rot('y90',helix(10,2,5,5))
l2=c23(arc(15,0,360*3,s=len(l1)-1))
l3=extrude_wave2path(l1,l2)
s1=sphere(30)
l4=plos(c_(s1),l3,[0,0,1])
l5=plos(c_(s1),l3,[0,1,0])
fileopen(f'''
color("blue") p_line3d({l3},.3);
color("magenta") p_line3d({l4},.3);
color("cyan") p_line3d({l5},.3);
''')
```



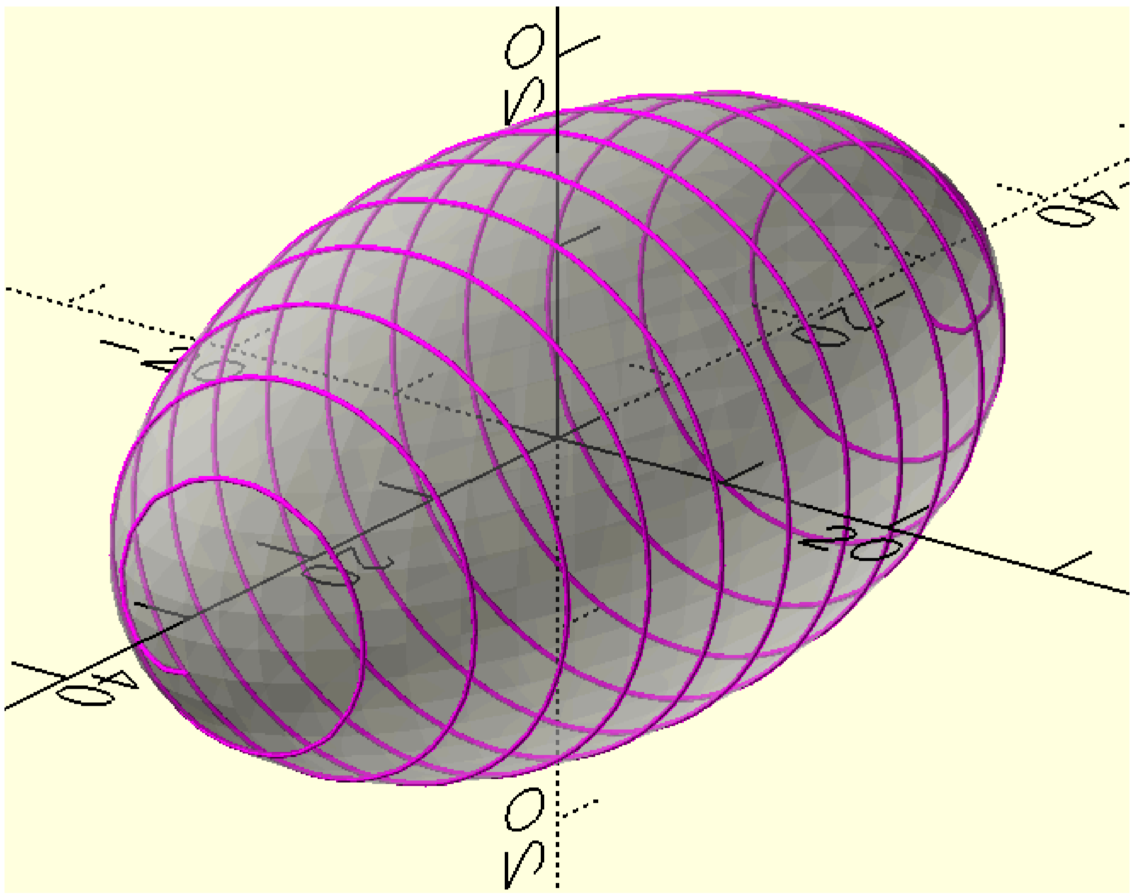
In [46]:

```
l1=translate([-5*12/2,0,0],rot('y90',helix(5,5,12,5)))
s1=rsz3dc(sphere(30),[61,30,30])
l2=plos_v(c_(s1),l1,[0,0,0])
fileopen(f'''
//color("blue") p_line3d({l1},.3);
color("magenta") p_line3d({l2},.3);
%{swp(s1)}
''')
```



In [47]:

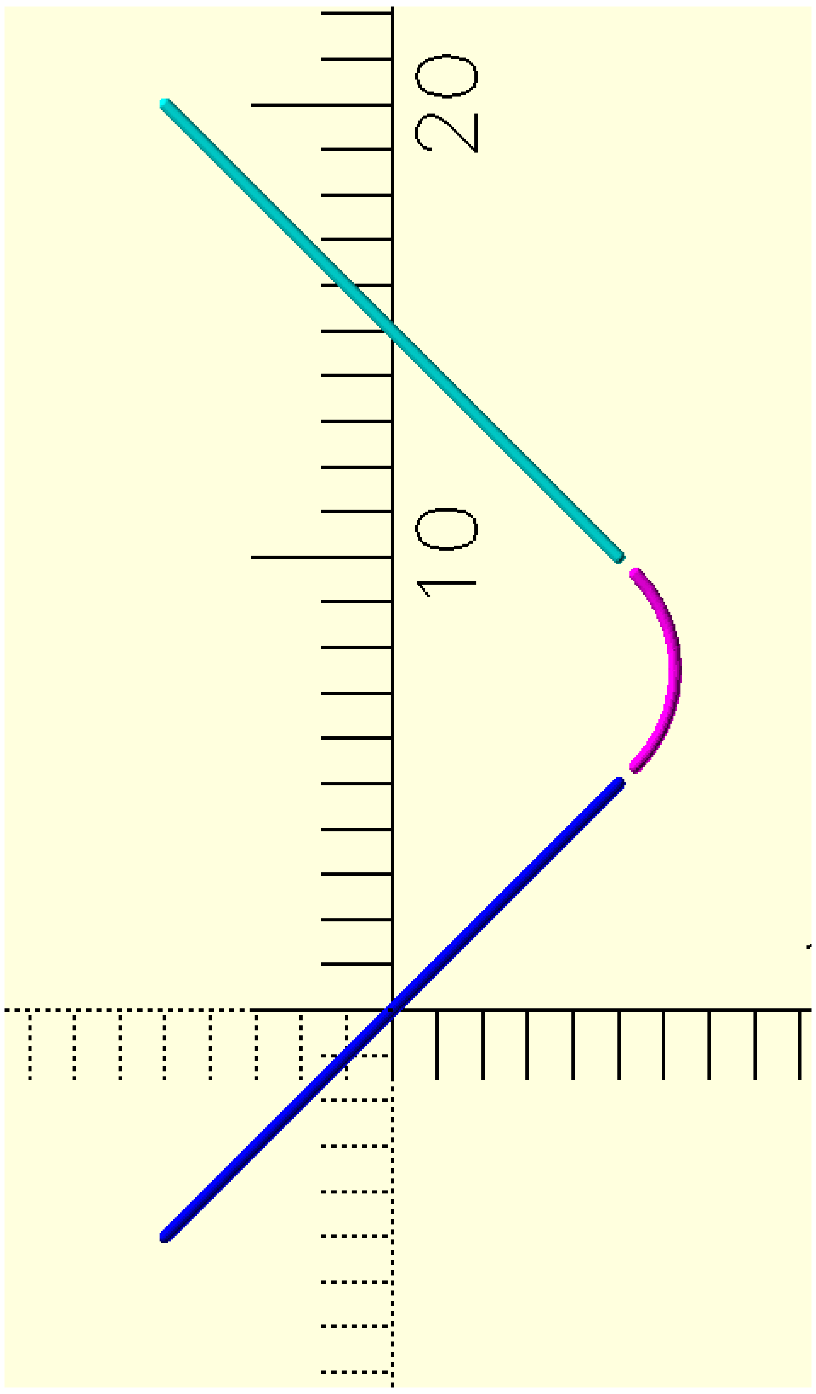
```
l1=translate([-5*12/2,0,0],rot('y90',helix(1,5,12,5)))

s1=rsz3dc(sphere(30),[61,30,30])
l2=plos_v_1(c_(s1),l1,[[0,0,0],[1,0,0]])

fileopen(f'''
//color("blue") p_line3d({l1},.3);
color("magenta") p_line3d({l2},.3);
%{swp(s1)}
''')
```



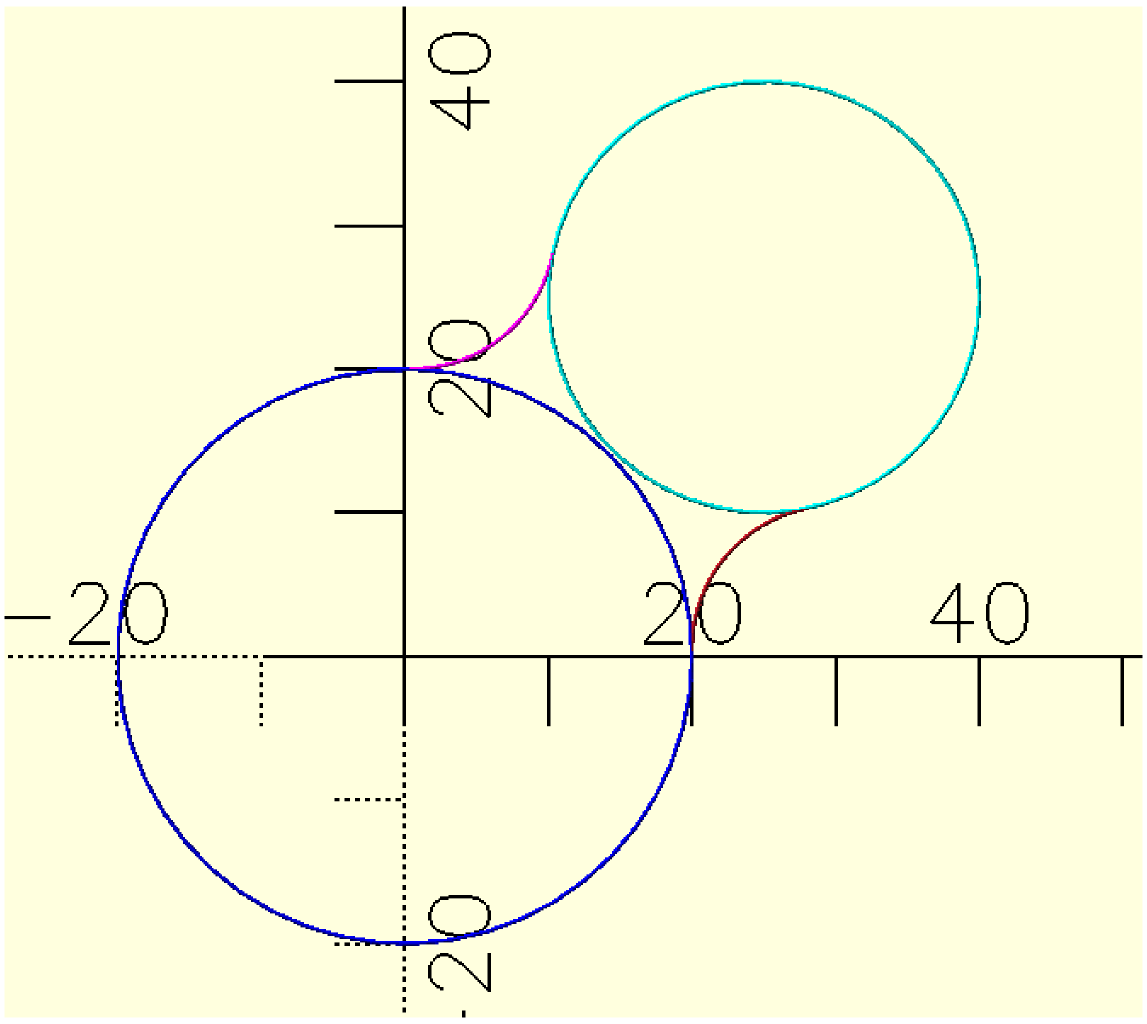
Fillets in 2d

In [48]:

```
# fillets between 2 intersecting lines
l1=point_vector([-5,-5],[10,10])
l2=point_vector([-5,20],[10,-10])
l3=fillet_intersection_lines(l1,l2,r=3)
fileopen(f'''
color("blue") p_line3d({l1},.3);
color("cyan") p_line3d({l2},.3);
color("magenta") p_line3d({l3},.3);
''')
```

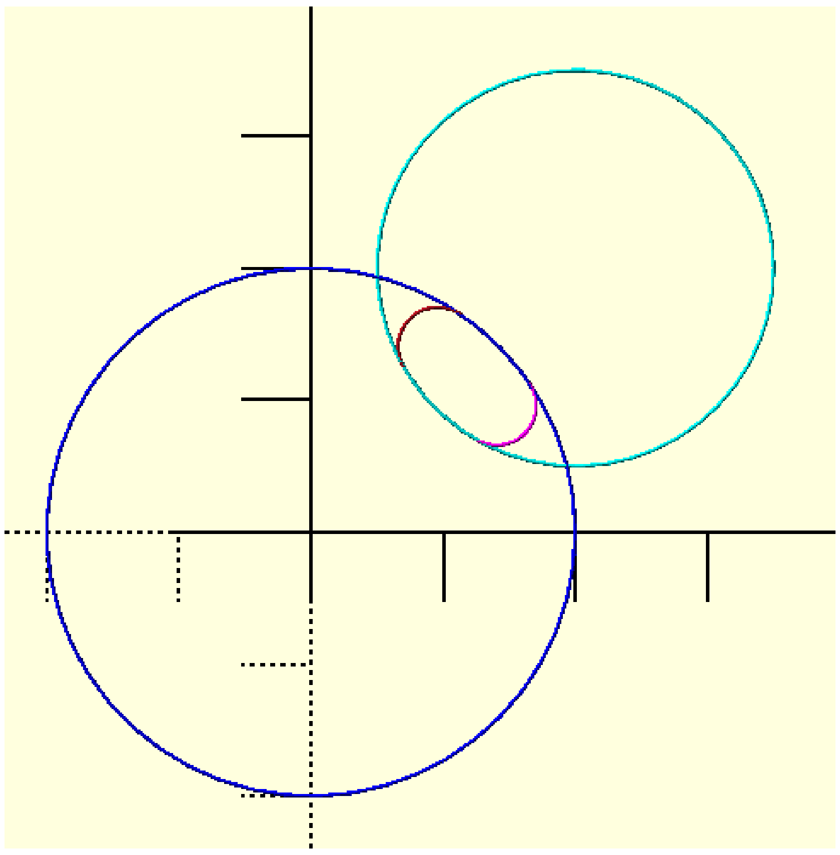


```
In [49]: # fillet between 2 circles
         c1=circle(20)
         c2=circle(15,[25,25])
         f1=two_cir_tarc(c2,c1,r=10)
         f2=two_cir_tarc(c1,c2,r=10)
         fileopen(f'''
         color("blue") p_line3dc({c1},.3);
         color("cyan") p_line3dc({c2},.3);
         color("magenta") p_line3d({f1},.3);
         color("brown") p_line3d({f2},.3);
         ''')
```



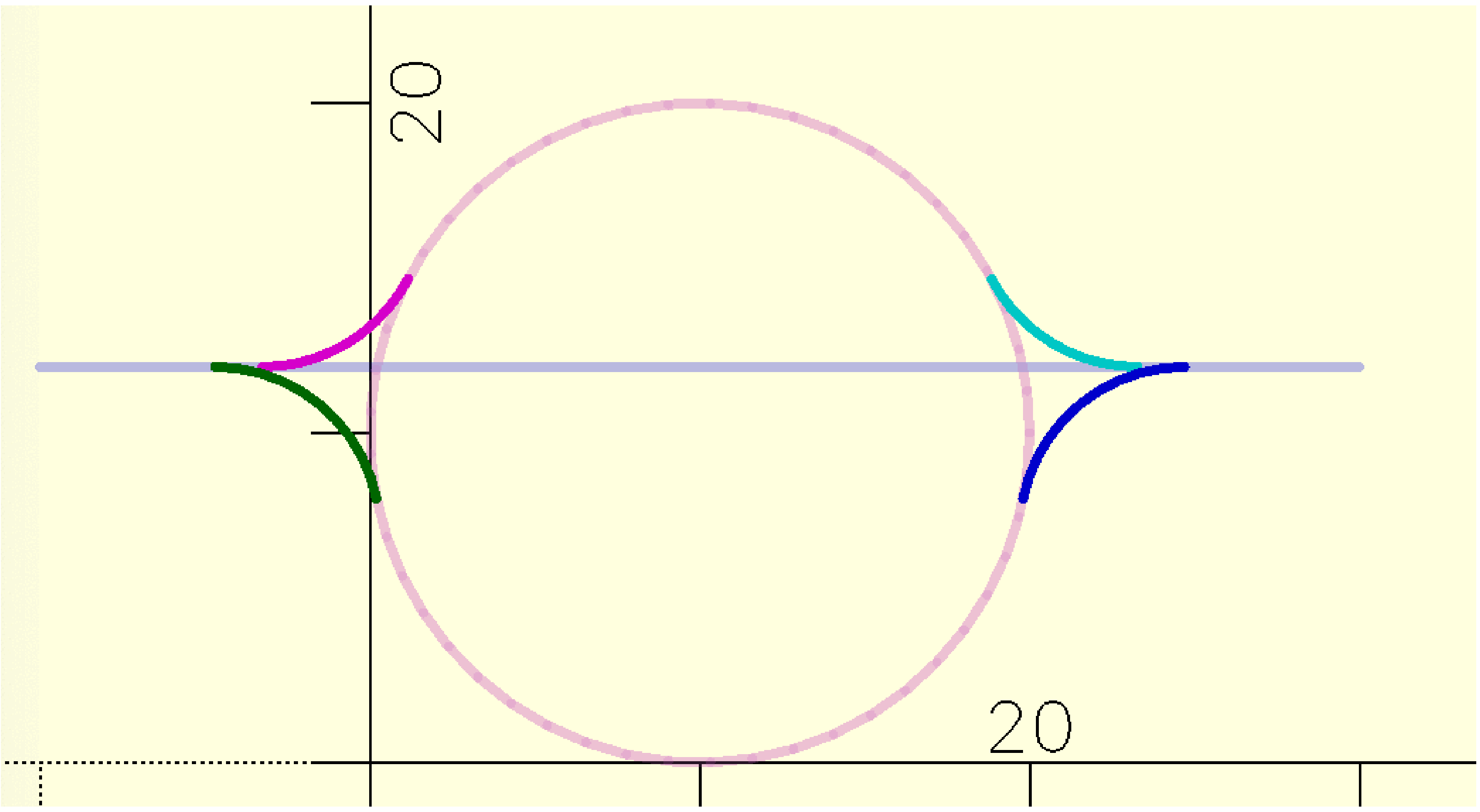
In [50]:

```
# fillet between 2 arcs
c1=circle(20)
c2=circle(15,[20,20])
f1=two_cir_tarc_internal(c2,c1,r=3)
f2=two_cir_tarc_internal(c1,c2,r=3)
fileopen(f'''
color("blue") p_line3dc({c1},.3);
color("cyan") p_line3dc({c2},.3);
color("magenta") p_line3d({f1},.3);
color("brown") p_line3d({f2},.3);
''')
```



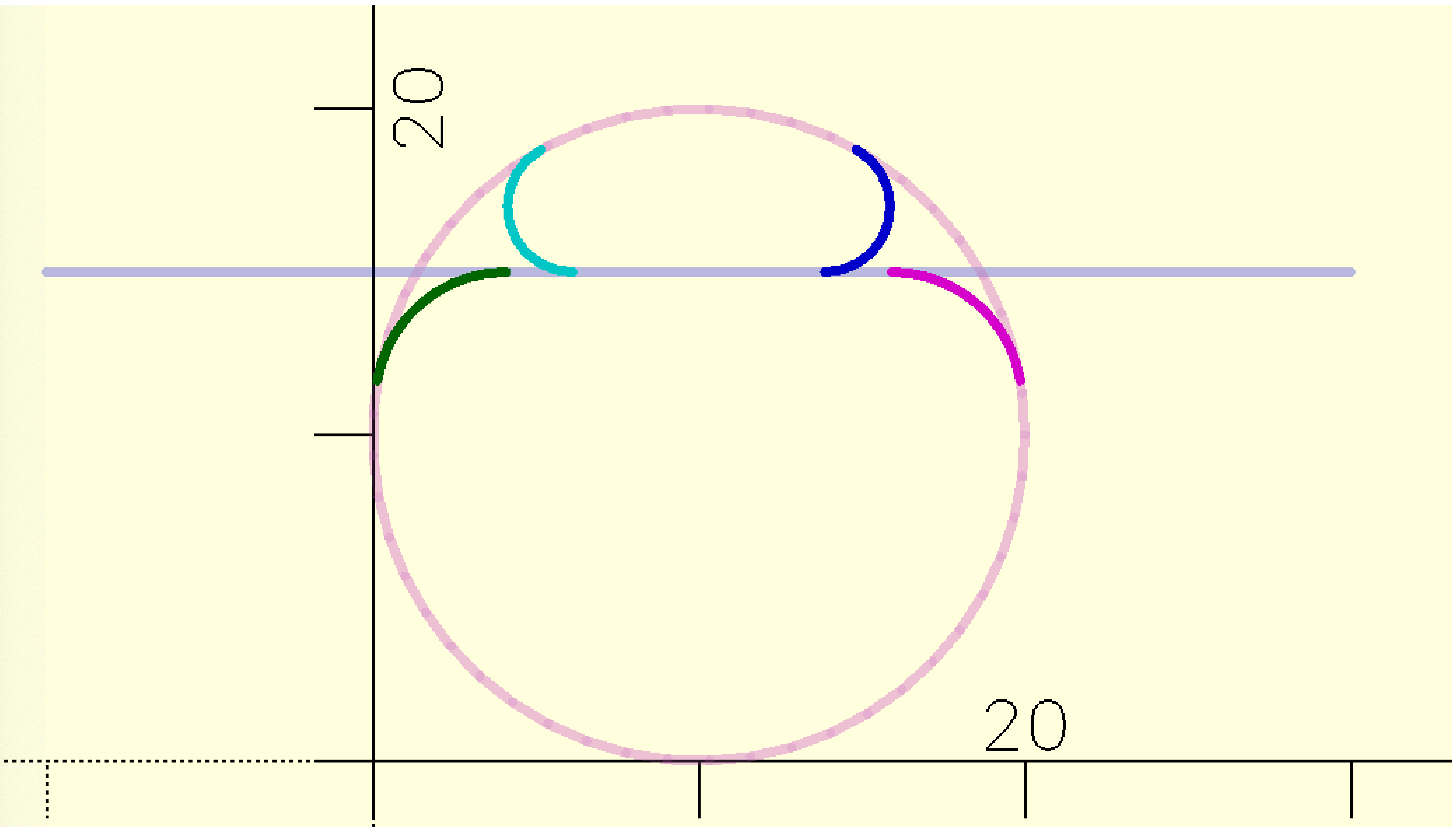
In [51]:

```
# fillet between line and circle (outside)
h=12
line=[[-10,h],[30,h]]
cir1=circle(10,[10,10])
r2=5
s=20
fillet1=fillet_line_circle(line,cir1,r2,1)
fillet2=fillet_line_circle(line,cir1,r2,2)
fillet3=fillet_line_circle(line,cir1,r2,3)
fillet4=fillet_line_circle(line,cir1,r2,4)
fileopen(f'''
color("blue",.1)p_line({line},.3);
color("violet",.2)p_line({cir1},.3);
color("cyan")p_lineo({fillet1},.3);
color("blue")p_lineo({fillet2},.3);
color("magenta")p_lineo({fillet3},.3);
color("green")p_lineo({fillet4},.3);
''')
```



In [52]:

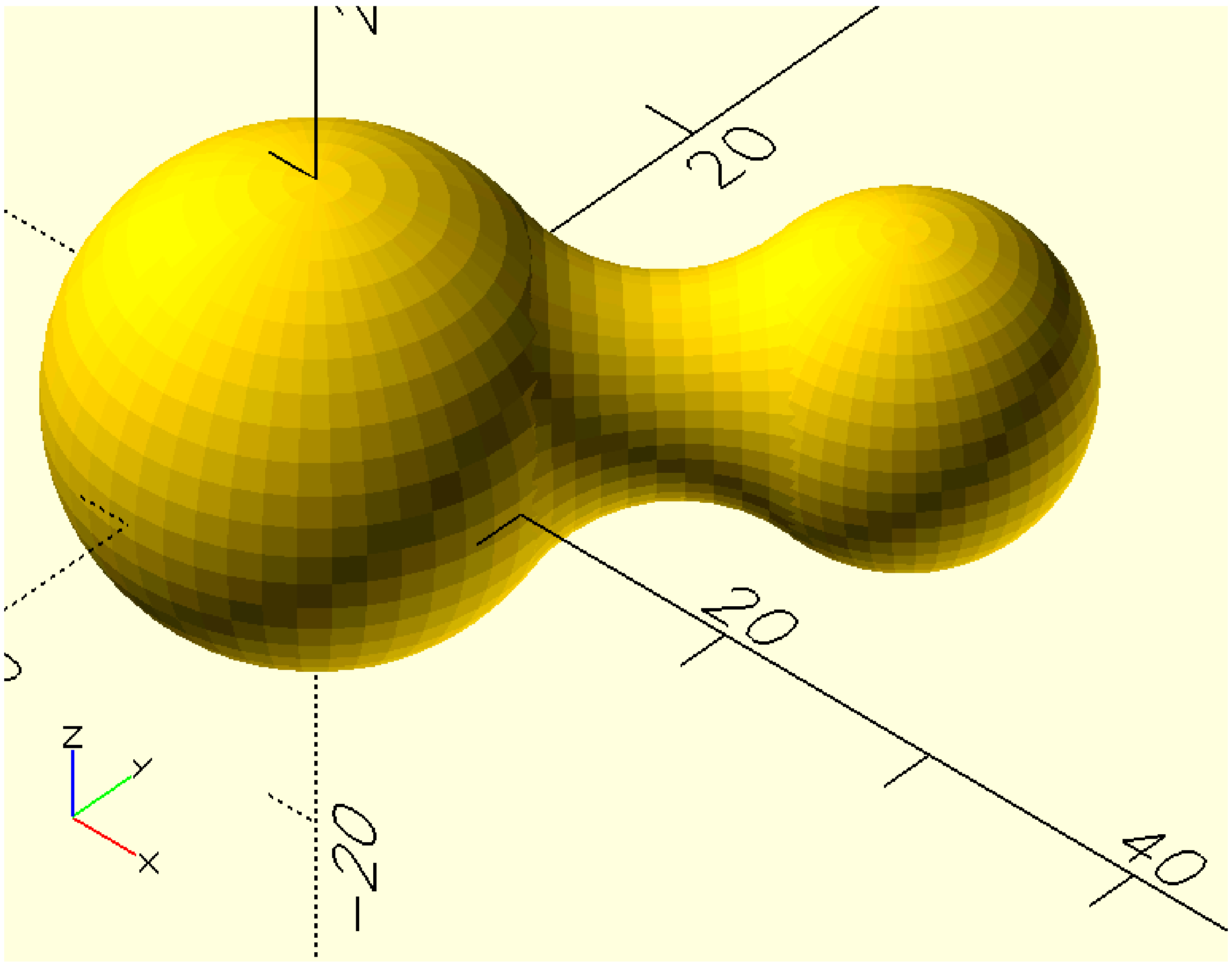
```
# fillet between line and circle (inside)
h=15
line=[[-10,h],[30,h]]
cir1=circle(10,[10,10])
s=20
fillet5=fillet_line_circle_internal(line,cir1,2,1)
fillet6=fillet_line_circle_internal(line,cir1,4,2)
fillet7=fillet_line_circle_internal(line,cir1,2,3)
fillet8=fillet_line_circle_internal(line,cir1,4,4)
fileopen(f'''
color("blue",.1)p_line({line},.3);
color("violet",.2)p_line({cir1},.3);
color("blue")p_lineo({fillet5},.3);
color("magenta")p_lineo({fillet6},.3);
color("cyan")p_lineo({fillet7},.3);
color("green")p_lineo({fillet8},.3);
''')
```



Fillets in 3d

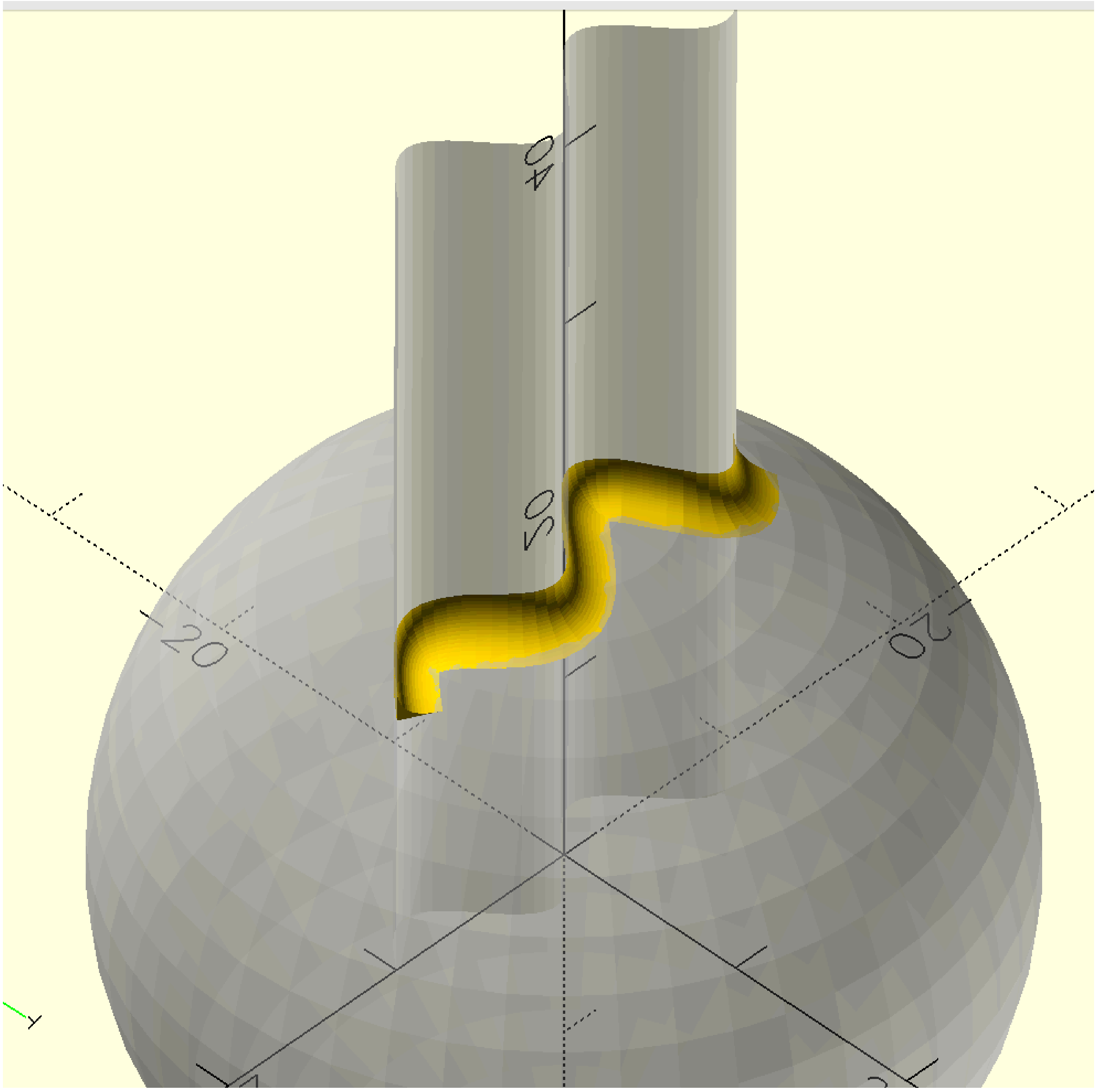
In [53]:

```
# fillet between 2 spheres
s1=sphere(10)
s2=sphere(7,[15,15,0])
f1=fillet_2spheres(s1,s2,7,s1=10,s2=40)
fileopen(f'''
{swp(s1)}
{swp(s2)}
{swp(f1)}
''')
```



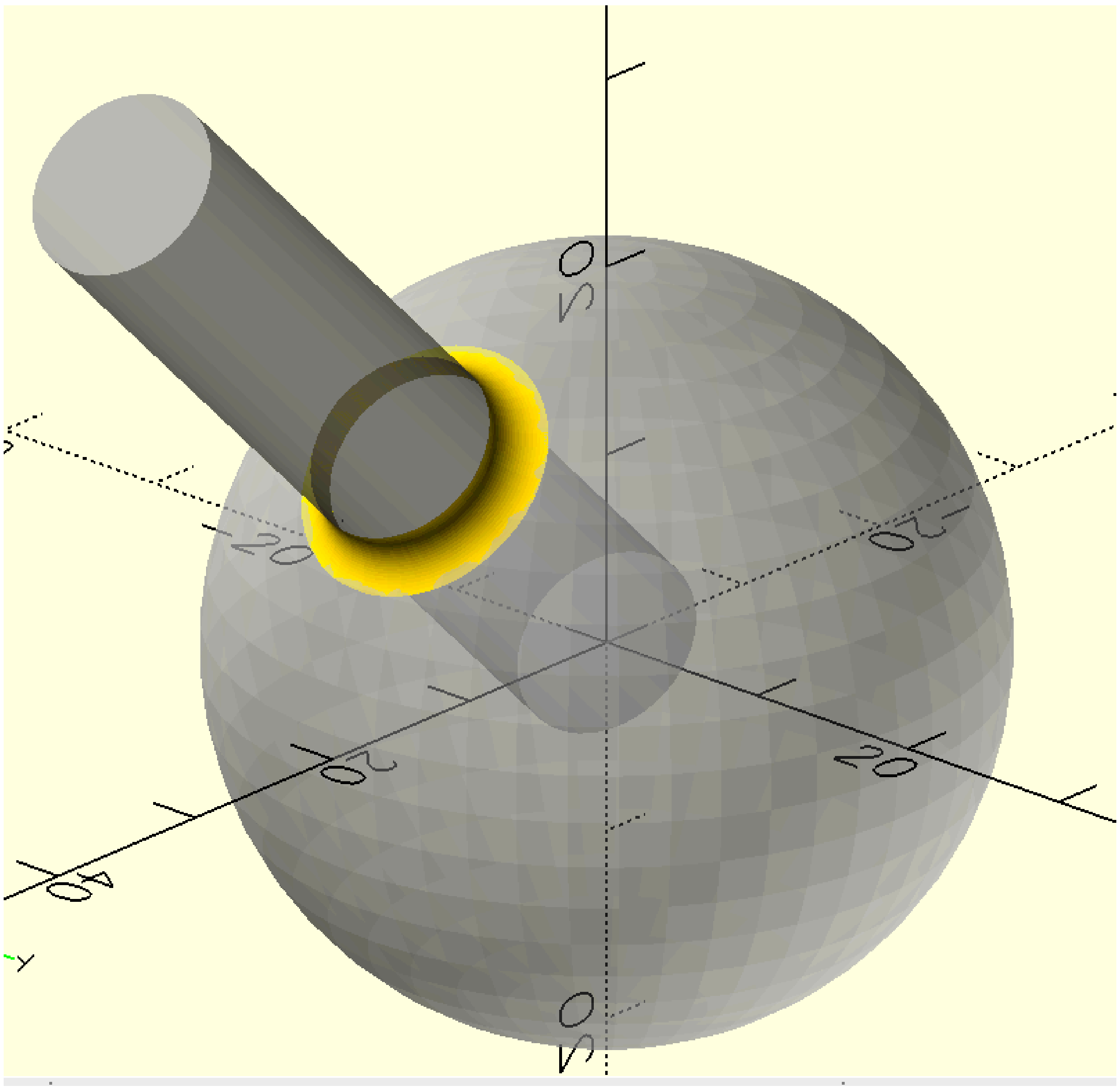
In [54]:

```
# fillet at the intersection of a solid and a surface
s1=sphere(20)
l1=translate([-10,0,0],sinewave(20,2,2,50))
s2=surface_line_vector(l1,[5,5,50])
f1=ip_fillet(s1,s2,2,2)
fileopen(f'''
%{swp(s1)}
%{swp_surf(s2)}
{swp(f1)}
''')
```



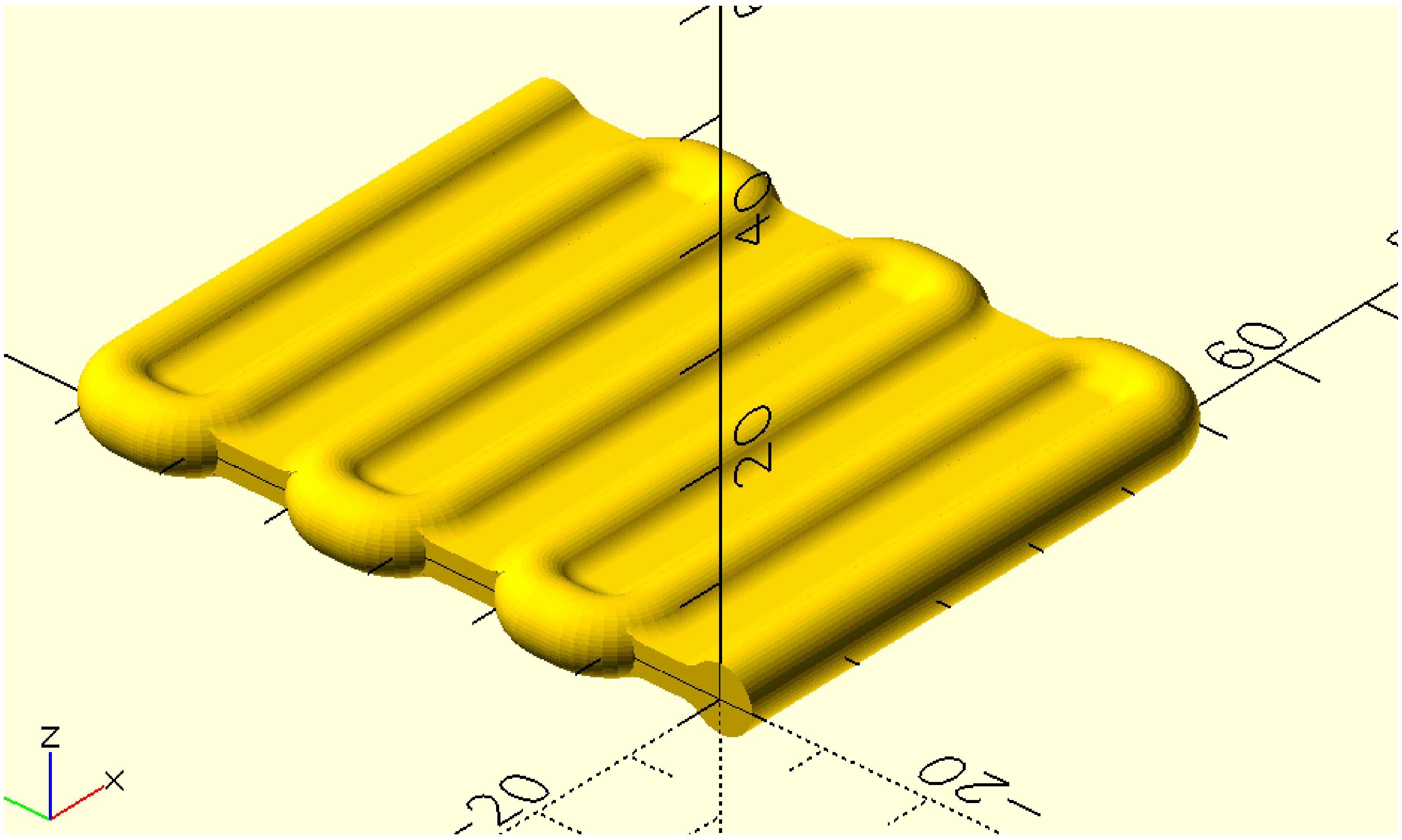
In [55]:

```
# fillets at the insection of 2 solids
s1=sphere(20)
c1=rot('y45',cylinder(r=5,h=50,s=50))
f1=ip_fillet_closed(s1,c1,-2,2)
fileopen(f'''
%{swp(s1)}
%{swp(c1)}
{swp_c(f1)}
''')
```

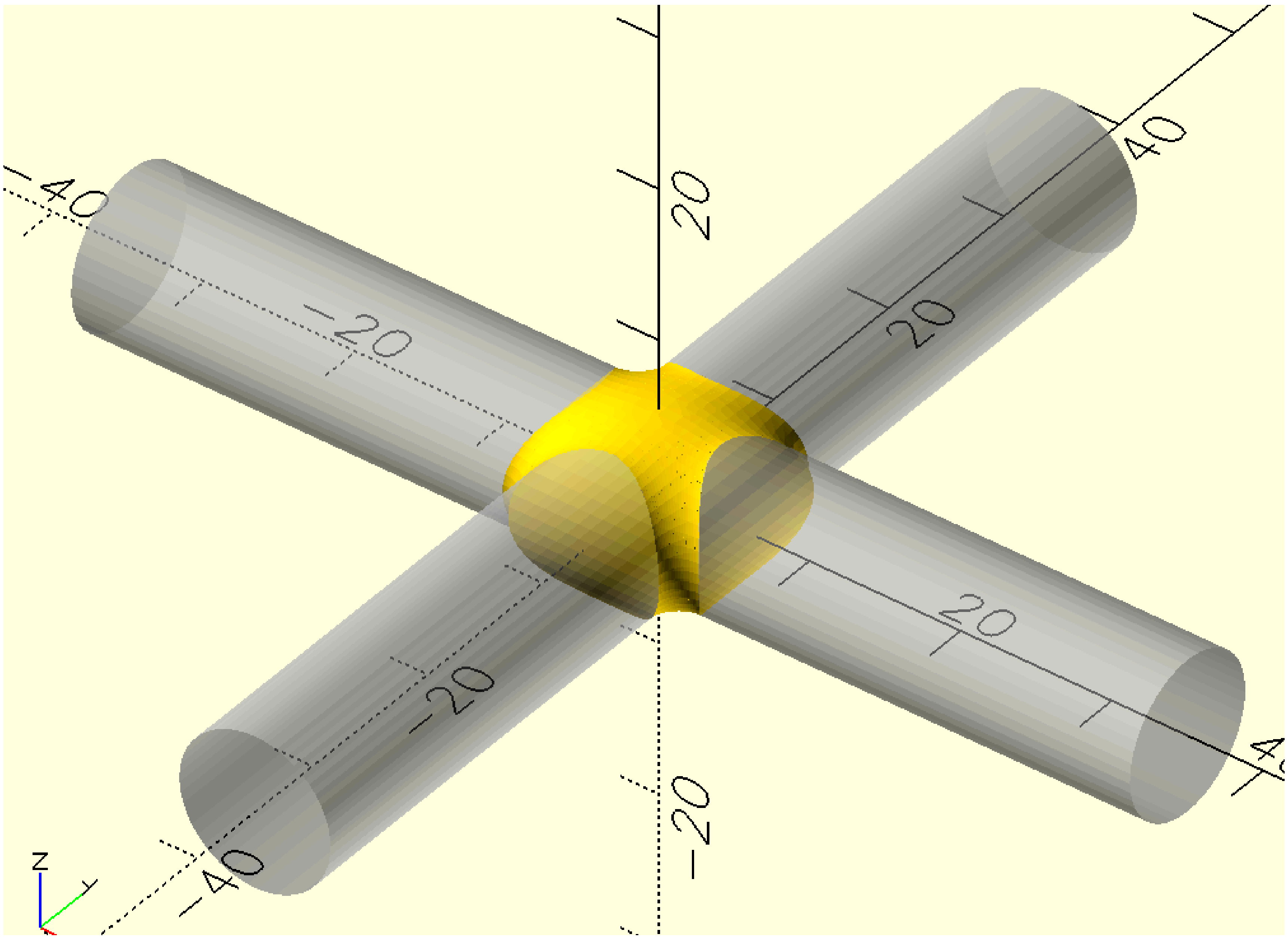


In [57]:

```
# Complex fillets
l1=c23(cr2dt([[0,0],[50,0,4],[0,10,4],[-50,0,4],[0,10,4],
         [50,0,4],[0,10,4],[-50,0,4],[0,10,4],
         [50,0,4],[0,10,4],[-50,0,4],[0,10,4],[50,0]],22))
s1=square([50,60])
c1=circle(3)
sol1=path_extrude_open(c1,l1)
p1=cr2dt([[-3,-1.5],[3,0],[0,3],[-3,0]],10)
p2=cr2dt([[-3,-3],[3,0,1],[0,6,1],[-3,0]],10)
sol2=prism(s1,p1)
sol3=prism(s1,p2)
l2=point_vector([-5,1.5],[5,0])
l3=point_vector([-5,-1.5],[5,0])
f1=fillet_line_circle(l2,c1,2.5,3,s=21)
p0=s_int1([l2]+seg(c1))[0]
f1=[p0]+f1
f2=c32(flip(mirror_line(c23(f1),[0,1,0],[0,0,0])))
f3=c32(flip(mirror_line(c23(f1),[1,0,0],[0,0,0])))
f4=c32(flip(mirror_line(c23(f3),[0,1,0],[0,0,0])))
s2=path_extrude_open(f1,l1)
s3=path_extrude_open(f2,l1)
s4=path_extrude_open(f3,l1)
s5=path_extrude_open(f4,l1)

fileopen(f'''
{swp(sol1)}
{swp(sol2)}
intersection(){{
{swp(sol3)}
for(p={[s2,s3,s4,s5]})swp(p);
}}

''')
```



In [ ]:
```
# another approach to create fillets

s2=translate([0,35,0],rot('x90',cylinder(r=5,h=70)))
s3=translate([-35,0,0],rot('y90',cylinder(r=5,h=70)))

p1=corner_radius_with_turtle([[1.5,0],[-1.5,0,1.5],[0,1.5]],20)
s4=[translate([0,35,0],rot('x90',cylinder(r=(5+x),h=70))) for (x,y) in p1]
s5=[translate([-35,0,0],rot('y90',cylinder(r=(5+y),h=70))) for (x,y) in p1]

fileopen(f'''
%{swp(s2)}
%{swp(s3)}
for(i=[0:19])
hull(){{
    intersection(){{
        swp({s4}[i]);
        swp({s5}[i]);
                }}
    intersection(){{
        swp({s4}[i+1]);
        swp({s5}[i+1]);
                    }}
        }}

''')
```



In [ ]:
```
s2=linear_extrude(circle(10,s=200),30)
s3=translate([10,0,5],linear_extrude(circle(5,s=100),20))
p1=corner_radius_with_turtle([[1,0],[-1,0,1],[0,1]],20)
s4=[ translate([10,0,5-x],linear_extrude(offset(circle(5,s=100),x),20+2*x)) for (x,y) in p1]
```

```
s5=[linear_extrude( offset(circle(10,s=200),y),30) for(x,y) in p1]

fileopen(f'''
{swp(s2)}
{swp(s3)}
for(i=[0:19])
hull(){{
intersection(){{
swp({s4}[i]);
swp({s5}[i]);
}}

intersection(){{
swp({s4}[i+1]);
swp({s5}[i+1]);
}}
}}
    ''')
```