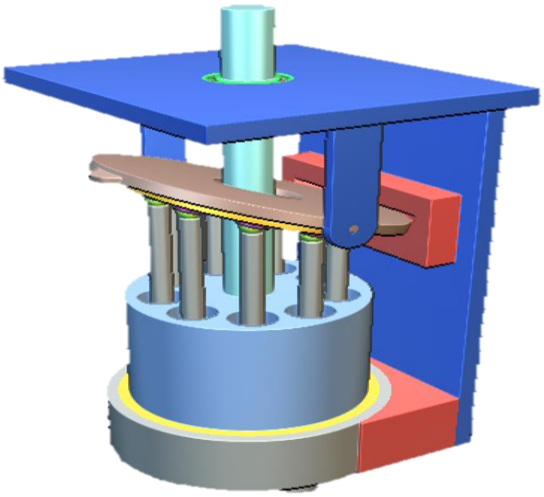
Case Study Cyber Physical Production Systems using AM

Group_12: Pump It Up

"<u>Axial Piston Pump</u>"



<u>Advisor</u>*: *Prof. Dr. Ing. Stefan Scherbarth

Ahmad Gharyib (00814085)

Jayraj Chauhan (22103804)

Mukund Patel (22104433)

Nachiket Kachoriya (22106621)

Pradhyuman Rathore (00736102)

March 2023 - July 2023

Table of Contents

1. Introduction

In this tutorial, explanation of Lua script in IceSL software is given, to generate 3D model of Axial Piston Pump. IceSL is software to generate 3D models using Lua script as programming language, to give user visual feedback on how model will look in 3D printing, with different voxel size and dimension. Here, the generated Axial Piston Pump is parametric. So, the user can change the quantity of component and size of the pump.

1.1. Parts in Axial Piston Pump

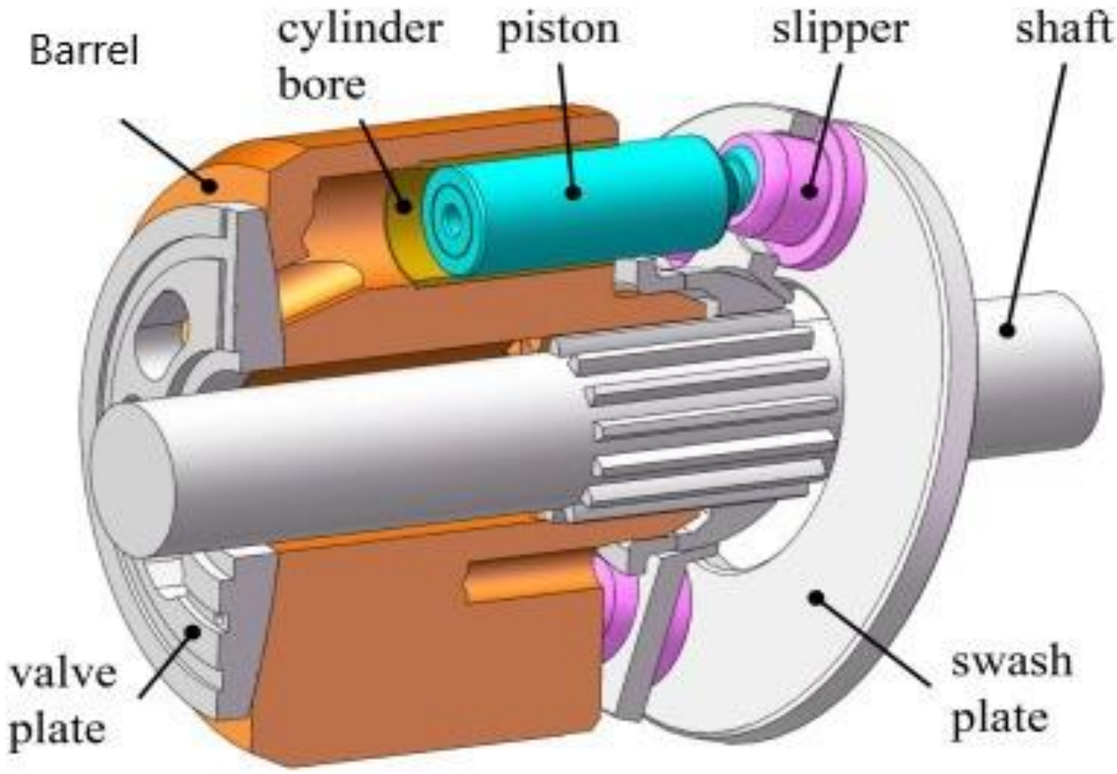


Figure 1.1: Parts of Axial piston pump

Source: https://www.sciencedirect.com/science/article/abs/pii/S0043164820302775

1.2. Nomenclature

n_c = number of cylinders/pistons
r_c = radius of cylinder bore/hollow part
beta_S = Swashplate angle
r_ppc = piston pitch circle radius
r_B = barrel radius
adjusted_r_c = radius for cylinder bore to avoid collision
h_p = piston height

h_B = barrel height

r_p = piston radius(piston radius is slightly less then cylinder bore radius for smooth translation motion.

rod_h = height of connecting rod

r_sl = radius of sphere for slipper

dw = factor that is total height of slider with sphere in vertical direction

r_s = swashplate radius

h_s = height of swash plate

h_F_3 = valve plate thickness

Note: Following image will give you a brief information of different radius which we used in given Lua script.

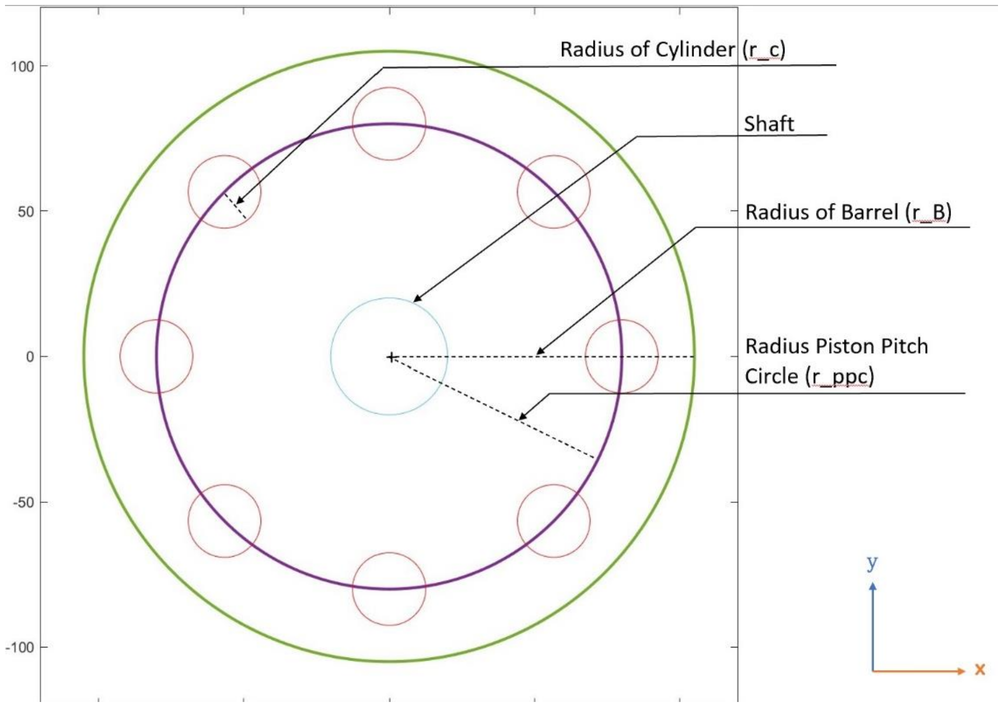
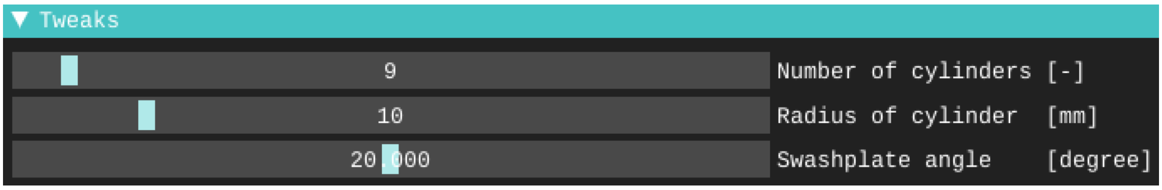


Figure 1.2: MATLAB generated idea of some part's radius

2. Lua Script

2.1. Input Parameter

For recreation of the Axial Piston Pump Model, one required the basic input parameter from which we will get another parameter to generate 3D Model. It helps to generate smooth and parametric geometry. In IceSL, there will be one tweak box which allows the user to change input parameters as shown below.



1. The first parameter is the number of cylinders or pistons.
2. The user can change the radius of cylinders/pistons with the help of a second parameter which later decides the barrel diameter and other parameter.
3. The third parameter helps you to change the angle of swash plate which will change the stroke length as well.

This can be generated with the help of code which is shown here:

```
 function Axial_Piston_Pump(n_c,r_c,beta_s)

n_c = ui_number("Number of cylinders [-]",9,3,100);   -- Desired number of pistons/cylinders

r_c = ui_number("Radius of cylinder  [mm]",10,2,50);   -- Desired radius of cylinder

beta_s = ui_scalar("Swashplate angle    [degree]",20,10,30);   -- Desired angle of Swashplate
```

Step_1: Create a function name with Axial Piston Pump with parameters: number of cylinder/piston(n_c), radius of cylinder/piston(r_c), swash plate angle(beta_s).

Step_2: Define the input parameters and design them as UI scaler variable, to give tweak box on display so user can directly change the input parameters without looking into the code.

2.2 Calculation for required Parameters

Step_3: Defining the pitch circle radius from number of pistons and piston radius and giving condition to increase the piston pitch circle radius to avoid collision of pistons when number of pistons is less than 5.

Step_4: Getting value of Barrel radius from piston pitch circle and piston radius and to be noted that 0.6 is here in code is clearance.

```
--- Calculation of the piston pitch circle radius (r_ppc)
--- Piston Pitch Circle (ppc) is circle that passes through all piston centers
    if n_c>4 then
        r_ppc = (n_c * r_c) / (2*math.pi)
    elseif n_c<=4 then
        r_ppc = (n_c * r_c) / (1.1*math.pi)
    end

--- Calculation of the "Barrel radius" based on the cylinder pitch circle radius
    r_B = r_ppc + 0.6 * r_c
```

Step_5: Using cylinder piston factor, calculating the new piston radius from desired arc length to actual arc length to avoid overlapping of pistons.

Step_6: Calculation of stroke length for getting piston height and barrel height too.

```
--- Calculation of the "desired arc distance between two Cylinders" with reference to actual arc
length to avoid collision between 2 adjucent pistons
    Cylinder_spacing_factor = 2.5   -- Factor for cylinder spacing to avoid overlaping of pistons
    desired_arc_distance = (2 * math.pi * r_ppc) / (n_c * Cylinder_spacing_factor)

--- Calculation for the "actual arc length between two Cylinders"
    actual_arc_length = (2 * math.pi * r_ppc) / n_c

--- Adjustment for the "r_c (cylinder radius)" based on the desired and actual arc distances
    adjusted_r_c = r_c * desired_arc_distance / actual_arc_length   -- This radius will be used to
avoid collision

--- Calculation for "Barrel height" and "Piston height" from Stroke length
    Stroke_length =  (r_ppc) * math.tan(math.rad(beta_s))+2*adjusted_r_c
    h_p = Stroke_length   -- Piston height from the Stroke length
    h_B = Stroke_length + h_p  -- Barrel height depends on Stroke Length
```

2.3 Formation of Barrel, Cylinders, Pistons, Rods and Slippers

Step_7: Creation of barrel cylinder (without hollow part) using.

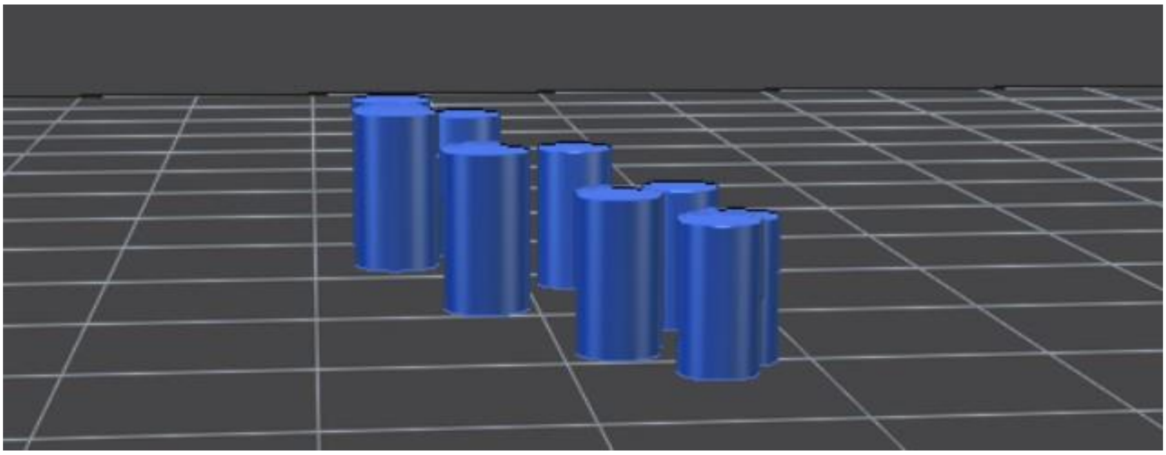
Step_8: Making for loop to generate pistons with swash angle and making pathway with some clearance for pistons.

Step_9: Define x, y and z coordinates which represent the center point of pistons and cylinders in 3-dimensions, for proper positioning of pistons, cylinders, rods and sliders.

Step_10: To create hollow parts in barrel for movement of piston within it, we use x and y coordinates and give some clearance to give pathway for piston.

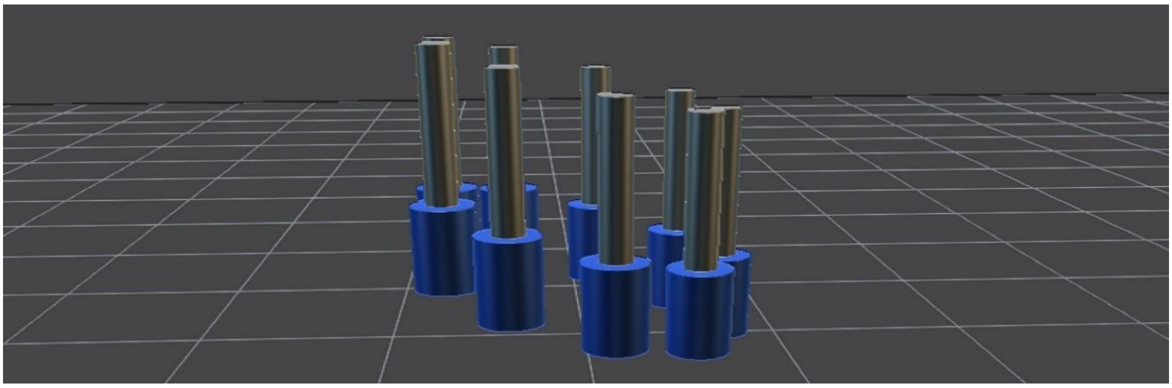
Step_11: Generation of pistons.

```
--=========================================================================================--
--::Creation of Barrel (Teal Blue), Cylinders, Pistons (Blue), Rods (Gray) and Slippers (Pink) ::--
--=========================================================================================--

    Barrel = translate(0,0,-r_ppc*math.tan(math.rad(beta_s)))* cylinder(r_B,h_B)

--- Generation of the Cylinders and add Pistons and their position using polar cordinates

    for i = 1, n_c do
        angle = (i - 1) * (360/n_c)  -- Angle for each Cylinder

        x = r_ppc*math.cos(math.rad(angle))     -- x cordinates for pistons and cylinders position
        y = r_ppc * math.sin(math.rad(angle))   -- y cordinates for pistons and cylinders position
        z = -x*math.tan(math.rad(beta_s))       -- z cordinates for pistons,rods,slippers position

--- Translation and creation of the Cylinder in barrel
    Cylinder = translate(x,y,-r_ppc*math.tan(math.rad(beta_s))) * cylinder(adjusted_r_c,h_B)

--- Creation of the piston with a diameter slightly less than adjusted_r_c
    r_p = adjusted_r_c - 0.4

    Piston = translate(x,y,z) * cylinder(r_p, h_p)

emit(Piston,19)
```



Step_12: Calculate the connecting rod height and build the rod.

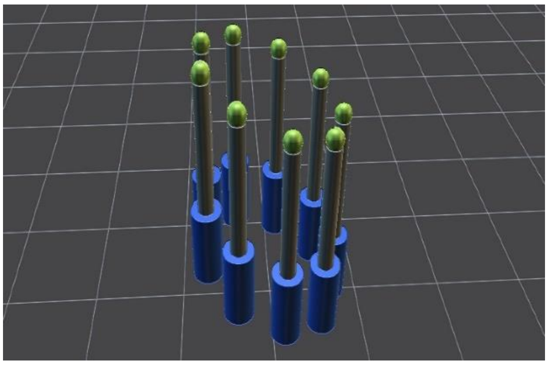
```
    rod_h = (h_B)*1.5  -- Connecting rod length
    rod = translate(x,y,z+h_p) * cylinder(r_p/2,rod_h-h_p)  -- Distribution and translation of of rods according to piston positions

    emit(rod)
```

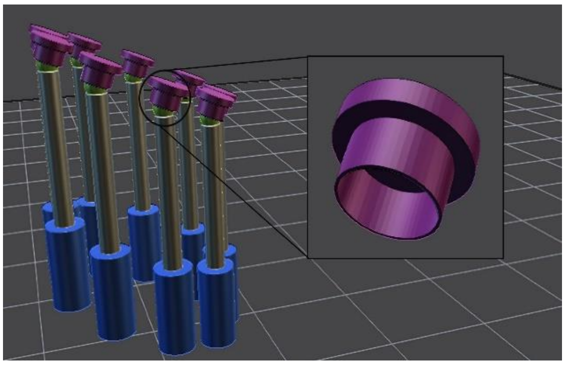


Step_13: Then calculate sphere dimension and create slider as well with reference to rod height for translation

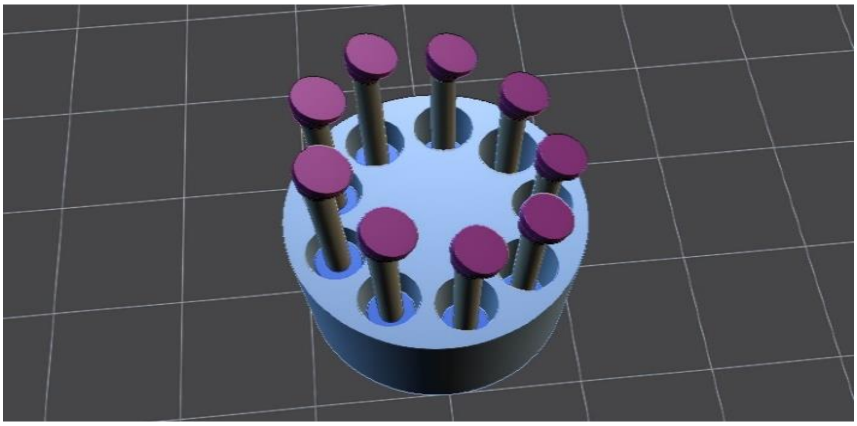
```
    slip = translate(x,y,z+rod_h) * sphere((r_p/2)*1.3)  -- Distribution and translation of the slippers

--- Creation of the Slippers
    r_sl = (r_p/2) * 1.1  -- Sphere radius

    slip = translate(x,y,z+rod_h+r_sl/1.5) * sphere(r_sl)  -- Distribution and translation of the slippers

        slip1 = cylinder(r_sl*1.3,r_sl*1.3)
        slip2 = cylinder(r_sl*1.2,r_sl*1.3)

        slip1 = difference(slip1,slip2)
        slip3 = translate(0,0,1.3*r_sl)*cylinder(1.8*r_sl,r_sl/1.5)

    dw = 1.3*r_sl+2*(r_sl/1.5)+0.2

    slider = rotate(0,beta_s,0) * union(slip1,slip3)
    slider2 = translate(x,y,z+rod_h+r_sl-(r_sl/3)) * slider

emit(slider2,3)  -- Output Shoes/Slider

--- Subtraction of the Cylinder from the Barrel
    Barrel = difference(Barrel, Cylinder)

emit(Barrel,18)  -- Output of Barrel
```

Step_14: emit the slider.





Step_15: Then take difference of cylinders to create hollow part in Barrel and emit it.

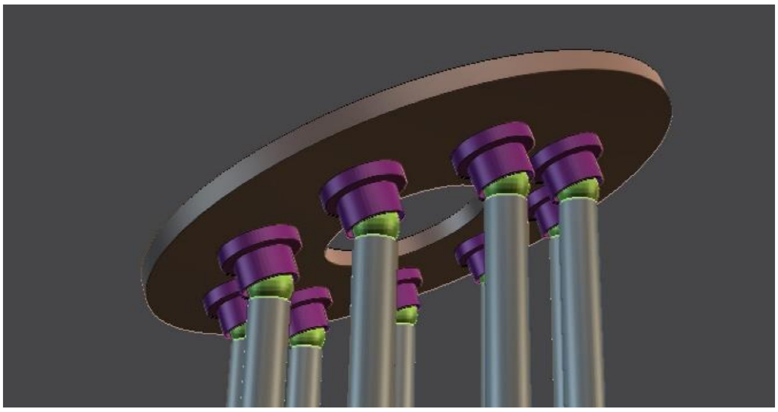


2.4 Formation of Swash plate and Shaft

Step_16: Calculate Swash plate radius with the help of barrel radius and swashplate angle.

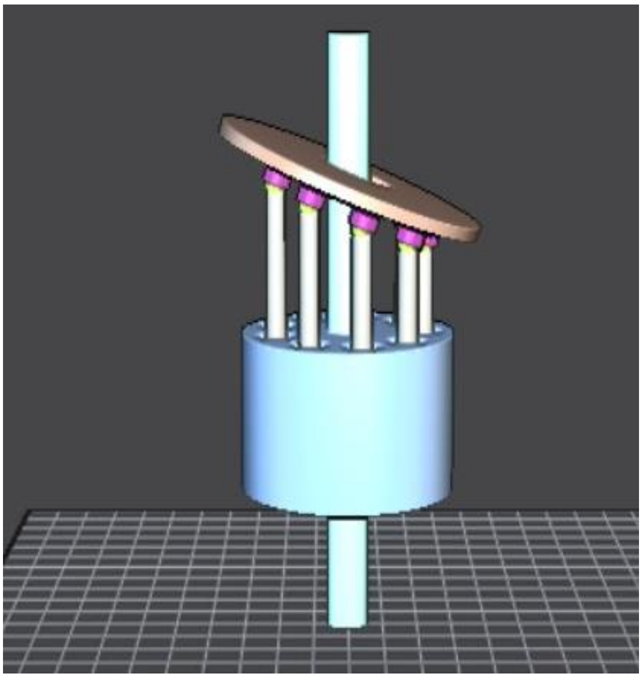
Step_17: Because the swash plate is fixed one need to give shaft way through, so create hollow disk by difference again.

```
--- Calculation for "Swashplate radius"
    r_s = r_B*1.2 / math.cos(math.rad(beta_s)) -- Diameter of the swash plate
    h_s = (r_p/1.5)  -- Height of the swash plate

    swash_plate = translate(0, 0, rod_h+dw) * rotate(0, beta_s, 0) * cylinder(r_s, h_s)  -- For translation and rotation of swashplate

    Shaft_hole = translate(0, 0,rod_h+h_s) * rotate(0, beta_s, 0) * cylinder(r_B/2.7,h_s*3)  -- Hole to avoide connection between shaft and swashplate

emit(difference(swash_plate,Shaft_hole),2)  -- Output of Swash plate
```



Step_18: Create shaft which radius is 5th of barrel radius and height 3 times height

```
-- Creation of the Shaft
 shaft = translate(0,0,-1.5*h_B)*cylinder(r_B/5,3*rod_h)  -- For generation of a middle shaft
 emit(shaft,14)
```



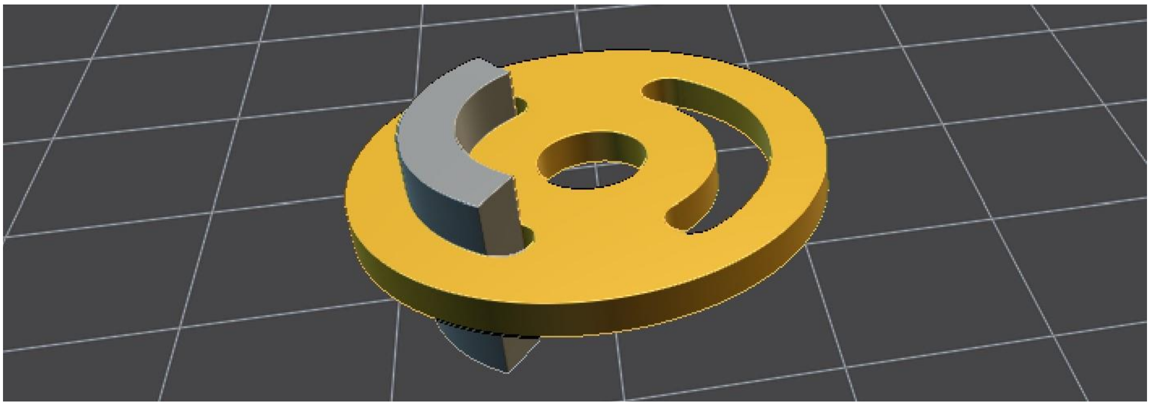
2.5 Formation of Valve Plate

Step_19: For generation of valve plate define parameter from previous piston and barrel parameter

```
--- Calculation for "Valve plate"
    h_F_3 = (h_B-h_p)/4 + 0.2  -- Valve Plate thickness/height

    Valve_P = translate(0,0,-h_F_3-r_ppc*math.tan(math.rad(beta_s))) * cylinder(r_B*1.1,h_F_3)

--- Calculation for "Inlet and Outlet ports" in valve plate

    r_inlet = r_p * 0.8  -- Radius of cylinder in inlet port

    Port = {}
    phi = 70  -- Angle between two ports
    angle = 90-phi/2
```

Step_20: Now to create an arc with inner and outer radius at angle from -55 to 55 degree, take 2 for loop for both arcs to create partial valve plate arc.

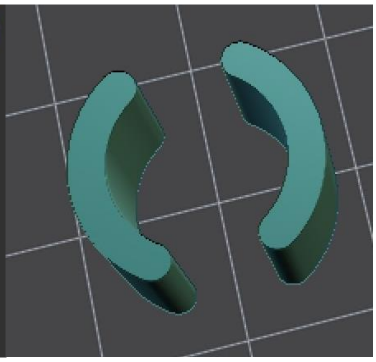
```
    for i = -angle,angle do

        x =  (r_ppc-r_inlet) * math.cos(2 * math.pi * i / 360)
        y =  (r_ppc-r_inlet) * math.sin(2 * math.pi * i / 360)
        Port[#Port + 1]= v(x,y)

    end

    for i = angle,-angle,-1 do

        x =  (r_ppc+r_inlet) * math.cos(2 * math.pi * i / 360)
        y =  (r_ppc+r_inlet) * math.sin(2 * math.pi * i / 360)
        Port[#Port + 1]= v(x,y)

    end

  Partial_Inlet = linear_extrude(v(0,0,-3.7*h_F_3-r_ppc*math.tan(math.rad(beta_s))),Port)  -- Linier extusion of port
```



[Above code creates just gray part here, whole part is just for understanding]

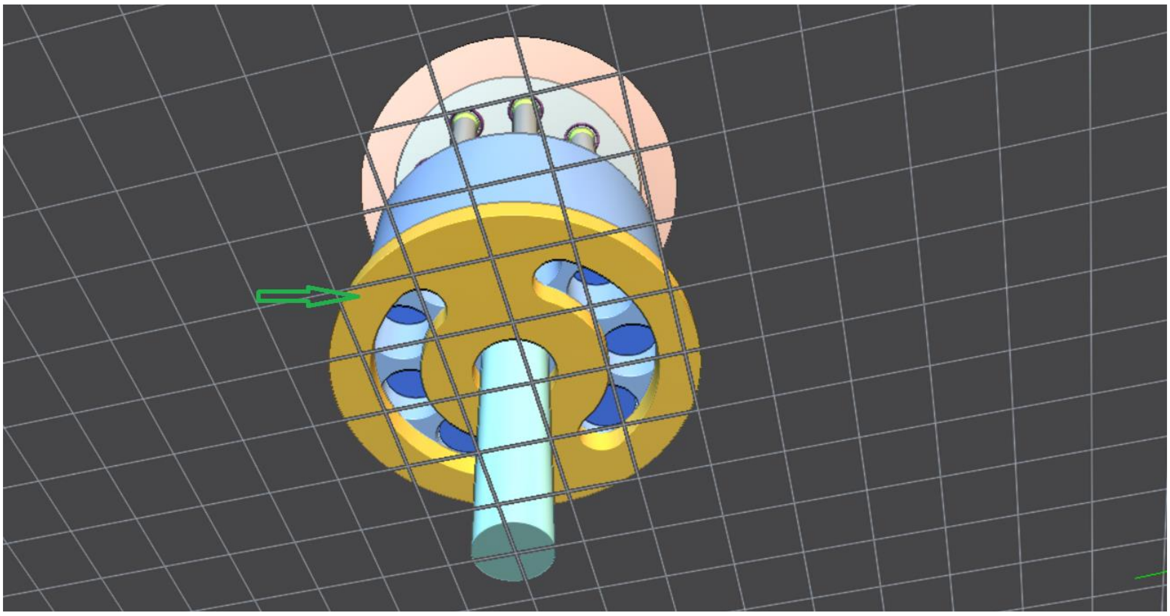
Step_21: At the end of the arc, we put cylinder on each side. Then we make "union" operation to create the Inlet port solid part's arc, then we rotate it 180 degrees to create solid outlet port's arc.

```
    c1 = rotate(0,0,angle) * translate(r_ppc,0,-3.7*h_F_3-r_ppc*math.tan(math.rad(beta_s))) * cylinder(r_inlet,3.7*h_F_3)  -- Cylinder to give roundshape to port
    C2 = rotate(0,0,-angle) * translate(r_ppc,0,-3.7*h_F_3-r_ppc*math.tan(math.rad(beta_s))) * cylinder(r_inlet,3.7*h_F_3)  -- Cylinder to give roundshape to port

    Inlet_Port = union{c1,C2,Partial_Inlet}  -- Inlet port
    Outlet_Port = (rotate(0,0,180) * Inlet_Port)  -- Outlet port

    Ports = union{Inlet_Port,Outlet_Port}  -- Union of both ports
```



Step_22: Then take difference of whole assembly from other cylinder disk, so it will generate final valve plate as.

```
    Valve_Plate = (difference(Valve_P,Ports))  -- Valve plate with ports

    V_P = (rotate(0,0,90)*Valve_Plate)  -- Output of Valve Plate

    F_hole1 = translate(0,0,-5*h_F_3-r_ppc*math.tan(math.rad(beta_s)))*cylinder(0.9*r_B/5+1.35*h_s,5*h_F_3)  -- Hole to give way for middle shaft and bearing

    V_P3 = (difference(V_P,F_hole1))

emit(V_P3,8)    -- Output of Valve Plate with center hole
```

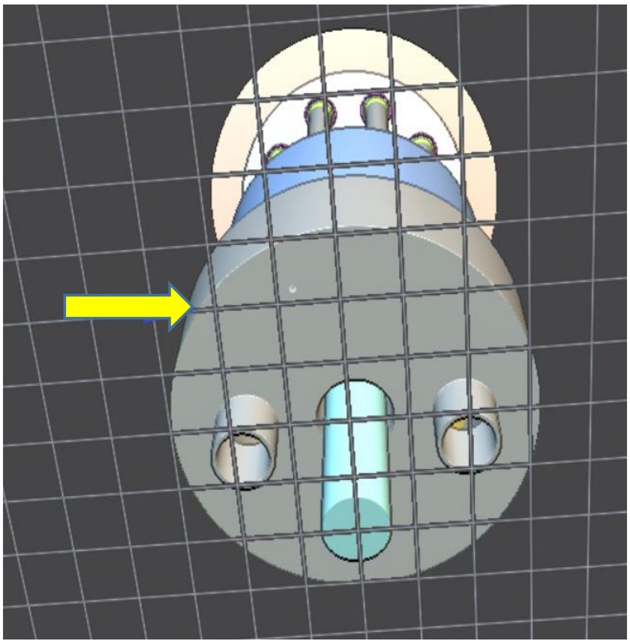


2.6 Formation of Flange

Step_23: For Generation of flange, we take reference of valve plate and at the end we put 2 pipes for inlet and outlet. To generate flange we took one cylinder and then remove valve plate from it.

```
--===================================================================================================--
-----::::: Creation of Flange (Represents Grey part under the Barrel in Model) and Shaft :::::-----
--===================================================================================================--

    F_1 = translate(0,0,-4*h_F_3-r_ppc*math.tan(math.rad(beta_s))) * cylinder(r_B*1.2,4*h_F_3)  -- Simple cylinder for Flange

    F_2 = (difference(F_1,Valve_Plate))  -- To create space for Valve plate in flange

    F_3 = difference(F_2,Ports)  -- To create inlet outlet port in flange

    F_31=(rotate(0,0,90)*F_3)

```

```
--- Creation of inlet and outlet Pipes

    c1 = cylinder(r_B/5,2*h_F_3)  -- Outer cylinder to generate pipe
    c2 = cylinder(r_B/6,2*h_F_3)  -- Inner cylinder to generate pipe

    Inlet_P = (translate(0,r_ppc,-6*h_F_3-r_ppc*math.tan(math.rad(beta_s)))*difference(c1,c2))  -- Inlet pipe
    Outlet_P = rotate(0,0,180)*Inlet_P  -- Outlet pipe
    I_O_Pipes= union{Inlet_P,Outlet_P}  -- Union of both pipes

emit(I_O_Pipes)  -- Output of inlet and outlet Pipes

--- Generation of holes so fluid can flow through pipes in flange

    c3 = translate(0,r_ppc,-6*h_F_3-r_ppc*math.tan(math.rad(beta_s)))*cylinder(r_B/6,5*h_F_3)  -- To generate hole at inlet port in flange
    c4 = rotate(0,0,180)*c3  -- To generate hole at outlet port in flange
    cyl = union{c3,c4}  -- Union of both holes

    Flange = difference(F_31,cyl)

emit(difference(Flange,F_hole1))  -- Output of flange
```



2.7 Formation of Support

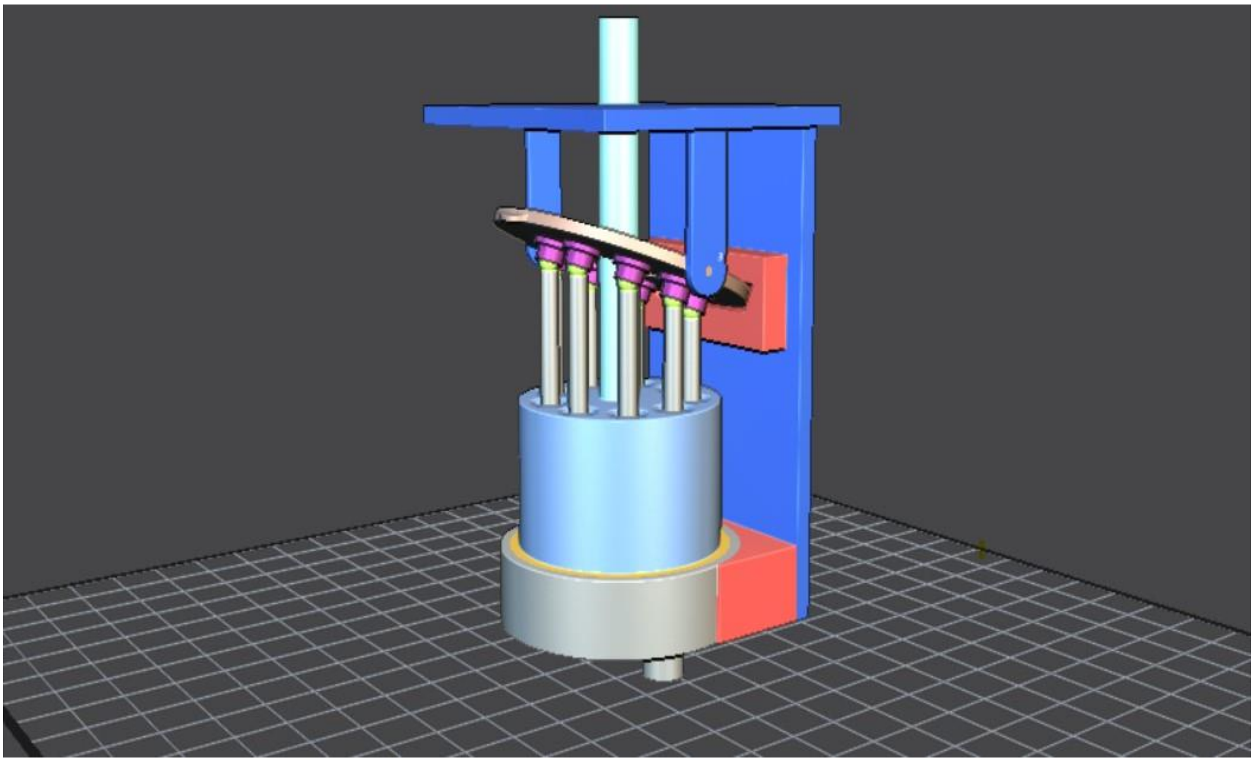
Step_24: Then we generate support for physical model with base plate, side plate (give swash plate support), pivot-hinge assembly (helps to change swash plate angle), handle, flange holder.

```
--=================================================================================================--
----::::::::::::::::::::::::::::::::::: Creation of Supports :::::::::::::::::::::::::::::::::::-----
--=================================================================================================--

--- Creation of Base Plate
    support_A = translate(r_B*1.2+(h_s*1.5),0,-r_ppc*math.tan(math.rad(beta_s))-4*h_F_3) * cube(h_s*1.5,2.4*r_B,h_p+4*h_F_3+1.7*rod_h-((r_s*1.1)*math.sin(math.rad(beta_s)))+h_s*1.5)

--- Creation of side Plate to give swashplate support
    support_A1 = translate(h_s*0.75,0,-r_ppc*math.tan(math.rad(beta_s))+h_p+1.7*rod_h-((r_s*1.1)*math.sin(math.rad(beta_s)))+h_s*1.5) * cube(2*r_s+h_s*1.5,2.4*r_s,h_s*1.5)

--- Creation of hole to in support_A1 for shaft way and bearing

    F_hole2 = translate(0,0,-r_ppc*math.tan(math.rad(beta_s))+h_p+1.7*rod_h-((r_s*1.1)*math.sin(math.rad(beta_s)))+h_s*1.5) * cylinder(0.9*r_B/5+1.35*h_s,5*h_F_3)

emit(difference(support_A1,F_hole2),7)  -- Output of swashplate support

--- Creation of support for swashplate to hold it when it changes angle
    sa = translate(r_B*1.2+(h_s/4),0,rod_h-((r_s*1.1)*math.sin(math.rad(beta_s))))*cube(h_s*4,2*r_B,h_s*7)

    swash_plate1 = translate(0, 0, rod_h+dw-h_s/2) * rotate(0, beta_s, 0) * cylinder(r_s,2*h_s) -- define new bigger swashplate just to generate holding groove

emit(difference{sa,swash_plate1,support_A},6)  -- Output of Swashplate holder

emit(support_A,7)  -- Output of Base plate
```

```
--- Creation of Pivot-Hinge Assembly to change swashplate angle
    P1 = translate(0,-r_s,rod_h+dw+h_s/2)*rotate(90,0,0)*cylinder(h_s/3,r_s/15)  -- Pivot on swashplate
    P2 = rotate(0,0,180)*P1  -- Pivot on swashplate
    P = union {P1,P2}  -- Union of both pivots

emit(P,2)  -- output of pivots

    circle_p = translate(0,-r_s,rod_h+dw+h_s/2)*rotate(90,0,0)*cylinder(r_B/5+h_s/22.4,r_s/15)  -- Hinge round part
    circle_p_hole = translate(0,-r_s,rod_h+dw+h_s/2)*rotate(90,0,0)*cylinder(h_s/2.8,r_s/15)  -- Hole for Pivot in Hinge

    CP = (difference(circle_p,circle_p_hole))  -- create hole for pivot in hinge

    Plate_P = translate(0,-r_s-r_s/30,rod_h+dw+h_s/2+h_s/2.8)*cube(r_B/2.5,r_s/15,-r_ppc*math.tan(math.rad(beta_s))+((r_s*1.1)*math.sin(math.rad(beta_s)))+h_s*8)  -- Hinge

    PS = union{Plate_P,CP}  -- Total part of Hinge
    PS1= rotate(0,0,180)*PS  -- Other side of Hinge
    Pivot_support = union{PS,PS1}  -- Union of both Hinges

emit(Pivot_support,7)  -- Output of Hinges
```

```
--- Creation of Handle on swash plate to change angle of swash plate
    H1 = translate(-r_B*1.2-(r_p*1.3)/2+r_s/7,0,rod_h+dw+r_s*math.sin(math.rad(beta_s)))*cylinder(r_s/7,h_s/2)

    SP = translate(0, 0, rod_h+dw+h_s) * rotate(0, beta_s, 0) * cylinder(r_s, h_s)  -- Imaginary swashplate to cut H1 from top side

    H_circle = difference(H1,SP)  -- cutting of H1(Handle)

emit(H_circle,2)  -- Output of little handle on swashplate to vary it's angle

--- Creation of Support for Flange
    H_S = translate(r_B*1.2+(h_s*1.5)/2-r_B/2,0,-4*h_F_3-r_ppc*math.tan(math.rad(beta_s)))*cube(r_B,2.4*r_B,4*h_F_3)  -- Simple cube for support Flange

emit(difference(H_S,F_1),6)  -- Difference of Flange from cube so it will generate round shape support for flange
```

3. Whole assembly with Supports:



4. STL file, G-code and other setting

4.1.Creating STL File

In Order to print the pump, the user need to generate the STL file of the pumps.

- As shown in the figure, In the service menu select “Export to mesh”.
- In the meshing method to use select “Dual contouring”.
- Change the voxel size accordingly (more voxel size less resolution, smaller file size of stl file)
- Click to run and save the STL file in the desired folder

4.2 Creating G-Code

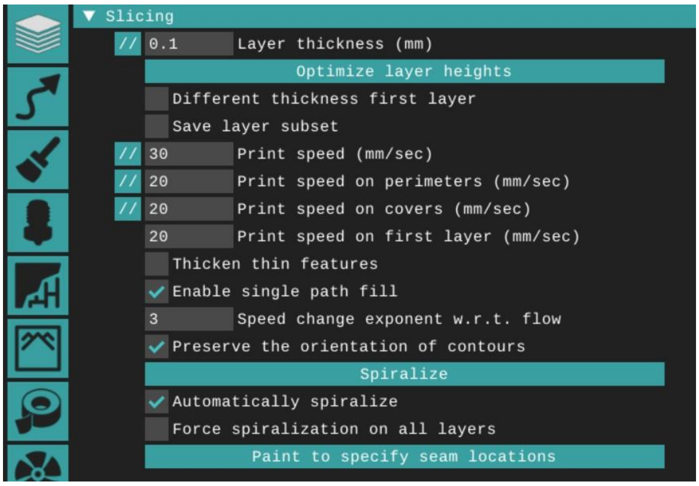
Also, the user can generate the G-code for the printer directly from the service menu

- Select “export to G-Code” and select proper profile quality, printer model and material
- Press on “slicing” to further edit the parameters like print speed, layer thickness, etc.
- Click on “Slice!” and now, one can save the g-code

4.3 Slicing

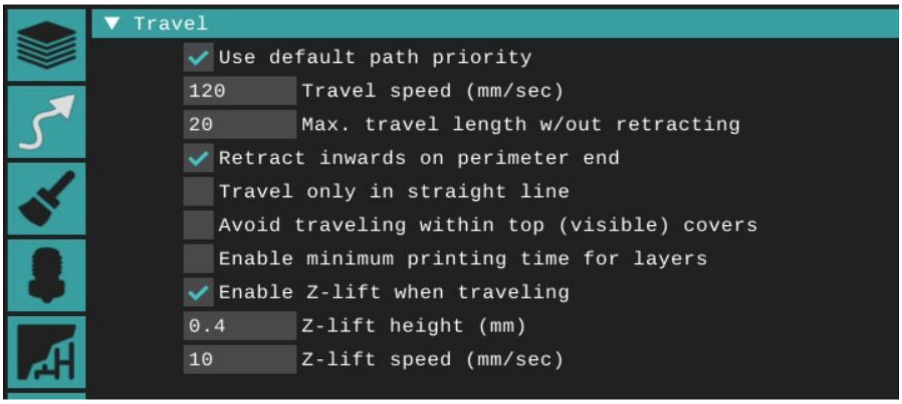
- In IceSL, slicing entails breaking down a 3D model into a number of 2D levels or slices.
- It produces G-code that the 3D printer can use to comprehend and follow in order to build the model layer by layer.

- Slicing enables the printing process to be optimized by ensuring precise layer placement and effective material use.
- IceSL's slicing capabilities are essential for producing prints with precise details and smooth surfaces.
- To make the sliced model compatible with a range of 3D printers, the file can be exported in several file types, such as G-code or STL.



4.4 Travel

- When the print head moves without pushing material out, this is referred to as travel.
- It reduces collisions and speeds up printing.
- Print speed and quality are impacted by travel path and distance.
- The pace and quality of travel can be changed.
- The total distance traveled can be seen.
- Minimize the printing time as per choice



4.5 Other settings

There are many settings like brush, extrude, supports, cooling, etc. which are useful to change printing properties as per your requirements.

5. Hints for the assembly of the model

5.1 Parts List

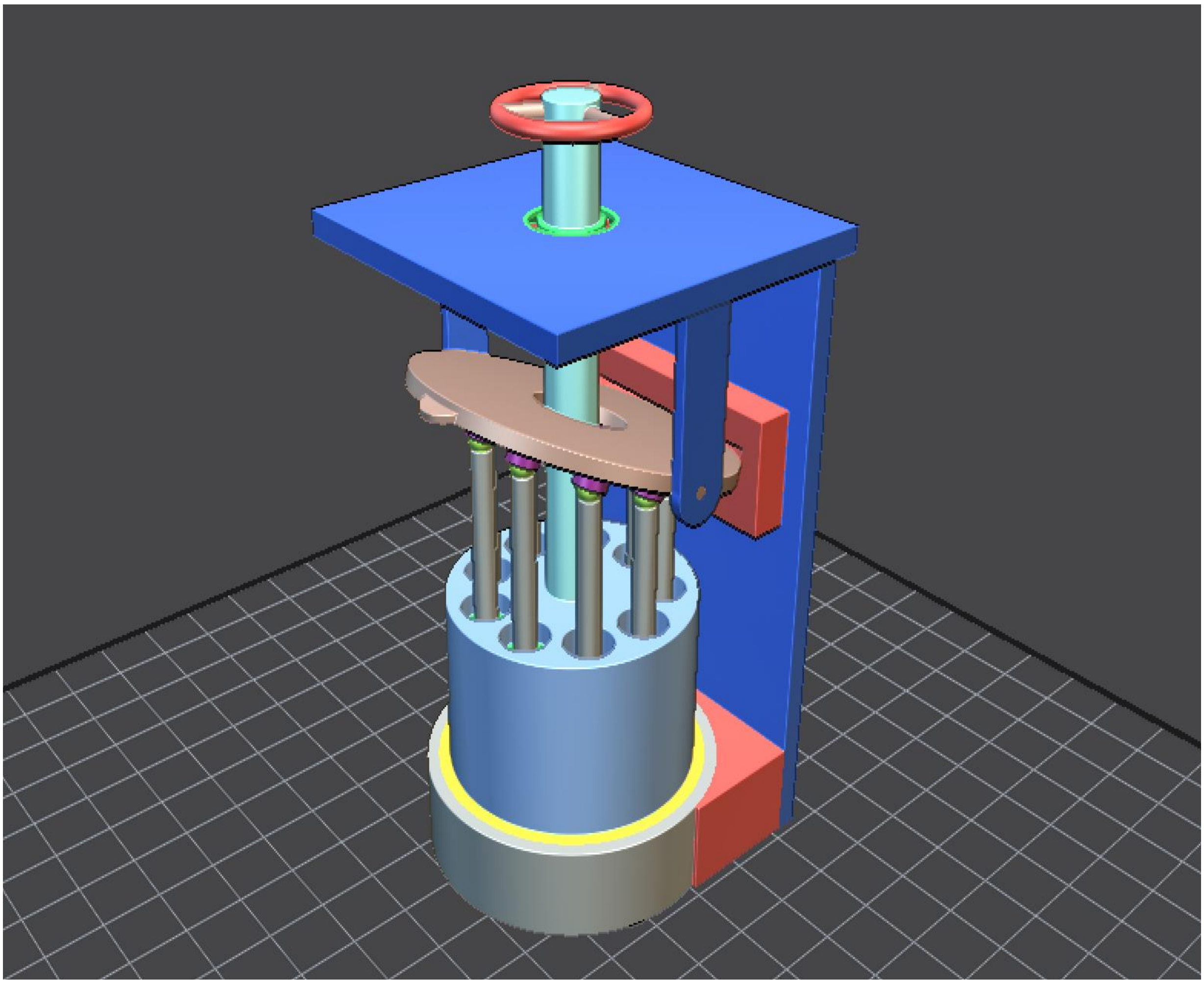
1. Pistons
2. Barrel with bores
3. Connecting rods
4. Slippers
5. Swashplate
6. Shaft
7. Valve plate
8. Flange
9. Support with hinge, handle
10. 2 x Bearing
11. 1 x Wooden Board
12. 4 x (screw, nuts, washers)

5.2 Tools for assembly

1. Electric screwdriver
2. Drill tip
3. Glue
4. Glue Gun
5. Wrench
6. Sandpaper

5.3 Hints for Assembly

- After printing, drill four holes to attach the support of pump to wooden board.
- Insert shaft from one side then, insert swashplate and fixed it to support with hinge.
- Afterwards, insert spring in barrel with pistons, and slide whole assembly through shaft or you can slide the barrel and add springs-pistons later.
- Now, one can add valve plate, flange accordingly



[Whole Assembly for demonstration model]