CASE STUDY CYBER PHYSICAL PRODUCTION SYSTEMS USING AM

INVOLUTE GEAR: EFFECT OF PROFILE SHIFT

Guide for developing an Profile shifted Involute gear with 3D Printing using LUA Script in IceSL

Group No: 17

Bennet Kurian - 22300722
Sarath Satheesh - 22301967
Sajin Saji - 22300624

Research Advisor:

Prof. Dr. Ing. Stefan Scherbarth

TECHNISCHE HOCHSCHULE DEGGENDORF

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This tutorial depicts about developing Involute Profile Shifted Gears for 3D printing using Lua script in IceSL. This User Guide takes users step by step through the process of generating gears and their fixings needed to mesh with variable parameters. The manual is structured to enable users get acquainted with some parameters and adjusting the results correspondingly. The entire parametric model is shown on the Lua platform, it contains all necessary instructions how to extract out ".stl" files and G-code which can be used in different operating systems and 3D printing of objects as well. The instance involves modelling for two gears and fixing them on a common shaft pin, along with this is optimizing gear performance for various 3D printers or materials that makes this a comprehensive resource for anyone involved in gear design and manufacturing.

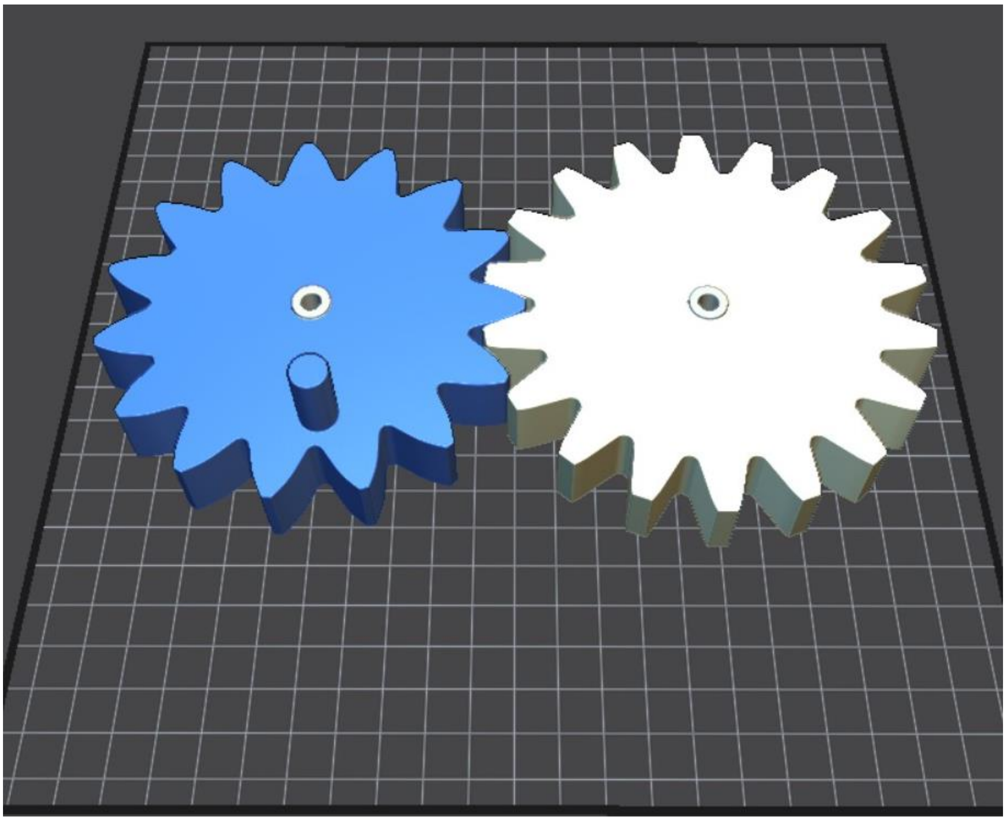


Fig. 1: View of Involute Gear with Profile Shift

2. OVERVIEW OF SCRIPT

The script, "Involute Gear Effect of Profile Shift.lua," is such that it allows the users to produce entire gears which are profile-shifted in their parameters that can be customized and then fit for 3D printing. The .lua file could only be edited through IceSL Forge, IceSL Slicer-supporting editors. It consists of four distinct parts:

1. Input parameters
2. Functions of Involute gear with Profile shift
3. Extrude function
4. Other components.

3. INPUT PARAMETERS

Defining parameters is a crucial step in the process. In IceSL, the tweak box allows users to input parameters and see real-time changes in the 3D model.

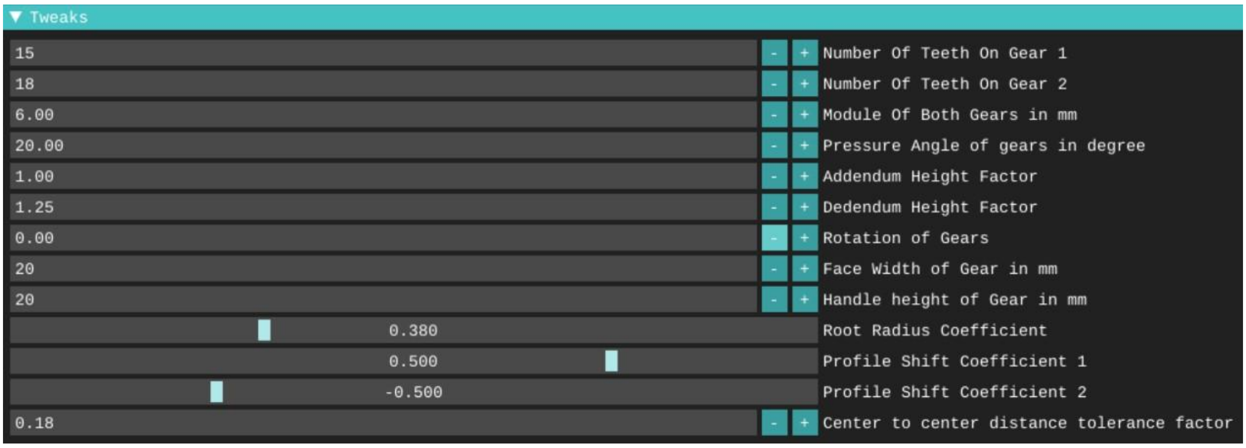
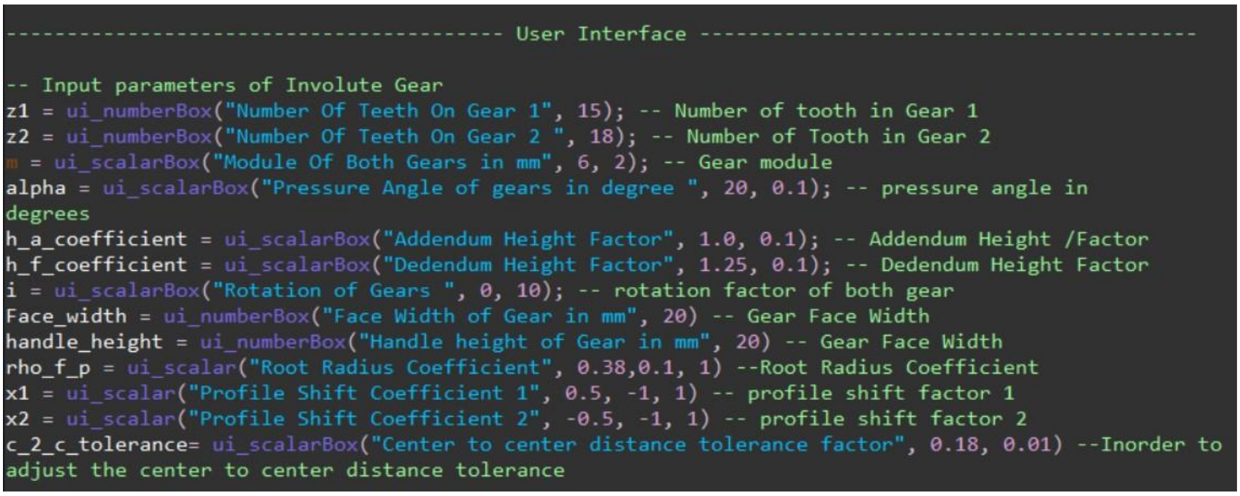


Fig. 2: Tweak Box

Users can adjust the default values, as well as the upper and lower limits of the tweak box, within the user interface section of the script. This flexibility ensures that the model can be precisely tailored to meet specific requirements.



```
----------------------------------------- User Interface -----------------------------------------

-- Input parameters of Involute Gear
z1 = ui_numberBox("Number Of Teeth On Gear 1", 15); -- Number of tooth in Gear 1
z2 = ui_numberBox("Number Of Teeth On Gear 2 ", 18); -- Number of Tooth in Gear 2
m = ui_scalarBox("Module Of Both Gears in mm", 6, 2); -- Gear module
alpha = ui_scalarBox("Pressure Angle of gears in degree ", 20, 0.1); -- pressure angle in
degrees
h_a_coefficient = ui_scalarBox("Addendum Height Factor", 1.0, 0.1); -- Addendum Height /Factor
h_f_coefficient = ui_scalarBox("Dedendum Height Factor", 1.25, 0.1); -- Dedendum Height Factor
i = ui_scalarBox("Rotation of Gears ", 0, 10); -- rotation factor of both gear
Face_width = ui_numberBox("Face Width of Gear in mm", 20) -- Gear Face Width
handle_height = ui_numberBox("Handle height of Gear in mm", 20) -- Gear Face Width
rho_f_p = ui_scalar("Root Radius Coefficient", 0.38,0.1, 1) --Root Radius Coefficient
x1 = ui_scalar("Profile Shift Coefficient 1", 0.5, -1, 1) -- profile shift factor 1
x2 = ui_scalar("Profile Shift Coefficient 2", -0.5, -1, 1) -- profile shift factor 2
c_2_c_tolerance= ui_scalarBox("Center to center distance tolerance factor", 0.18, 0.01) --Inorder to
adjust the center to center distance tolerance
```

Fig. 3: Input Parameters

4. FUNCTIONS OF INVOLUTE GEAR WITH PROFILE SHIFT

4.1 Involute Curve of Gear

This function is denoted as ('inv_c'), it calculates a point on the involute curve for a gear. The involute curve is defined by the base radius (`r_b`) and the involute angle (`inv_alpha`). It uses trigonometric functions to determine the x and y coordinates of the involute point, which are then returned as a vector.

```lua
-- Calculating Involute Curve
-- r_b is the base radius
-- inv_alpha is the involute angle
-- inv_c is the involute curve
inv_c = function(r_b, inv_alpha)
    inv1 = v(r_b * (math.sin(inv_alpha) - inv_alpha * math.cos(inv_alpha)), r_b * (math.cos
(inv_alpha) + inv_alpha * math.sin(inv_alpha)))
    return inv1
end
```

Fig. 4: Involute Curve

4.2 Rotation

This function rotates a given coordinate by a specified angle (`rotate`). It applies the rotation transformation using cosine and sine functions, returning the new rotated coordinates as a vector.

```lua
-- Function of Rotation
Rotation = function(rotate, coordinate)
    R = v(math.cos(rotate) * coordinate.x + math.sin(rotate) * coordinate.y,
        math.cos(rotate) * coordinate.y - math.sin(rotate) * coordinate.x)
    return R
end
```

Fig. 5: Rotation

4.3 Angle of Involute

This function is denoted as ‘roll_angle_psi’ and it calculates the roll angle ratio between two angles, `psi_b` and `psi_a`. It computes the ratio by taking the square root of the difference of their squares, normalized by the square of `psi_b`. The result represents the ratio of the two roll angles.

```lua
-- Assign function of angle of involute in roll_angle_psi

roll_angle_psi = function(psi_b, psi_a)
    R_a = (math.sqrt((psi_a * psi_a - psi_b * psi_b) / (psi_b * psi_b)))
    return R_a
end
```

Fig. 6: Angle of Involute

4.4 Circle

This function ‘circle’ calculates a point on a circle given its center coordinates (`a`, `b`), radius (`r`), and an angle (`angle`). It uses trigonometric functions to determine the x and y coordinates of the point on the circle, returning these coordinates as a vector.

```lua
-- Calculate the angle subtended by the gear tooth profile at a specified point on the base circle or the entire circle

function Circle(a, b, r, angle)
    return v(a + r * math.cos(angle), b + r * math.sin(angle))
end
```

Fig. 7: Points on Circle

4.5 Mirror

This function mirrors a given coordinate across the y-axis. It achieves this by negating the x-coordinate while keeping the y-coordinate unchanged. The mirrored coordinates are returned as a vector.

```
-- function of Mirror
Mirror = function(coordinate)
    M = v(-coordinate.x, coordinate.y)
    return M
end
```

Fig. 8: Mirror Function

4.6 Profile Shift

This function denoted as 'prof_slop' and it calculates the Profile Shift of a gear given by a set of points (`prof`). It computes the slope by taking the difference in y-coordinates and dividing by the difference in x-coordinates between the first and fifth points in the profile.
Profile_slop({v(1, 2), v(2, 3), v(3, 4), v(4, 5), v(5, 6)})

```
-- Function of Profile shift
Profile_slop = function(prof)
    P_s = ((prof[5].y - prof[1].y) / (prof[5].x - prof[1].x))
    return P_s
end
```

Fig. 9: Function of Profile Shift

4.7 Pressure Angle

This function calculates the pressure angle for a gear given the module (`m`), profile shift (`x`), pressure angle (`alpha`), and number of teeth (`z`). It converts the pressure angle from degrees to radians and uses it in a formula to calculate the angle between the sides of a gear tooth.

```
-- Function to calculate the pressure angle: the angle formed between the two sides of a gear tooth.

function angle_gear(m, x, alpha, z)
   alpha = alpha * math.pi / 180
   al = (((math.pi * m / 2) + 2 * m * x * math.tan(alpha)) / (z * m / 2) + 2 * math.tan(alpha) - 2 * alpha)
   return al
end
```

Fig. 10: Function of Pressure angle

5. EXTRUDE FUNCTION

Extrude function takes five arguements profile, angle_deg, extrusion_direction, scaling_factors, z_steps.

This function extrudes a given profile along a specified direction, applying rotation and scaling at each step. The profile is defined by a set of points, and the extrusion parameters include the rotation angle in degrees, the extrusion direction, scaling factors, and the number of steps in the extrusion. It returns a polyhedron representing the extruded shape.

```lua
-- Create a linear extrude by scaling and extruding a given profile in a specified direction.
function extrude(profile, angle_deg, extrusion_direction, scaling_factors, z_steps)
    local n_points = #profile --Number of points
    local angle_rad = angle_deg / 180 * math.pi --Convert angle from degrees to radian
    local vertices = {} -- table to hold the vertices of the extruded shape

    for j = 0, z_steps - 1 do -- Loop over each step in the extrusion
        local phi = angle_rad * j / (z_steps - 1) -- -- Calculate the rotation angle for this  z step
        local vectordirection = extrusion_direction * j / (z_steps - 1) ---- Calculate the position shift for this step
        local scalefactor = (scaling_factors - v(1, 1, 1)) * (j / (z_steps - 1)) + v(1, 1, 1) ---- Calculate the scaling factor for this step
        for i = 1, n_points - 1 do  -- Loop over each point in the profile

-- Calculate the position of the vertex after rotation, scaling, and translation
            vertices[i + j * n_points] = v(vectordirection.x + scalefactor.x *
                                    (profile[i].x * math.cos(phi) - profile[i].y * math.sin(phi)), vectordirection.y +
                scalefactor.y * (profile[i].x * math.sin(phi) + profile[i].y * math.cos(phi)), vectordirection.z * scalefactor.z)
        end
        table.insert(vertices, vertices[1 + j * n_points]) -- Close the loop by connecting to the first vertex of this step
    end
    local vertex_sum_start = v(0, 0, 0) -- Initialize sum of start vertices
    local vertex_sum_end = v(0, 0, 0) -- Initialize sum of end vertices
    for i = 1, n_points - 1 do -- Loop over each point in the profile
        vertex_sum_start = vertex_sum_start + vertices[i]
        vertex_sum_end = vertex_sum_end + vertices[i + n_points * (z_steps - 1)] -- -- Sum up the start and end vertices
end
    table.insert(vertices, vertex_sum_start / (n_points - 1))
    table.insert(vertices, vertex_sum_end / (n_points - 1))
-- Calculate the average start and end vertexes and insert
    local triangles = {} -- Table to hold the triangles of the extruded sha
    local k = 1 -- The indexing on the table with vertices starts with zero.
    for j = 0, z_steps - 2 do -- -- Loop over each step in the extrusion
        for i = 0, n_points - 2 do -- -- -- Loop over each point in the profile
-- Define two triangles for each quad formed by consecutive points in consecutive steps
            triangles[k] = v(i, i + 1, i + n_points) + v(1, 1, 1) * n_points * j
            triangles[k + 1] = v(i + 1, i + n_points + 1, i + n_points) + v(1, 1, 1) * n_points * j
            k = k + 2
        end
    end
    for i = 0, n_points - 2 do
        triangles[k] = v(i + 1, i, n_points * z_steps)
        k = k + 1
    end
    for i = 0, n_points - 2 do -- Loop over each point in the profile
        -- Define triangles connecting the last step to the start vertex
        triangles[k] = v(i + n_points * (z_steps - 1), i + 1 + n_points * (z_steps - 1), n_points * z_steps + 1)
        k = k + 1
    end
    return polyhedron(vertices, triangles)
end
```

Fig. 11: Extrude Function

5.1 Fillet radius

This function calculates the center point of the fillet radius between two profile points. It determines the slope of the profile and uses it to find the angle and coordinates of the center of the fillet circle, considering the fillet radius (`r_c`) and root radius (`r_r`).

```lua
--- Calculation of centre point of Fillet radius between two profile points
function Fillet_radius(prof1, r_c, r_r)
    local slop = (prof1[2].y - prof1[1].y) / (prof1[2].x - prof1[1].x)
    local slop_ang = math.atan(slop)
   -- -- Finds the point on the involute curve that is parallel to the tangent at the specified profile
point.
    local x = prof1[1].x + r_c * math.cos(slop_ang + math.pi / 2)
    local y = prof1[1].y + r_c * math.sin(slop_ang + math.pi / 2)
    local d = (y - slop * x) / math.sqrt(slop * slop + 1)
    local th1 = math.asin(d / (r_c + r_r)) + slop_ang
    -- Returns v: Vector representing the center point of the fillet radius.
    return v((r_c + r_r) * math.cos(th1), (r_c + r_r) * math.sin(th1))
end
```

Fig 12: Fillet Radius

5.2 Function of Single tooth Gear with generate full profile

This function generates the full profile of a gear. It takes various parameters such as the number of teeth (`z`), module (`m`), pressure angle (`alpha_rad`), profile shift (`x`), addendum height factor (`h_a_coeff`), dedendum height factor (`h_f_coeff`), and fillet radius coefficient (`rho_f`). It calculates the gear's dimensions, involute curve points, Centre distance, start/end angles for the fillet circles and fillet points, then assembles these into a complete gear profile.

```lua
function gear(z, m, alpha_rad, x, h_a_coeff, h_f_coeff, rho_f) -- Function Generates full profile of gear
    local xy = {}
    c_c=x1+x2    cl = (c_c+c_2_c_tolerance )* m    a = (z1 + z2) * m / 2 + cl    --Calculating centre distance
    alpha_rad = alpha * math.pi / 180 -- Convert pressure angle of gear to radians
    D_p = z * m -- Calculating pitch  diameter     [1]
    R_p = D_p / 2--Calculating reference radius
    D_base = D_p * math.cos(alpha_rad)-- Calculating diameter of base circle of the gear [2]
    r_b = D_base / 2-- Calculating base radius
    d_a = D_p + 2 * m * h_a_coefficient + 2 * m * x -- Calculating addendum diameter with profile shift [3]
    r_a = d_a / 2 -- Calculating addendum radius
    h_a = m * h_a_coefficient -- Calculating addendum height (h_a_coeff is equal to  Addendum Height Factor)
    d_f = D_p - 2 * m * h_f_coefficient + 2 * m * x --Calculating dedendum diameter with profile shift factor [3]
    r_f = d_f / 2    -- Calculating root_radius
    h_r = m * h_f_coefficient -- dedendum height (h_f_coeff = Dedendum Height Factor)
    r_c = rho_f * m -- Calculating Fillet root radius Coefficient [2]
    True_dia = math.sqrt(math.pow(D_p * math.sin(alpha_rad) - 2 * (h_a - (m * x) - h_r * (1 - math.sin(alpha_rad))), 2) + D_base * D_base)--  True dia to calculate true involute diameter
    True_rad = True_dia / 2 -- true involute Radius
    tooth_angle = m * ((math.pi / 2) + 2 * x * math.tan(alpha_rad)) / R_p + 2 * math.tan(alpha_rad) - 2 * alpha_rad -- Angle between two involute Curve
    Start_SOI = roll_angle_psi(r_b, True_rad) -- Starting angle of Involute curve    -- Start_SOI is the Diameter at start of Involute
    End_SOI = roll_angle_psi(r_b, r_a) -- ending angle of involute curve    -- End_SOI is the Diameter at end of Involute
    Points = 15   ----Number of points for Better Accuracy    -- Calculating involute curve points we for maintaining the continuty of the curve
    local involute = {} ---- Table to store involute curve points
    for i = 1, Points do -- Calculate the involute point for each step and store it in the involute table
        involute[i] = inv_c(r_b, (Start_SOI + (End_SOI - Start_SOI) * i / Points))
    end
    Profile_slop_inv = Profile_slop(involute) --Calculate the slope of profile using the involute points
    pressure_angle = math.atan(Profile_slop_inv) -- Calculation of pressure angle from the profile slope
    center_a = {}
    center_a[1] = Fillet_radius(involute, r_c, r_f) -- --Calculate the centre of the fillet circle
    Start_fillet = 2 * math.pi + math.atan(center_a[1].y / center_a[1].x)
    End_fillet = 3 * math.pi / 2 + pressure_angle-- Calculate the start and and angle of the fillet    -- start generating full gear profile including fillets
    for i = 1, z do -- Loop over each tooth of the Gear
        for j = 1, Points do -- Calculate the points for the fillet and apply rotation for each tooth
            xy[#xy + 1] = Rotation(2 * math.pi * i / z, Circle(center_a[1].x, center_a[1].y, r_c, (Start_fillet + (End_fillet - Start_fillet) * j / Points)))
        end
        for j = 1, Points do -- Loop over each point in the left involute -- Calculate the points for the left involute and apply rotation for each tooth
            xy[#xy + 1] = Rotation(2 * math.pi * i / z, inv_c(r_b, (Start_SOI + (End_SOI - Start_SOI) * j / Points)))
        end
        for j = Points, 1, -1 do
            -- -- Loop over each point in the right involute in reverse order        -- Calculate the points for the right involute, mirror, rotate for each tooth
            xy[#xy + 1] = Rotation(2 * math.pi * i / z, Rotation(tooth_angle, Mirror(
                inv_c(r_b, (Start_SOI + (End_SOI - Start_SOI) * j / Points)))))
        end
        for j = Points, 1, -1 do -- Loop over each point in the right fillet in reverse order        -- Calculate the points for the right fillet, mirror, rotate for each tooth
            xy[#xy + 1] = Rotation(2 * math.pi * i / z, Rotation(tooth_angle, Mirror(
                Circle(center_a[1].x, center_a[1].y, r_c, (Start_fillet + (End_fillet - Start_fillet) * j / Points)))))
        end
    end
    xy[#xy + 1] = xy[1] -- Close the profile by connecting to the first point
    return xy -- -- Return the generated gear profile
end
```

Fig. 13: Generation of Gear full Profile

5.3 Translate

This function applies a translation to a geometry by specified amounts in the x, y, and z directions. The function moves the geometry to a new location without altering its shape or orientation.

6. OTHER COMPONENTS

6.1 Shaft Pins

Shaft Pins is made up from cylinder hole with 5 unit height and it is subtracts from gear.

```
shaft1 = ccylinder(5, 2*Face_width) -- shaft hole of Gear1
Gear1 = difference(Gear_1, shaft1)
---Create assembly pins for gear 1

base = cylinder(r_f/4, Face_width/4)
h1 = cylinder(2, Face_width/4)
base = difference(base, h1)
p1 = cylinder(4.9, 1.3*Face_width)
p2 = cylinder(2.5, 1.3*Face_width)
p = difference(p1, p2)
pin = union(base, p)

handle = translate(r_f-0.1*z1*m,0,0)*cylinder(z1*m/15,handle_height);
Gear1=union(Gear1, handle);

--place the pin and Gear 1 in the model
emit(translate(0, 0, -1.25*Face_width) * pin)
emit(translate(80, -15, 0) * rotation_2 * rotation_1 * Gear1, 7)
-- creating the Gear1 which will then mesh with the Gear2
shaft2 = ccylinder(5, 2*Face_width) -- Shaft hole of Gear2
-- place a pin assembly for Gear 2
emit(translate(0, a, -1.25*Face_width) * pin)
```

Fig. 14: Generation of shaft Pins

6.2 Connecting Stand

Connecting Stand is made up of create a rectangular prism with subtracting two holes.

```
-- Create a stand for holding this pin
stand= translate(0,a/2,-25)*cube(r_f*0.45,a*0.95,Face_width/4);

stand=difference(stand,translate(0, a/4, -25) * p2)
stand=difference(stand,translate(0, a*0.75, -25) * p2)
emit(stand)
```

Fig. 15: Stand

7. SLICING AND G-CODE

The 3D model can be exported as either an ".stl" file or G-code for 3D printing, based on the user's preference. The following step-by-step instructions will be helpful for this process.

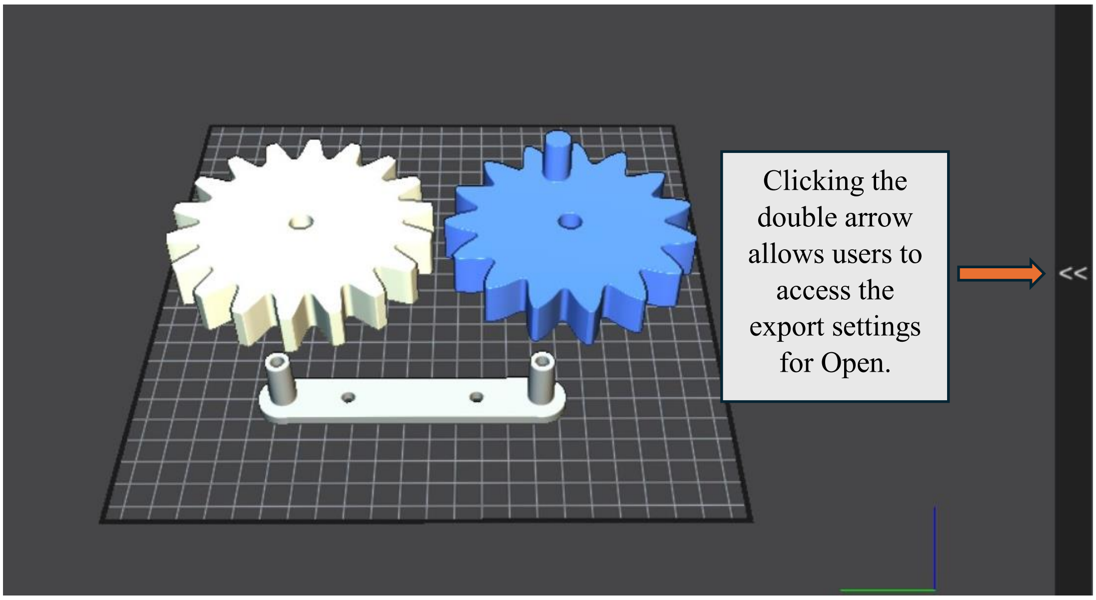


Fig. 16: Slicing Step 1

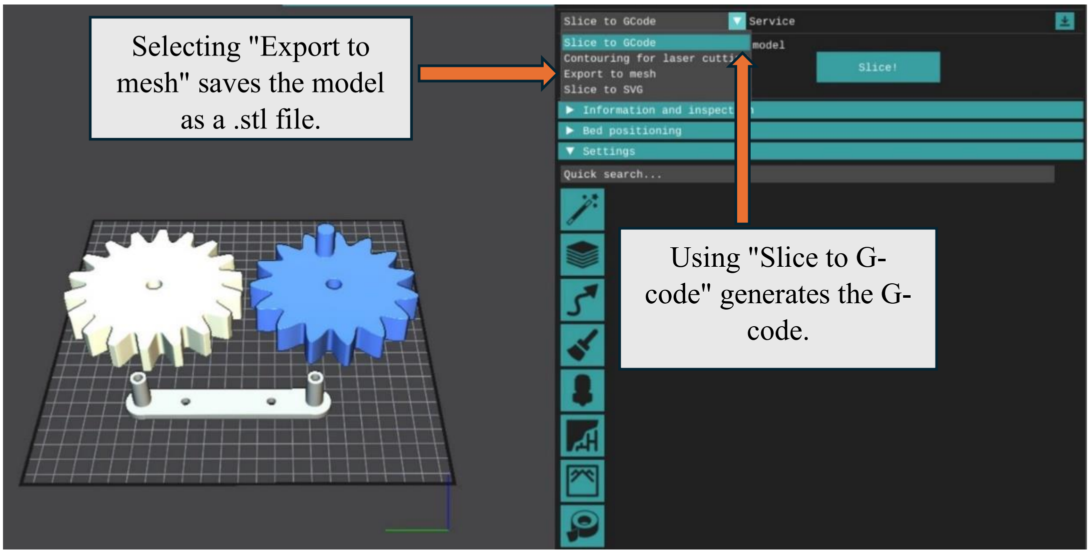


Fig. 17: Slicing Step 2

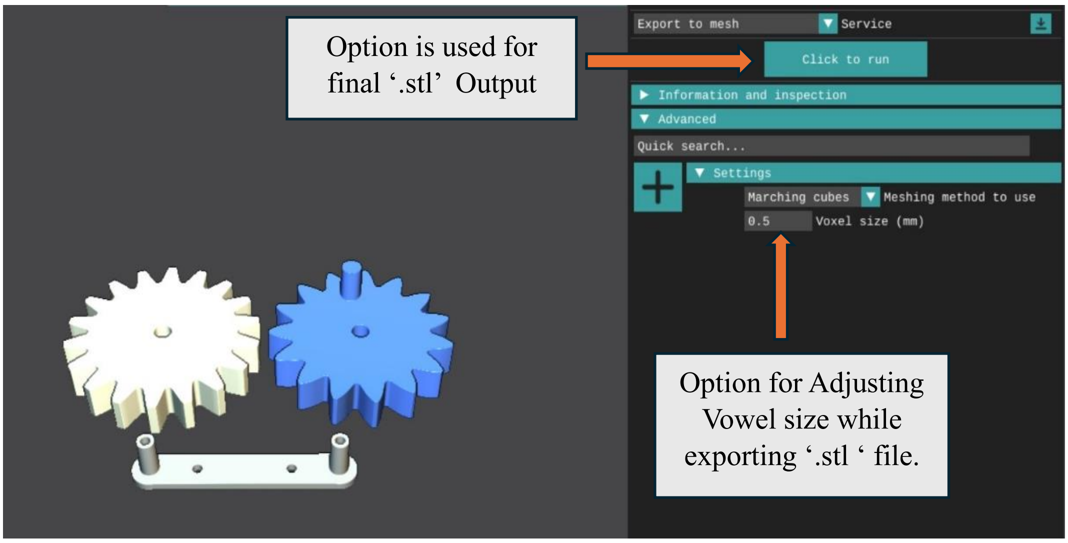


Fig. 18: Adjusting voxel size and run

8. ASSEMBLY OF THE PARTS

A hint for assembly of the printed parts can be seen in the following figures below,

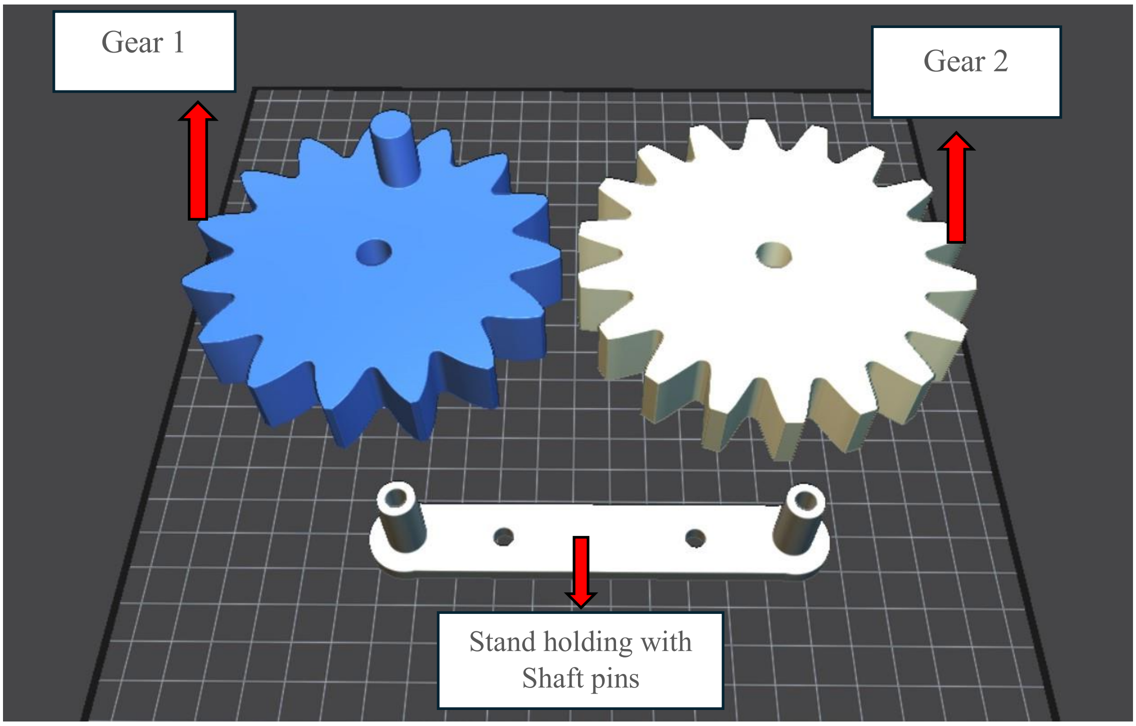


Fig. 19: Components

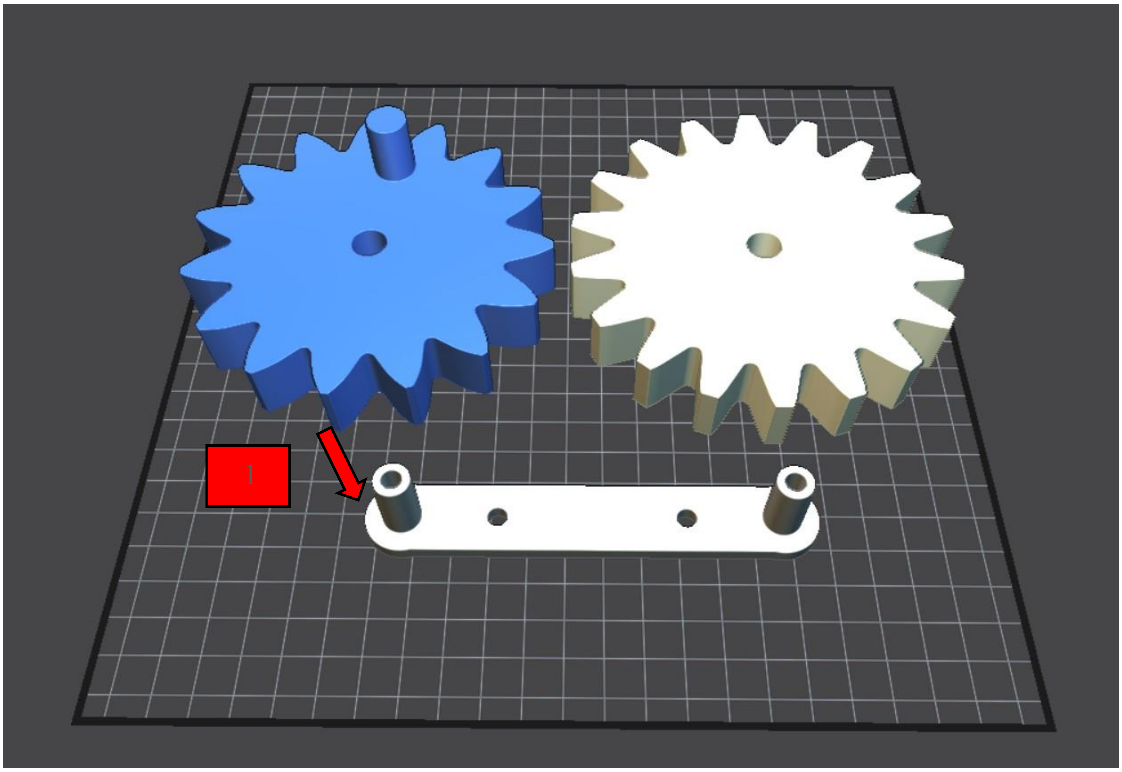


Fig. 20: Assembly of Gear 1 from components

The red arrow shows, where the first and second gear placed in the shaft of the stand

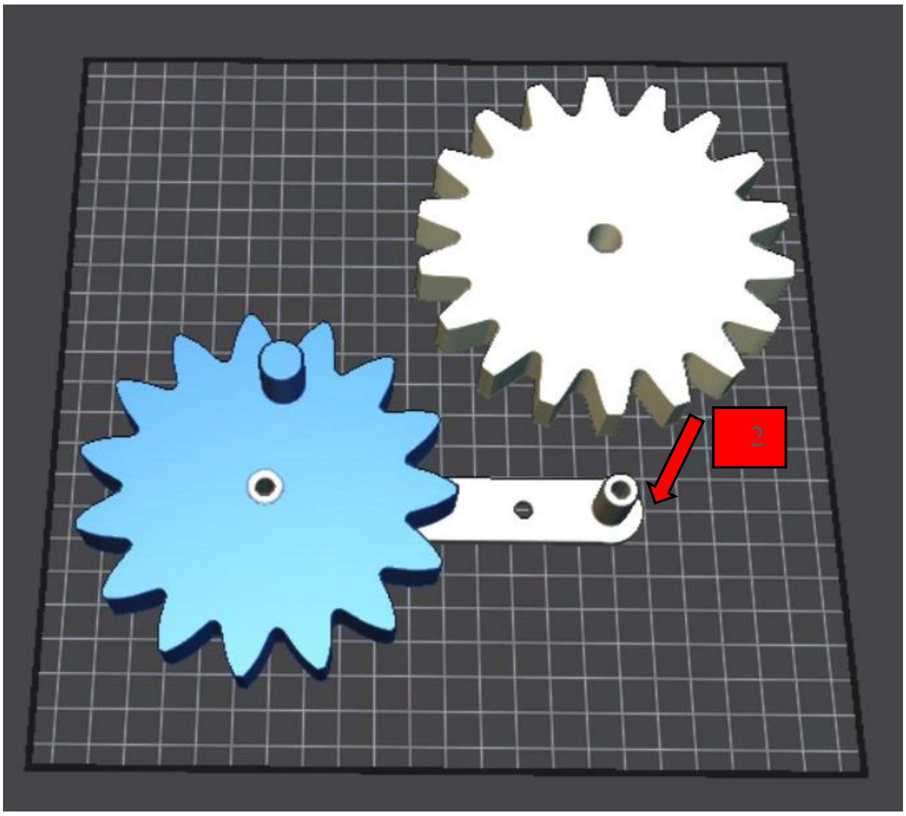


Fig. 21: Assembly of Gear 2 from the components

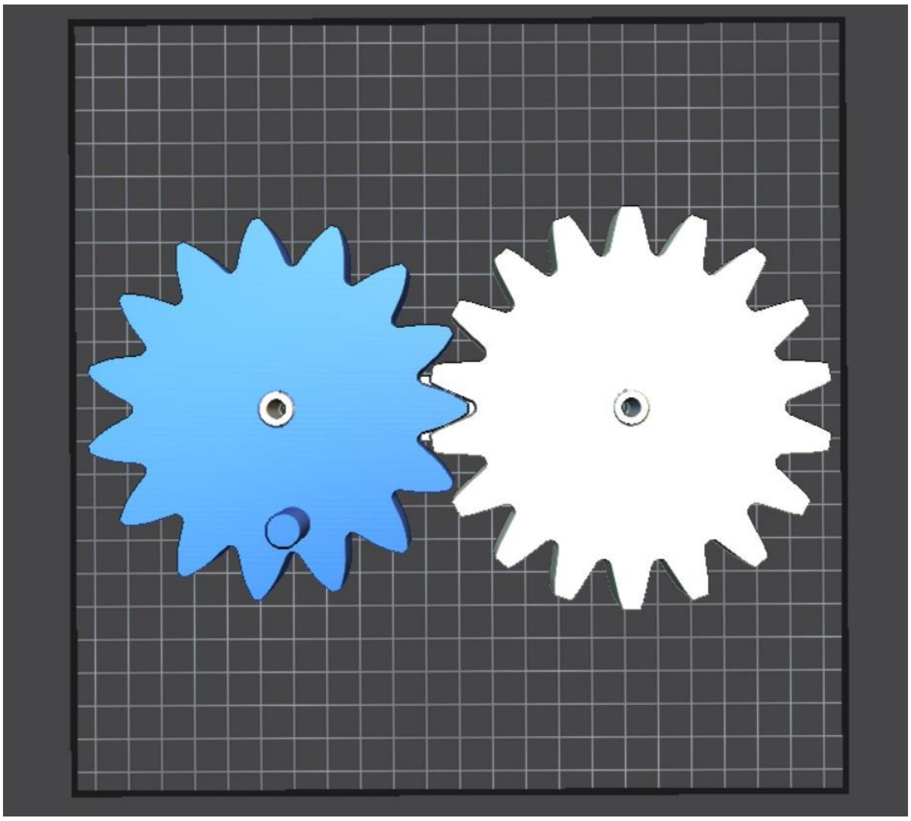
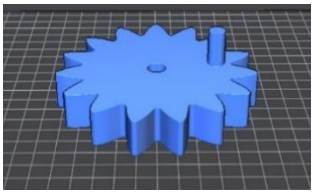
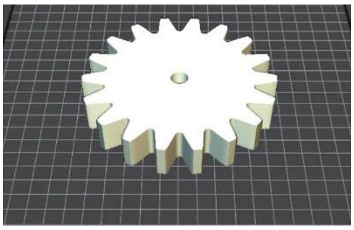
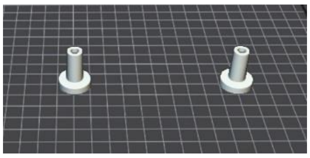


Fig. 22: Complete Assembled Gear

9. LIST OF PARTS

Name of the part	Quantity	Picture
Gear with handle	1	
Gear	1	
Shaft Pins	2	
Spax Screws	2	
Wooden plate	1	
Stand hold with two shaft pins	1