24_final_report

Project Background:

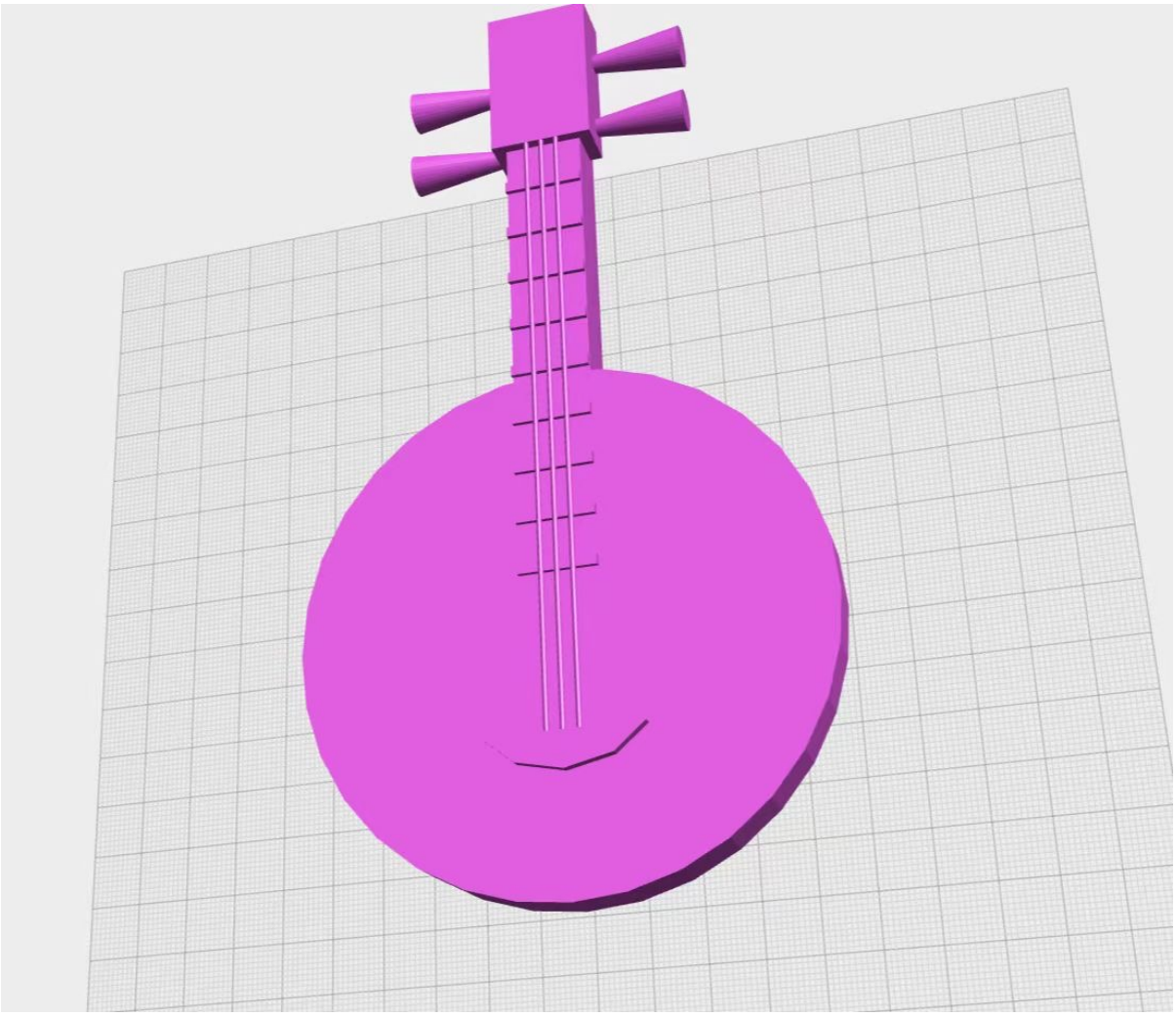
The moon lute, or "yueqin" is a traditional Chinese string instrument with a rich cultural heritage. Known for its distinctive round body and crisp and soft sound, the moon lute plays a significant role in Peking opera and folk traditions music.
Our project aims to create a 3D CG content of the moon lute by openSCAD and Three.js. Through this project, we hope to contribute to the preservation of the traditional Chinese instrument, offering a digital tribute to a cherished cultural artifact.

Topic Analysis:

Modelling

Phase 1:

Reuth utilized OpenSCAD to develop an initial model of a moon guitar, which serves as the foundational base for subsequent enhancements.
This general model incorporates all the critical components of a moon guitar, including the main body, neck, four pegs, three strings, a crescent shape, and several frets. All our work will build upon and evolve from this primary model.

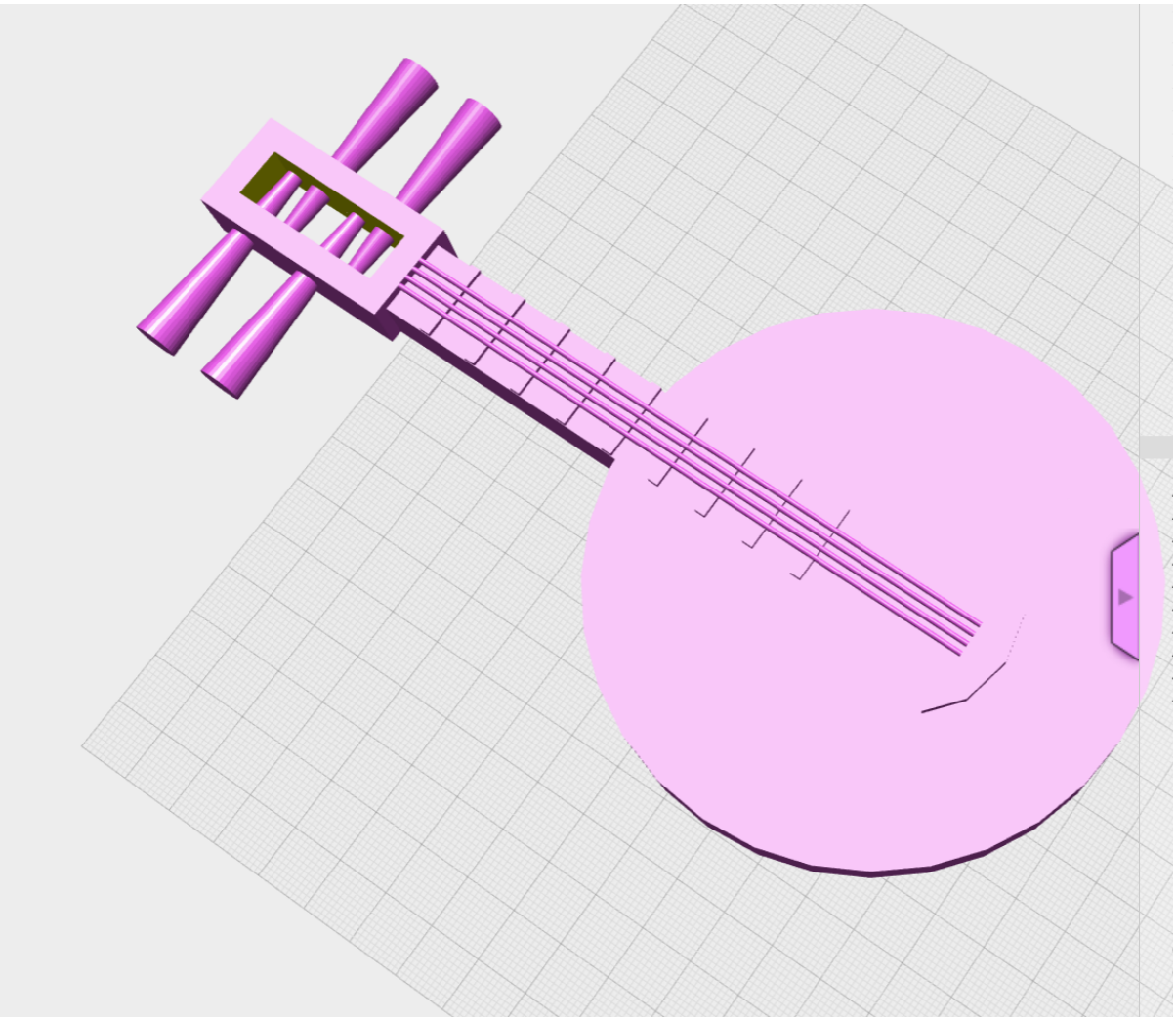


Below shows the codes of our first draft.

```
function main(){
return union
}
```

Phase 2:

Hollow out the head of the Yueqin and adjust the peg to make it fit the prototype structure.
This adjustment gives the instrument a more lively and less cumbersome appearance.

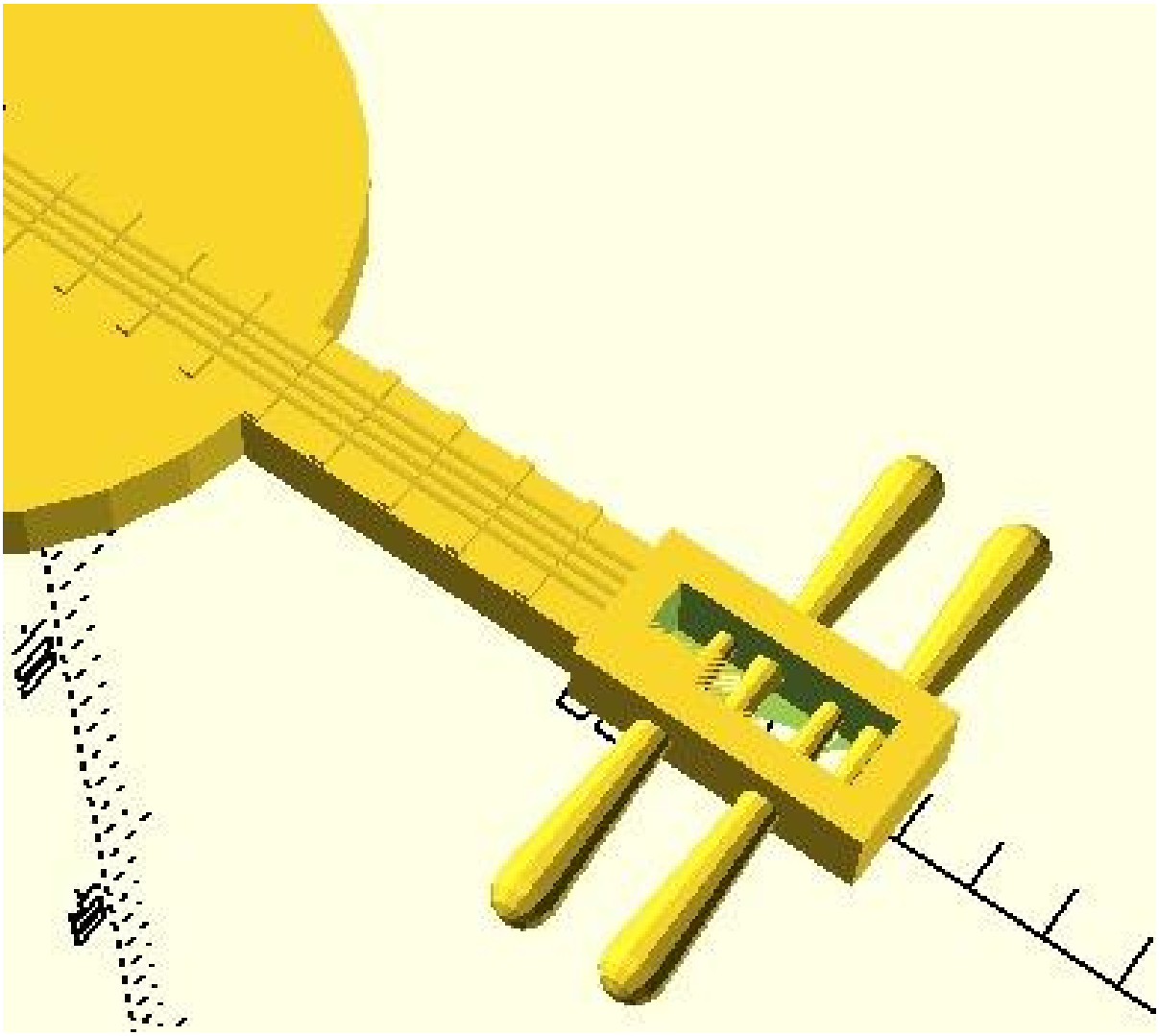


To achieve this, we remove a smaller cube from the head using difference() function.

```
module hole() {
  translate([-5, 105, 5])
    cube([10, 30, 12]);
}

module headWithHole() {
  difference() {
    head();
    hole();
  }
}
```

Phase 3

Add details to the head of the Yueqin to make it curved. And add peg details to make its head be a half sphere.



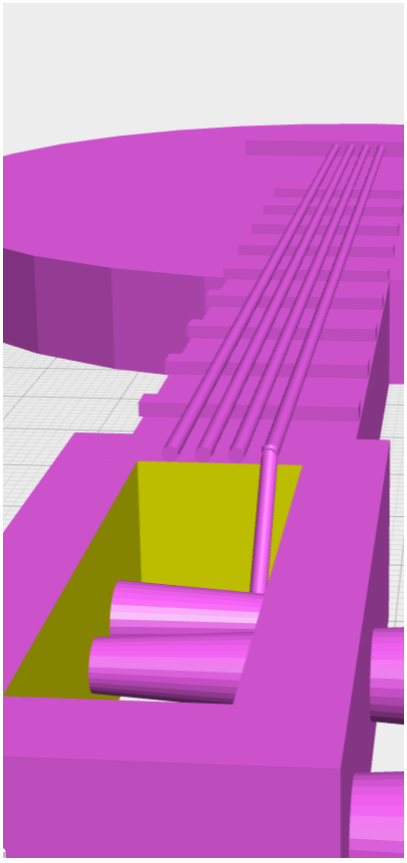
```
module curve () {
    union () {
        difference(){
         // main
        translate ([0, 120, 5 + 10 * sqrt (11)]){
            rotate ([0, 90, 0])
                cylinder (r = 40, h = 20, center = true, $fn = 200);
        }
        // block 1
        translate ([-10, 100, 5])
           cube ([30, 100, 80]);

        translate ([-10, 140, -10])

           cube ([30, 100, 120]);
        //block 2
        translate ([-15, 20, -40])
                cube ([40, 80, 120]);
        //block 3
         translate([-5, 105, -40])
```

```
                cube([10, 30, 120]);
        }
    }
}
```

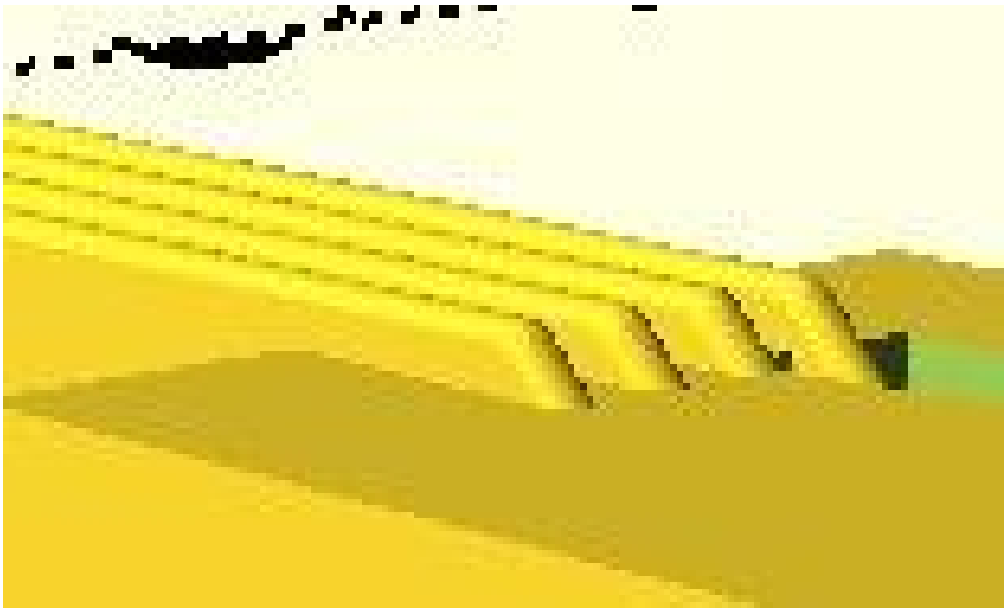
Phase 4

Adjust the strings so that they can turn naturally and connect with the corresponding pegs.

The initial plan was to simply splice two cylinders, but the result of this was that the intersection of the two cylinders was very rough.



Then tried to use hull() to make the two cylinders connect naturally, but it failed in the end because it would turn the two cylinders into sheets.

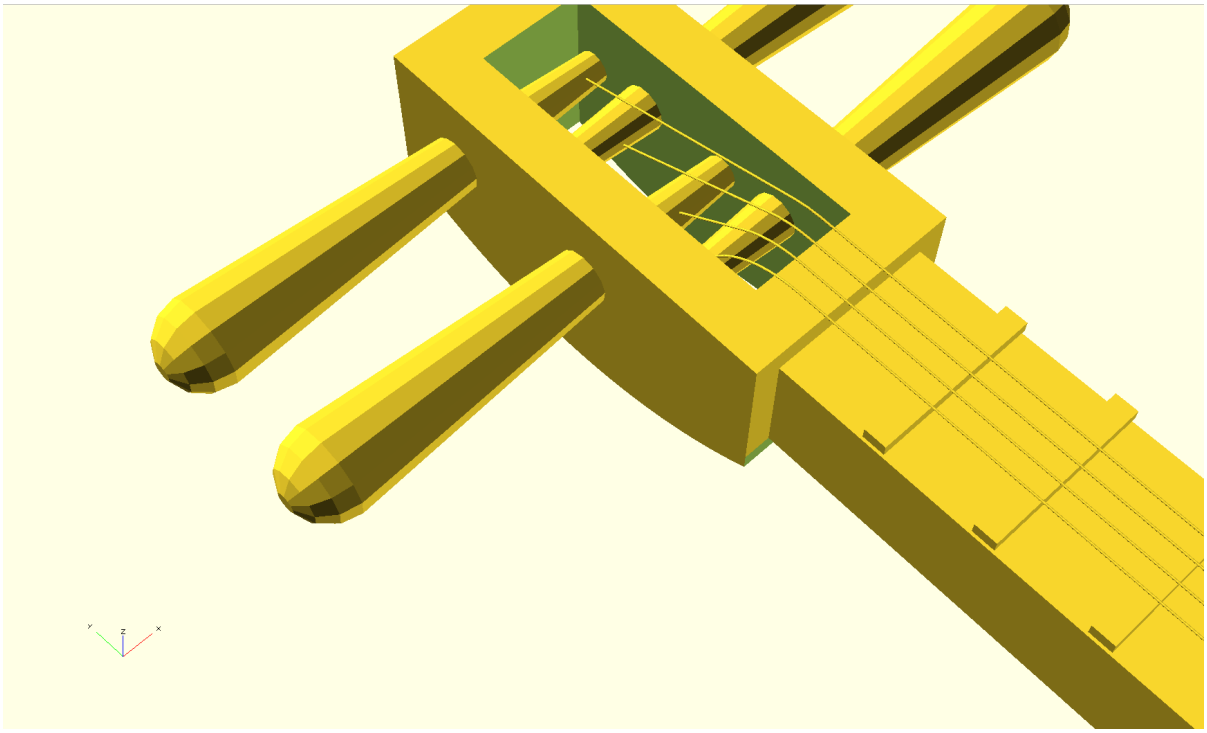


Finally, we used wiring.scad in The Belfry OpenScad Library, v2.0 to simulate the strings.
According to the documentation, the `wire_bundle()` function requires a minimum of three points to define a path and subsequently generates a 3D model of a wire bundle that adheres to this specified path.

```
wire_bundle(path, wires, [wirediam], [rounding], [wirenum=], [corner_steps=]);
```

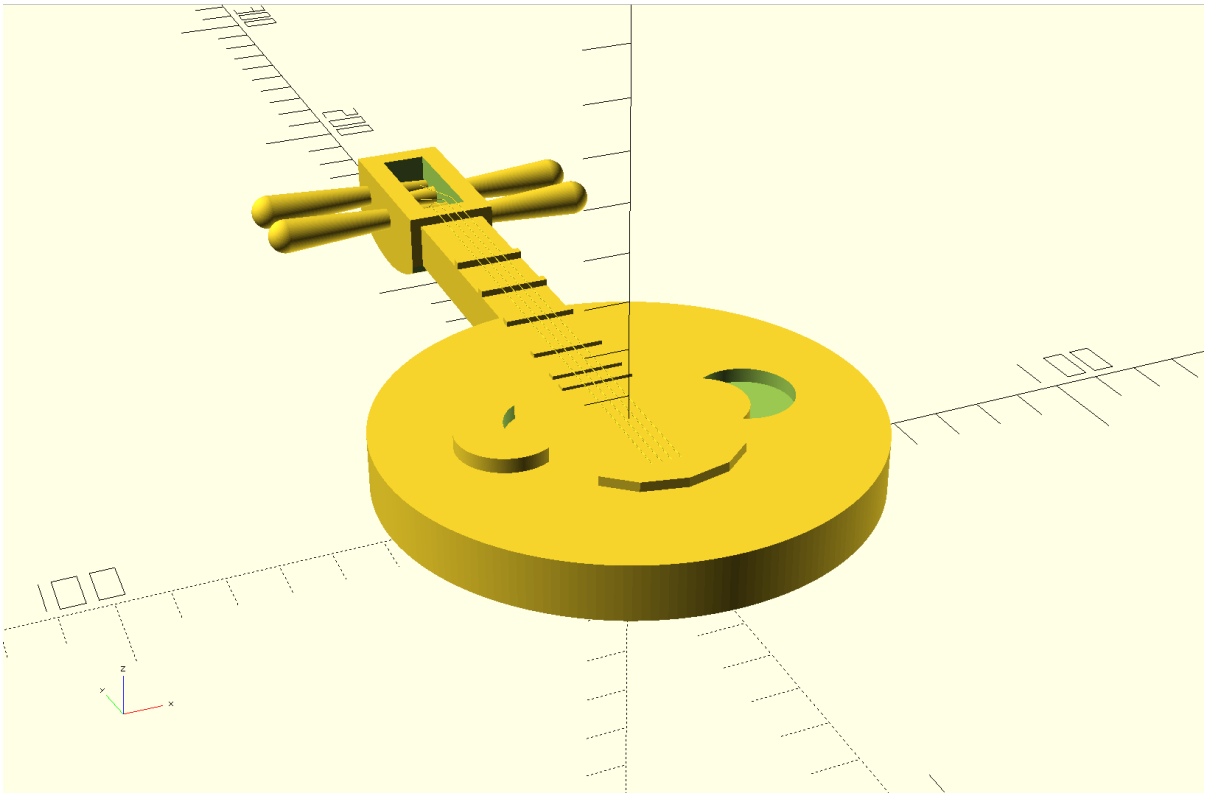
By using this function, we successfully connect the strings smoothly and create a curved effect on the strings, making them appear softer.

```
wire_bundle([[-3,113.5,10], [-3,107.5,17.3], [-3, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[-1,118.5,10], [-1,107.5,17.3], [-1, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[1,125.5,10], [1,107.5,17.3], [1, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[3,132.5,10.8], [3,107.5,17.3], [3, -25, 15]],
wirediam=0.2,rounding=10,wires=1);
```



Phase 5

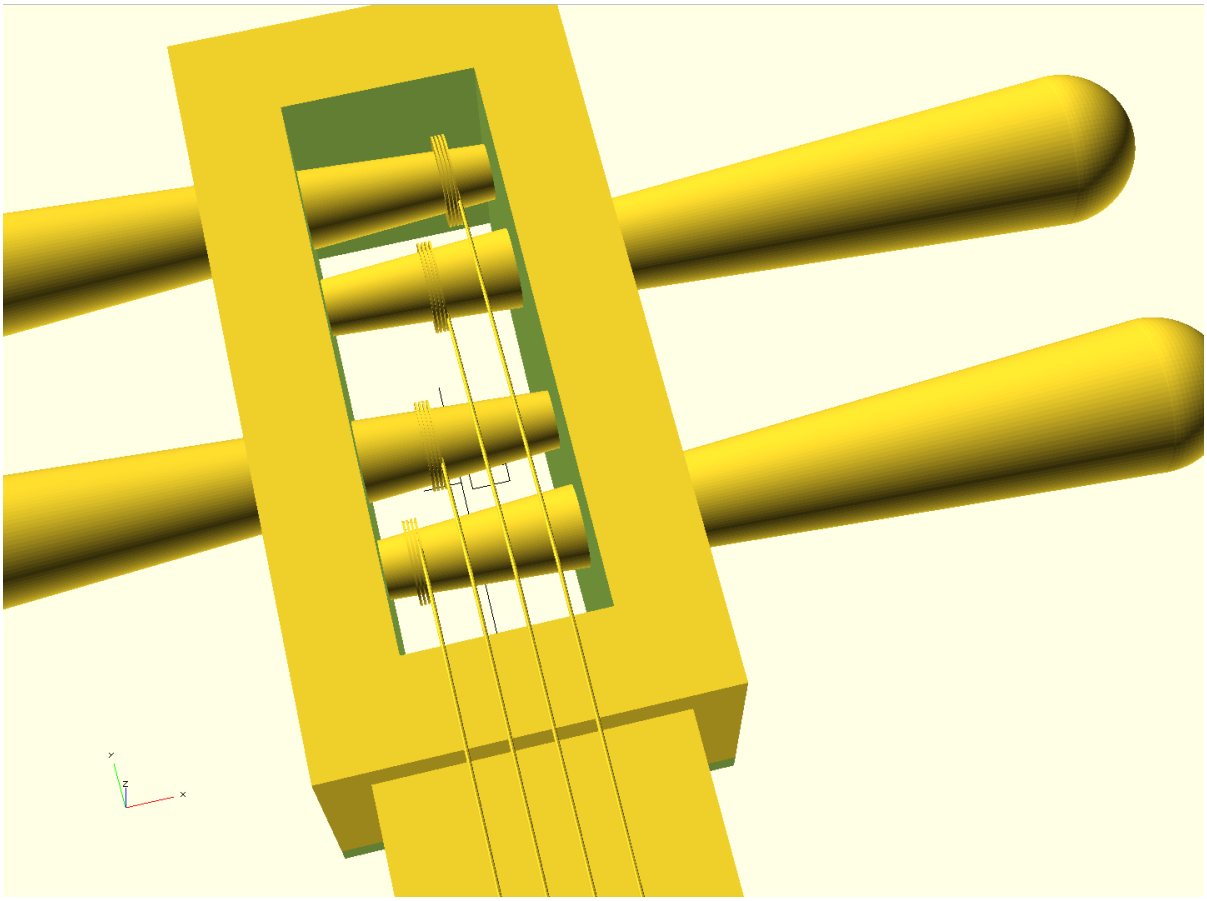
Decorate the soundboard of the Yueqin and add $fn to make the model more refined.



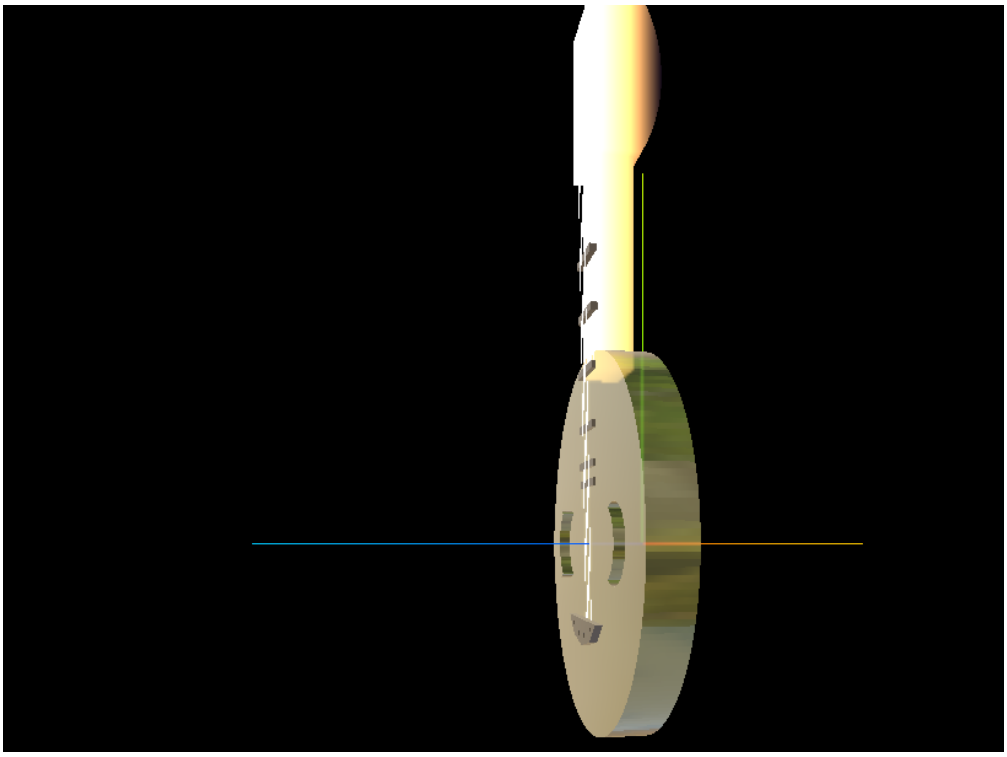
Phase 6

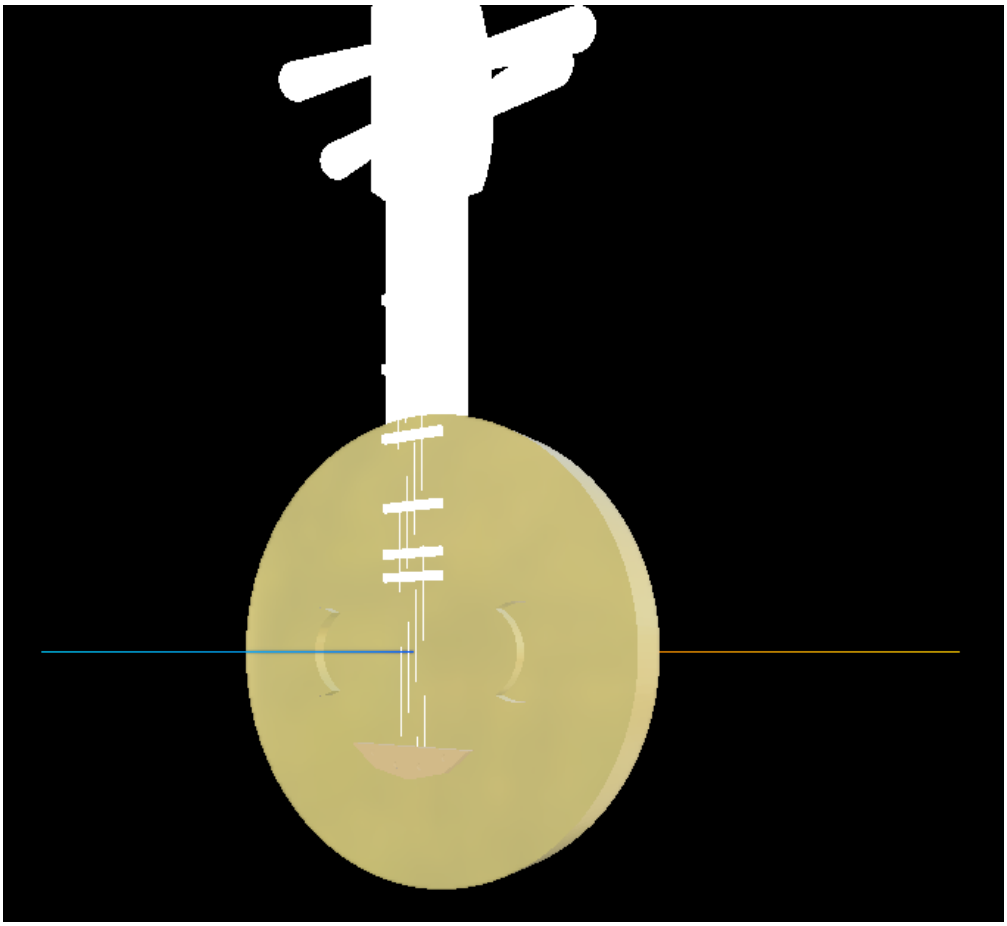
At first, we tried wire_budle() to wrap the strings around the pegs, however since pegs are rotated cylinders, it is too hard to obtain the accurate coordinates to design paths.
So we decided to make use of for loop to simulate a coil shape. The for loop iteratively generates and modifies cylinders. Each iteration translates a pair of cylinders along the z-axis by an increasing offset (`0.1 * i`), which progressively stacks them with a slight vertical displacement. This results in a spring-like 3D structure.

```
    translate([-2.9, 112, 10.07]) {
        rotate([0, 270, 0]) {
            /*cylinder(r1 = 1.6, r2 = 1.65 h = 2);*/
            for (i = [0: 2: 6]) {
                difference() {
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r = 2.15, center = false, $fn = 100);

                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r1 = 2.10 , center = false, $fn = 100);
                }
            }
        }
    }
```



Rendering:





Animation:

The moon guitar rotates periodically around its X, Y, and Z axes. Upon loading, the moon guitar zooms in from a distance before starting its rotation.

Contribute description:

Lijiang JIANG：

Modeling

- The model in Phase 2 [Hollow out the head of the Yueqin and adjust the peg to make it fit the prototype structure]
- Phase 4[Adjust the strings so that they can turn naturally and connect with the corresponding pegs](two cylinders version and final version with wiring.scad)
- Phase 5 [Decorate the soundboard of the Yueqin and add $fn to make the model more refined].

Other:

- Split the integrated code into components for rendering use.

Report:

- Project Background part
- Topic Analysis part, Modeling part
- partially Contribute description part
- partially Summary and reflection part.

Buyi LYU：

Modeling:

- Building the initial prototype of the moon guitar in OpenJSCAD, including the guitar body, head, strings, frets, and tuning pegs.

Animation:

- Create a loading animation: the moon guitar zooms in from a distance. Make the background image rotate over a period of time.

Zaoshen ZHANG：

Rendering:

- The model in Phase of Rendering， the disassembled parts of the model were imported into THREE.js for rendering and appropriate colors were adjusted. Later, new light sources (ambient light and sun-like light sources) were added to the original code to make the model surface more clearly visible.I initially used some color matching in the Yueqin but it didn't work well. The color was eventually changed to wood by the other members.

WEI Youlin：

Modeling:

- The model in Phase 3 [Add details to the head of the Yueqin to make it curved. And add peg details to make its head be a half sphere]
- Phase 4 [Adjust the strings so that they can turn naturally and connect with the corresponding pegs] (hull() version)
- Phase 6 [Add the wire wrapped around the peg].

Rendering:
Rendering the light yellow version of moon guitar.

Report:

- Formatting report

Code:

Modelling:

```
include <BOSL2/std.scad>
include <BOSL2/wiring.scad>
module first() {
  union() {
    // body
    //translate([0, 0, 3.5])
      //cylinder(r = 50, h = 11.5);

    // frets
    translate([-7.5, 15, 14.65])
    cube([15, 2, 1]);

    translate([-7.5, 20, 14.75])
    cube([15, 2, 1]);

    translate([-7.5, 30, 14.9])
    cube([15, 2, 1]);

    translate([-7.5, 45, 14.15])
    cube([15, 2, 2]);

    translate([-7.5, 60, 14.39])
    cube([15, 2, 2]);

    translate([-7.5, 75, 14.6])
    cube([15, 2, 2]);

    //string
```

```
    wire_bundle([[-3,113.5,10], [-3,107.5,17.3], [-3, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[-1,118.5,10], [-1,107.5,17.3], [-1, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[1,125.5,10], [1,107.5,17.3], [1, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    wire_bundle([[3,132.5,10.8], [3,107.5,17.3], [3, -25, 15]],
wirediam=0.2,rounding=10,wires=1);

    // tuning pegs with spherical tops
    for (k = [[5, 118, 10], [5, 133, 10], [-5, 112, 10], [-5, 127, 10]]) {
      translate([k[0], k[1], k[2]])
      rotate([0, k[0] > 0 ? 270 : 90, 0]) {
        cylinder(r1 = 1.5, r2 = 4, h = 40,$fn=100);
        translate([0, 0, 40])  // moving to the top of the cylinder
          sphere(r = 4,$fn=100);
      }
    }


    translate([0, -20, 17])
      rotate([180, 0, 0])
        linear_extrude(height = 3)
          polygon([
            [-15, 0], [-9, 5], [0, 7], [9, 5], [15, 0], [0, 0]
          ]);
  }
}

module winding() {
    translate([-2.9, 112, 10.07]) {
        rotate([0, 270, 0]) {
            /*cylinder(r1 = 1.6, r2 = 1.65 h = 2);*/
            for (i = [0: 2: 6]) {
                difference() {
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r = 2.15, center = false, $fn = 100);

                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r1 = 2.10 , center = false, $fn = 100);
                }
            }
        }
    }
   translate ([-1,118,10.07]){
       rotate([0, 270, 0]){
           for (i = [0: 2: 6]) {
                difference() {
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r = 2.3, center = false, $fn = 100);
```

```
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r1 = 2.25 , center = false, $fn = 100);
            }
        }
    }
}
  translate ([1,127,9.8]){
      rotate ([0 , 270, 0])
        //cylinder (r = 2.5, h = 2);
      for (i = [0: 2: 6]) {
                difference() {
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r = 2.4, center = false, $fn = 100);

                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r1 = 2.35 , center = false, $fn = 100);
      }
   }
}

   translate ([3,133, 10.0]){
       rotate ([0, 270, 0])
       //cylinder (r = 2.6, h = 2);
       for (i = [0: 2: 6]) {
                difference() {
                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r = 2.45, center = false, $fn = 100);

                    translate([0, 0, 0.1 * i])
                        cylinder(h = 0.12, r1 = 2.40 , center = false, $fn = 100);
        }
      }
   }
}

module head() {
  union() {
    // neck
    translate([-7.5, 45, 5])
      cube([15, 80, 10]);
    // head
    color("red"){
    translate([-10, 100, 5])
      cube([20, 40, 12]);
    }

  }
}

module curve () {
    union () {
        difference(){
```

```
         // main
        translate ([0, 120, 5 + 10 * sqrt (11)]){
            rotate ([0, 90, 0])
                cylinder (r = 40, h = 20, center = true, $fn = 200);
        }
        // block 1
        translate ([-10, 100, 5])
           cube ([30, 100, 80]);

        translate ([-10, 140, -10])

           cube ([30, 100, 120]);
        //block 2
        translate ([-15, 20, -40])
                cube ([40, 80, 120]);
        //block 3
         translate([-5, 105, -40])
               cube([10, 30, 120]);
    }
  }
}

module hole() {
  translate([-5, 105, 5])
    cube([10, 30, 12]);
}

module headWithHole() {
  difference() {
    head();
    hole();
  }
}

module moonL(){
  difference() {
    translate([-25,0,15])
    cylinder(r = 10, h = 3,$fn=200);
    translate([-15,0,15])
    cylinder(r = 10, h = 3,$fn=200);
  }
}

module moonR(){
  difference() {
    translate([25,0,12])
    cylinder(r = 10, h = 3, $fn=200);
    translate([15,0,12])
    cylinder(r = 10, h = 3,$fn=200);
  }
}
/*
module test(){
```

```
    color("red"){
        winding();
        }
}
*/
//body with moon
module bodyMoon(){
  difference() {
    translate([0, 0, 3.5])
    cylinder(r = 50, h = 11.5,$fn=200);
    moonR();
 }
}

module main() {
 translate ([0, 0, 0]){
  union() {
    winding();
    bodyMoon();
    //test();

    first();
    headWithHole();
    curve();
    moonL();
  }
}
}


main();
```

Experimental results:

We successfully create a 3D CG content of the moon lute by openSCAD and Three.js.

Summary and reflection:

While the project achieved its primary objectives, there were still some parts related to accurately capturing the Yueqin's details and translating them into a digital format can be improved. Future iterations of the project could explore additional features such as interactive elements, or virtual reality performance to enhance user engagement and educational value.