FINAL PEOJECT

Group 5

Team Members

Zhengxuan Li, Weiyu Ouyang, Yizhen Sun, Youyou Wu

CONTENT

About project----------1
Modeling process----------2-16
Rendering----------17-27
Animation----------28
VR----------29
RESPONSIBLE PART of PROJECT----------30-31

ABOUT PORJECT

Water lily pastry

Sweetheart pastry

Peach blossom pastry

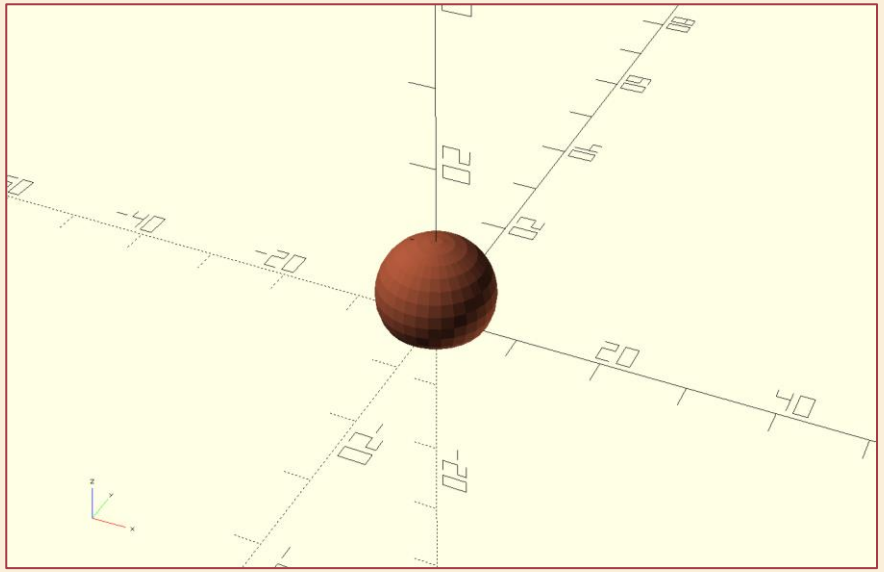
Mooncake

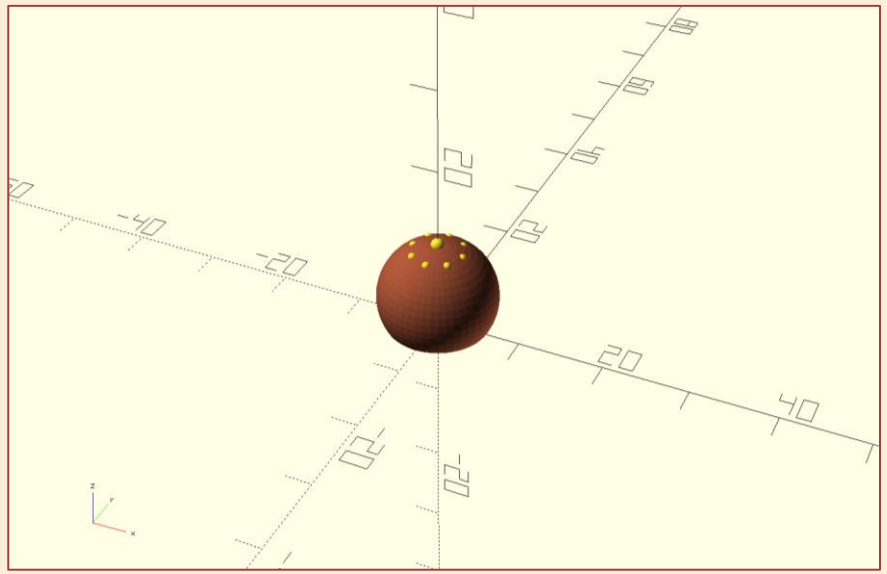
Chinese New Year pastries embody Chinese traditional culture, symbolizing luck and reunion. With rich flavors and festive joy, they bring warmth and tradition to every celebration.

Each member of our group chose a delicious Chinese new year pastries, to model, namely Water lily pastry, Sweetheart pastry, Peach blossom pastry and Mooncake. Next is our modeling process.

1. Water lily pastry

MODELING PROCESS

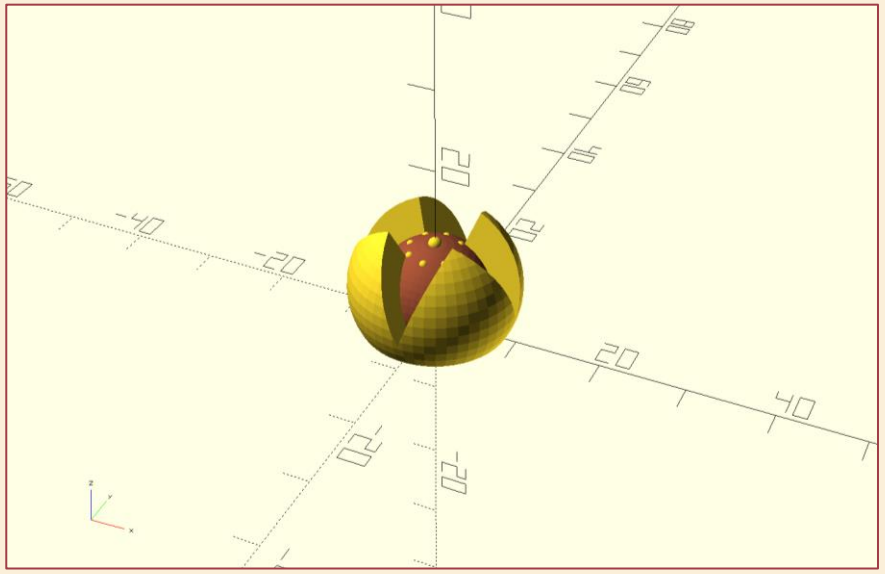


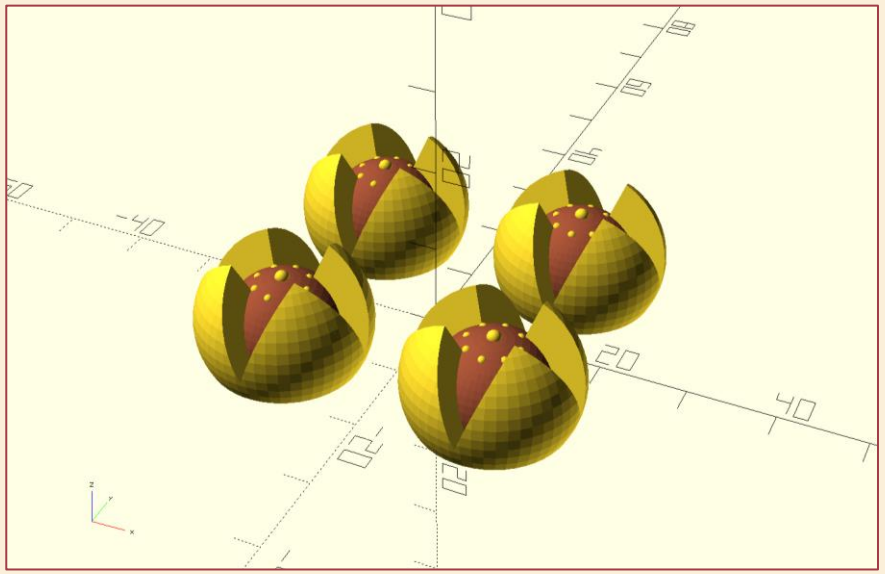


Step1.Use the sphere minus the bottom to form the filling for the pastry.

Step2. Sprinkle some garnishes on top of the filling to give it a more sophisticated look.

MODELING PROCESS

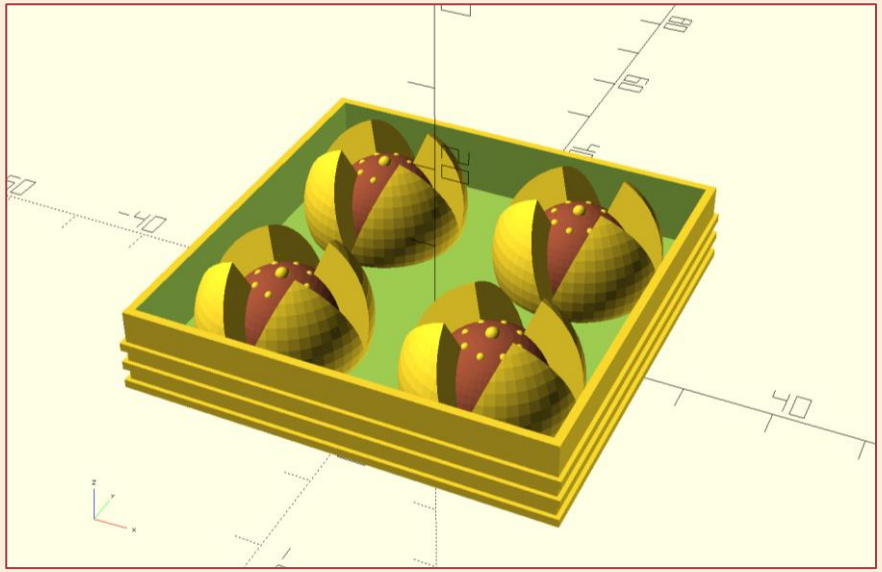


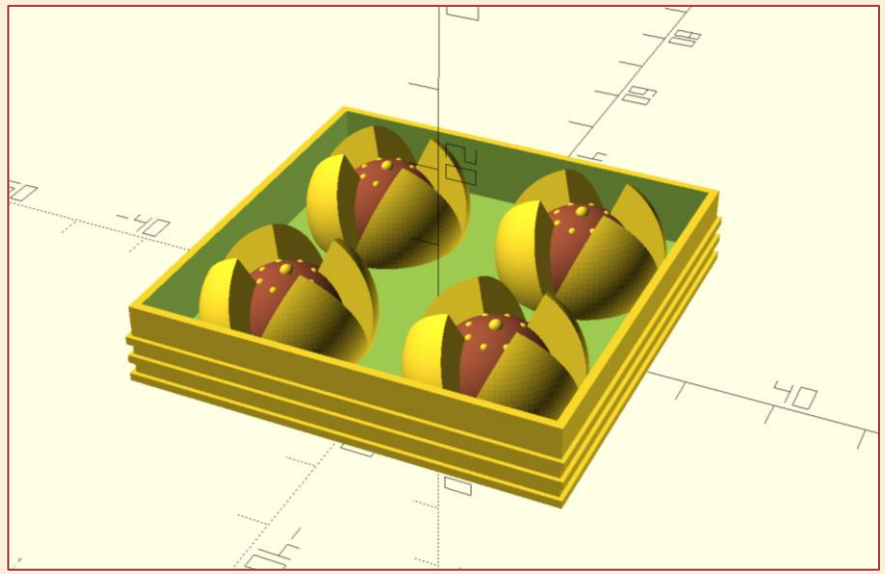


Step3. Use Boolean operations to add snack cracked puff pastry.

Step4. Generate multiple snacks via module() and loops.

MODELING PROCESS



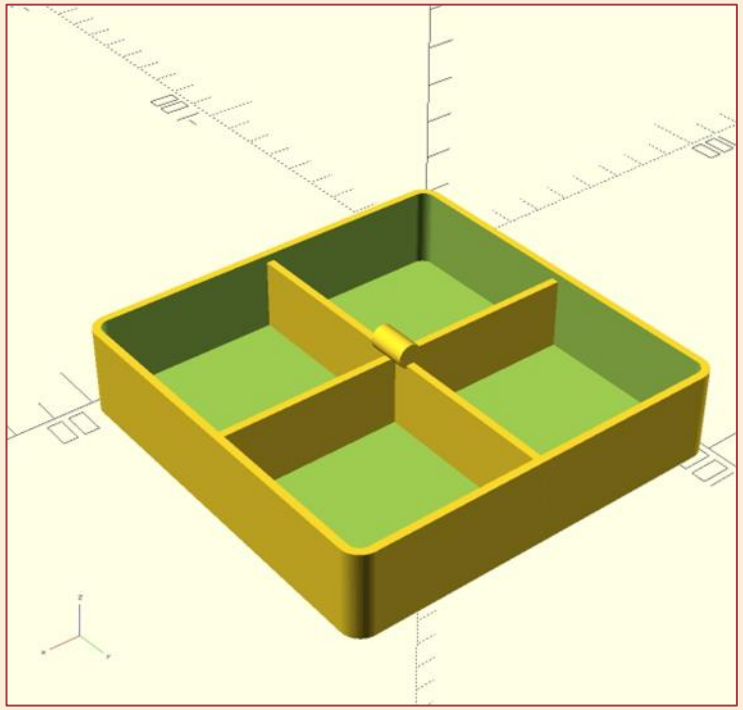


Step5.Use the sphere minus the bottom to form the filling for the pastry.

Step6. Increase the number of modeling surfaces to make the treats look smoother .

2. Sweetheart pastry

MODELING PROCESS



Step1.

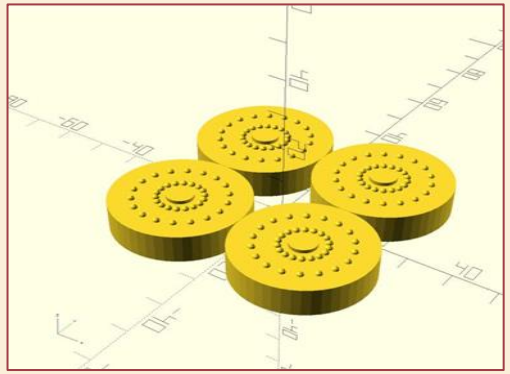
Build the box.

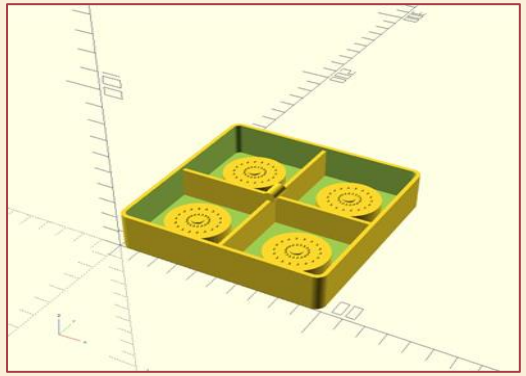
Add cross-shaped dividers.

Create the lid.

Add a small cylindrical handle.

MODELING PROCESS





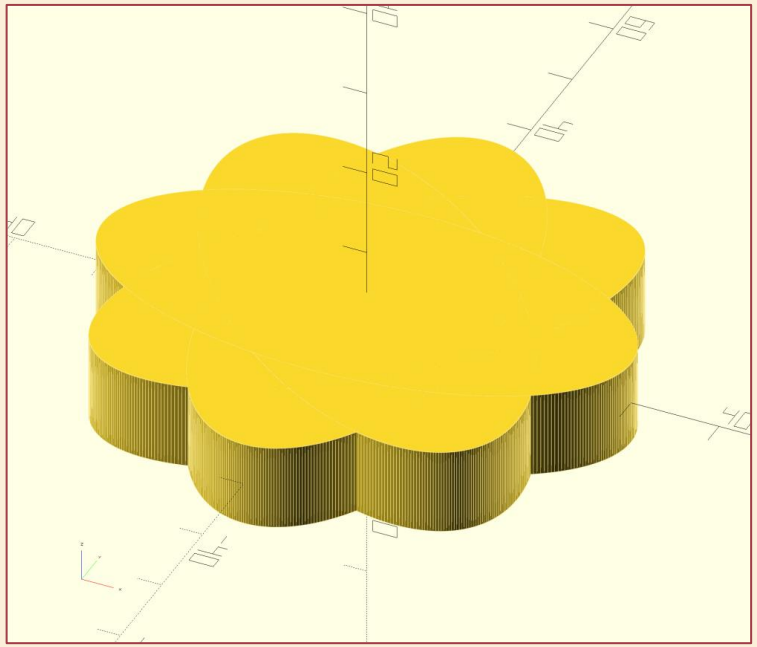
Step2.Model the pastries.

Step3.Arrange the pastries.

Step4.Assemble the complete model.

3. Peach blossom pastry

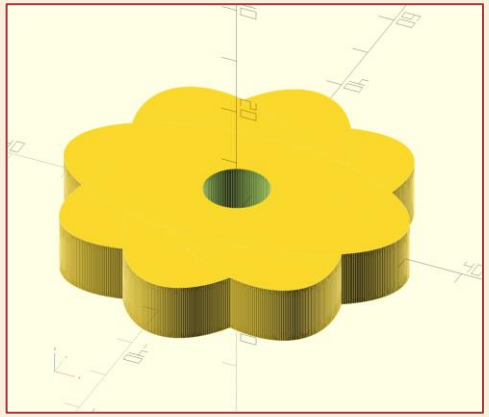
MODELING PROCESS

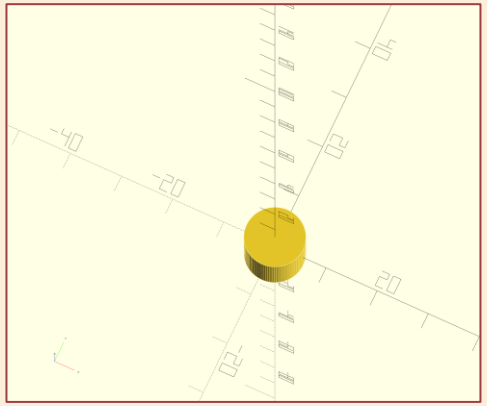


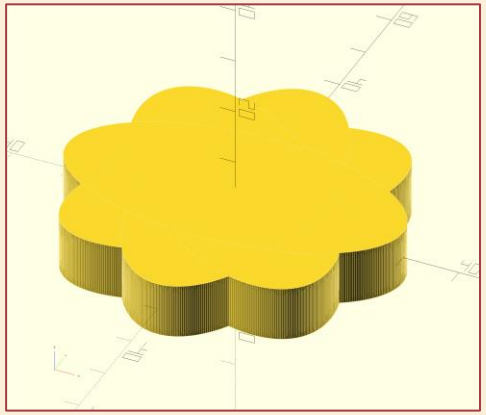
Step1.

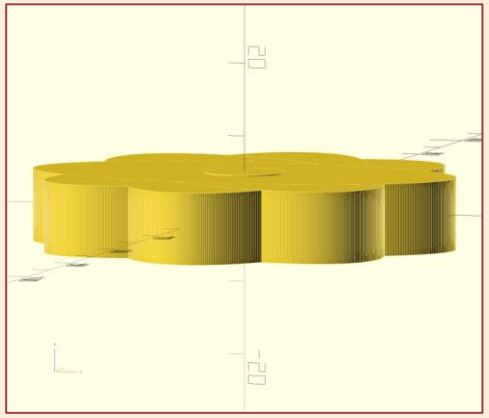
Simulate the shape of pastry petals

MODELING PROCESS







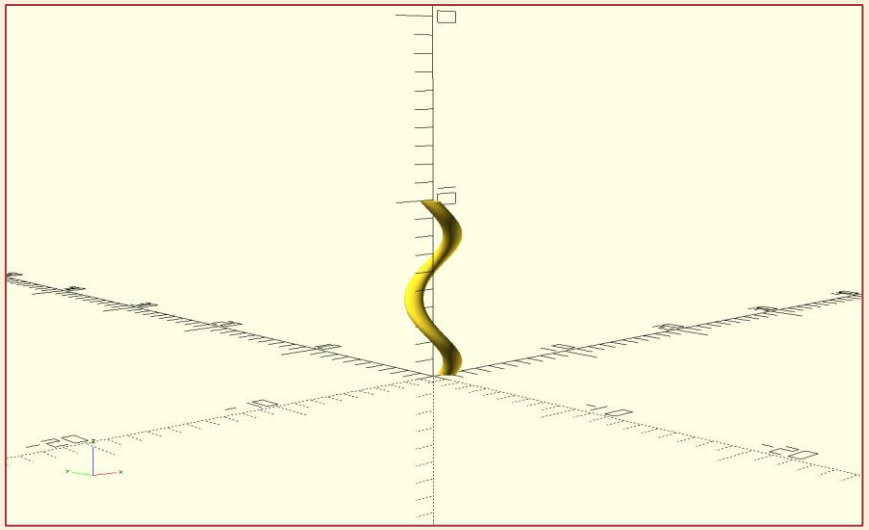


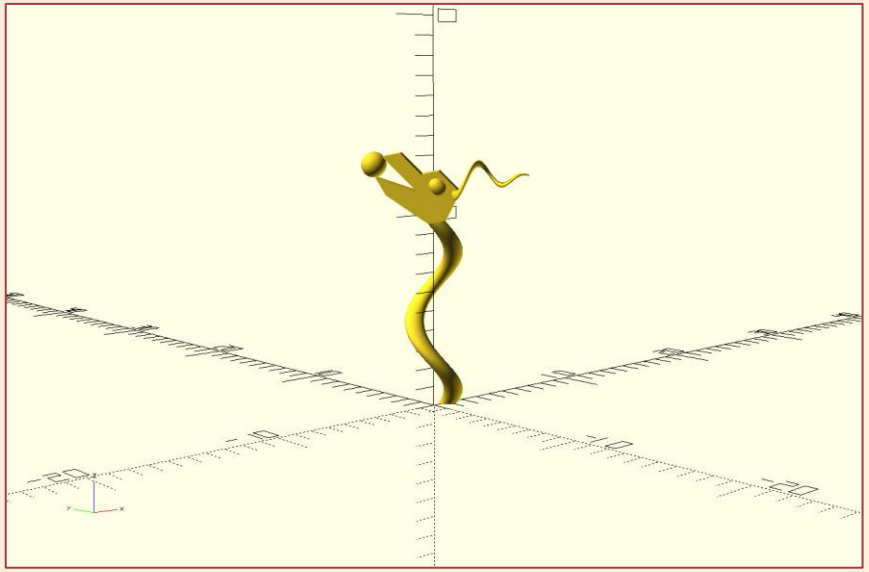
Step2.Leave the filling in the middle of the model.

Step3.Use the sphere minus the filling of the pastry.

4. Mooncake

MODELING PROCESS

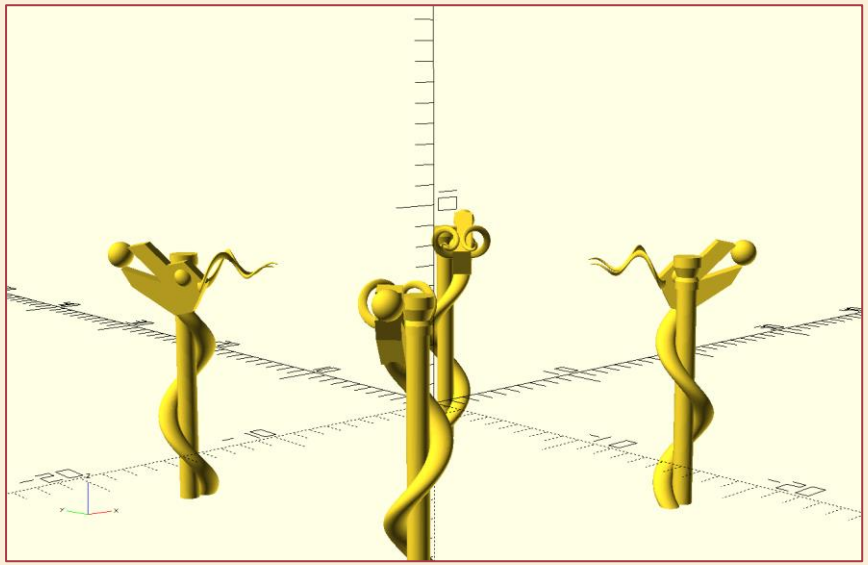


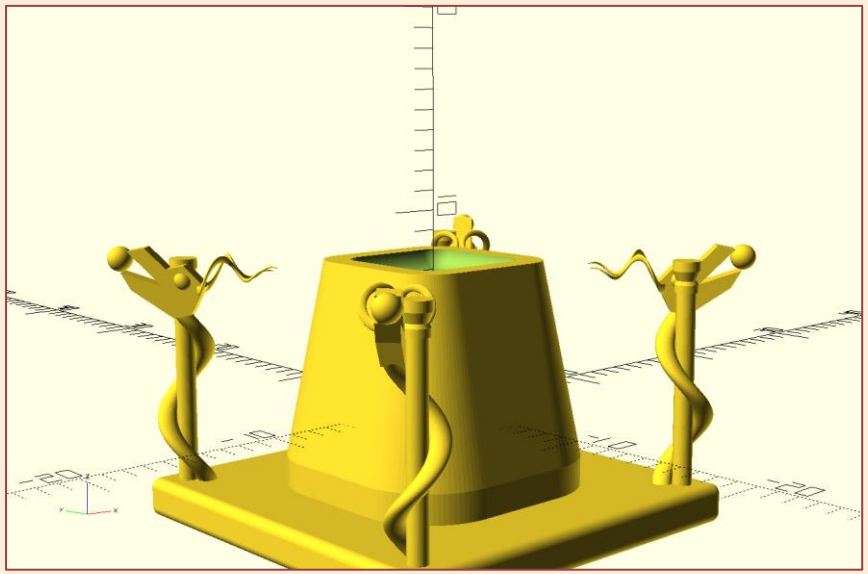


1.It all starts from a simple linear_extrude...

2.It reminds me of some kind of dragons, so why not? Just some transformation of simple geometry. Only one thing to say is that I use linear_extrude, moving a polygon to create the head.

MODELING PROCESS

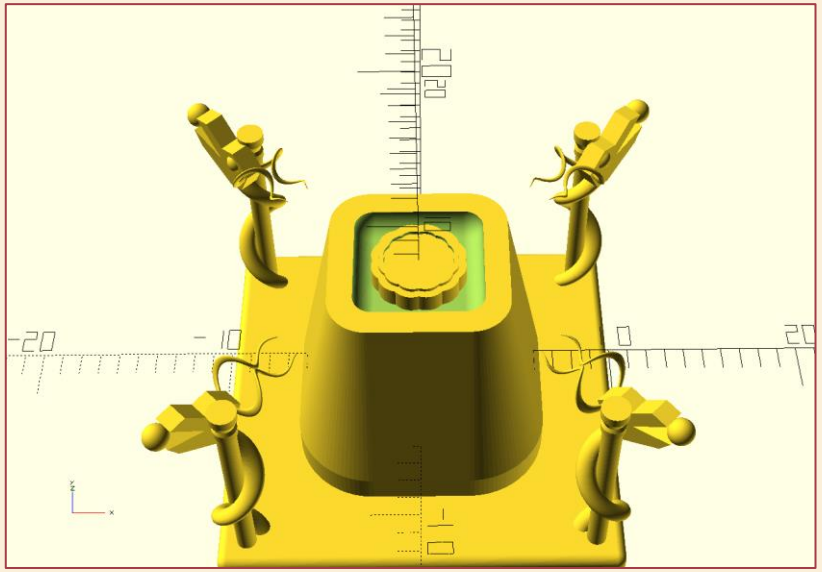


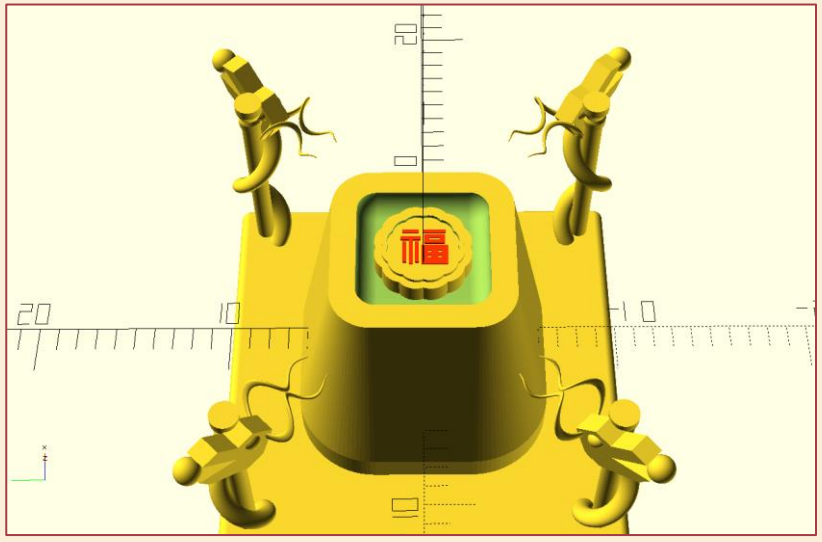


3.Create a pillar with some cylinder and mirror the whole object, twice.

4.I love minkowski, don’t know how exactly it works (I mean the mathematical calculation behind) but it help me made smoother geometries. Also use difference to create the hollow on the top.

MODELING PROCESS

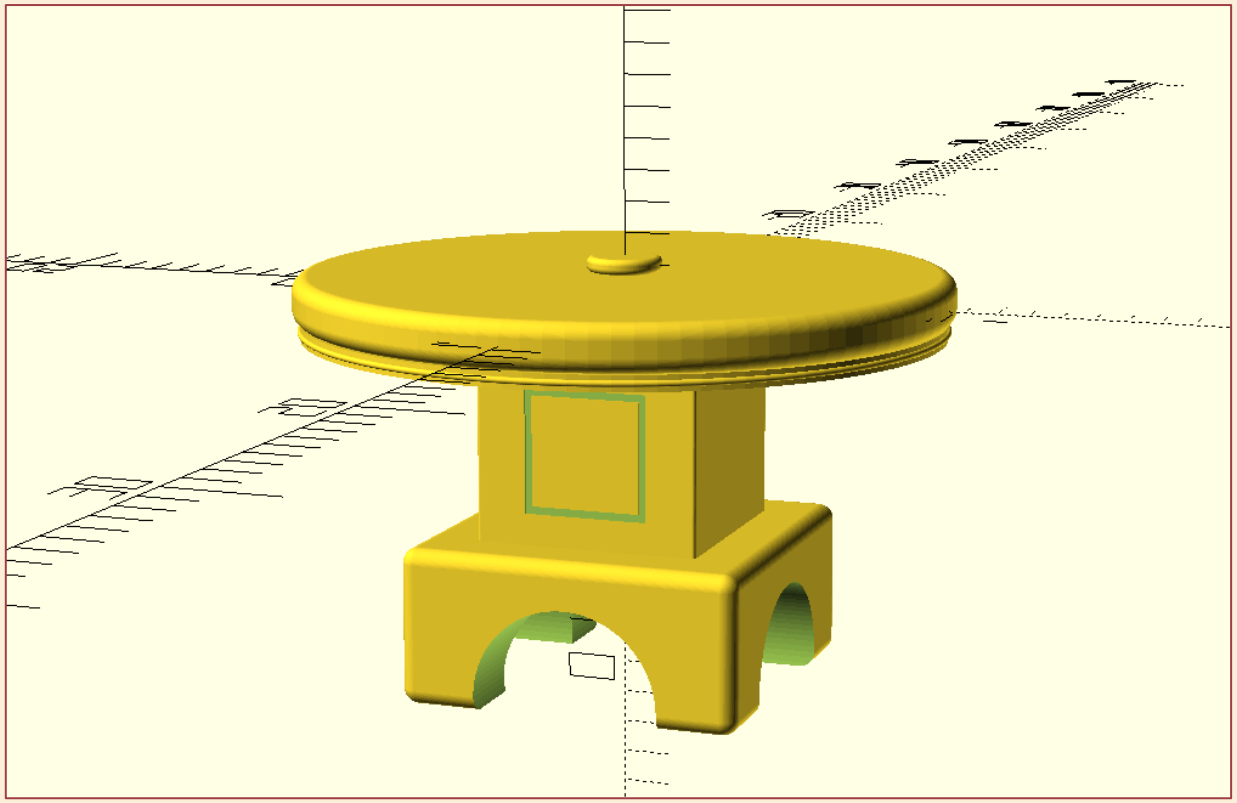




5.Use a for loop to create the ruffles. Resize it twice and difference them to get the pattern above.

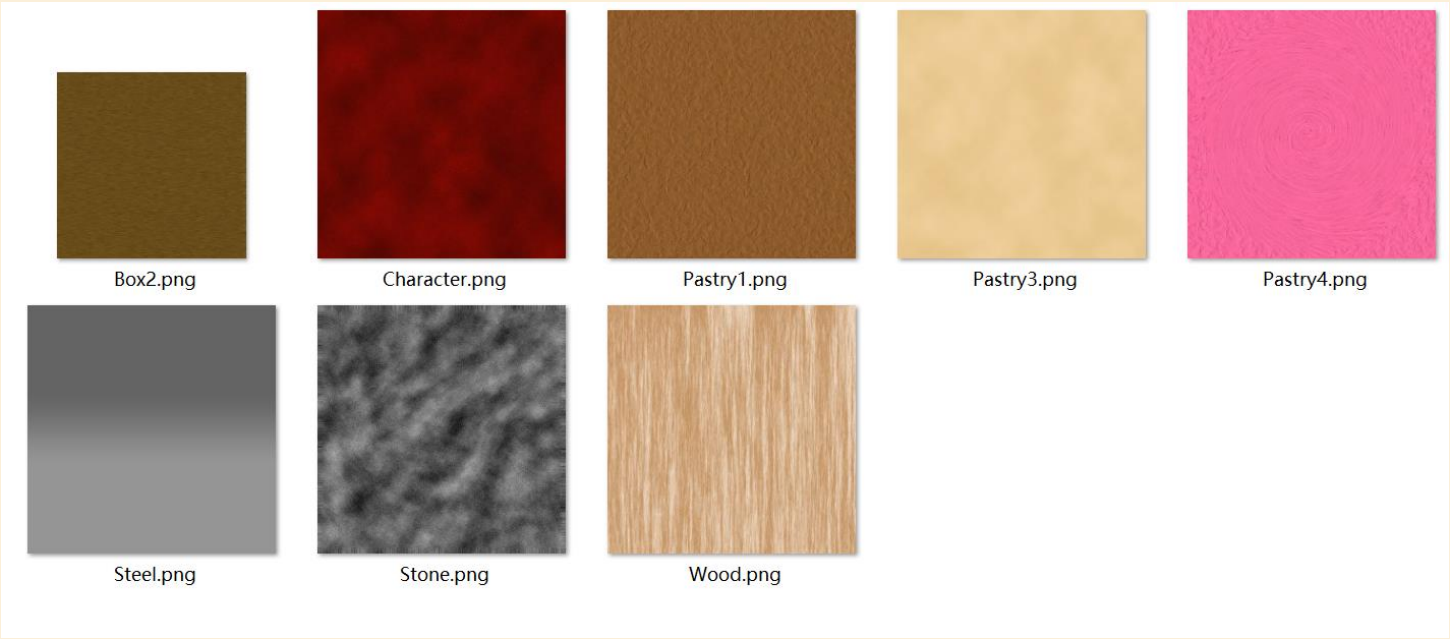
6.I find how to create English characters easily but hey, Chinese pastry will be better with Chinese character. So I make some cylinders and form a character using them.

MODELING PROCESS



7.Create a simple table. Again I love minkowski.

RENDERING



8.Make some textures using Photoshop.

RENDERING

```html
<!DOCTYPE html>
<html lang = "en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>demo</title>
        <style>
            body{
                margin: 0;
            }
        </style>
        <script type="importmap">
            {
                "imports": {
                    "three": "https://cdn.jsdelivr.net/npm/three@0.174.0/build/three.module.js",
                    "jsm/": "https://cdn.jsdelivr.net/npm/three@0.174.0/examples/jsm/"
                }
            }
        </script>
        <script src="https://cdn.jsdelivr.net/npm/gsap@3.11.1/dist/gsap.min.js"></script>
    </head>
    <body>
        <script type="module" src="index.js"></script>
    </body>
</html>
```

9.Write a simple HTML, adding a import map for Three.js library and Three.js example modules. Also import GSAP and load index.js as a JavaScript module.

RENDERING

```
import * as THREE from "three";
import { STLLoader } from "jsm/loaders/STLLoader.js";
import { OrbitControls } from "jsm/controls/OrbitControls.js";

const scene = new THREE.Scene();

/*Renderer*/
const w = window.innerWidth;
const h = window.innerHeight;
const renderer = new THREE.WebGLRenderer({ antialias: true });
renderer.setSize(w, h);
document.body.appendChild(renderer.domElement);

/*Camera*/
const fov = 75;
const aspect = w / h;
const near = 0.1;
const far = 100;
const camera = new THREE.PerspectiveCamera(fov, aspect, near, far);
camera.position.set(0, 0, 5);

/*Light*/
const ambientLight = new THREE.AmbientLight(0xffffff, 0.4);
scene.add(ambientLight);
const dirlight = new THREE.DirectionalLight(0xffffff, 1);
dirlight.position.set(0, 20, 0);
scene.add(dirlight);
```

10.Start writing the JavaScript module. Create scene, renderer, camera, and light.

RENDERING

```
/*Texture*/
const textureloader = new THREE.TextureLoader();
    const Rind = textureloader.load('Texture/LZX/Rind.jpg');
    const Kernel = textureloader.load('Texture/LZX/Kernel.jpg');
    const Box2 = textureloader.load('Texture/SYZ/Box2.png');
    const Character = textureloader.load('Texture/SYZ/Character.png');
    const Pastry1 = textureloader.load('Texture/SYZ/Pastry1.png');
    const Box1 = textureloader.load('Texture/SYZ/Stone.png');
    const Pastry3 = textureloader.load('Texture/SYZ/Pastry3.png');
    const Box3 = textureloader.load('Texture/SYZ/Steel.png');
    const Pastry4 = textureloader.load('Texture/SYZ/Pastry4.png');
    const Wood = textureloader.load('Texture/SYZ/Wood.png');

/*Background*/
textureloader.load('Texture/LZX/Background.png', function(texture){
    scene.background = texture;
})

/*UV*/
function CylinderUV(geometry) {
    geometry.computeBoundingBox();
    const center = geometry.boundingBox.getCenter(new THREE.Vector3());
    const size = geometry.boundingBox.getSize(new THREE.Vector3());

    geometry.setAttribute('uv', new THREE.BufferAttribute(new Float32Array(geometry.attributes.position.count * 2), 2));

    for (let i = 0; i < geometry.attributes.position.count; i++) {
      const vertex = new THREE.Vector3(
        geometry.attributes.position.getX(i),
        geometry.attributes.position.getY(i),
        geometry.attributes.position.getZ(i)
      );

      vertex.sub(center).divide(size);

      const u = 0.5 + Math.atan2(vertex.z, vertex.x) / (2 * Math.PI);
      const v = (vertex.y + 0.5);

      geometry.attributes.uv.setXY(i, u, v);
    }
```

11.Load those texture I created and LZX provided. Make a background. Use two functions to find uv of the models.

RENDERING

```
function SphereUV(geometry){
    geometry.computeBoundingBox();
    const bboxSize = geometry.boundingBox.getSize(new THREE.Vector3());

    geometry.setAttribute('uv', new THREE.BufferAttribute(new Float32Array(geometry.attributes.position.count * 2), 2));

    for (let i = 0; i < geometry.attributes.position.count; i++) {
        const x = geometry.attributes.position.getX(i);
        const y = geometry.attributes.position.getY(i);
        const z = geometry.attributes.position.getZ(i);

        const absX = Math.abs(x / bboxSize.x);
        const absY = Math.abs(y / bboxSize.y);
        const absZ = Math.abs(z / bboxSize.z);

        if (absX >= absY && absX >= absZ) {
        geometry.attributes.uv.setXY(i, (z - geometry.boundingBox.min.z) / bboxSize.z,
                                        (y - geometry.boundingBox.min.y) / bboxSize.y);
        } else if (absY >= absX && absY >= absZ) {
        geometry.attributes.uv.setXY(i, (x - geometry.boundingBox.min.x) / bboxSize.x,
                                        (z - geometry.boundingBox.min.z) / bboxSize.z);
        } else {
        geometry.attributes.uv.setXY(i, (x - geometry.boundingBox.min.x) / bboxSize.x,
                                        (y - geometry.boundingBox.min.y) / bboxSize.y);
        }
    }
}
```

11.Load those texture I created and LZX provided. Make a background. Use two functions to find uv of the models.

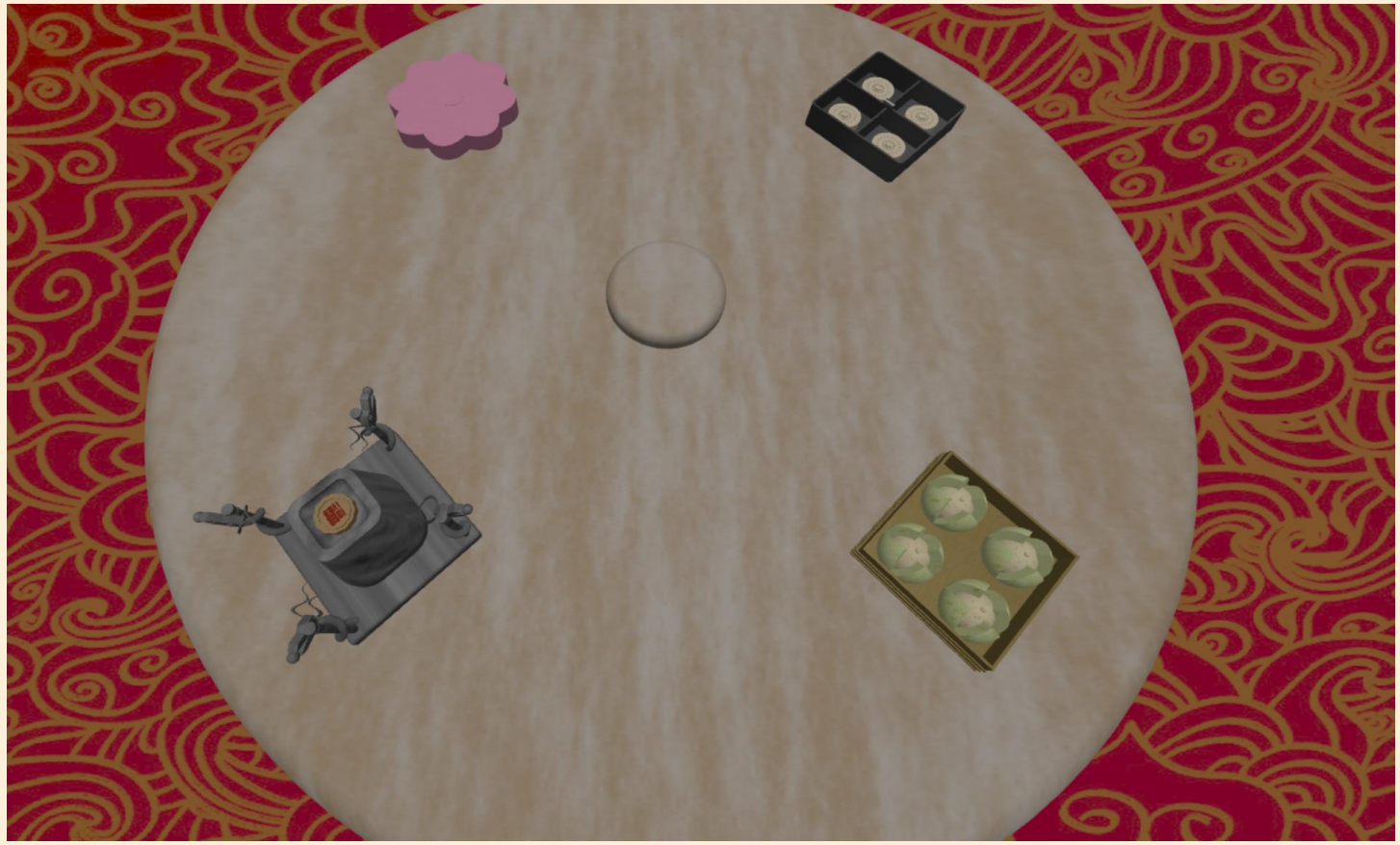
RENDERING

```
/*Models*/
const loader = new STLLoader();
const PastryS = new THREE.Group();
const PastryL = new THREE.Group();
const PastryO = new THREE.Group();
const PastryW = new THREE.Group();
const Table = new THREE.Group();
    /*SYZ*/
        /*Character*/
        loader.load('Model/SYZ/Character.stl', function(geometry){
            CylinderUV(geometry);

            const material = new THREE.MeshStandardMaterial({
                map: Character,
                roughness: 0.5,
                metalness: 0.7,
            });
            const mesh = new THREE.Mesh(geometry, material);
            mesh.scale.set(0.1, 0.1, 0.1);

            mesh.rotation.x -= 1.5;
            mesh.rotation.z += 0.785;

            PastryS.add(mesh);
        });

        /*Pastry1*/
        loader.load('Model/SYZ/Pastry1.stl', function(geometry){
            CylinderUV(geometry);

            const material = new THREE.MeshStandardMaterial({
                map: Pastry1,
                roughness: 0.8,
                metalness: 0.1,
            });
            const mesh = new THREE.Mesh(geometry, material);
            mesh.scale.set(0.1, 0.1, 0.1);

            mesh.rotation.x -= 1.5;

            PastryS.add(mesh);
```

```
        /*Box1*/
        loader.load('Model/SYZ/Box1.stl', function(geometry){
            SphereUV(geometry);

            const material = new THREE.MeshStandardMaterial({
                map: Box1,
                roughness: 0.8,
                metalness: 0.6,
            });
            const mesh = new THREE.Mesh(geometry, material);
            mesh.scale.set(0.1, 0.1, 0.1);

            mesh.rotation.x -= 1.5;
            mesh.rotation.z -= 2.355;

            PastryS.add(mesh);
            PastryS.position.set(-4, -1.1, -1);
            scene.add(PastryS);
        });

        /*Table*/
        loader.load('Model/SYZ/Table.stl', function(geometry){
            CylinderUV(geometry);

            const material = new THREE.MeshStandardMaterial({
                map: Wood,
                roughness: 0.8,
                metalness: 0.5,
            });
            const mesh = new THREE.Mesh(geometry, material);
            mesh.scale.set(0.8, 0.8, 0.8);

            mesh.rotation.x -= 1.5;
            mesh.rotation.z -= 3.14;

            Table.add(mesh);
            Table.position.set(0, -2, -5);
            scene.add(Table);
        });
```

12.Load all the models, and set their attributes. Divide them into groups to use later. Additionally add groups into the scene.

RENDERING



RENDERING

```
/*Control*/
const controls = new OrbitControls(camera, renderer.domElement);
controls.enableDamping = true;
controls.dampingFactor = 0.25;
controls.screenSpacePanning = true;

/*Choose Pastry*/
const raycaster = new THREE.Raycaster();
const mouse = new THREE.Vector2();
const targetPosition = new THREE.Vector3(0, 0, 0);

window.addEventListener('click', onMouseClick, false);
window.addEventListener('keydown', function(event){
    if(event.key ==='Escape'){
        resetScene();
        window.addEventListener('click', onMouseClick, false);
    }
})

/*Rotate*/
let isRotating = false;
let rotationTween = null;

window.addEventListener('keydown', function(event){
    if(event.key === ' '){
        if (!isRotating) {
            rotateScene();
        } else {
            stopRotation();
        }
    }
});
```

13.Make four animations — — mouse control, click group to choose and focus on it, press esc to reset scene, press space to start and stop rotating. Set the render loop.

RENDERING

```
function rotateScene() {
    isRotating = true;
    rotationTween = gsap.to(scene.rotation, {
        y: "+=" + Math.PI * 20,
        duration: 5,
        repeat: -1,
        ease: "linear"
    });
}

function stopRotation() {
    isRotating = false;
    if (rotationTween) {
        rotationTween.kill();
        rotationTween = null;
    }
}

function onMouseClick(event){
    mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
    mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;
    raycaster.setFromCamera(mouse, camera);

    const intersects = raycaster.intersectObjects(scene.children, true);

    if(intersects.length > 0){
        const selectedObject = intersects[0].object;
        const parentGroup = findParentGroup(selectedObject);

        if(parentGroup){
            focusOnGroup(parentGroup);
            window.removeEventListener('click', onMouseClick);
        }
    }
}
```

13.Make four animations — — mouse control, click group to choose and focus on it, press esc to reset scene, press space to start and stop rotating. Set the render loop.

RENDERING

```
function findParentGroup(object){
    let current = object;
    while(current.parent){
        if(current.parent instanceof THREE.Group){
            return current.parent;
        }
        current = current.parent;
    }
}

function focusOnGroup(group){
    if (!group) return;

    [PastryS, PastryL, PastryO, PastryW, Table].forEach(g =>{
        g.visible = (g === group);
    });

    group.getWorldPosition(targetPosition);

    controls.enabled = false;
    gsap.to(camera.position, {
        x: targetPosition.x+5,
        y: targetPosition.y+5,
        z: targetPosition.z+5,
        duration: 1,
        ease: "power2.out",
        onUpdate: () => {
            controls.target.copy(targetPosition);
            controls.update();
        },
        onComplete: () => {
            controls.enabled = true;
        }
    });
}
```

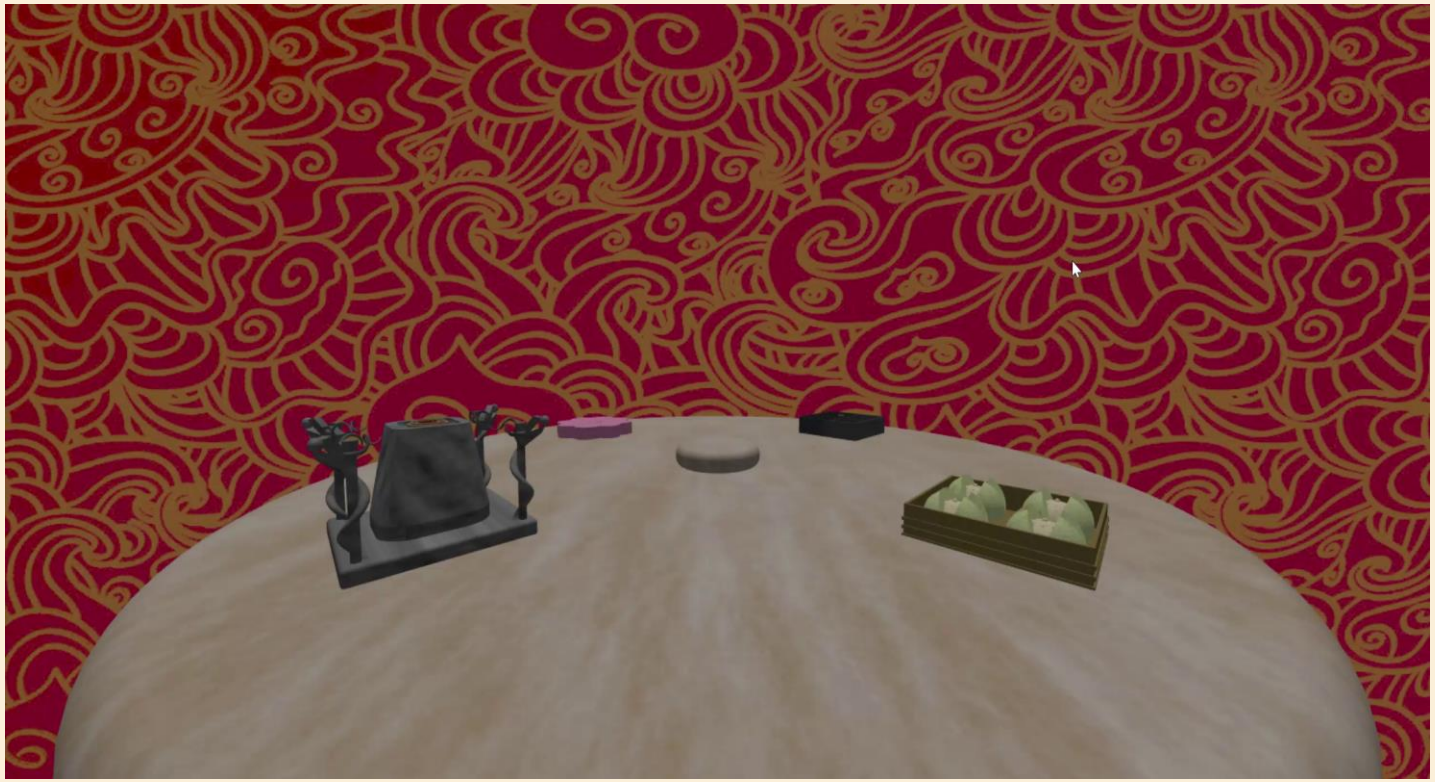
13.Make four animations — — mouse control, click group to choose and focus on it, press esc to reset scene, press space to start and stop rotating. Set the render loop.

RENDERING

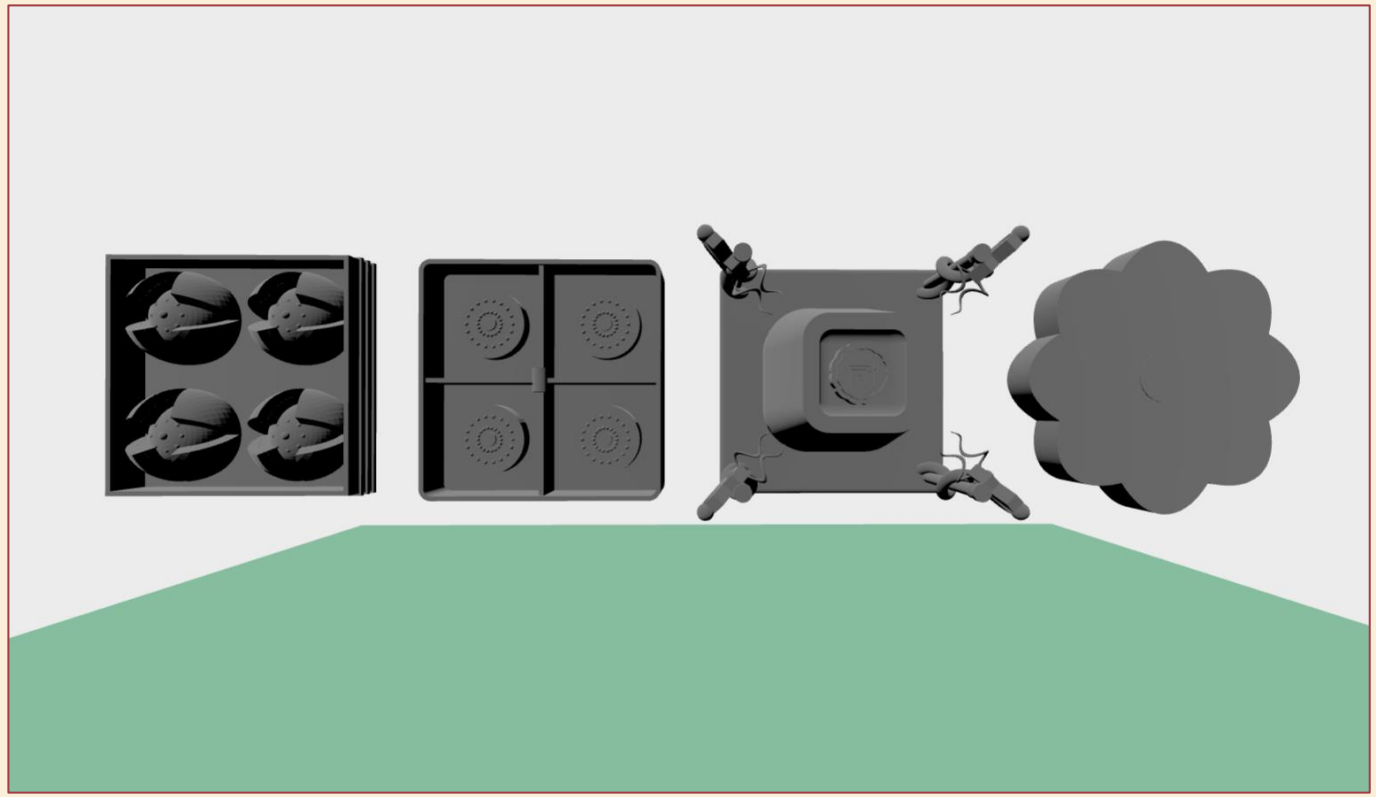
```
function resetScene(){
    [PastryS, PastryL, PastryO, PastryW, Table].forEach(model =>{
        model.visible = true;
    });

    const newTargetPosition = new THREE.Vector3(0, 0, 0);

    controls.enabled = false;
    gsap.to(camera.position, {
        x: 0,
        y: 0,
        z: 5,
        duration: 1,
        ease: "power2.out",
        onUpdate: () => {
            controls.target.copy(new THREE.Vector3(0, 0, 0));
            controls.update();
        },
        onComplete: () => {
            controls.enabled = true;
        }
    });
}

/*Animation*/
function animate(){
    requestAnimationFrame(animate);
    renderer.render(scene, camera);
    controls.update();
}

animate();
```

13.Make four animations — — mouse control, click group to choose and focus on it, press esc to reset scene, press space to start and stop rotating. Set the render loop.

ANIMATION



VR



We also import the four models into Glitch to get the VR effect (which is really fun).

RESPONSIBLE PART of PROJECT

Zhengxuan Li (Levi):

1.Create a model of Water Lily Crisp.
2. Import the model into three.js.
3. Adds a background image to a web page.
4. create model animation effect and GUI panel (may not be used in the end).
5. import the model into VR-browser: Glitch.
6. Layout the report PPT.

Weiyu Ouyang (Aiden):

1.Create a model of sweetheart pastry .
2.Be responsible for writing the final report.
3.Create a theme template for PowerPoint presentations.
4.Participate in the production of the final PPT.

RESPONSIBLE PART of PROJECT

Youyou Wu: (Brongcer):

1.Create a model of peach blossom pastry.
2.Designed the color card matching for others modeling work.
3.Participate in the production of the final PPT.
4.Be responsible for the writing of the final presentation.
5.Discuss the final presentation with the associate supervisor.

Yizhen Sun (Yizhen):

1. 2 models, mooncake with box and the table in "\Final\Model\SYZ“
2. Every ".png" textures in "\Final\Texture\SYZ“
3. Whole index.html and index.js in "\Final“
4. PPT and Word report for the above

Thank you for listening