G1_final_report

Project Background:

Pastries are familiar to everybody. People always enjoy them while watching TV, playing videogames and so on, or during teatime.

Our project focuses on generating a 3D CG of various of pastries by OpenSCAD and Three.js. Through the project, we would like to show delicious pastries via our cooperation and skills of computer graphics.

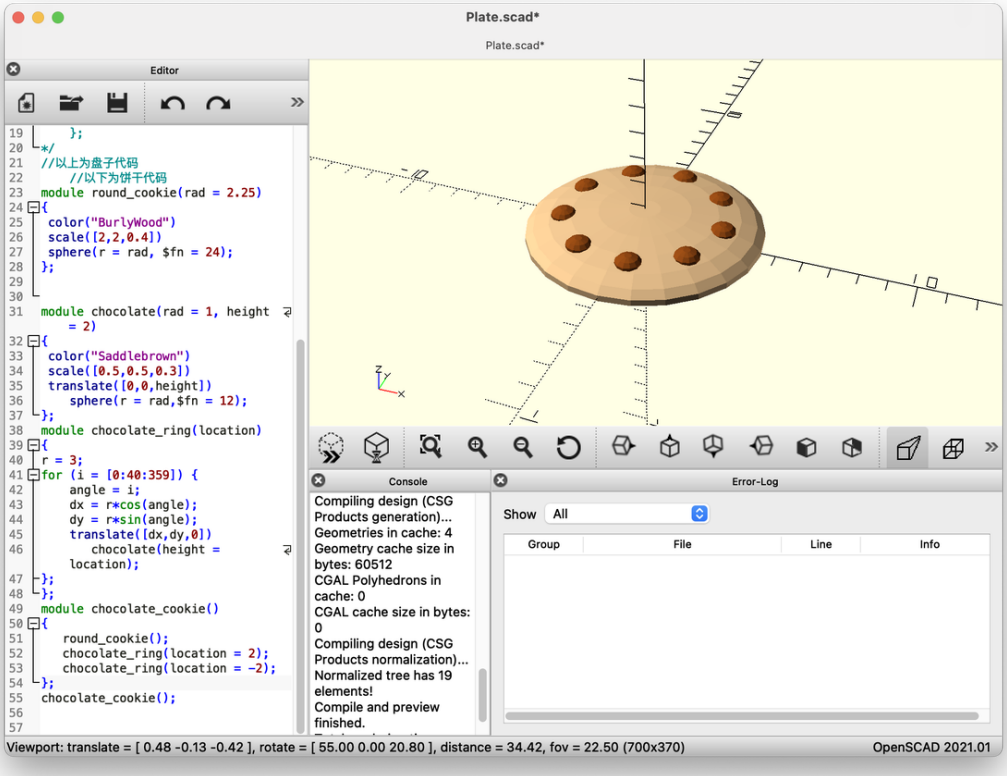
Topic Analysis:

Modeling:

We totally create 7 models, including a cookie, a bear cookie, a sliced cake, a cupcake, 2 different mooncakes and a plate.

1. Cookie

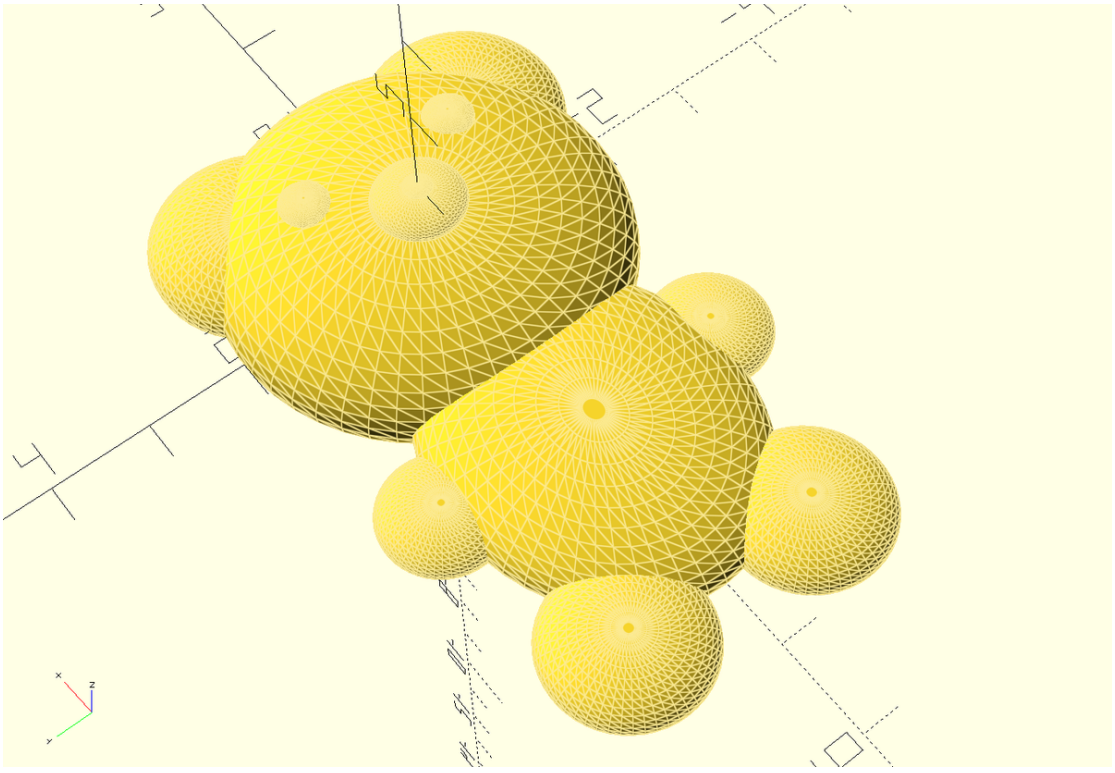
To create a basic chocolate cookie in OpenSCAD, we learn that the sphere() function can be used to create both the chocolate and the cookie base by using scale() function. We also realize that the for() loop function can be used to create chocolate rings. Here is the code. (Note: 1.Cookie shares the same file as 6.Plate)



2. Bear cookie

To make cookies more special, we try to model a different and unique shape. Bear cookies are great choices because they have shapes like bears which are more complicated than normal cookies.

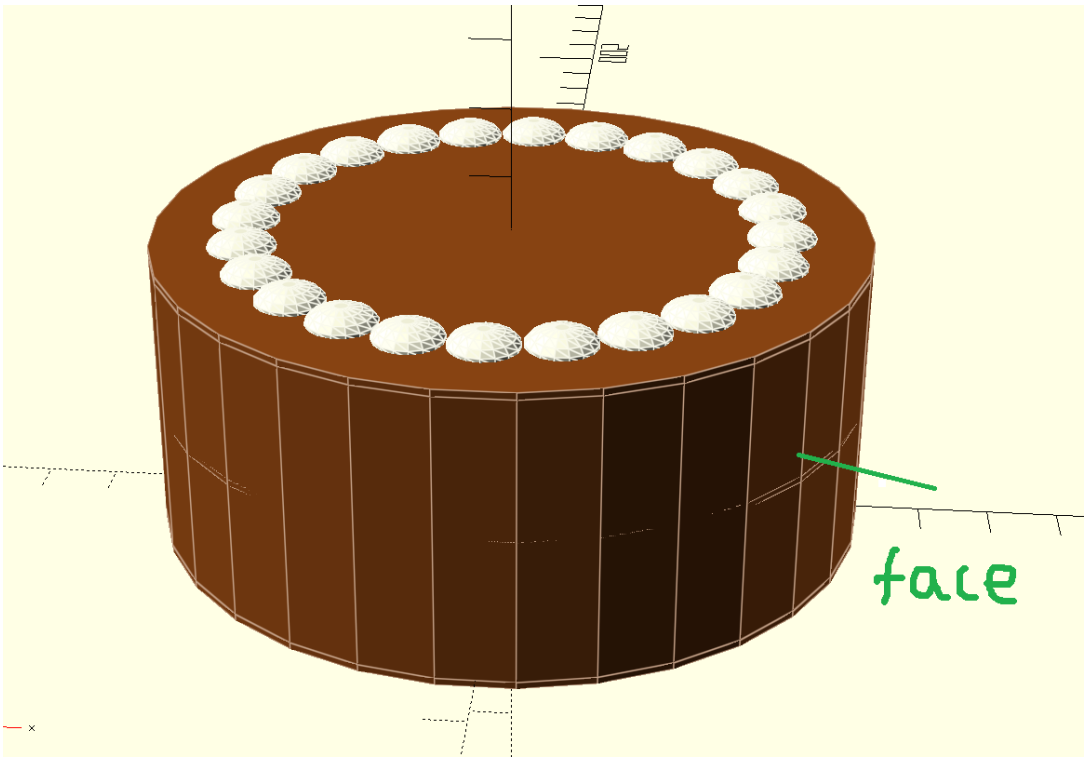
In considering how to model a bear cookie in OpenSCAD, the cookie should be separated to every single geometry. We find that bear's all parts like ears, head, body, arms and legs can be ellipsoids. Therefore, we use sphere() and scale() functions to create bear's parts, then combine them by using function union(). Here is the model.



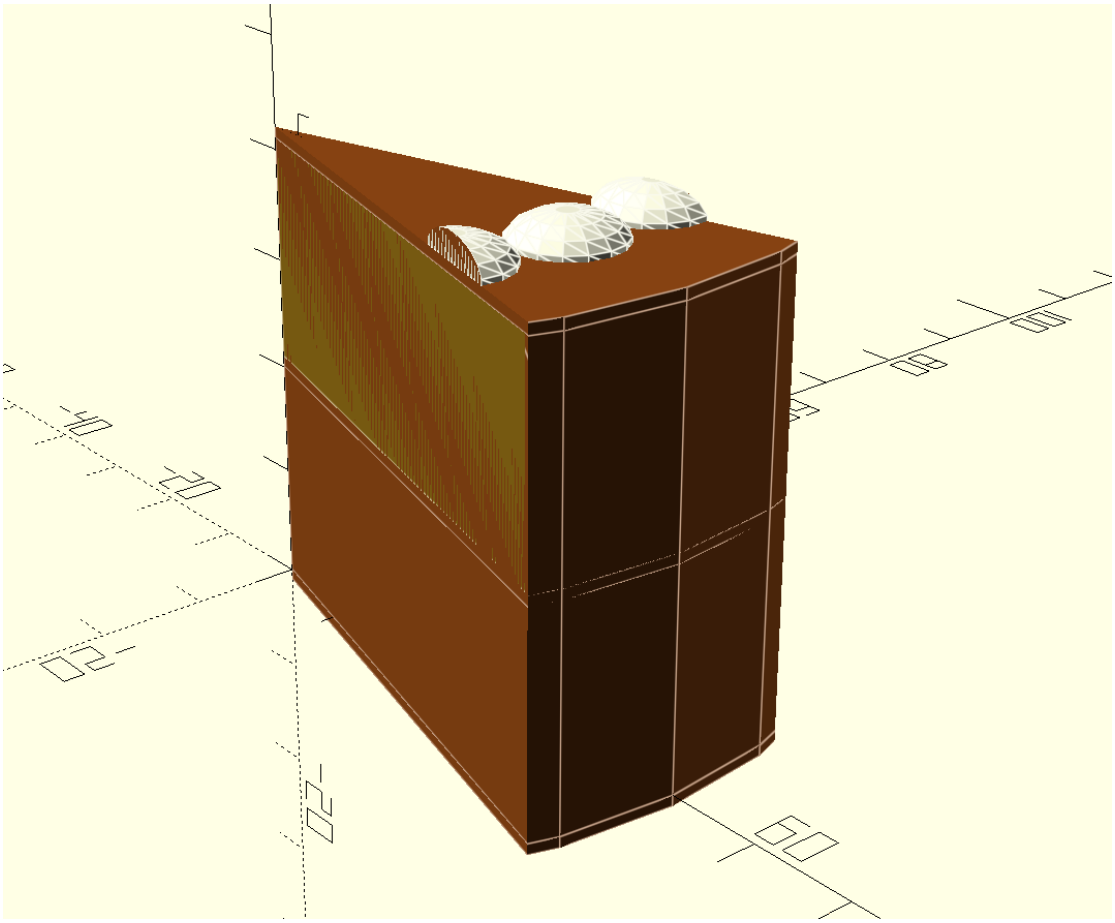
3. Sliced cake

Cake's shape looks like a cylinder, and we would like to the sliced cake. So, its basic shape and geometry are easy.

First, we choose to model the whole cake first by using cylinder() function. It is noted that the cake has many layers, but it doesn't matter, all of them can be seen as cylinders. As for its face which should be a cover like chocolate, we do difference() between two cylinder with the same height but different radius. Then, we add a cycle of creams via using for loops, sphere() and scale() function. In the end, we use union() function to combine all parts. Here is the model.



After that, we slice the cake by using intersection() function. We do intersection between the cylinder and a bigger polygon with function linear_extrude(). Here is our final model of the cake.



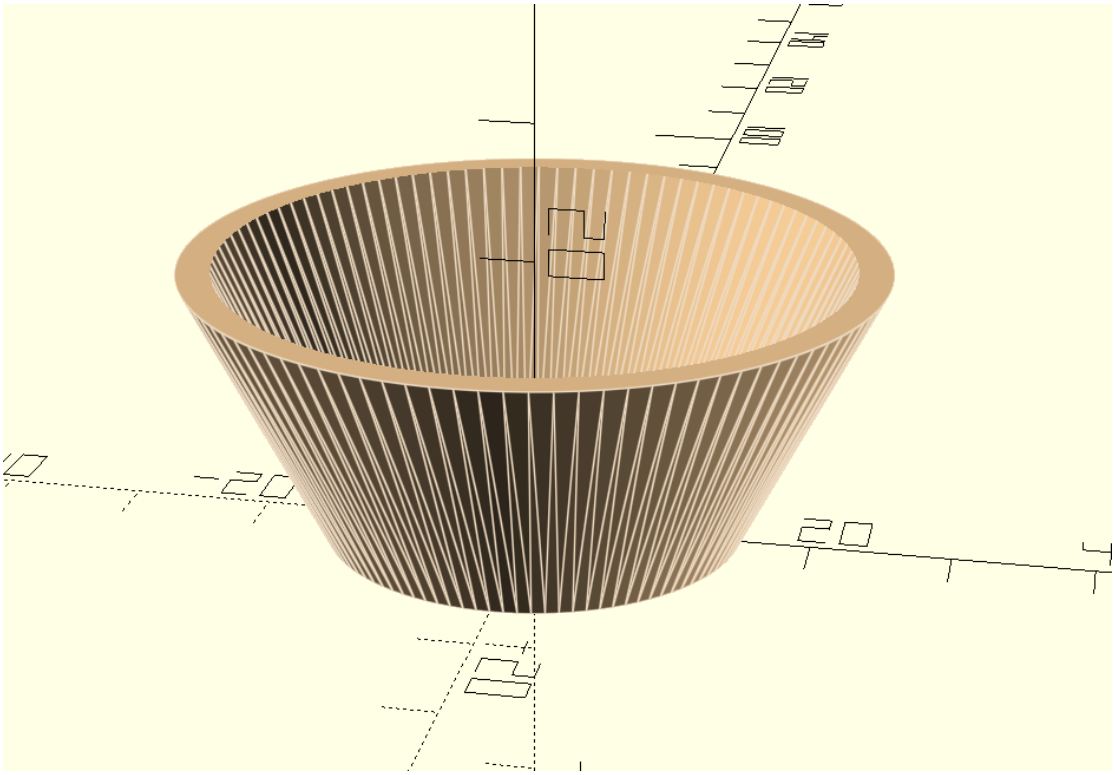
4. Cupcake

Cupcake is more challenging than sliced cake. It has many complicated parts and modeling it with OpenSCAD seems quite difficult.

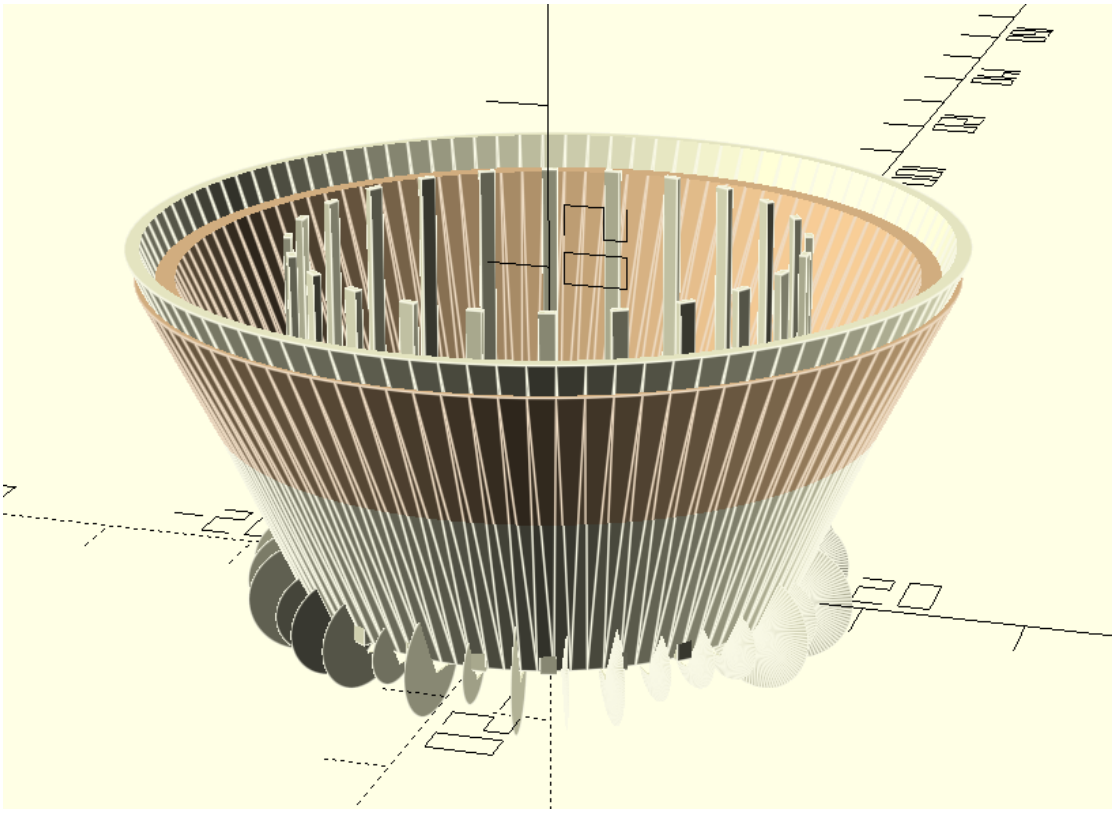
To overcome this, we divide cupcake into several parts -- base, wrapper,

frosting, cherry and sprinkles.

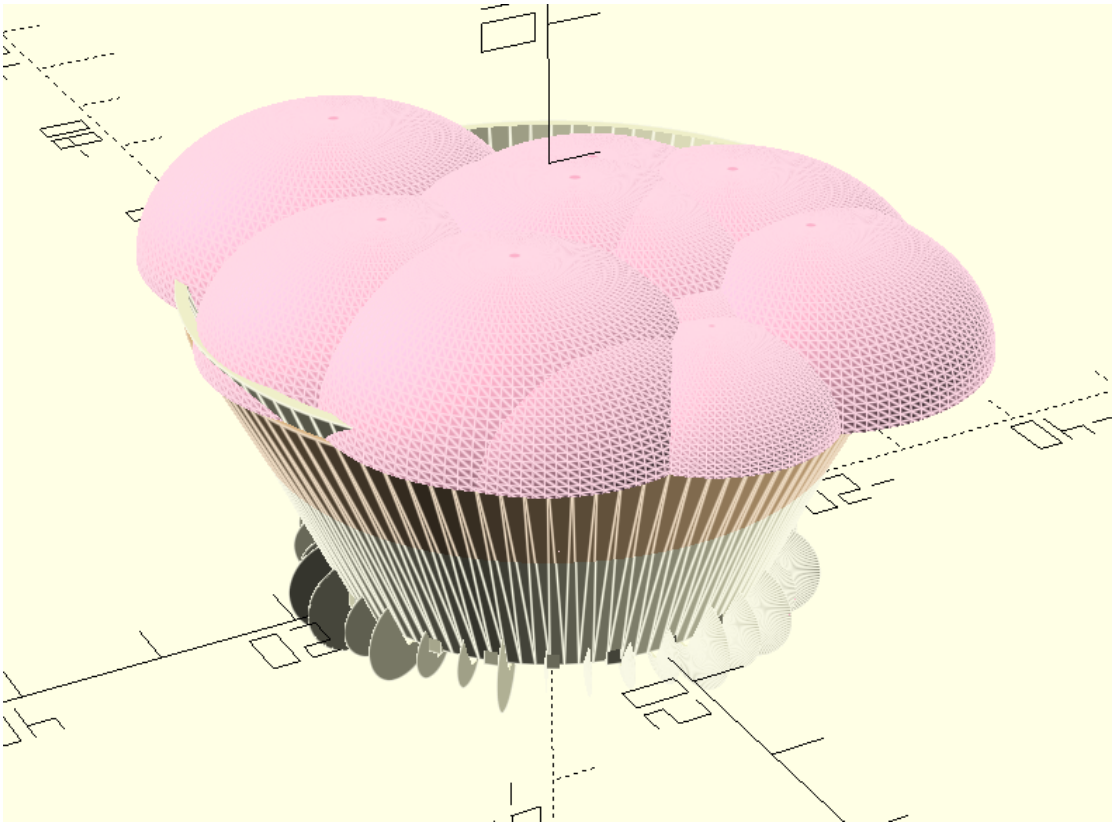
First of all, cupcake's base can be seen as a ring of a cylinder by using difference() function.



Then, the wrapper needs the same cylinder as the base to support the whole cupcake. In addition, we add some decorations by using some small cubes and cylinders, then use for loop to encircle the wrapper.

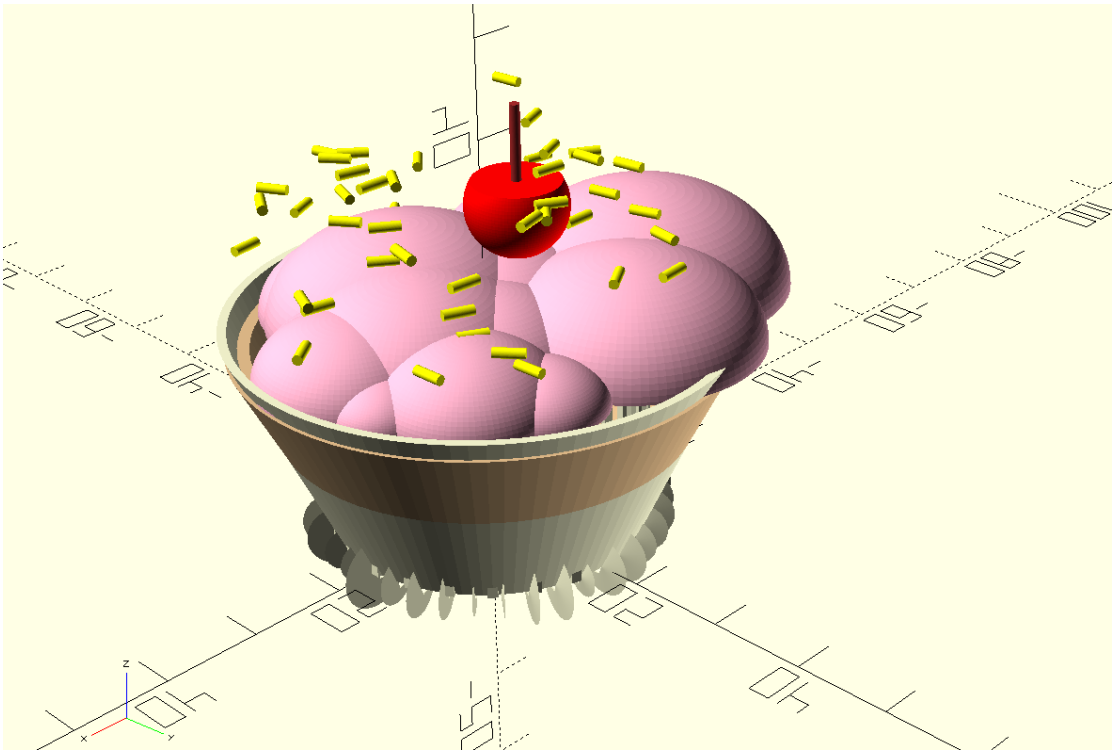


Next, we need to add cakes into the wrapper. According to sample pictures, they can be spheres, thus we use sphere() function and for loop to generate.



To make the cupcake more delicious, we add cherry and some sprinkles. The former is created by using difference() function between a sphere and a cube and combine it with a cylinder. The later are just cylinders. We make it in different angles and positions by using rands() functions and spread them by using for loop.

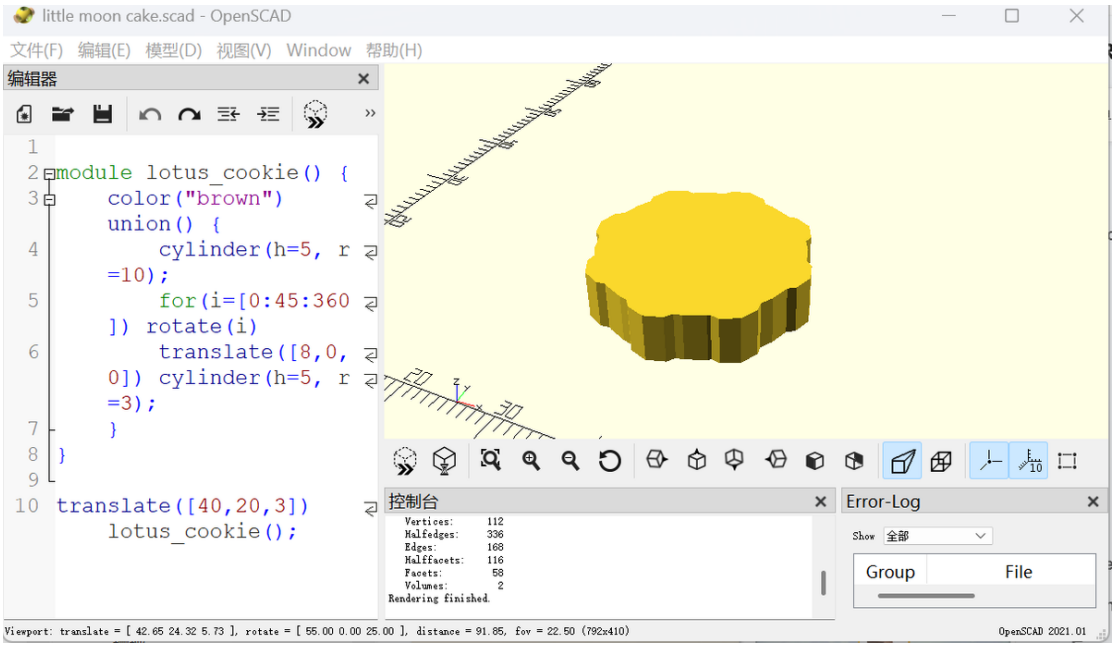
Finally, the cupcake is successfully modeled by combining all parts through union() function.



5. Simple Mooncake

When modeling a simple mooncake in OpenSCAD, the proper use of the for

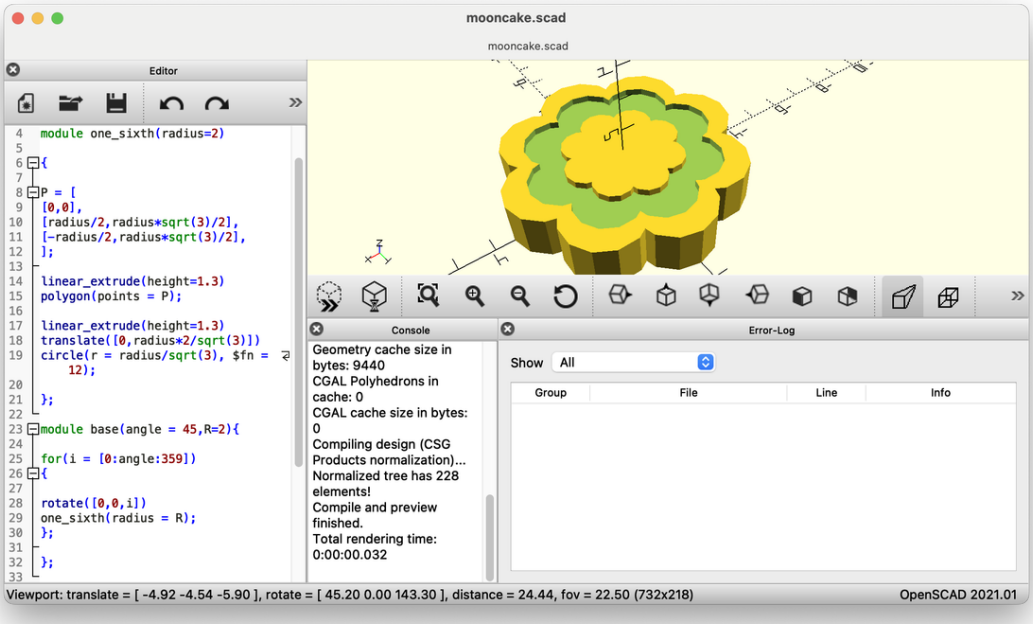
loop function can achieve a specific rotation to build the side of the mooncake. Use cylinder to build the body and then combine them with union () to get a simple mooncake. Here's model and the code.



6. Mooncake

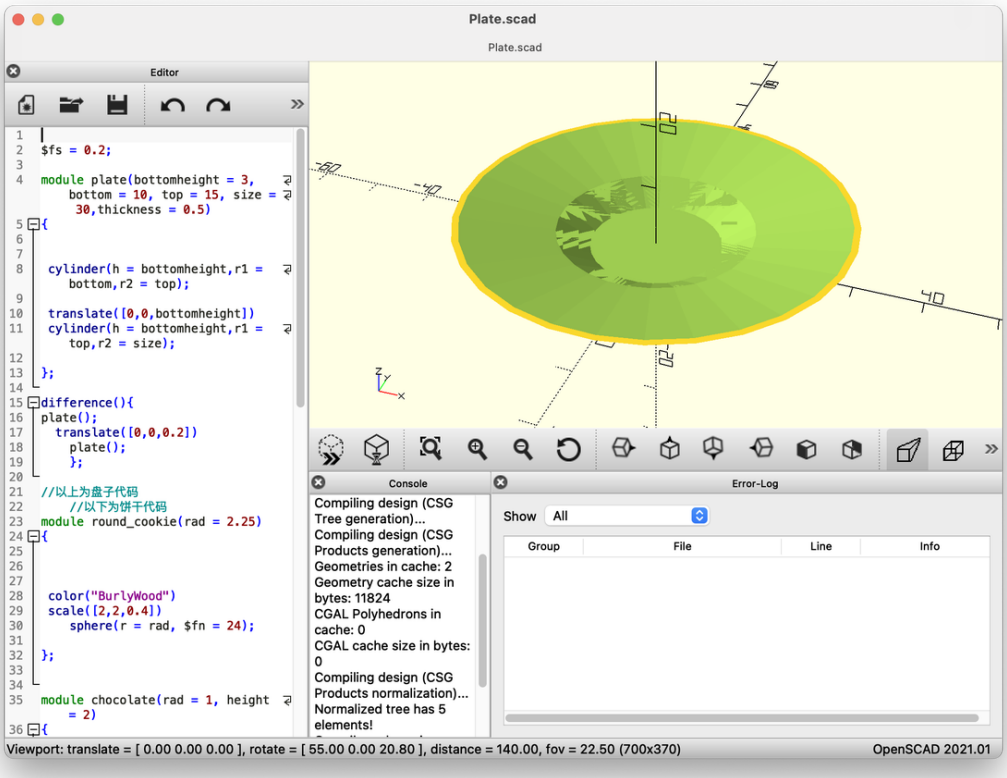
When modelling the mooncake in OpenSCAD, we find that the for() loop function can achieve repetition and rotation of a specific pattern while difference() function can model the difference of depth within the mooncake without building extra geometries.

Therefore, we firstly use polygon() and circle() function to create a pattern and give it depth using linear_extrude() function. Then we use for() loop function to complete the basic geometry of the mooncake. Here is the code.



7. Plate

To model plate in OpenSCAD, we find that the cylinder() function can be used to create trapezoidal cylinder by modifying radius of the top and the bottom circle. Then we can use difference() function to create a thin plate. Here is the code. (Note: 1. Cookie shares the same file as 6. Plate)



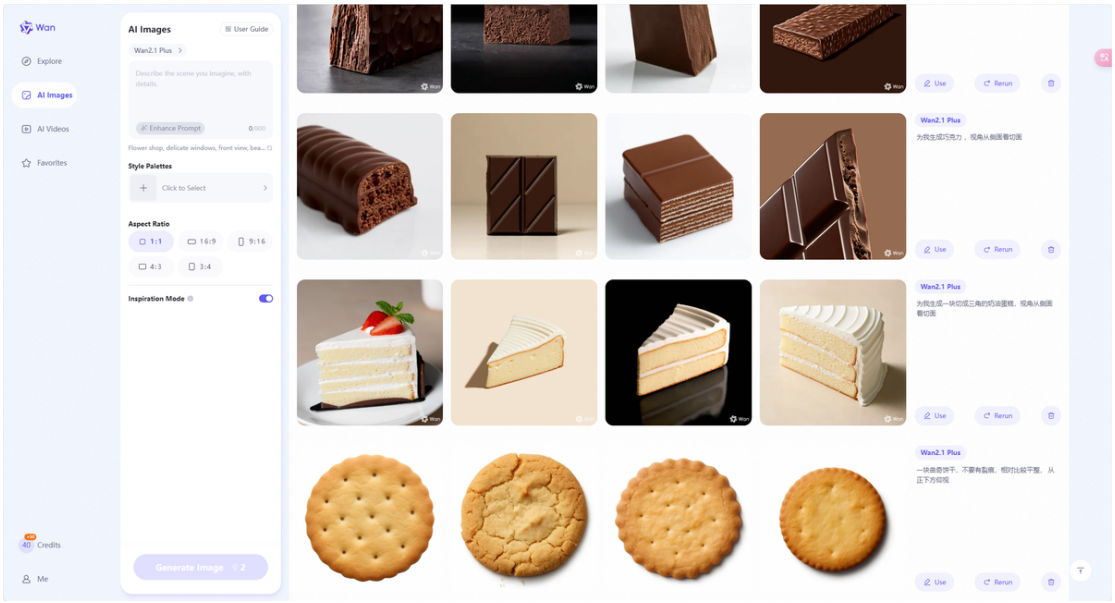
Texture:

1. Tools : Photoshop , Wan AI

Wan AI is used in order to get high-definition texture map raw material. Then use Photoshop to extract, transform and process. In order to make the model more realistic, we need to make a seamless texture

2. Wan AI

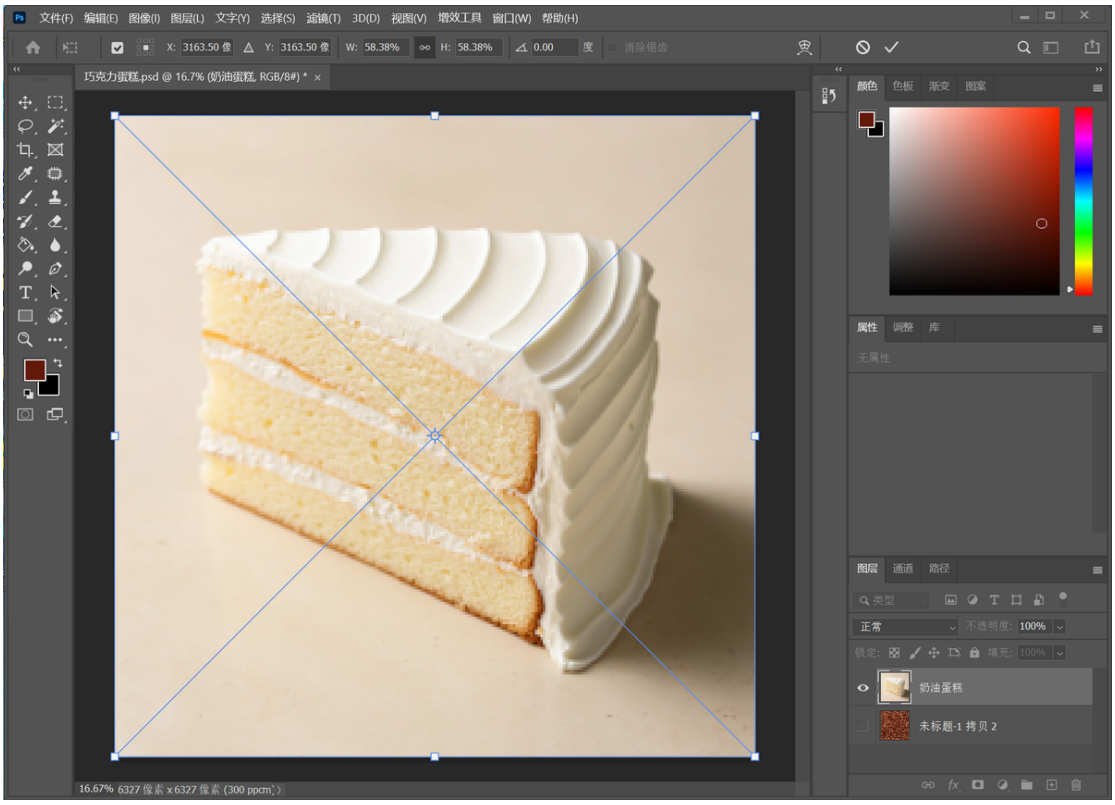
First, use reasonable prompts to tell the AI to generate high-definition food images for me. According to the modeling situation of students in the group, create different pictures to extract different texture maps



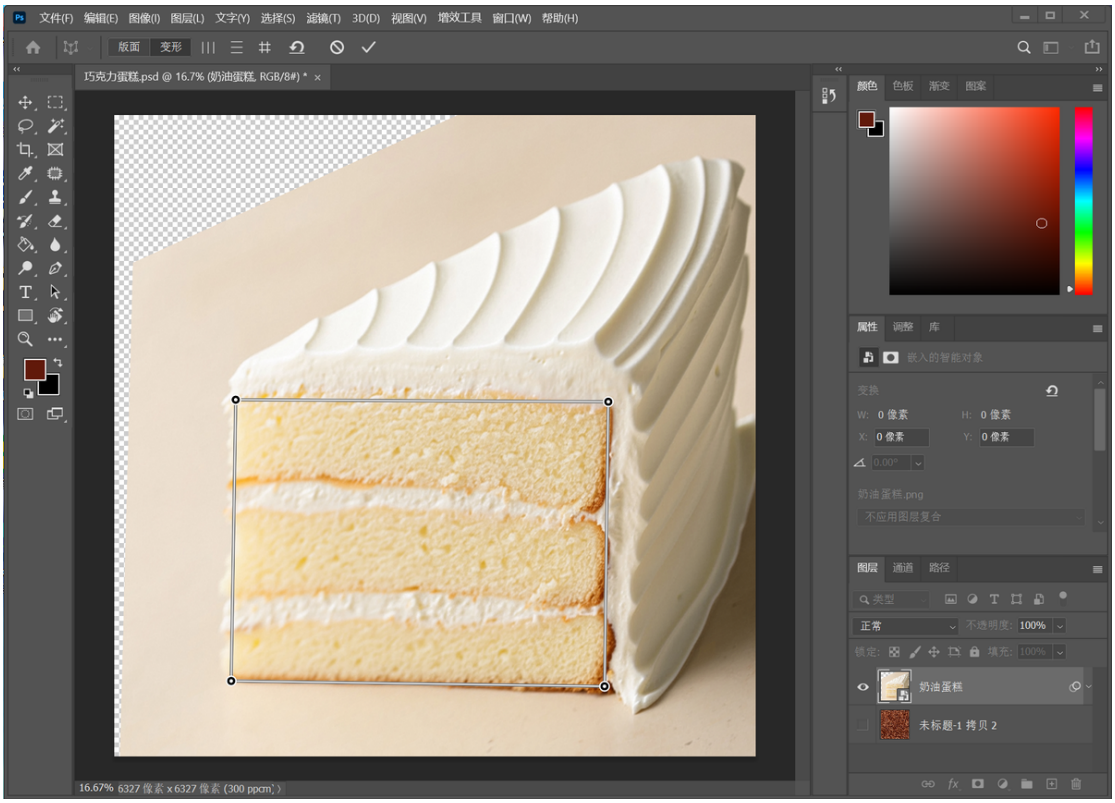
3. Photoshop

- **Prelliminary work**

In Photoshop, open the image, adjust the appropriate canvas size.



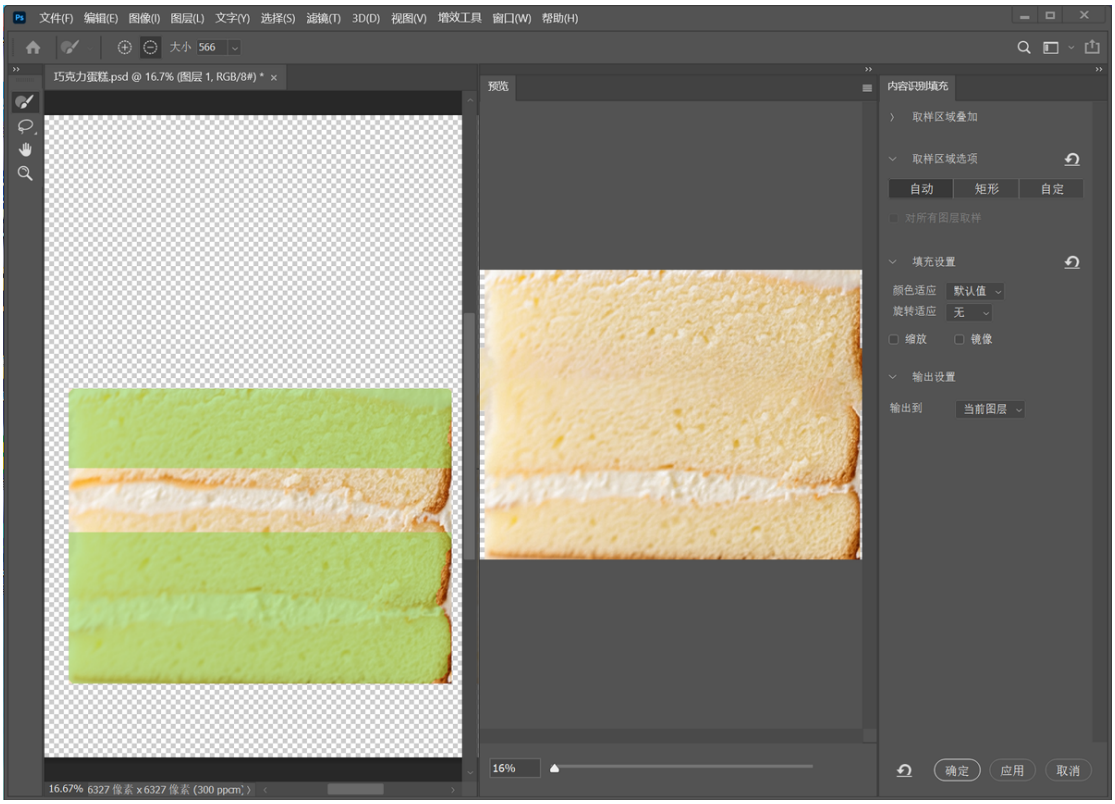
Use the Perspective Deformation tool to correct the contents of the box selection.



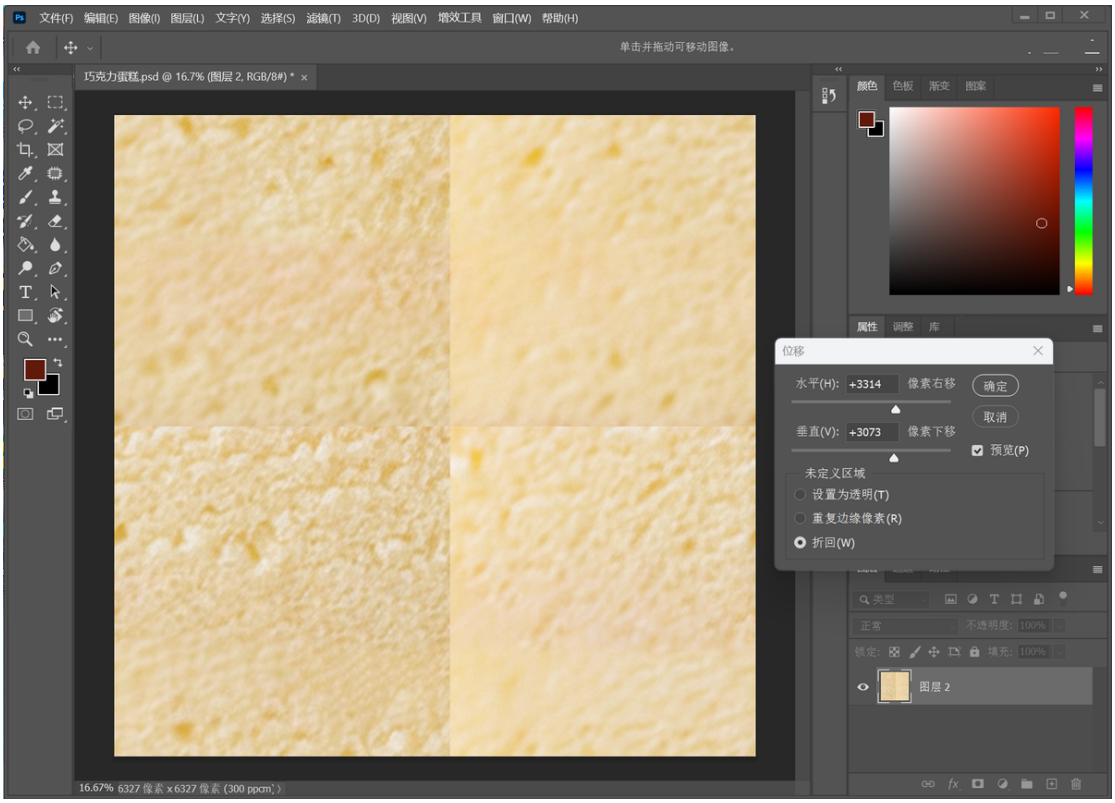
Cut the desired contents onto the new layer. Then the preliminary work is finished.

- **Seamless pattern**

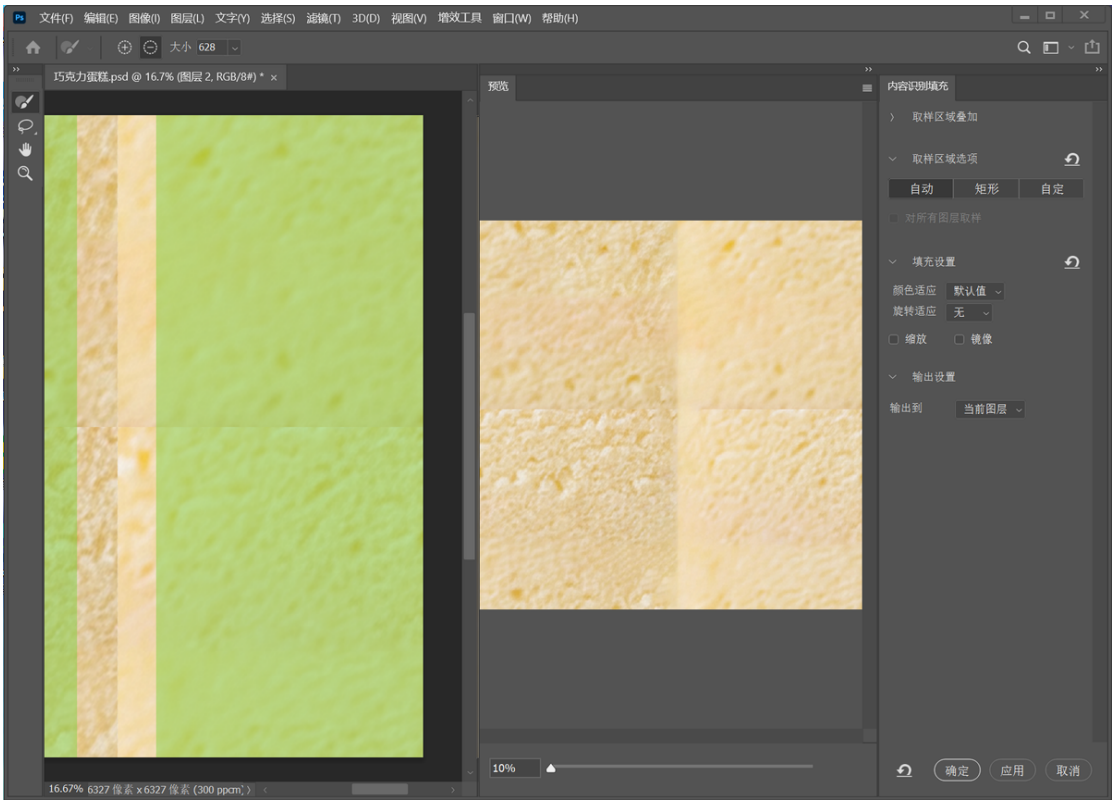
Use the Content-Aware Fill tool to process the image and make the pattern more uniform. Repeat this step to handle the entire image.



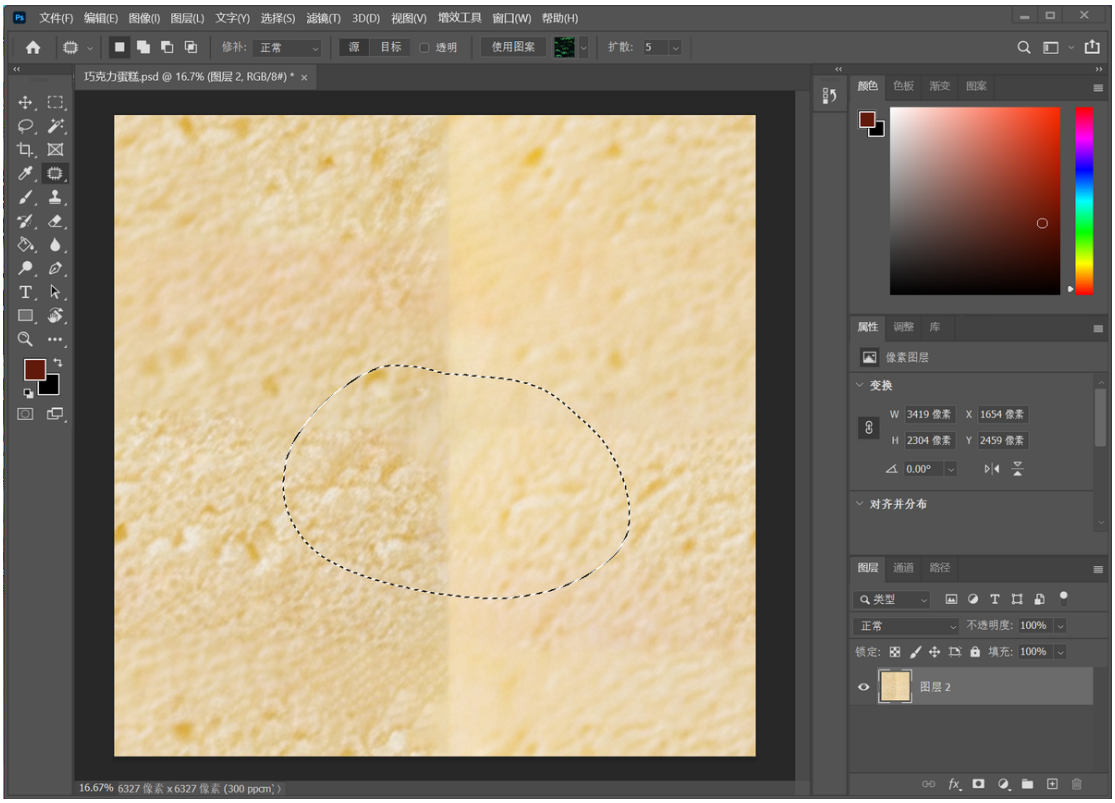
Crop a square section to fill the canvas. Apply the Offset filter, setting the X and Y offsets to half the canvas size.



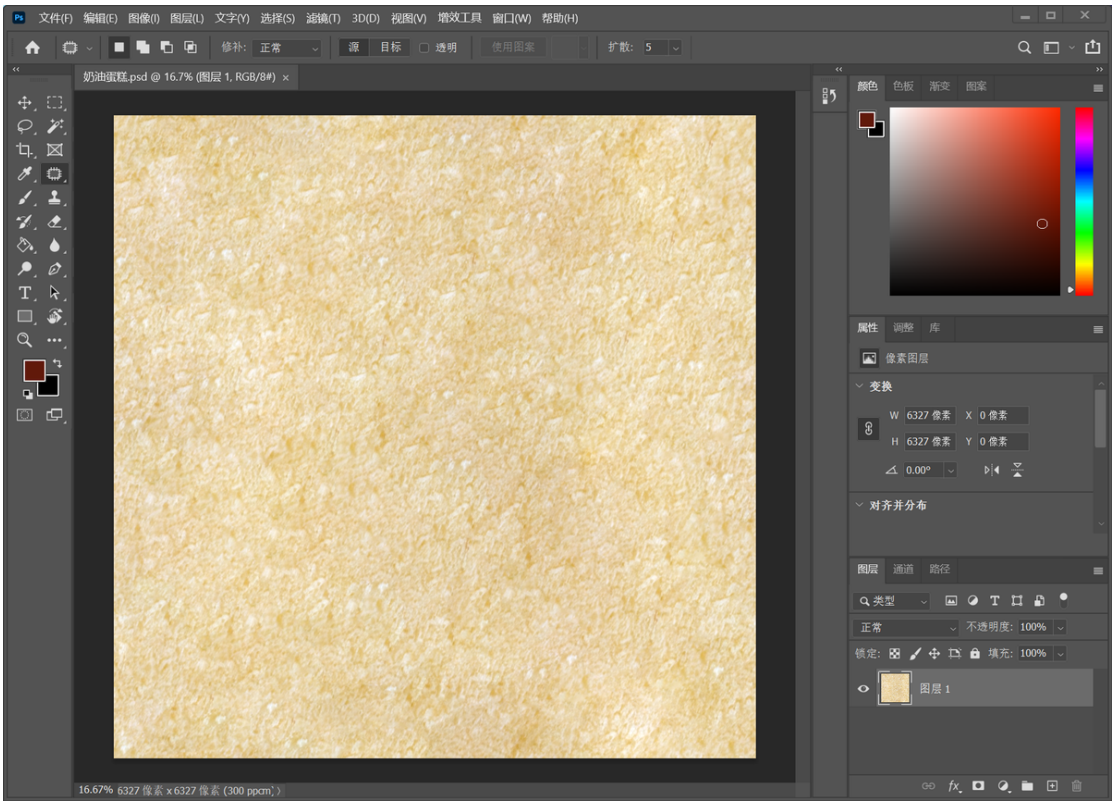
Use Content-Aware Fill again to fix the cross-shaped seam in the middle.



Use the Patch tool repeatedly to repair small flaws.



Finally, you get a seamless map.

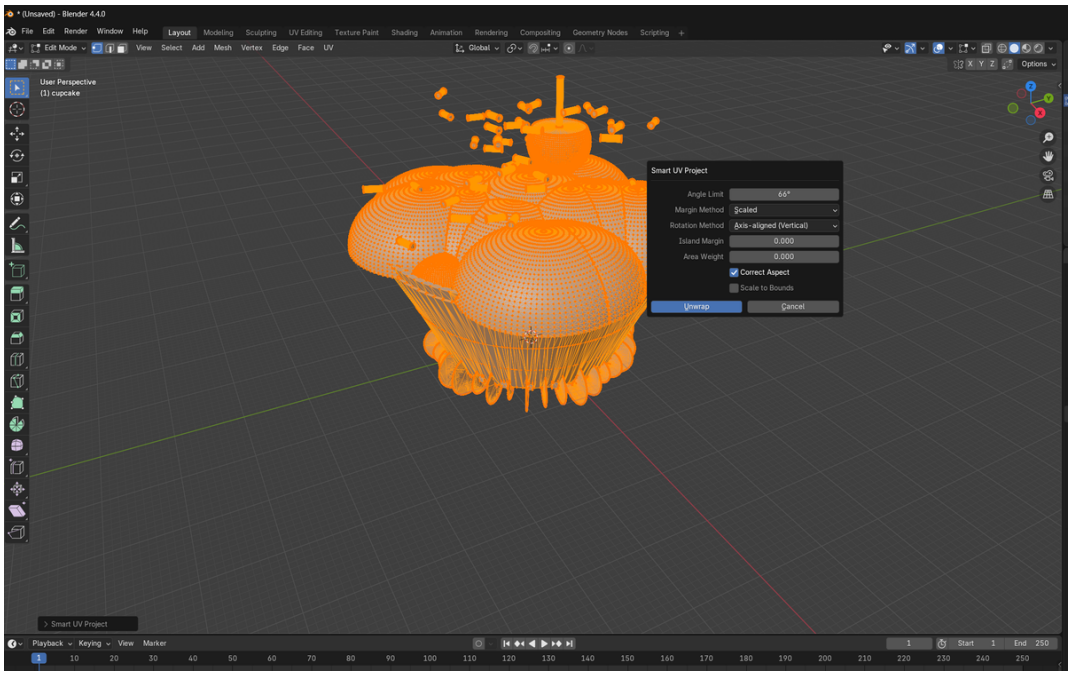


The following texture map can be obtained by repeating the operation on multiple images.

Rendering:

1. UV Wrapping in Blender

We import stl file into blender and create UV wrapping, then export as obj file for later animation process.



2. The initial setting: light, renderer and scene.

```
CGViteProject > src > JS main.js > ...
import './style.css'
import * as THREE from 'three';
import { OrbitControls } from 'three/examples/jsm/controls/OrbitControls';
import { OBJLoader } from 'three/examples/jsm/loaders/OBJLoader';
const scene = new THREE.Scene();
const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);
const renderer = new THREE.WebGLRenderer({
  canvas: document.querySelector('#bg'),
});

renderer.setPixelRatio(window.devicePixelRatio);
renderer.setSize(window.innerWidth, window.innerHeight);
camera.position.set(20, 30, 10);

// Add Lighting
const pointLight = new THREE.PointLight(0xffffff);
pointLight.position.set(0, 10, 10);
scene.add(pointLight);

const ambientLight = new THREE.AmbientLight(0xffffff);
scene.add(ambientLight);

const lightHelper = new THREE.PointLightHelper(pointLight);
const gridHelper = new THREE.GridHelper(200, 50);
scene.add(gridHelper);
scene.add(lightHelper);

// Add controls
const controls = new OrbitControls(camera, renderer.domElement);

// Load a background image
const myTexture = new THREE.TextureLoader().load('src/textures/colorful.jpg');
scene.background = myTexture;
```

3. The plate is being rendered. The texture, position and size of the plate is being set.

```
// Add a plate from obj in the middle of the scene
function loadPlateModel(url, position, scale) {
  const loader = new OBJLoader();
  loader.load(url, function (object) {
    const texture = new THREE.TextureLoader().load('src/textures/plate.png');
    object.traverse(function (child) {
      if (child instanceof THREE.Mesh) {
        child.material = new THREE.MeshStandardMaterial({ map: texture });
      }
    });
    object.position.copy(position);
    object.scale.set(scale.x, scale.y, scale.z);
    object.rotation.set(Math.PI / 1000, 0, 0);
    scene.add(object);
  });
}

// Adjust the plate position
loadPlateModel('src/models/plate.obj', new THREE.Vector3(0, -0.5, 0), new THREE.Vector3(0.5, 0.5, 0.5));
```

4. Render models into the place. For convenience, all the models are being placed into the "pastryModels" list. Then, each model is rendered with a texture.

```
// Adjust the plate position
loadPlateModel('src/models/plate.obj', new THREE.Vector3(0, -0.5, 0), new THREE.Vector3(0.5, 0.5, 0.5));

// Variables for pastries
const pastries = [];
const fallSpeed = 0.01;
const bounceFactor = 0.7;
const plateY = -0.5;
const pastryCount = 5; // Number of each type of pastry
const gravity = -0.001;

const pastryModels = [
  { name: 'Cookie', scale: new THREE.Vector3(0.2, 0.2, 0.2) },
  { name: 'bearCookie', scale: new THREE.Vector3(0.4, 0.4, 0.4) },
  { name: 'cake', scale: new THREE.Vector3(0.1, 0.1, 0.1) },
  { name: 'cupcake', scale: new THREE.Vector3(0.1, 0.1, 0.1) },
  { name: 'mooncake', scale: new THREE.Vector3(0.3, 0.3, 0.3) }
];

function loadPastryModel(modelInfo) {
  const loader = new OBJLoader();
  loader.load(`src/models/${modelInfo.name}.obj`, function (object) {
    const texture = new THREE.TextureLoader().load(`src/textures/${modelInfo.name}.png`);
    object.traverse(function (child) {
      if (child instanceof THREE.Mesh) {
        child.material = new THREE.MeshStandardMaterial({ map: texture });
      }
    });

    for (let i = 0; i < pastryCount; i++) {
      const pastry = object.clone();
      pastry.position.set(
        THREE.MathUtils.randFloatSpread(20),
        10 + Math.random() * 20,
        THREE.MathUtils.randFloatSpread(10)
      );
      pastry.scale.copy(modelInfo.scale);
      pastry.rotation.set(-Math.PI / 2, 0, 0);
      scene.add(pastry);

      pastries.push({ mesh: pastry, velocity: 0 });
    }
  });
}
```

Animation:

Bouncing: we found various elements needed to be considered to make the motion more realistic. The gravity forced on it, the time when the model reaches the ground, then, 30 percent of energy is lost during the bouncing.

```
// Animate function
function animate() {
  requestAnimationFrame(animate);
  pastries.forEach((pastry) => {
    // Apply gravity
    pastry.velocity += gravity;
    pastry.mesh.position.y += pastry.velocity;

    // Check for bounce
    if (pastry.mesh.position.y <= plateY + 1) {
      pastry.mesh.position.y = plateY + 1;
      pastry.velocity *= -bounceFactor;

      // Add a minimum bounce threshold to eventually stop bouncing
      if (Math.abs(pastry.velocity) < 0.01) {
        pastry.velocity = 0;
      }
    }

    // Rotate the pastry only if it's still moving
    if (Math.abs(pastry.velocity) > 0.01) {
      pastry.mesh.rotation.x += 0.01;
      pastry.mesh.rotation.y += 0.01;
    }
  });
  controls.update();
  renderer.render(scene, camera);
}
animate();
```

Experimental results:

We finally accomplish the project successfully. Palatable pastries with nice textures and rendering move smoothly and vividly in our project.

Summary and reflection:

Although the current phase of the project has Wbeen completed, there are still a long way to go. Further study of the project can focus on more realistic models, more detailed textures and rendering and more vivid animations. To achieve these goals, we could study and try new tools like Maya or Blender which are more powerful than OpenSCAD. Other technology like ray-tracking and visual effects can be also considered to enhance the recent project to give users better visual experience.

Contribution:

Bangqiao WANG (Nozomi):

Modeling:

- The whole process of modeling bear cookie

- The whole process of modeling cake

Power Point:

- Modeling parts (bear cookie, cake and cupcake)

Report:

- The general frame
- Modeling parts (bear cookie, cake and cupcake)
- Experimental results part
- Summary and reflection part
- Final check of the whole report

Kunliang RAO (Kunliang):

Modeling:

- The whole process of modeling simple mooncake

Texture:

- The whole process of making all texture maps

Report:

- Modeling parts (simple mooncake)
- Process of making texture maps

Power Point:

- Texture Part
- Modeling parts (simple mooncake)

Shiming HUANG (Poetty):

Modeling:

- Modelling of the Cookie, Mooncake and Plate

Power Point:

- The general frame
- Modeling parts (cookie, mooncake and plate)

Report:

- Modeling parts (cookie, mooncake and plate)

Yangfan BI (Evan):

Texture:

- Apply all textures to models.

Report:

- Rendering part.
- Animation part.

Yujie FENG (David):

Modeling:

- Cupcake

Rendering:

- Texture UV Wrapping in Blender
- Lighting

Animating:

- Import object in three.js
- Gravity, Bouncing Physics

Power Point:

- Rendering part
- Animation part

Report:

- Animation part
- Texture part