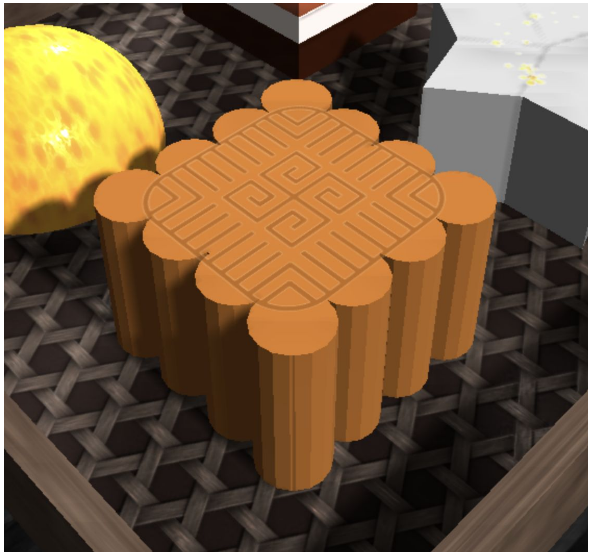
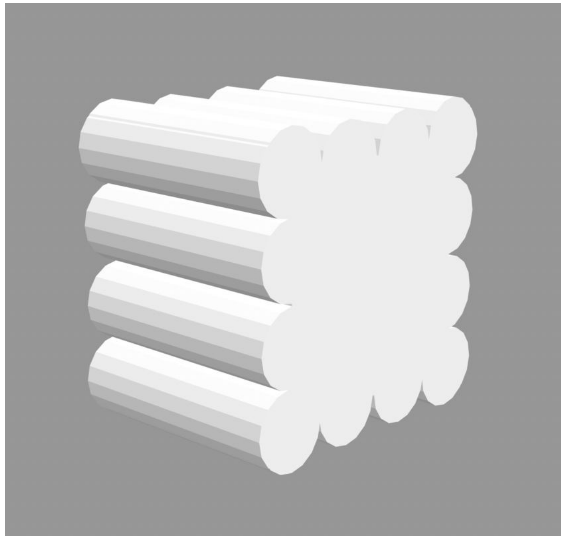
Render.

Mooncake

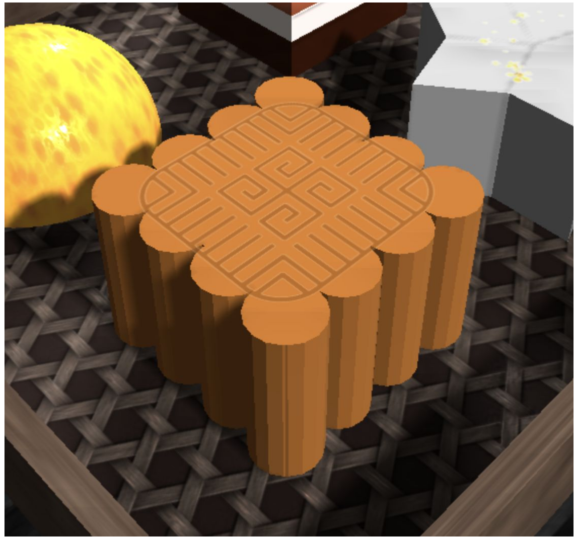
My Render



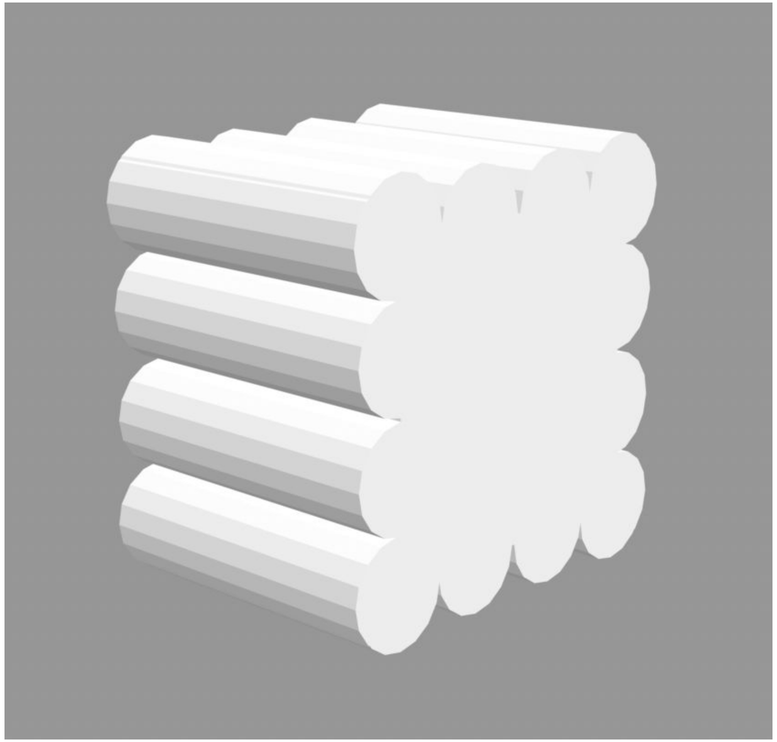


STL Model
Import the stl model to three.js via stl loader

Apply Material
Apple material by THREE.MeshPhysicalMaterial



Add Details
USe transparent texture map to add mooncake pattern



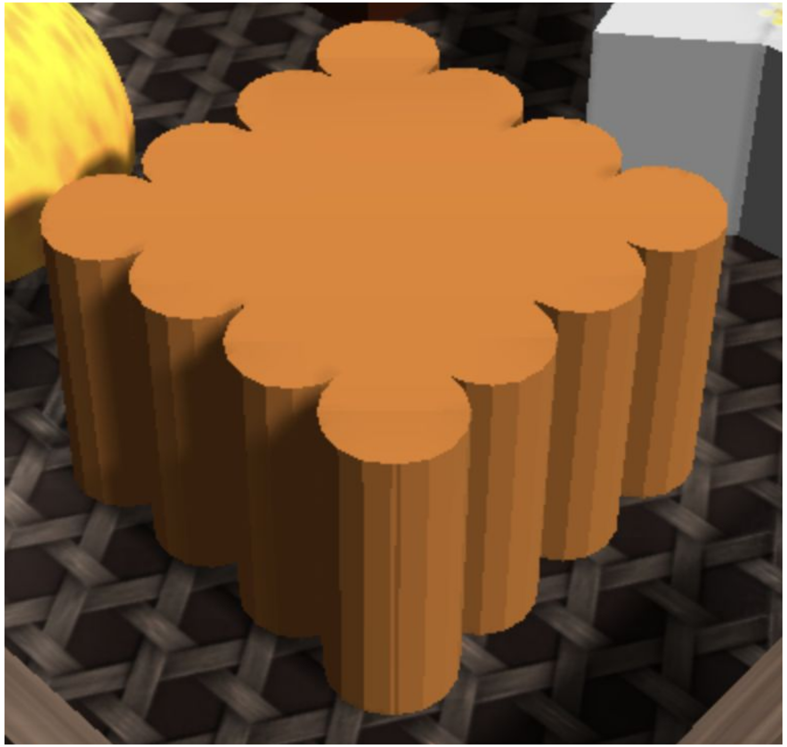
STL Model

Code implementation

```
import { STLLoader } from
'https://unpkg.com/three@0.126.1/exampl
es/jsm/loaders/STLLoader';

const loader = new STLLoader();

loader.load('stlModel/cake.stl',
function(cakeGeometry){
   const cakeMesh = new
THREE.Mesh(cakeGeometry,cakeMaterial)
   …
   scene.add(cakeMesh);
});
```

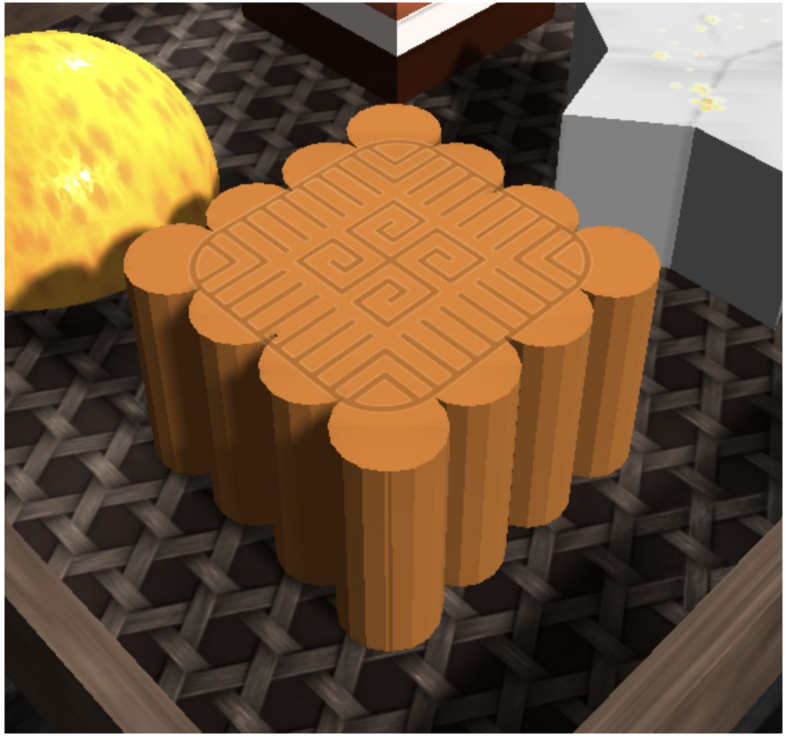


Apply Material

Code implementation

```
const cakeMaterial = new
THREE.MeshPhysicalMaterial({
    color: 0x914e13,
    roughness: 1,
});
```

Reasoning: `MeshPhysicalMaterial` can interact with three.js lights. Advantage over `MeshBasicMaterial`, many of shaderfrog's shaders, and texture mapping(not supported).



Add Detail

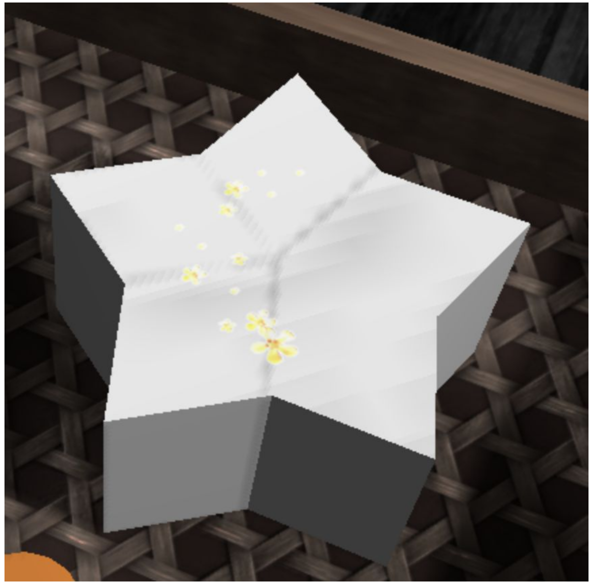
Code implementation

```
const cpGeometry = new
THREE.PlaneGeometry( 1.25, 1.25 );
     const cpMaterial = new
THREE.MeshStandardMaterial( {
     map: pattern,
     transparent: true, opacity: 0.6
});
const cp = new THREE.Mesh( cpGeometry,
cpMaterial);
scene.add( cp );
```

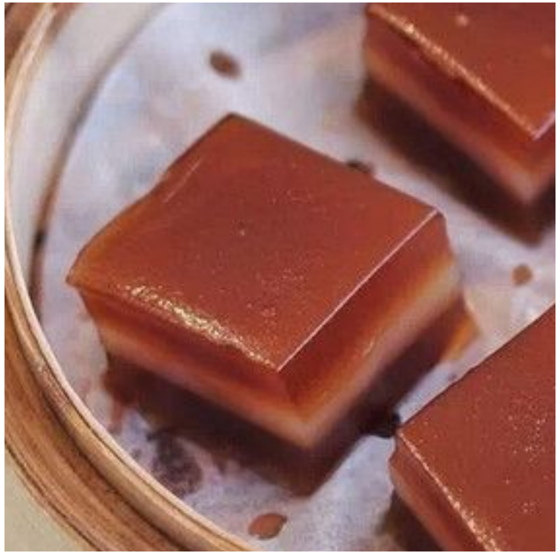
Reasoning: use another object(plane) to apply texture that is impossible for stl files

Sweet Osmanthus Cake

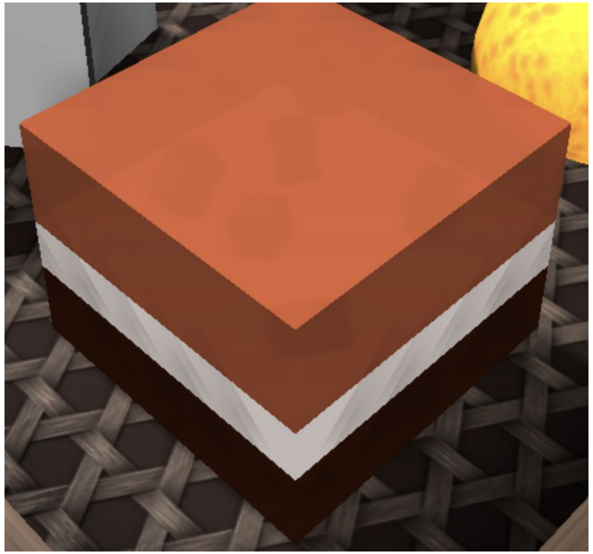
My Render

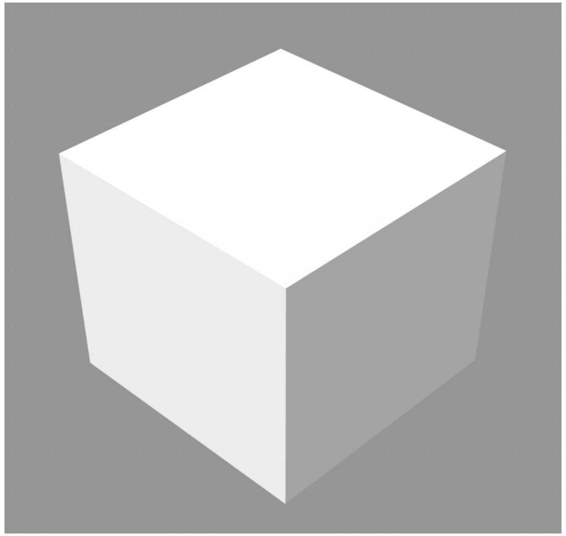


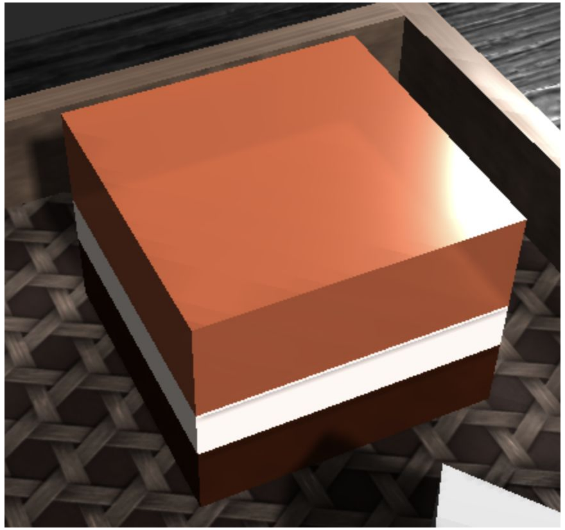
Red Dates Cake

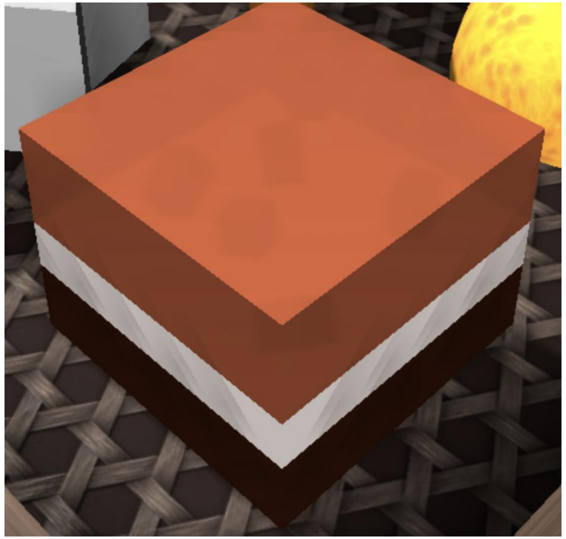


My Render





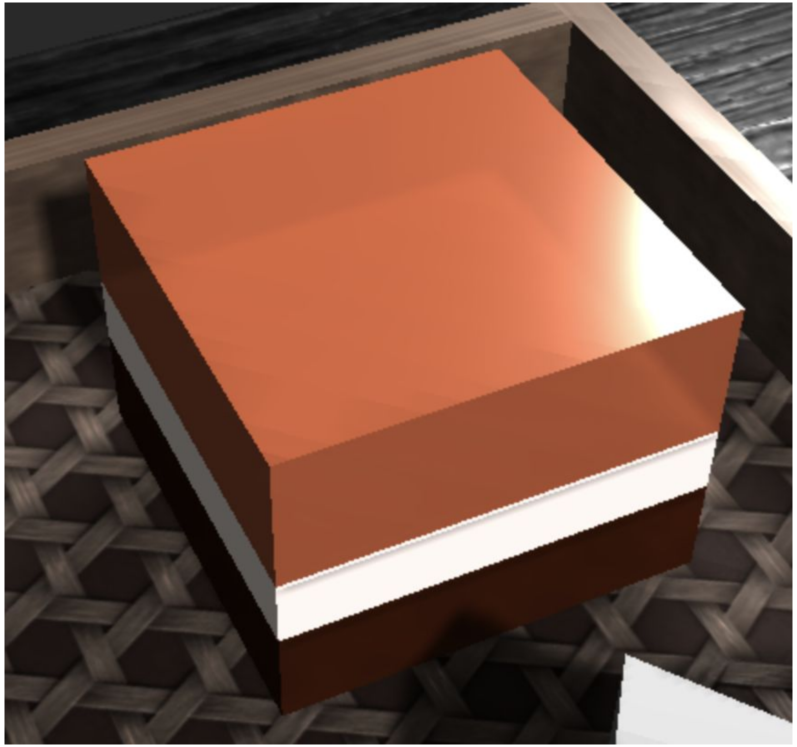




STL Model
Import the stl model to three.js via stl loader

Apply Material
Apply material by THREE.MeshPhysicalMaterial

Add Details
Create multiple cube objects to mimic red dates



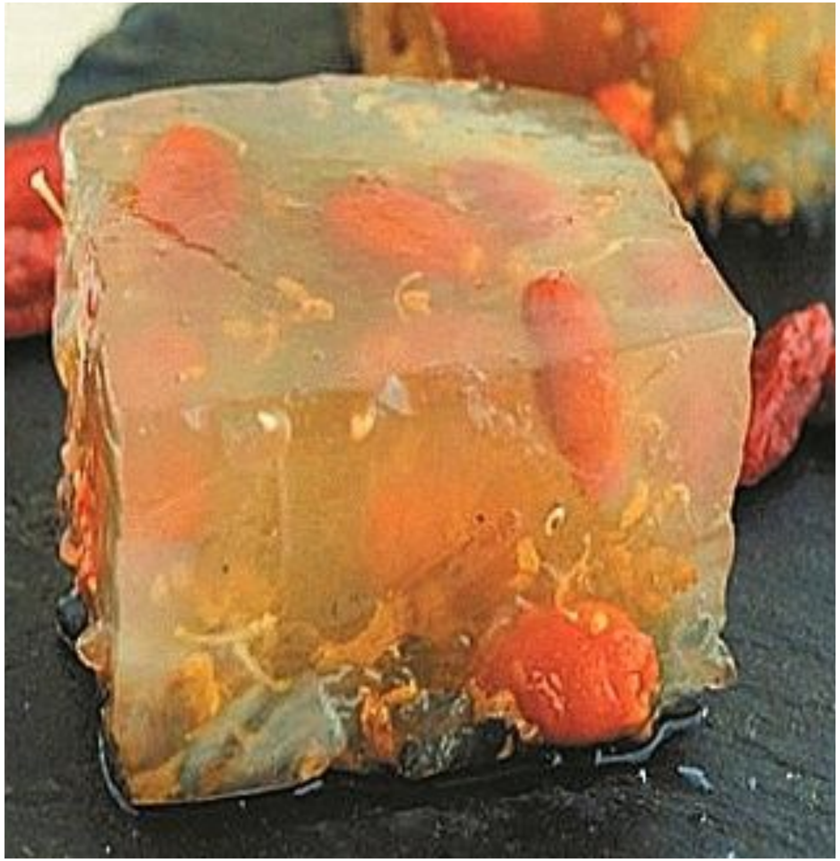
Apply Material

Code implementation

```
const material3 = new
THREE.MeshPhysicalMaterial({
   color: 0x944328,
   roughness: 0.2,
   transmission: 0.1,
   thickness: 0.4,
   clearcoat: 1.0
})
```

Reasoning:

`transmission: 0.1` makes model semi-transparent.
`clearcoat: 1.0` reflective surface
`thickness: 0.4` sense of volume



Add Details

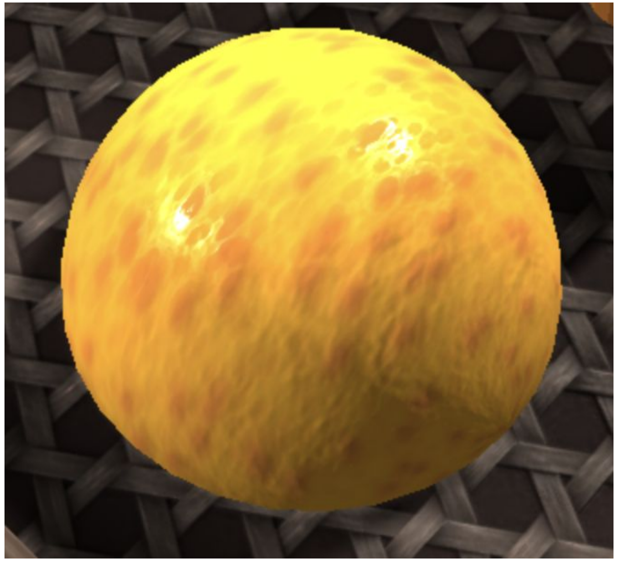
Code implementation

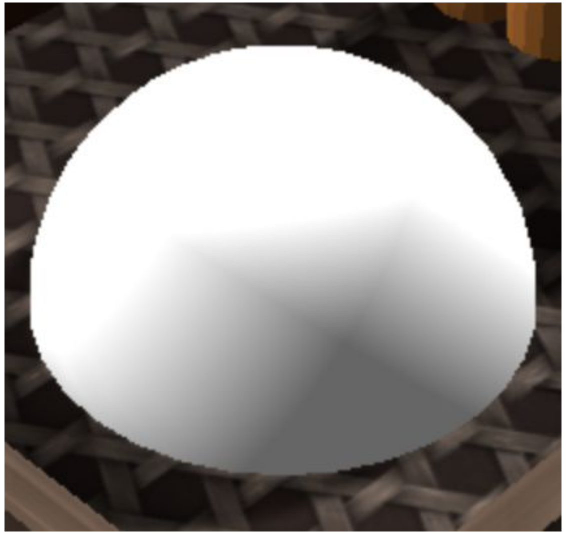
```
const chunk = new THREE.Mesh( new
THREE.BoxGeometry( 0.14, 0.14, 0.2 ),
     new THREE.MeshStandardMaterial( {
color: 0x30130c } ) );
     chunk.position.set( 1, 0.68, 1 );
     chunk.rotation.set( 1, 0.68, 1 );
     scene.add( chunk );
```

Reasoning:
Inspiration for another chinese pastries

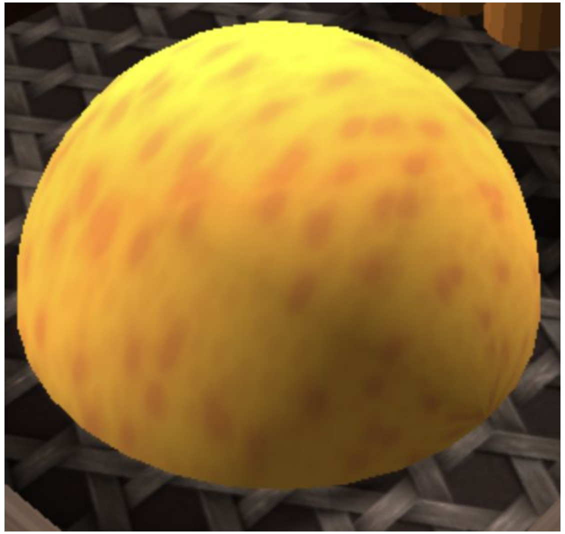
Su Pi Bing

My Render

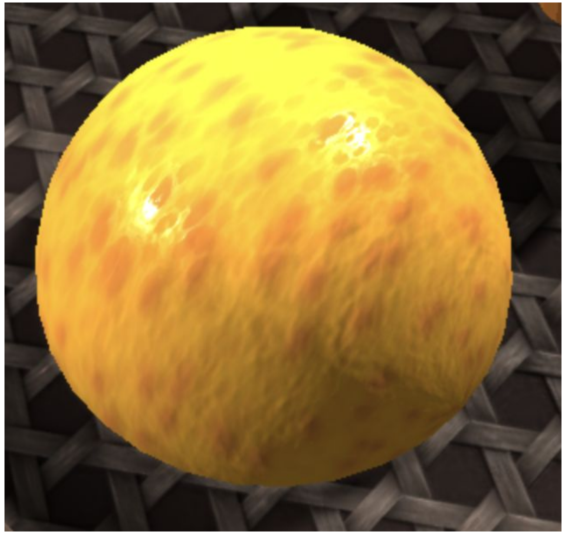




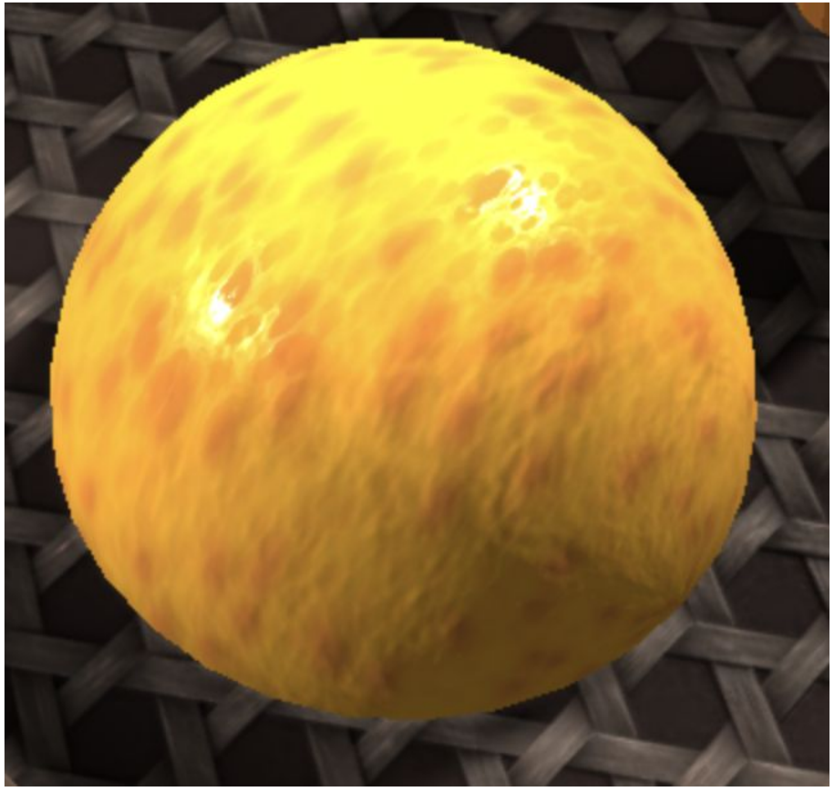
Three.js Sphere
Create the basic Sphere Geometry



Apply Color
Apply simple color texture to the sphere



Apply Texture
Apply normal, roughness, AO maps to the sphere

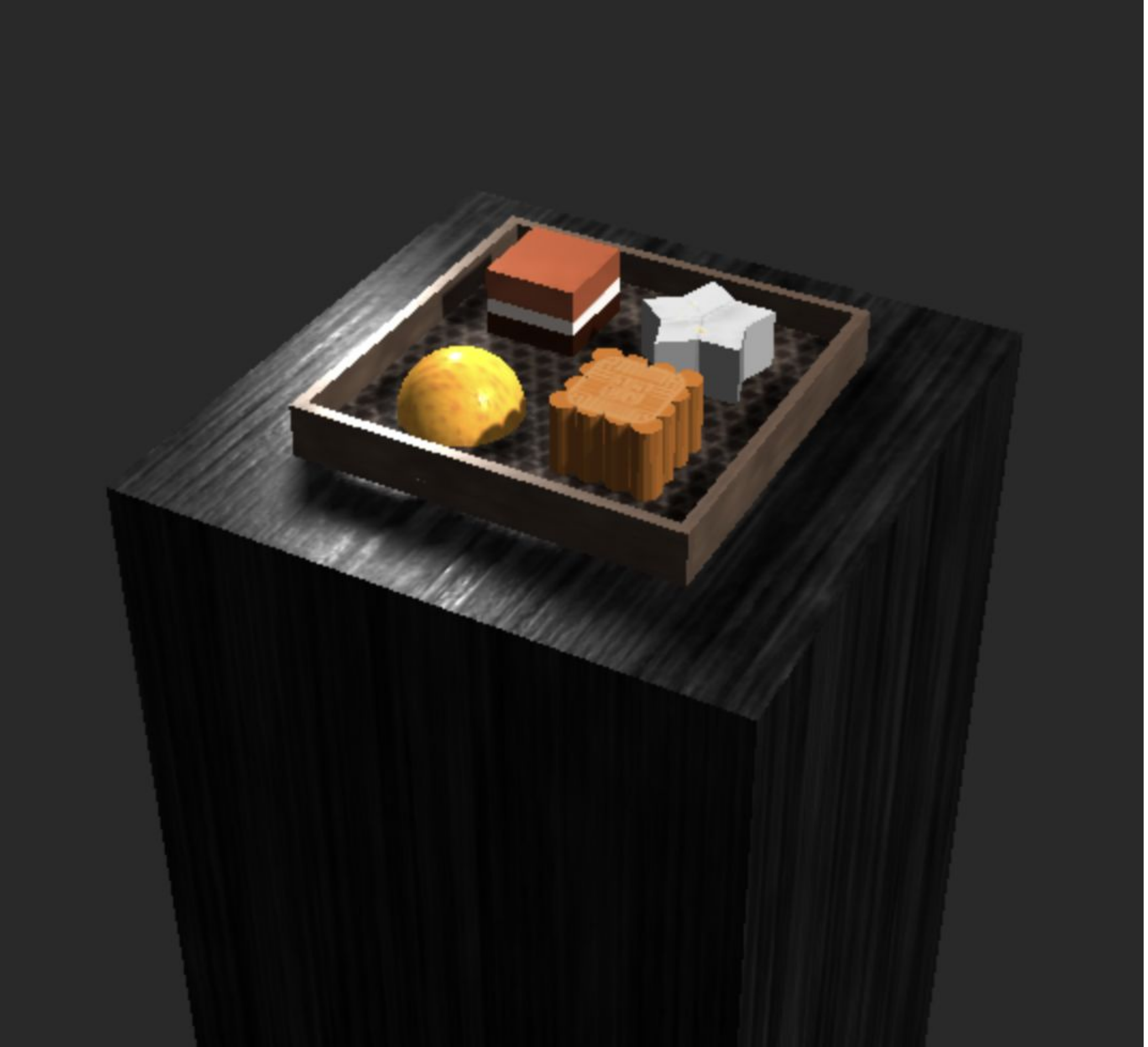


Apply Material

Code implementation

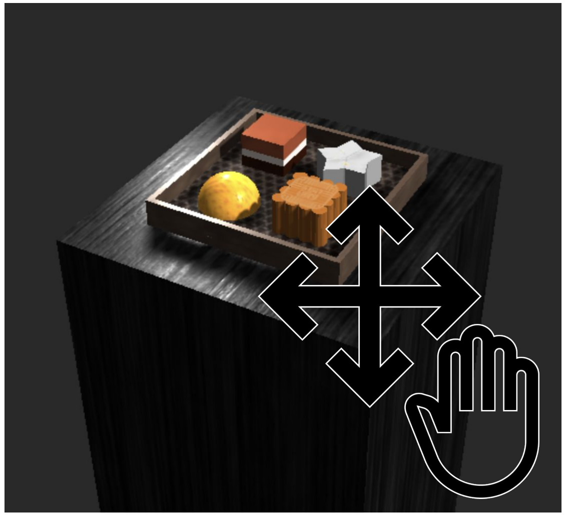
```
const ballMaterial = new
THREE.MeshStandardMaterial( {
     map: baseBall,
     normalMap: normalBall,
     roughnessMap: roughBall,
     roughness: 1.0,
     aoMap: aoBall
});
```

Reasoning:
Use normal, roughness, AO maps on top of base color map create a surface texture that is more realistic.



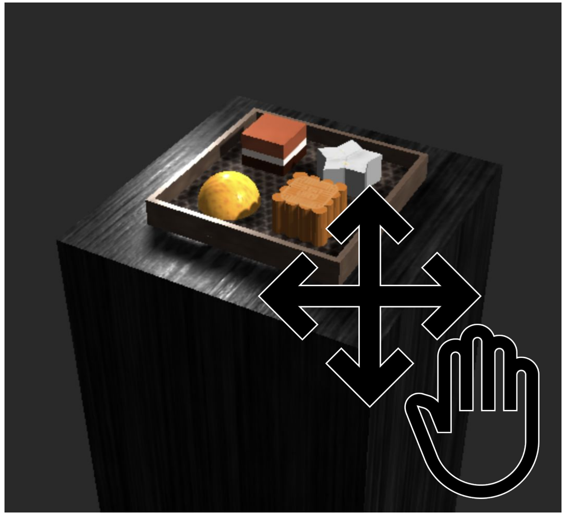
Show case.

Animation.



Manual Drag
Viewer can use their mouse cursor to manually drag the scene around

Auto Camera Rotate
When the mouse cursor is inactive, the camera will rotate around the models



Animation.

Code Implementation

```
import { OrbitControls } from
'https://unpkg.com/three@0.126.1/example
s/jsm/controls/OrbitControls.js';

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

controls.enablePan = false;
```

Manual Drag
Viewer can use their mouse cursor to manually drag the scene around

Animation.

Code Implementation

```
controls.autoRotate = true;
controls.autoRotateSpeed = 1.8;
controls.enablePan = false;
controls.addEventListener('start',
function(){
    controls.autoRotate = false;});
controls.addEventListener('end',
function(){
    setTimeout(() => {
        controls.autoRotate = true;
    }, 1000);});
```

Auto Camera Rotate
When the mouse cursor is inactive, the camera will rotate around the models

Thank
You For
Watching.