PACSSR-301007

Final Project Report

Tsukumo Jiang, Wency Wei, Justin Wu, Eric Yin, Austin Zhang

3/21/2025

1 Project Overview

This project implements an interactive 3D visualization system through three core technical components:

1.1 Procedural Modeling

1. **Cookie Designs**: Three distinct styles created using OpenSCAD's CSG operations
   - Classic: Cylindrical base with radial crack patterns
   - Modern: Cubic form with grid-aligned circular depressions
   - Artistic: Floral geometry with petal-edge detailing
2. **Snowman Candy**: Composite structure combining
   - Hierarchical spherical components (7 primitives total)
   - Cylindrical base platform
   - Position-controlled accessory placement
3. **Heart-shaped Cake**: Mathematical surface generation using parametric equations

$$\begin{cases} x = 16\sin^3(t) \\ y = 13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t) \end{cases}$$

4. **Plate Model**: Multi-layer construction with intentional segment variation

1.2 Three.js Animation & Controls

1. **Interactive Features**:
   - Dropdown menu for model selection
   - Toggle buttons for jump/rotation animations
   - Background switcher (solid/gradient/starry)
   - Pause/resume functionality
2. **Animation System**:
   - Vertical oscillation:
   - Continuous rotation:
   - Combined motion through superposition
3. **Technical Implementation**:
   - STL model loading with MeshPhongMaterial
   - Real-time shader switching for backgrounds
   - Event-driven UI state management

1.3 VR Realization

1. **A-Frame Integration**:
   - WebXR-enabled VR scene setup
   - GLTF model optimization for VRAM efficiency
   - Position-controlled entity placement
2. **Immersive Environment**:
   - Ambient + directional lighting configuration
   - Head-tracking camera controls
   - Gaze-based interaction cursor
3. **Cross-Platform Compatibility**:
   - Responsive viewport scaling
   - Mobile HMD alignment
   - Frame-rate optimized rendering

The system demonstrates a complete pipeline from procedural asset creation to interactive web visualization, supporting both conventional displays and VR headsets through shared model resources and complementary rendering subsystems.

2 Implementation Approach

2.1 Modeling Through OpenSCAD

1. The Cookie uses several different functions to generate different parts of the cookie:
   - **Cylinder()**: Utilize the cylinder() function to generate the primary thickness of the cookie, with the height representing the thickness and the radius denoted as r.
   - **The upper portion of the "arch" structure**: Form a sphere(r=r) and then stretch or compress it lengthwise with scale([1,1,domeHeight/r]), so that the sphere becomes a hemispherical structure of domeHeight. By intersecting intersection() and a cube(), only the upper part of the sphere will be retained, resulting in a smooth cookie "vault". union() combines the cylinder and arched parts into a complete cookie base.
   - **ChocolateChips(...) module**: Iterate through the coordinates in the positions array (pos[0], pos[1]) and determine if it is less than the radius r of the cookie (if (hypot(...)). ¡= r)) to determine whether the chocolate bean should fall on the surface of the cookie. Simulate each chocolate bean using sphere(r=3), and translate() places them on the corresponding coordinates and sits them on the surface of the cookie (thickness + 0.8*domeHeight).

- **Function hypot(x, y) = sqrt(x*x + y*y)**: This is a simple function used to calculate the Euclidean distance (that is, $\sqrt{x^2 + y^2}$. Used to determine the circumference of the chocolate bean's position relative to the center of the cookie.

2. The Geometry Cookies(classic cookie, modern cookie, artistic cookie) were created using OpenSCAD, focusing on procedural modeling techniques to ensure scalability and flexibility. Each cookie model was designed with the following approach:

   - Classic cookie: **Base**: A circular disc created using the cylinder primitive, with a radius of 20 and a height of 5. **Texture**: 12 small holes evenly distributed along the edge using a for loop and rotate operation, simulating the natural cracks of a traditional cookie.
   - Modern cookie: **Base**: A square shape created using the cube primitive, with a side length of 40 and a height of 5, centered in the scene. **Geometric Pattern**: 25 circular grooves evenly distributed across the surface using nested for loops, creating a modern grid-like texture.
   - Artistic cookie: **Base**: A flower shape created using the cylinder primitive and rotate operation, with 6 petals evenly distributed around the center. **Texture**: 12 small holes evenly distributed along the edges of the petals, enhancing the artistic appearance.

3. The Snowman Candy uses **combination of multiple geometries**:

   - **Hierarchical positioning**: Translation vectors (x, y, z) position components relative to parent coordinate system
   - **Primitive combination**: Union operation merges 7 distinct geometries (2 large spheres, 1 cylinder, 4 small spheres)

4. The cake model is created using OpenSCAD and consists of four main parts:

   - The heart-shaped body is positioned 10 units above the Z-axis, with its color set to pink (RGB value '[1, 0.7529, 0.7961]'). It is generated by calculating its 2D contour points using mathematical formulas:

$$\begin{cases} x = 16\sin^3(t) \\ y = 13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t) \end{cases}$$

   - The bottom part consists of two cylinders, each with a height of 10 and a radius of 20. The first cylinder is a default smooth cylinder, while the second cylinder has a lower segment count set by '$fs=6', making its surface rougher.
   - In the middle section, a cone is positioned 3 units above the Z-axis, with its color set to silver (RGB value '[0.7529, 0.7529, 0.7529]'). The cone has a top radius of 10, a bottom radius of 20, and a height of 10. It is centered using 'center=true'. This is the plate for the cake.

5. For the plate model, the code references the cylinder part above, modifying the height to 1 and the radius to 25. It also uses '$fs=6' to set a lower segment count, making its surface rougher.

2.2 Three.js Integration

1. Load STL 3D Model
   - Import STLLoader and specify the model path as 'models/xxx.stl'.
   - Use loader.load to load the model and pass the geometry to Mesh constructor to create a mesh.
   - Set the material as MeshPhongMaterial and add the mesh to the scene.
2. HTML Interface Setup:
   - Created `<select>` element with options for 5 models: Cookie, Cookie1, Cookie2, Cookie3, Snowman Candy, and Heart Cake
   - Positioned button with click event binding
3. Implement Jump Animation
   - Define the jump height (jumpHeight) and speed (jumpSpeed).
   - Use Math.sin(time) to simulate vertical movement and update the model's Y-coordinate in the animation loop.
   - When jumping is enabled, the model begins to jump.
4. Implement Rotation Animation
   - Define rotationSpeed to control the rotation speed.
   - In the animation loop, increment model.rotation.y to create rotation.
   - When rotating is enabled, the model starts to rotate.
5. Combine Jump and Rotation
   - Clicking the button sets both jumping and rotating to true.
   - The animation loop updates both position and rotation simultaneously.
6. Pause Animation
   - Click the pause button to set animationState.paused = true.
   - Check the paused state in the animation loop to stop position and rotation updates.
7. Switch Background
   - Define the backgrounds array, which includes different color options, gradients, and a starry sky.
   - Click the button to trigger the changeBackground function to cycle through different backgrounds.
   - Update scene.background or renderer behavior based on the background type.
8. Starry Background
   - Create the createStars function to generate 5000 random points as stars.

- Add the star point cloud to the scene to simulate a starry background.
- Slowly rotate the stars in the animation loop to enhance visual dynamics.

9. Gradient Background
    - Define ShaderMaterial using custom vertexShader and fragmentShader.
    - Use vUv.y in the fragment shader to interpolate between color1 and color2 for a vertical gradient.
    - Render the gradient background using renderer.render.

2.3 VR Realization

The VR implementation is based on the A-Frame framework, providing an immersive 3D model viewer. The specific implementation includes:

- **A-Frame Framework Integration**: The A-Frame library (version 1.7.0) is introduced, and the VR scene is created with VR mode enabled.
- **Model Loading and Placement**: GLTF-format model resources (cake, cookie1, cookie2, cookie3, snowman) are preloaded and placed in the scene using the <a-entity> tag, with position and scale properties configured.
- **Scene Setup**: The background is set to light gray (<a-sky>), and ambient and directional lights are added to ensure natural lighting effects.
- **Interactive Camera**: An interactive camera is created to allow users to control the viewpoint, and a cursor is added for gaze-based interactions.

3 Results

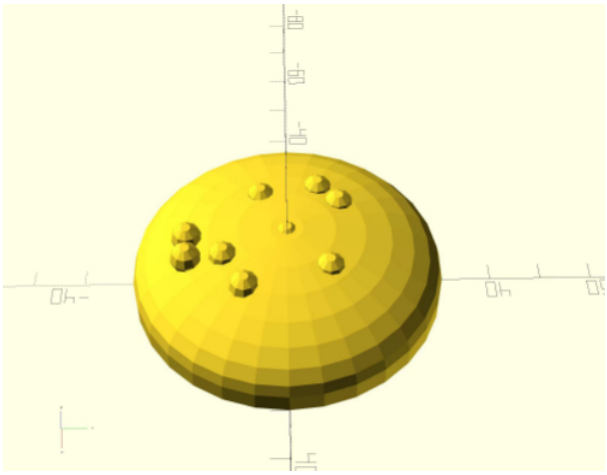

Figure 1: Cookie

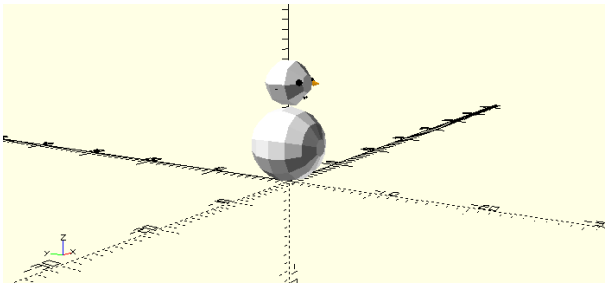

Figure 2: Snowman Candy

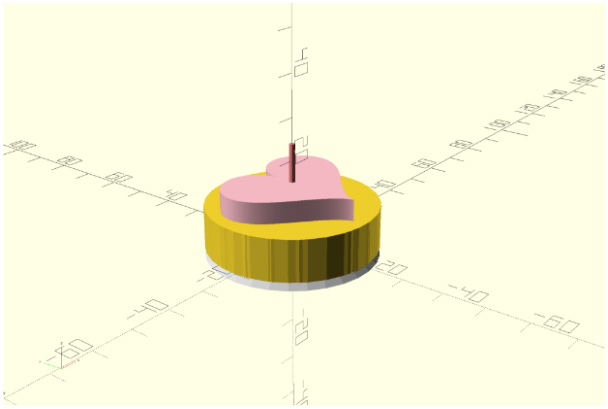

Figure 3: Cake

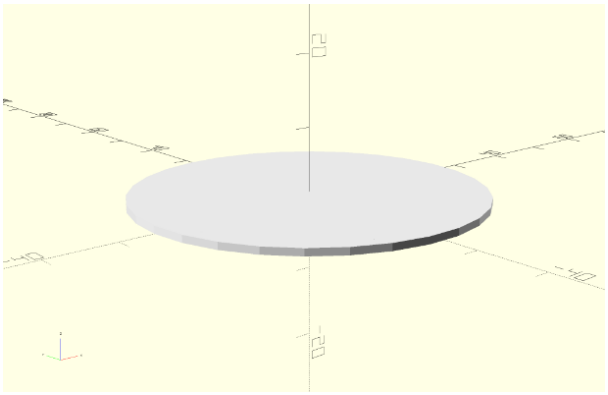

Figure 4: Plate

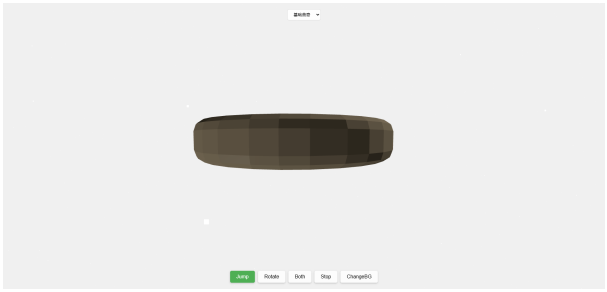

Figure 5: Jumping Cookie

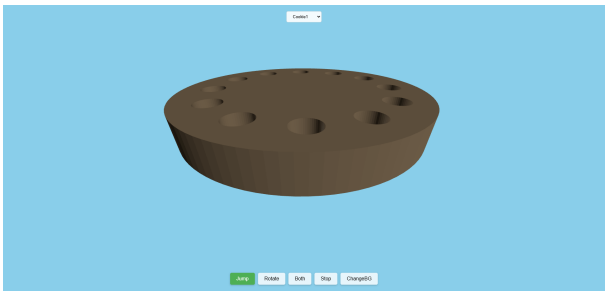

Figure 6: Jumping Cookie1 Blue

Figure 7: Jumping Cookie2 Pink

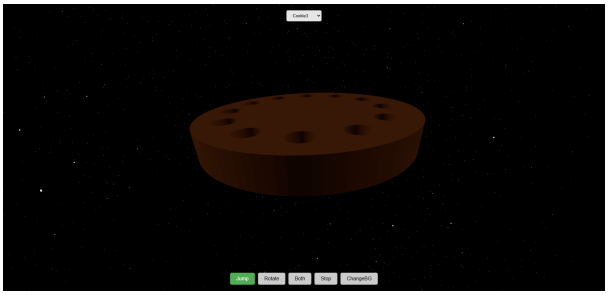

Figure 8: Jumping Cookie3 Starry

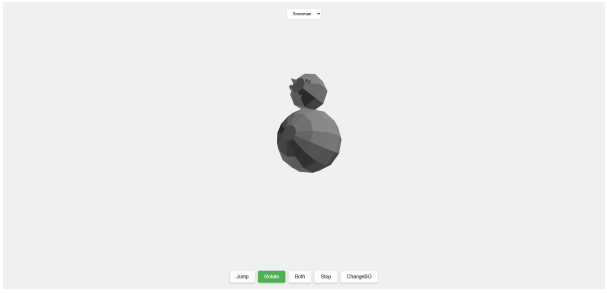

Figure 9: Rotating Snowman

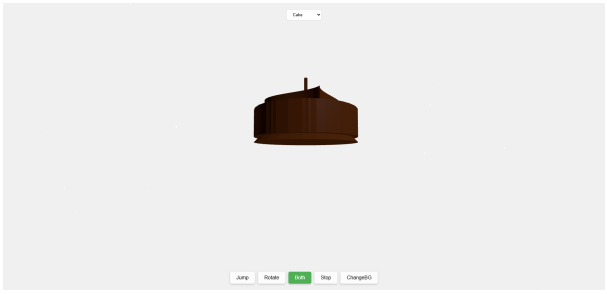

Figure 10: Jumping and Rotating Cake

4 Teamwork Split-up

Chengzi Jiang: Three.js animation, Github maintainance
Wency Wei: Cake modeling, Plate modeling, Powerpoint design
Zhuoyuan Wu: Geometry cookies modeling, VR production
Eric Yin: Cookie modeling, Screenshot management, Video Editing
Austin Zhang: Snowman modeling, Three.js Dropdown & Button Functions, LaTeX maintainance, Github maintainance