<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>3D Aquarium Experience</title>

<style>

body, html {

margin: 0;

padding: 0;

height: 100%;

overflow: hidden;

display: flex;

justify-content: center;

align-items: center;

background-color: #000;

color: #fff;

font-family: Arial, sans-serif;

}

#container {

position: relative;

width: 90vw;

height: 90vh;

perspective: 1500px;

}

.face {

position: absolute;

width: 100%;

height: 100%;

background-size: cover;

background-position: center;

opacity: 0.9;

transition: transform 0.8s ease-in-out;

}

.front { transform: translateZ(500px); }

.back { transform: translateZ(-500px) rotateY(180deg); background-color: #333; }

.top { transform: rotateX(90deg) translateZ(500px); }

.bottom { transform: rotateX(-90deg) translateZ(500px); background: url('floor.jpg'); }

.left { transform: rotateY(-90deg) translateZ(500px); }

.right { transform: rotateY(90deg) translateZ(500px); }

#cube {

position: absolute;

width: 100%;

height: 100%;

transform-style: preserve-3d;

transition: transform 0.8s ease-in-out;

}

.arrow {

position: fixed;

width: 30px;

height: 30px;

background-color: #fff;

color: #000;

text-align: center;

line-height: 30px;

border-radius: 50%;

cursor: pointer;

z-index: 10;

}

.up { top: 10px; left: 50%; transform: translateX(-50%); }

.down { bottom: 10px; left: 50%; transform: translateX(-50%); }

.left { top: 50%; left: 10px; transform: translateY(-50%); }

.right { top: 50%; right: 10px; transform: translateY(-50%); }

#clock {

position: fixed;

top: 10px;

right: 10px;

background-color: rgba(0, 0, 0, 0.5);

padding: 10px;

border-radius: 8px;

font-size: 18px;

z-index: 20;

}

</style>

</head>

<body>

<div id="clock">09:00 AM | Clicks: 0</div>

<div id="container">

<div id="cube">

<div class="face front" style="background: url('glass.jpg');"></div>

<div class="face back"></div>

<div class="face top" style="background: url('water\_surface.jpg');"></div>

<div class="face bottom"></div>

<div class="face left" style="background: url('water.jpg');"></div>

<div class="face right" style="background: url('water.jpg');"></div>

</div>

</div>

<div class="arrow up">↑</div>

<div class="arrow down">↓</div>

<div class="arrow left">←</div>

<div class="arrow right">→</div>

<audio id="splash" src="splash.mp3" preload="auto"></audio>

<script>

const cube = document.getElementById('cube');

const splash = document.getElementById('splash');

const clock = document.getElementById('clock');

let angleX = 0;

let angleY = 0;

let clicks = 0;

let hours = 9;

document.querySelector('.up').addEventListener('click', () => {

angleX -= 90;

updateCube();

});

document.querySelector('.down').addEventListener('click', () => {

angleX += 90;

updateCube();

});

document.querySelector('.left').addEventListener('click', () => {

angleY -= 90;

updateCube();

});

document.querySelector('.right').addEventListener('click', () => {

angleY += 90;

updateCube();

});

function updateCube() {

cube.style.transform = `rotateX(${angleX}deg) rotateY(${angleY}deg)`;

playSplashSound();

updateClock();

}

function playSplashSound() {

splash.currentTime = 0;

splash.play();

}

function updateClock() {

clicks++;

hours++;

if (hours > 21) {

hours = 9;

document.body.style.backgroundColor = document.body.style.backgroundColor === '#000' ? '#222' : '#000';

}

const period = hours < 12 || hours === 24 ? 'AM' : 'PM';

const displayHours = hours % 12 === 0 ? 12 : hours % 12;

clock.textContent = `${displayHours}:00 ${period} | Clicks: ${clicks}`;

}

</script>

</body>

</html>

1. 保留上面这段代码里的“**右上角的时钟与点击次数计数器**：每点击一次箭头，时钟前进一个小时，且记录点击次数。完成12次点击后会切换到一个新的空间，颜色不同但结构相同。”这个功能
2. <!DOCTYPE html>
3. <html lang="en">
4. <head>
5. <meta charset="UTF-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1.0">
7. <title>Interactive 3D Cube</title>
8. <style>
9. body, html {
10. margin: 0;
11. height: 100%;
12. overflow: hidden;
13. display: flex;
14. justify-content: center;
15. align-items: center;
16. background-color: #333;
17. }
18. #cube {
19. width: 300px;
20. height: 300px;
21. transform-style: preserve-3d;
22. transition: transform 1s;
23. }
24. .face {
25. position: absolute;
26. width: 300px;
27. height: 300px;
28. background-color: rgba(255, 255, 255, 0.9);
29. border: 2px solid #fff;
30. font-size: 30px;
31. color: #333;
32. display: flex;
33. justify-content: center;
34. align-items: center;
35. transition: background-color 0.5s, transform 1s;
36. }
37. /\* 每个面的背景或内容 \*/
38. .front {
39. background-image: url('front.jpg'); /\* 示例图片 \*/
40. background-size: cover;
41. transform: translateZ(150px);
42. }
43. .back {
44. background-image: url('back.jpg');
45. background-size: cover;
46. transform: rotateY(180deg) translateZ(150px);
47. }
48. .left {
49. background-color: lightblue;
50. transform: rotateY(-90deg) translateZ(150px);
51. }
52. .right {
53. background-color: lightgreen;
54. transform: rotateY(90deg) translateZ(150px);
55. }
56. .top {
57. background-color: lightpink;
58. transform: rotateX(90deg) translateZ(150px);
59. }
60. .bottom {
61. background-color: lightyellow;
62. transform: rotateX(-90deg) translateZ(150px);
63. }
64. .controls {
65. position: absolute;
66. top: 20px;
67. right: 20px;
68. font-size: 24px;
69. color: #fff;
70. text-align: right;
71. }
72. .arrows {
73. position: absolute;
74. bottom: 20px;
75. left: 50%;
76. transform: translateX(-50%);
77. display: flex;
78. gap: 20px;
79. }
80. .arrow {
81. cursor: pointer;
82. font-size: 36px;
83. color: #fff;
84. }
85. .arrow:hover {
86. color: #ff0;
87. }
88. .expanded {
89. transform: translateZ(300px) scale(1.5);
90. }
91. </style>
92. </head>
93. <body>
94. <div id="cube">
95. <div class="face front">Front</div>
96. <div class="face back">Back</div>
97. <div class="face left">Left</div>
98. <div class="face right">Right</div>
99. <div class="face top">Top</div>
100. <div class="face bottom">Bottom</div>
101. </div>
102. <div class="controls">
103. <div id="clock">09:00</div>
104. <div id="clicks">Clicks: 0</div>
105. </div>
106. <div class="arrows">
107. <div class="arrow" id="left-arrow">◄</div>
108. <div class="arrow" id="right-arrow">►</div>
109. <div class="arrow" id="up-arrow">▲</div>
110. </div>
111. <script>
112. let currentAngle = {x: 0, y: 0};
113. let clicks = 0;
114. let hours = 9;
115. let expandedFace = null;
116. const cube = document.getElementById('cube');
117. const clock = document.getElementById('clock');
118. const clicksDisplay = document.getElementById('clicks');
119. const arrowLeft = document.getElementById('left-arrow');
120. const arrowRight = document.getElementById('right-arrow');
121. const arrowUp = document.getElementById('up-arrow');
122. const faces = document.querySelectorAll('.face');
123. function updateClock() {
124. hours = (hours % 12) + 1;
125. clock.innerText = `${hours.toString().padStart(2, '0')}:00`;
126. clicks++;
127. clicksDisplay.innerText = `Clicks: ${clicks}`;
128. if (hours === 9) {
129. document.body.style.backgroundColor = document.body.style.backgroundColor === 'black' ? '#333' : 'black';
130. }
131. }
132. function rotateCube() {
133. cube.style.transform = `rotateX(${currentAngle.x}deg) rotateY(${currentAngle.y}deg)`;
134. }
135. function expandFace(face) {
136. face.classList.add('expanded');
137. }
138. function resetExpansion() {
139. if (expandedFace) {
140. expandedFace.classList.remove('expanded');
141. expandedFace = null;
142. }
143. }
144. function handleArrowClick(direction) {
145. resetExpansion();
146. if (direction === 'left') {
147. currentAngle.y -= 90;
148. } else if (direction === 'right') {
149. currentAngle.y += 90;
150. } else if (direction === 'up') {
151. currentAngle.x -= 90;
152. }
153. if (currentAngle.x % 360 === 0 && currentAngle.y % 360 === 0) {
154. arrowUp.dataset.target = 'top';
155. } else if (currentAngle.x % 360 === -90) {
156. arrowUp.dataset.target = 'back';
157. } else if (currentAngle.x % 360 === -180) {
158. arrowUp.dataset.target = 'bottom';
159. } else if (currentAngle.x % 360 === -270) {
160. arrowUp.dataset.target = 'front';
161. }
162. rotateCube();
163. updateClock();
164. }
165. arrowLeft.addEventListener('click', () => handleArrowClick('left'));
166. arrowRight.addEventListener('click', () => handleArrowClick('right'));
167. arrowUp.addEventListener('click', () => handleArrowClick('up'));
168. window.addEventListener('wheel', (event) => {
169. const delta = Math.sign(event.deltaY);
170. if (delta < 0) {
171. expandFace(faces[delta]);
172. } else {
173. resetExpansion();
174. }
175. });
176. </script>
177. </body>
178. </html>

保留这段代码中“· **箭头与视图切换**：点击左右或上箭头时，立方体旋转相应的角度，切换视图。

· · **视图扩展与归位**：滚动鼠标中键可以将视图向外扩展，点击箭头切换视图时会归位。

· · **无限切换**：从正视图点击上箭头可以到达顶视图，依次切换至背视图、底视图和正视图，左右视图也可无限切换。”的功能，但是**箭头与视图切换只能12次，到达十二次就进入另一个除了背景颜色和文字以外一模一样的空间，所以你需要在这个空间上给出不同的文字标记让我知道这是另一个空间。到达24次之后就会归零，从0点开始，同时回到一开始的空间，一直重复**

3.<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>3D Room with Interactions</title>

<style>

body, html {

margin: 0;

overflow: hidden;

height: 100%;

}

#scene {

width: 100%;

height: 100%;

position: relative;

perspective: 1000px;

transform-style: preserve-3d;

}

.cube {

width: 1920px;

height: 1080px;

position: absolute;

transform-style: preserve-3d;

}

.face {

position: absolute;

width: 1920px;

height: 1080px;

backface-visibility: hidden;

}

/\* Assigning background images or colors to each face \*/

#front { background: url('front.jpg') center/cover; }

#back { background: url('back.jpg') center/cover; transform: rotateY(180deg) translateZ(960px); }

#left { background: url('left.jpg') center/cover; transform: rotateY(-90deg) translateZ(960px); }

#right { background: url('right.jpg') center/cover; transform: rotateY(90deg) translateZ(960px); }

#top { background: url('top.jpg') center/cover; transform: rotateX(90deg) translateZ(540px); }

#bottom { background: url('bottom.jpg') center/cover; transform: rotateX(-90deg) translateZ(540px); }

.arrow {

position: absolute;

width: 50px;

height: 50px;

background-color: rgba(255, 255, 255, 0.5);

cursor: pointer;

z-index: 10;

display: flex;

justify-content: center;

align-items: center;

font-size: 24px;

color: #000;

}

/\* Positioning the arrows \*/

#arrow-up { top: 10px; left: 50%; transform: translateX(-50%); }

#arrow-down { bottom: 10px; left: 50%; transform: translateX(-50%); }

#arrow-left { top: 50%; left: 10px; transform: translateY(-50%); }

#arrow-right { top: 50%; right: 10px; transform: translateY(-50%); }

</style>

</head>

<body>

<div id="scene">

<div id="cube" class="cube">

<div id="front" class="face"></div>

<div id="back" class="face"></div>

<div id="left" class="face"></div>

<div id="right" class="face"></div>

<div id="top" class="face"></div>

<div id="bottom" class="face"></div>

</div>

<div id="arrow-up" class="arrow">↑</div>

<div id="arrow-down" class="arrow">↓</div>

<div id="arrow-left" class="arrow">←</div>

<div id="arrow-right" class="arrow">→</div>

</div>

<script>

const cube = document.getElementById('cube');

let rotateX = 0;

let rotateY = 0;

const updateTransform = () => {

cube.style.transform = `rotateX(${rotateX}deg) rotateY(${rotateY}deg)`;

};

document.getElementById('arrow-up').addEventListener('click', () => {

rotateX -= 90;

updateTransform();

});

document.getElementById('arrow-down').addEventListener('click', () => {

rotateX += 90;

updateTransform();

});

document.getElementById('arrow-left').addEventListener('click', () => {

rotateY -= 90;

updateTransform();

});

document.getElementById('arrow-right').addEventListener('click', () => {

rotateY += 90;

updateTransform();

});

document.addEventListener('wheel', (event) => {

const delta = Math.sign(event.deltaY);

const currentScale = parseFloat(getComputedStyle(cube).transform.match(/matrix.\*\((.+)\)/)[1].split(', ')[0]);

const newScale = Math.max(1, currentScale + delta \* 0.1);

cube.style.transform += ` scale(${newScale})`;

});

const resetScale = () => {

cube.style.transform = `rotateX(${rotateX}deg) rotateY(${rotateY}deg) scale(1)`;

};

document.getElementById('arrow-up').addEventListener('click', resetScale);

document.getElementById('arrow-down').addEventListener('click', resetScale);

document.getElementById('arrow-left').addEventListener('click', resetScale);

document.getElementById('arrow-right').addEventListener('click', resetScale);

</script>

</body>

</html>

保留上面这段代码中“· **设置相机位置和视角**：相机（用户视角）始终位于空间中心，且每个平面的大小都固定为1920px\*1080px。

· **调整每个平面的位置和大小**：每个平面与相机的距离和角度确保在用户视角内正好填满视口，从而基本看不到空隙，但是能看到透视。

”

4.<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cube with Fixed Planes</title>

<style>

body { margin: 0; }

canvas { display: block; }

</style>

</head>

<body>

<script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r128/three.min.js"></script>

<script>

// Scene setup

const scene = new THREE.Scene();

const camera = new THREE.PerspectiveCamera(75, window.innerWidth / window.innerHeight, 0.1, 1000);

const renderer = new THREE.WebGLRenderer();

renderer.setSize(window.innerWidth, window.innerHeight);

document.body.appendChild(renderer.domElement);

// Dimensions of the cube

const width = 1920;

const height = 1080;

const depth = 1920; // Adjust this to ensure all planes are enclosed

// Create the six faces of the cube

const createFace = (width, height, color) => {

const geometry = new THREE.PlaneGeometry(width, height);

const material = new THREE.MeshBasicMaterial({ color, side: THREE.DoubleSide });

return new THREE.Mesh(geometry, material);

};

const frontFace = createFace(width, height, 0xff0000); // Red

const backFace = createFace(width, height, 0x00ff00); // Green

const leftFace = createFace(width, height, 0x0000ff); // Blue

const rightFace = createFace(width, height, 0xffff00); // Yellow

const topFace = createFace(width, height, 0xff00ff); // Magenta

const bottomFace = createFace(width, height, 0x00ffff); // Cyan

// Position the faces

frontFace.position.z = depth / 2;

backFace.position.z = -depth / 2;

leftFace.position.x = -width / 2;

leftFace.rotation.y = Math.PI / 2;

rightFace.position.x = width / 2;

rightFace.rotation.y = -Math.PI / 2;

topFace.position.y = height / 2;

topFace.rotation.x = -Math.PI / 2;

bottomFace.position.y = -height / 2;

bottomFace.rotation.x = Math.PI / 2;

// Add the faces to the scene

scene.add(frontFace);

scene.add(backFace);

scene.add(leftFace);

scene.add(rightFace);

scene.add(topFace);

scene.add(bottomFace);

// Position the camera at the center of the cube

camera.position.z = depth / 2;

// Handle window resizing

window.addEventListener('resize', () => {

camera.aspect = window.innerWidth / window.innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(window.innerWidth, window.innerHeight);

});

// Render loop

const animate = () => {

requestAnimationFrame(animate);

renderer.render(scene, camera);

};

animate();

</script>

</body>

</html>

保留着段代码中每个平面大小固定：所有平面（front, back, left, right, top, bottom）的大小都设置为1920px\*1080px。 空间闭合，确保每个平面都完全封闭在视角中。 视角和相机设置：用户的视角（相机）始终处于立方体中心。还有这段代码中空间的大小透视、颜色的设置

1. 将这四段代码根据各自要求保留的功能完美结合成一个符合主题的优化过的水族馆艺术网站

老师 你啥时候有空？我找ai生成了几个我比较想要的效果，但是细节上我得靠自己调整，上想调整但是