如果要将水体的厚度设置为其最大水位高度，可以将 **extrudedHeight** 属性设置为 **currentWaterHeight**，而不是 **currentWaterHeight + waterThickness**。这样，水体的厚度将直接等于当前的水位高度。

以下是更新后的代码示例：

html

复制代码

<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Cesium with Dynamic Water Effect</title> <script src="https://cesium.com/downloads/cesiumjs/releases/1.89/Build/Cesium/Cesium.js"></script> <link href="https://cesium.com/downloads/cesiumjs/releases/1.89/Build/Cesium/Widgets/widgets.css" rel="stylesheet"> <style> html, body, #cesiumContainer { width: 100%; height: 100%; margin: 0; padding: 0; overflow: hidden; } #controlPanel { position: absolute; top: 10px; right: 10px; background: rgba(255, 255, 255, 0.8); padding: 10px; border-radius: 8px; box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.5); } #controlPanel button { display: block; width: 100px; margin-bottom: 5px; } #controlPanel input { width: 90px; margin-bottom: 5px; } </style> </head> <body> <div id="cesiumContainer"></div> <div id="controlPanel"> <input type="number" id="heightControl" placeholder="Target Height (m)" value="500"> <input type="number" id="speedControl" placeholder="Speed (m/s)" value="1"> <button id="raiseWater">Raise Water</button> <button id="lowerWater">Lower Water</button> </div> <script> // Initialize Cesium Viewer const viewer = new Cesium.Viewer('cesiumContainer', { terrainProvider: Cesium.createWorldTerrain(), animation: true, timeline: true }); // Variables to manage water height and control states let currentWaterHeight = 0; let targetWaterHeight = 500; let isRaising = false; let isLowering = false; const maxHeight = 500; // Function to set view to user's current location function setViewToCurrentLocation(position) { const latitude = position.coords.latitude; const longitude = position.coords.longitude; viewer.camera.setView({ destination: Cesium.Cartesian3.fromDegrees(longitude, latitude, 10000), orientation: { heading: Cesium.Math.toRadians(0.0), pitch: Cesium.Math.toRadians(-45.0), roll: 0.0 } }); // Create the water entity with water material const waterEntity = viewer.entities.add({ name: 'Dynamic Water', polygon: { hierarchy: Cesium.Cartesian3.fromDegreesArray([ longitude - 0.01, latitude - 0.01, longitude + 0.01, latitude - 0.01, longitude + 0.01, latitude + 0.01, longitude - 0.01, latitude + 0.01 ]), height: new Cesium.CallbackProperty(() => currentWaterHeight, false), extrudedHeight: new Cesium.CallbackProperty(() => currentWaterHeight, false), material: new Cesium.ImageMaterialProperty({ image: 'https://cesium.com/downloads/cesiumjs/releases/1.89/Build/Cesium/Assets/Textures/waterNormals.jpg', transparent: true, color: Cesium.Color.BLUE.withAlpha(0.5), repeat: new Cesium.Cartesian2(10, 10) }) } }); viewer.zoomTo(waterEntity); // Event listeners for buttons document.getElementById('raiseWater').addEventListener('click', () => { isRaising = !isRaising; if (isRaising) { isLowering = false; } }); document.getElementById('lowerWater').addEventListener('click', () => { isLowering = !isLowering; if (isLowering) { isRaising = false; } }); // Update loop viewer.clock.onTick.addEventListener(() => { const speed = parseFloat(document.getElementById('speedControl').value) || 1; targetWaterHeight = parseFloat(document.getElementById('heightControl').value) || maxHeight; if (isRaising && currentWaterHeight < targetWaterHeight) { currentWaterHeight = Math.min(currentWaterHeight + speed, targetWaterHeight); } else if (isLowering && currentWaterHeight > 0) { currentWaterHeight = Math.max(currentWaterHeight - speed, 0); } }); } // Get the user's current position if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(setViewToCurrentLocation, function (error) { console.error("Error getting location: ", error); alert("Unable to retrieve your location."); }); } else { alert("Geolocation is not supported by this browser."); } </script> </body> </html>

**说明：**

1. **初始化Cesium Viewer**：使用**Cesium.Viewer**初始化一个Cesium Viewer实例，并加载世界地形数据。
2. **获取用户当前位置**：
   * 使用浏览器的**navigator.geolocation.getCurrentPosition** API获取用户的当前位置。
   * 如果成功获取位置，调用**setViewToCurrentLocation**函数设置视角并添加水淹效果。
3. **创建悬浮窗和按钮**：
   * 使用HTML和CSS创建一个悬浮窗，其中包含两个按钮和两个输入框。
   * 一个输入框用于设置目标高度（单位：米），另一个输入框用于设置速度（单位：米/秒）。
4. **控制水位上升和下降**：
   * 在**setViewToCurrentLocation**函数中，根据用户的经纬度设置视角。
   * 使用**Cesium.CallbackProperty**来定义水的高度随时间变化。
   * 创建一个水实体，定义多边形的**height**和**extrudedHeight**属性，使水体成为一个立体对象，并添加水体材质。
   * 使用Cesium的**ImageMaterialProperty**来加载水面法线贴图，模拟真实水面效果。
5. **动态更新水位**：
   * 在**viewer.clock.onTick**回调函数中，根据速度和目标高度动态更新**currentWaterHeight**。
   * 添加按钮点击事件监听器，通过切换**isRaising**和**isLowering**变量来控制水位的上升和下降。
   * **extrudedHeight**属性设置为**currentWaterHeight**，这样水体的厚度直接等于当前的水位高度。

通过以上更新，现在水体的厚度将直接等于当前的水位高度，实现了立体的水体效果。