ARScene Documentation

Table of Contents

Introduction

The `ARScene` class provides a framework for building augmented reality (AR) scenes using the Three.js library. It enables the integration of 3D models into AR environments and includes functionality for user interaction and callbacks.

Constructor

javascript

```
const arScene = new ARScene(
  containerId,
  modelPath,
  cameraCallback,
  onlineCallback,
  commentCallback,
  saveCallback,
  streamCallback
);
```

- **containerId (string):** The ID of the HTML element that will contain the AR scene.
- **modelPath (string):** The file path or URL of the 3D model to be loaded into the AR scene.

- **cameraCallback (function):** Callback function for camera-related actions.
- **onlineCallback (function):** Callback function for online-related actions.
- **commentCallback (function):** Callback function for comment-related actions.
- **saveCallback (function):** Callback function for save-related actions.
- **streamCallback (function):** Callback function for stream-related actions.

Methods

setup

javascript
```
arScene.setup();
```

- Initializes the AR scene by calling the following setup methods:
  - `setupScene`: Creates a Three.js scene with a specified background color.
  - `setupRenderer`: Creates a WebGL renderer and attaches it to the specified HTML container.
  - `setupCamera`: Sets up a perspective camera.
  - `setupControls`: Initializes camera controls for AR and orbit controls.
  - `new LightingSetup`: Configures lighting in the scene.

setupScene

javascript
```
arScene.setupScene();
```

- Creates a new Three.js scene with a white background.

setupCamera

javascript
```
arScene.setupCamera();
```

- Sets up a perspective camera with default parameters.

setupRenderer

javascript
```
arScene.setupRenderer();
```

- Creates a WebGL renderer, attaches it to the specified container, and sets up XR capabilities.

setupControls

javascript
```
arScene.setupControls();
```

- Sets up camera controls using OrbitControls and adds an XR controller to the scene.

setupAnimator

javascript
```
arScene.setupAnimator();
```

- Creates an Animator instance for the loaded 3D model, associating it with the scene, camera, and renderer.

loadModel

javascript
```
arScene.loadModel(modelPath);
```

- Loads a 3D model from the specified path and adds it to the scene. Sets up animation and UI after the model is loaded.

setupUI

javascript
```
arScene.setupUI();
```

- Sets up the user interface, including the placement of icons and the integration of AR and VR buttons.

setCameraCallback

javascript
```
arScene.setCameraCallback(callback);
```

- Sets a callback function for camera-related actions.

setupEventListeners

javascript
```
arScene.setupEventListeners();
```

- Sets up event listeners for XR session start/end and window resize events.

animate

javascript
```
arScene.animate();
```

- Initiates the animation loop using the renderer's `setAnimationLoop` method.

render

javascript
```
arScene.render(timestamp, frame);
```

- Renders the scene, updating the 3D model's position based on hit test results and handling user interactions.

Animator Documentation

Table of Contents

Introduction

The `Animator` class facilitates animations and interactions with a 3D model in a Three.js scene. It includes methods for zooming, rotating, transitioning between standard views, and updating animations.

Constructor

javascript
```
const animator = new Animator(model, scene, camera, renderer);
```

- **model (THREE.Group):** The 3D model to be animated.
- **scene (THREE.Scene):** The Three.js scene containing the model.
- **camera (THREE.PerspectiveCamera):** The camera used to view the scene.
- **renderer (THREE.WebGLRenderer):** The WebGL renderer for rendering the scene.

Methods

zoom

javascript
```
animator.zoom();
```

- Initiates a zoom animation by scaling the model to its maximum scale or back to its original scale.

handleClick

javascript
```
animator.handleClick(event);
```

- Event handler for mouse clicks. Detects if the mouse click intersects with the model and triggers rotation if a intersection is found.

rotate

javascript
```
animator.rotate();
```

- Initiates a rotation animation by rotating the model around its Y-axis.

standardViews

javascript
```
animator.standardViews();
```

- Transitions the model through a sequence of standard views (front, top, side) with a smooth animation.

update

javascript
```
animator.update();
```

- Updates the TWEEN library, ensuring smooth animations over time. Should be called in the render loop.

IconManager Documentation

Table of Contents

Introduction

The `IconManager` class manages the user interface (UI) icons, bars, and their associated callbacks for an augmented reality (AR) scene. It includes methods for loading bars, setting callbacks, and handling various UI interactions.

Constructor

javascript
```
const iconManager = new IconManager(targetDiv, animator, scene);
```

- **targetDiv (string):** The ID of the HTML element that will contain the UI icons and bars.
- **animator (Animator):** An instance of the Animator class for managing animations.
- **scene (THREE.Scene):** The Three.js scene containing the AR elements.

Methods

loadBars

javascript
```
iconManager.loadBars();
```

- Creates and loads the bottom and top bars containing icons into the specified targetDiv.

setCallbackByNameTop

javascript
```
iconManager.setCallbackByNameTop(name, callback);
```

- Sets a callback function for a top bar icon based on its name.

setCallbackByNameBottom

javascript
```
iconManager.setCallbackByNameBottom(name, callback);
```

- Sets a callback function for a bottom bar icon based on its name.

test

javascript
```
iconManager.test();
```

- A test callback function that logs a message to the console.

fullscreen

javascript
```
iconManager.fullscreen();
```

- Toggles fullscreen mode for the specified targetDiv.

share

javascript
```
iconManager.share();
```

- Shares the current page using the Web Share API, if supported.

toggleFullscreen

javascript
```
iconManager.toggleFullscreen(targetDiv);
```

- Toggles fullscreen mode for the specified targetDiv.

requestFullscreen

javascript
```
iconManager.requestFullscreen(element);
```

- Requests fullscreen mode for the specified element.

exitFullscreen

javascript

```
iconManager.exitFullscreen();
```

- Exits fullscreen mode.