控制器（新版）

`v2.10.x` 版本之后发布了全新的控制器，增加了属性配置、方法、事件等。

并且重构了相机相关的所有方法。 `setCameraViewpoint` 、 `flyToObj` 等方法内部都是基于控制器的方法实现。

通过 next 标签安装

```sh
npm install soonspacejs@next
```

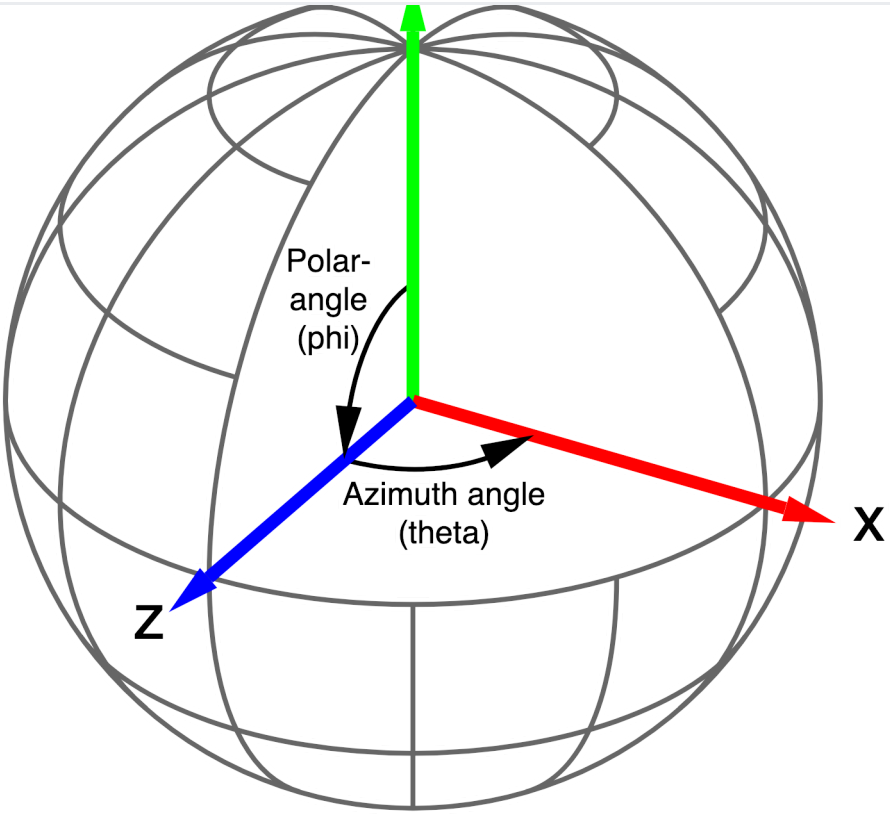
所有的属性读写、方法调用、事件监听都推荐以下方式。

```js
const { controls } = ssp;

// 修改属性
controls.enabled = true;
controls.minDistance = 3;

// 读取属性
controls.active;

// 方法调用
controls.rotateAzimuthTo(Math.PI / 2, true);

// 事件监听
controls.addEventListener('update', () => {
  console.log('控制器相机发生变化');
});
```

术语

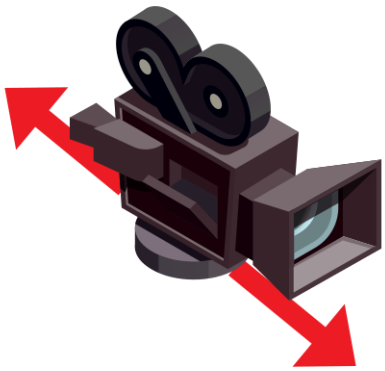
Orbit 旋转

控制器使用球坐标进行轨道旋转。

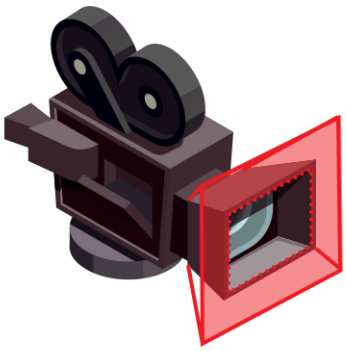


Dolly 和 Zoom

- Dolly 实际上是移动相机来改变每一帧中图像的组成（移动）。
- Zoom 包括改变镜头焦距。在 three.js 中，Zoom 实际上是改变相机的 FOV，而相机是静止的（不移动）。



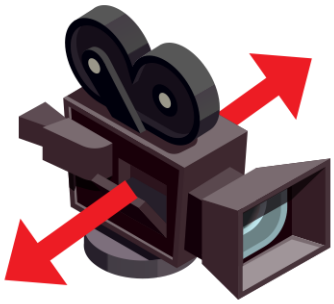
Dolly



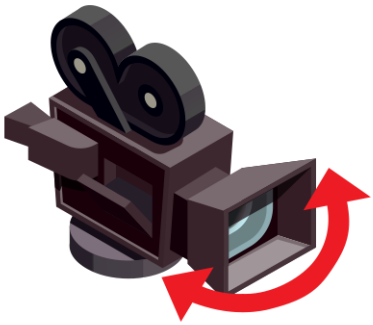
Zoom

Truck 和 Pan

- Truck 相机上下左右的平移操作。
- Pan 原地保持不动，只转动镜头。类似第一人称的操作。



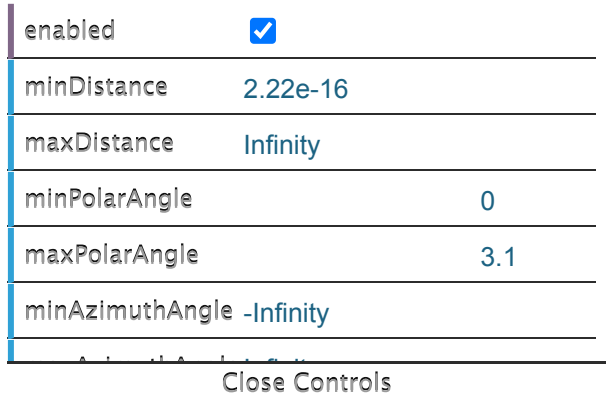
Truck



Pan

属性

样例:



objects 29
triangles 28734
vertices 86202
frametime 16.90 ms

属性	描述	类型	必
enabled	启用控制器	boolean	✗
camera	当前被控制的相机	透视或正交相机	✗
active	当前控制器是否是激活状态	boolean	✗
currentAction	当前 `ACTION`	ACTION	✗
distance	相机 `position` 与 `target` 的距离	number	✗
minDistance	Dolly 的最小距离，这个值必须大于0	number	✗
maxDistance	Dolly 的最大距离	number	✗
minZoom	Zoom 的最小值	number	✗
maxZoom	Zoom 的最大值	number	✗
polarAngle	当前的极角弧度	number	✗
minPolarAngle	最小极角弧度	number	✗

maxPolarAngle	最大极角弧度	number	✗
azimuthAngle	当前的方位角弧度	number	✗
minAzimuthAngle	最小方位角弧度	number	✗
maxAzimuthAngle	最大方位角弧度	number	✗
boundaryFriction	边界摩擦比	number	✗
boundaryEnclosesCamera	相机位置是否应该被封闭在边界内	boolean	✗
smoothTime	到达目标的过渡时间，以秒为单位。 数值越小，到达目标的速度越快	number	✗
draggingSmoothTime	操控时的过渡时间	number	✗
maxSpeed	最大速度	number	✗
azimuthRotateSpeed	方位角旋转速度	number	✗
polarRotateSpeed	极角旋转速度	number	✗
dollySpeed	鼠标滚轮时的相机移动速度	number	✗
truckSpeed	平移速度	number	✗
verticalDragToForward	拖拽时是否前后移动，默认为上下	boolean	✗
dollyToCursor	是否以光标点为 Dolly 目标	boolean	✗
dollyDragInverted	当通过拖动触发 Dolly 和 Zoom 时反转方向	boolean	✗
interactiveArea	在domElement中设置拖拽、 触摸和滚轮启用区域。 每个值都在0到1之间， 其中0表示屏幕的左/上，1表示屏幕的右/ 下	DOMRect	✗
colliderMeshes	与相机碰撞的场景对象	Object3D[]	✗
infinityDolly	启用无限 Dolly。 与minDistance和maxDistance一起使用	boolean	✗
restThreshold	相机减速时rest事件触发的速度	number	✗

```
import { MathUtils } from 'three';

// 角度转弧度
360 * MathUtils.DEG2RAD;

// 弧度转弧度
Math.PI * MathUtils.RAD2DEG;
```

- 每当 360 度的旋转被添加到 `azimuthAngle` 时，这是累积的。 `360º = 360 * MathUtils.DEG2RAD = Math.PI * 2` ， `720º = Math.PI * 4` 。

```
// 将 azimuthAngle 限制到 0 - Math.PI * 2 之间
controls.normalizeRotations();

console.log(controls.azimuthAngle);
```

- 注意 `colliderMeshes` 可能会降低性能。碰撞测试使用来自相机的 4 个光线投射器，因为近平面有 4 个角。
- 如果 Dolly 的距离小于（或大于）minDistance （或 maxDistance ），则 infinityDolly 将保持距离并推动目标（target）位置。

事件

`controls` 发出以下事件。

订阅方式：通过 `controls.addEventListener( 'eventname', function )`

如需取消订阅：请使用 `controls.removeEventListener( 'eventname', function )`

事件名称	触发时机
`'controlstart'`	当用户开始通过鼠标或手指操作时
`'control'`	当用户正在操作时
`'controlend'`	用户结束操作时
`'transitionstart'`	任意过渡开始时，用户操作或方法调用 `enableTransition = true`

`'update'`	当相机 `position` 发生变化
`'wake'`	当相机开始移动时
`'rest'`	当相机将要停止时 由 `restThreshold` 控制
`'sleep'`	当相机结束移动

- `mouseButtons.wheel` (鼠标滚轮控制)不发出 `controlstart` 和 `controlend` 。 `mouseButtons.wheel` 在内部使用滚动事件，并且滚动事件间歇性地发生。这意味着无法检测到 start 和 end。
- 由于阻尼， `sleep` 通常会在相机看起来已经停止移动的几秒钟后触发。如果你想在相机停止的时候做一些事情(例如，启用 UI，执行另一个过渡)，你可能想要 `rest` 事件。这可以使用 `restthreshold` 参数进行微调。

用户操作配置

`ACTION` 常量请从 `SoonSpace` 中获取

```js
import SoonSpace from 'soonspacejs';

const { ACTION } = SoonSpace;

controls.mouseButtons.left = ACTION.ROTATE;
controls.mouseButtons.right = ACTION.TRUCK;

controls.touches.one = ACTION.TOUCH_ROTATE;
controls.touches.two = ACTION.TOUCH_DOLLY_TRUCK;
```

表格中 * 表示默认值

鼠标键位	行为
`mouseButtons.left`	`ACTION.ROTATE` *| `ACTION.TRUCK` | `ACTION.OFFSET` | `ACTION.DOLLY` | `ACTION.ZOOM` | `ACTION.NONE`
`mouseButtons.right`	`ACTION.ROTATE` | `ACTION.TRUCK` *| `ACTION.OFFSET` | `ACTION.DOLLY` | `ACTION.ZOOM` | `ACTION.NONE`

mouseButtons.wheel	ACTION.ROTATE | ACTION.TRUCK | ACTION.OFFSET | ACTION.DOLLY *| ACTION.ZOOM | ACTION.NONE
mouseButtons.middle	ACTION.ROTATE | ACTION.TRUCK | ACTION.OFFSET | ACTION.DOLLY *| ACTION.ZOOM | ACTION.NONE

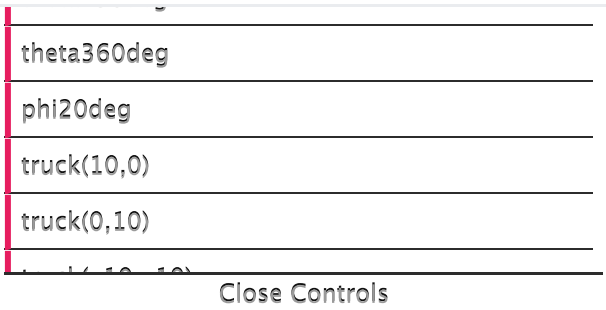
- `mouseButtons.wheel` 的默认行为是：
  - `DOLLY` 用于 `PerspectiveCamera`
  - `ZOOM` 用于 `OrthographicCamera`，并且无法设置 `DOLLY`.

触控操作	行为
touches.one	ACTION.TOUCH ROTATE *| ACTION.TOUCH_TRUCK | ACTION.TOUCH_OFFSET | ACTION.DOLLY
touches.two	ACTION.TOUCH DOLLY TRUCK | ACTION.TOUCH DOLLY OFFSET | ACTION.TOUCH DOLLY ROTATE | ACTION.TOUCH ZOOM TRUCK | ACTION.TOUCH ZOOM OFFSET | ACTION.TOUCH_ZOOM_ROTATE | ACTION.TOUCH DOLLY | ACTION.TOUCH ZOOM | ACTION.TOUCH ROTATE | ACTION.TOUCH_TRUCK | ACTION.TOUCH_OFFSET | ACTION.NONE
touches.three	ACTION.TOUCH DOLLY TRUCK | ACTION.TOUCH DOLLY OFFSET | ACTION.TOUCH DOLLY ROTATE | ACTION.TOUCH ZOOM TRUCK | ACTION.TOUCH ZOOM OFFSET | ACTION.TOUCH_ZOOM_ROTATE | ACTION.TOUCH ROTATE | ACTION.TOUCH_TRUCK | ACTION.TOUCH_OFFSET | ACTION.NONE

- `touches.two` 和 `touches.three` 的默认行为分别是：
  - `TOUCH_DOLLY_TRUCK` 用于 `PerspectiveCamera`
  - `TOUCH_ZOOM_TRUCK` 用于 `OrthographicCamera`，并且无法设置 `TOUCH_DOLLY_TRUCK` 和 `TOUCH_DOLLY`

方法

样例:



objects 29
triangles 28734
vertices 86202
frametime 20.00 ms

`enableTransition = true` 时，调整 `controls.smoothTime` 控制过渡时间

rotate

旋转方位角(水平)和极角(垂直)。每个值都被添加到当前值中。

定义

```ts
function rotate(azimuthAngle: number, polarAngle: number, enableTransition?:
```

属性	描述	类型	必填	默认值
azimuthAngle	方位角旋转（弧度）	number	✓	
polarAngle	极角旋转（弧度）	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

如果只是要旋转其中一个轴，只需将另一个参数设置为 `0`

rotateAzimuthTo

将方位角(水平)旋转到给定角度，并保持的极角(垂直)不变。

定义

```ts
function rotateAzimuthTo(azimuthAngle: number, enableTransition?: boolean): P
```

属性	描述	类型	必填	默认值
azimuthAngle	方位角旋转（弧度）	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

rotatePolarTo

将极角(垂直)旋转到给定角度，并保持的方位角(水平)不变。

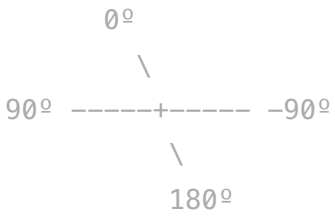
定义

```ts
function rotatePolarTo(polarAngle: number, enableTransition?: boolean): Promi
```

属性	描述	类型	必填	默认值
polarAngle	极角旋转（弧度）	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

rotateTo

将极角(垂直)和方位角(水平)旋转到给定角度



0° 表示朝向前方, 90° (`Math.PI / 2`) 表示朝向左边, -90° (`- Math.PI / 2`) 朝向右边, 180° (`Math.PI`) 朝向背面

极角

180° (`Math.PI`) 表示朝向天空/天花板, 90° (`Math.PI / 2`) 是水平, 0° 朝向地面/地板

定义

```ts
function rotateTo(azimuthAngle: number, polarAngle: number, enableTransition?
```

属性	描述	类型	必填	默认值
azimuthAngle	方位角旋转（弧度）	number	✓	
polarAngle	极角旋转（弧度）	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

dolly

将相机拉进或拉远

负值将拉远

定义

```ts
function dolly(distance: number, enableTransition?: boolean): Promise<void>;
```

属性	描述	类型	必填	默认值
distance	拉进（远）的距离	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

dollyTo

将相机拉进或拉远到给定的距离

定义

```ts
function dollyTo(distance: number, enableTransition?: boolean): Promise<void>
```

属性	描述	类型	必填	默认值
distance	给定的拉进（远）的距离	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

dollyInFixed

将相机拉进或拉远，但不改变目标（target）和相机之间的距离，而是移动目标（target）的位置。

样例

SoonSpace.js 2.x

dolly tocursor
infinitydolly
distance: min: 5, max: 5
distance: min: 3, max: 7
distance: min: 1, max: 10
dolly 1 dolly -1
dollyInFixed 1 dollyInFixed -1
reset

objects 31
triangles 30718
vertices 87291
frametime 16.60 ms

定义

```ts
function dollyInFixed(distance: number, enableTransition?: boolean): Promise<
```

属性	描述	类型	必填	默认值
distance	拉进（远）的距离（target）	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

zoom

相机变焦，该值添加到相机 zoom 属性上。

定义

```ts
function zoom(zoomStep: number, enableTransition?: boolean): Promise<void>;
```

zoomStep	变焦比例	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

你可以直接读取相机的 `zoom` 属性实现 in/out

```js
const zoomIn = () => controls.zoom(controls.camera.zoom / 2, true);
const zoomOut = () => controls.zoom(-controls.camera.zoom / 2, true);
```

zoomTo

相机变焦，该值会覆盖相机 `zoom` 属性。

通过设置 `minZoom` 和 `maxZoom` 限制

定义

```ts
function zoomTo(zoomStep: number, enableTransition?: boolean): Promise<void>;
```

属性	描述	类型	必填	默认值
zoomStep	变焦比例	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

truck(x, y, enableTransition)

基于当前方位角平移

定义

```ts
function truck(x: number, y: number, enableTransition?: boolean): Promise<voi
```

x	水平偏移量	number	✓	
y	垂直偏移量	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

lookInDirectionOf

看向给定点的方向

只改变相机的 `position`

定义

```ts
function lookInDirectionOf(x: number, y: number, z: number, enableTransition?
```

属性	描述	类型	必填	默认值
x	x 位置	number	✓	
y	y 位置	number	✓	
z	z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setFocalOffset

使用屏幕平行坐标设置焦点偏移

相机的旋转中心点不变

定义

```ts
function setFocalOffset(x: number, y: number, z: number, enableTransition?: b
```

x	x 位置	number	✓	
y	y 位置	number	✓	
z	z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setOrbitPoint

设置旋转中心点

相机的位置和朝向不变，一般配合 setFocalOffset 方法使用

定义

```ts
function setOrbitPoint(targetX: number, targetY: number, targetZ: number, ena
```

属性	描述	类型	必填	默认值
targetX	旋转 x 位置	number	✓	
targetY	旋转 y 位置	number	✓	
targetZ	旋转 z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

forward

向前或向后移动

定义

```ts
function forward(distance: number, enableTransition?: boolean): Promise<void>
```

distance	移动距离	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

moveTo

移动 `target` 到给定的点， `position` 到 `target` 的距离不会改变

相机的 `position` 和 朝向都会改变

定义

```ts
function moveTo(x: number, y: number, z: number, enableTransition?: boolean):
```

属性	描述	类型	必填	默认值
x	x 位置	number	✓	
y	y 位置	number	✓	
z	z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

elevate(height, enableTransition)

上下移动

定义

```ts
function elevate(height: number, enableTransition?: boolean): Promise<void>;
```

height	移动距离	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

fitToBox

使用相机最近的轴将视口与对象的包围盒或 `Box3` 对齐

flyToObj 方法基于此方法实现

定义

```ts
interface FitToOptions {
  cover: boolean;
  paddingLeft: number;
  paddingRight: number;
  paddingBottom: number;
  paddingTop: number;
}

function fitToBox(box3OrObject: Box3 | Object3D, enableTransition: boolean, {
```

属性	描述	类型	必填	默认值
box3OrObject	场景对象或 Box3	Box3 | Object3D	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false
options	选项	boolean	✗	{}

options

属性	描述	类型	必填	默认值
cover	是否填满屏幕	boolean	✗	false
paddingLeft	左侧填充距离	number	✗	0

paddingRight	右侧填充距离	number	✗	0
paddingBottom	底部填充距离	number	✗	0
paddingTop	顶部填充距离	number	✗	0

fitToSphere

将视口与对象包围球匹配

定义

```ts
function fitToSphere(sphereOrMesh: Sphere | Object3D, enableTransition: boole
```

属性	描述	类型	必填	默认值
sphereOrMesh	场景对象或 Sphere	Sphere | Object3D	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setLookAt

从 position 看向 target

setCameraViewpoint 方法基于此方法实现

定义

```ts
function setLookAt(positionX: number, positionY: number, positionZ: number, t
```

positionX	x 位置	number	✓	
positionY	x 位置	number	✓	
positionZ	x 位置	number	✓	
targetX	朝向 x 位置	number	✓	
targetY	朝向 y 位置	number	✓	
targetZ	朝向 z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

lerpLookAt

类似于 setLookAt，不过是基于两个状态之间插值

定义

```ts
function lerpLookAt(
  positionAX: number,
  positionAY: number,
  positionAZ: number,
  targetAX: number,
  targetAY: number,
  targetAZ: number,
  positionBX: number,
  positionBY: number,
  positionBZ: number,
  targetBX: number,
  targetBY: number,
  targetBZ: number,
  t: number,
  enableTransition?: boolean
): Promise<void>;
```

属性	描述	类型	必填	默认值
positionAX	原始 x 位置	number	✓	

positionAY	原始 x 位置	number	✓	
positionAZ	原始 x 位置	number	✓	
targetAX	原始朝向 x 位置	number	✓	
targetAY	原始朝向 y 位置	number	✓	
targetAZ	原始朝向 z 位置	number	✓	
positionBX	插值 x 位置	number	✓	
positionBY	插值 x 位置	number	✓	
positionBZ	插值 x 位置	number	✓	
targetBX	插值朝向 x 位置	number	✓	
targetBY	插值朝向 y 位置	number	✓	
targetBZ	插值朝向 z 位置	number	✓	
t	插值系数，必须在 0 - 1 之间	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setPosition

设置相机 `position` ，但会保持相机仍然看向 `target`

相机的位置和朝向都会改变

类似于 setLookAt，但是 `target` 保留

定义

```ts
function setPosition(positionX: number, positionY: number, positionZ: number,
```

positionX	x 位置	number	✓	
positionY	x 位置	number	✓	
positionZ	x 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setTarget

设置 `target`

相机的位置不会改变，会朝向新的目标点

类似于 setLookAt，但是 `position` 保留

定义

```ts
function setTarget(targetX: number, targetY: number, targetZ: number, enableT
```

属性	描述	类型	必填	默认值
targetX	朝向 x 位置	number	✓	
targetY	朝向 y 位置	number	✓	
targetZ	朝向 z 位置	number	✓	
enableTransition	是否开启平滑过渡	boolean	✗	false

setBoundary

设置摄像机 `target` 的边界框， `target` 无法超出这个边界

定义

属性	描述	类型	必填	默认值
box3	限制区域的 box3	Box3	✗	

- box3 可以通过以下方法获取

```js
// 或者手动创建
import { Box3 } from 'three';
const box3 = new Box3();
// 获取3d对象的box3
const objectBox3 = ssp.utils.getBoundingBox(object);
```

setViewport

设置画面的展示区域

定义

```ts
function setBoundary(x: number, y: number, width: number, height: number): vo
```

属性	描述	类型	必填	默认值
x	视口的最左边位置	number	✓	
y	视口的最底部位置	number	✓	
width	视口的宽度	number	✓	
height	视口的高度	number	✓	

getTarget

返回当前 target

getCameraViewpoint 方法基于此方法实现

定义

```ts
function getTarget(out?: Vector3, receiveEndValue?: boolean): Vector3;
```

属性	描述	类型	必填	默认值
out	接收数据的 Vector3	number	✗	
receiveEndValue	是否获取过渡结束的值	number	✗	true

getPosition

返回当前 `position`

getCameraViewpoint 方法基于此方法实现

定义

```ts
function getPosition(out?: Vector3, receiveEndValue?: boolean): Vector3;
```

属性	描述	类型	必填	默认值
out	接收数据的 Vector3	number	✗	
receiveEndValue	是否获取过渡结束的值	number	✗	true

getSpherical(out, receiveEndValue)

返回轨道的球坐标。

定义

```ts
function getSpherical(out?: Spherical, receiveEndValue?: boolean): Spherical;
```

out	接收数据的 Spherical	number	✗	
receiveEndValue	是否获取过渡结束的值	number	✗	true

getFocalOffset(out, receiveEndValue)

返回焦点偏移量，这是相机在屏幕平行坐标中平移的程度。

定义

```ts
function getFocalOffset(out?: Vector3, receiveEndValue?: boolean): Vector3;
```

属性	描述	类型	必填	默认值
out	接收数据的 Vector3	number	✗	
receiveEndValue	是否获取过渡结束的值	number	✗	true

saveState()

保存当前状态

定义

```ts
function saveState(): void;
```

normalizeRotations()

格式化方位角旋转角度到 0° - 360° 之间

定义

reset

重置回默认状态，可以配合 saveState 使用

定义

```ts
function reset(enableTransition?: boolean): Promise<void[]>;
```

属性	描述	类型	必填	默认值
enableTransition	是否开启平滑过渡	boolean	✗	false

addEventListener(type: string, listener: function)

添加一个事件监听

```js
const updateHandler = (event) => {
  event.type; // update
};

// 添加一个事件监听
controls.addEventListener('update', updateHandler);
// 移除一个事件监听
controls.removeEventListener('update', updateHandler);
// 移除该类型所有的事件监听
controls.removeAllEventListeners('update');
```

removeEventListener(type: string, listener: function)

移除一个事件监听

removeAllEventListeners(type: string)

← 控制器（废弃）

模型操作 →