


plugin-navigation

导航插件。

导航插件拥有三个部分的内容，分别为

- 导航相机(多视角跟踪相机) 
- 导航地图（将来）
- 导航控制器（将来）

安装

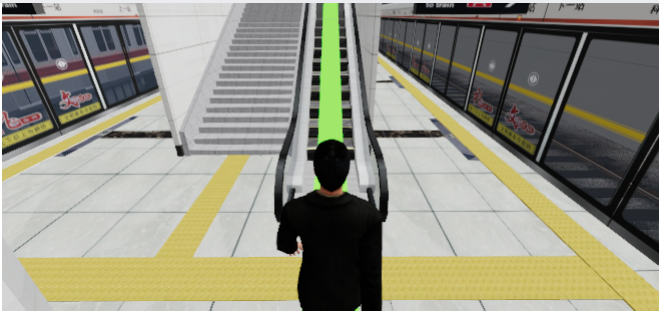
```
npm install @soonspacejs/plugin-navigation@next -S  
# or  
yarn add @soonspacejs/plugin-navigation@next -S
```

sh

导航相机

样例

☰ SoonSpace.js 2.x



视角深度	80
小地图正交相机	
切换相机	
地图缩放	10
Close Controls	

objects 1003
triangles 204002
vertices 342123
frametime 1.70 ms



使用方法

```
import SoonSpace from "soonspacejs";
import { NavigateCamera } from "@soonspacejs/plugin-navigation";

const ssp = new SoonSpace({
  el: "#view",
  options: {},
  event: {},
});

const navigateCamera = new NavigateCamera(ssp);

navigateCamera.active();
```

ts

NavigateCamera(ssp, camera?, options?, controls?)

- camera** 可以在初始化时传入，若不传入，在该类内部会自动创建一个默认的 透视投影相机(Perspective Camera)
- options** 为初始化设置参数，当调用 **resetOptions** 时会使用默认参数以及 **options** 参数合并后的值。

参数

NavigateCameraOptions 设置参数

属性	描述	类型
<code>disabledAnimate</code>	相机复原动画是否开启	<code>boolean</code>
<code>fixedOrientation</code>	是否锁定朝向（相当于禁用控制器操作）	<code>boolean</code>
<code>autoRestoreOrientation</code>	在操作后是否自动复原相机位置，传入 <code>number</code> 值时为定时复原，传入 <code>boolean</code> 值则表示立即复原或者不复原	<code>boolean</code>
<code>oppositeCamera</code>	是否反转相机	<code>boolean</code>
<code>oppositeType</code>	相机反转类型，支持不同轴或者平面反转，默认为基于 <code>y</code> 轴反转	<code>{ x: boolean; z: boolean }</code>
<code>vision</code>	视角设置，支持第一人称、第三人称、俯视角和左视角配置	<code>NAVIGATE_</code> <code>.FIRST_VI</code> <code>.THIRD_VI</code> <code>.UP_VISIO</code> <code>.LEFT_VIS</code>
<code>orientationTarget</code>	朝向目标设置，当设置为 <code>main</code> 时，相机跟踪当前跟踪对象的朝向，其他值类型还有 <code>Vector3</code> (固定朝向某个点)、 <code>Object3D</code> (固定朝向某个模型对象)、 <code>Euler</code> (固定朝向某个方向)	<code>main</code> <code>Ve</code> <code>Euler</code>
<code>orientationType</code>	朝向方式设置，支持相对朝向、绝对朝向和陀螺仪朝向，设置为相对朝向时，相机将跟随目标旋转	<code>NAVIGATE_</code> <code>.RELATIVE</code> 相对朝向 <code>.FIXED_OR</code> 绝对朝向 <code>.GYRO_ORI</code> 陀螺仪朝向
<code>distanceToTarget</code>	相机与跟踪目标之间距离	<code>number</code>
<code>rotationToTarget</code>	相机与跟踪目标之间的夹角向量，当设置为非第三人称时，该参数无效	<code>Vector3</code>

☰ SoonSpace.js 2.x

targetRotationFix	相机角度修正，支持动态传入方法返回	[number, EulerOrde [number, EulerOrde
isFixRotationRelativeTarget	是否相对跟踪对象进行角度修正， 设置为是时相机以跟踪目标朝向修正， 设置为否时相机以自身朝向修正	boolean
targetPositionFix	相机位置修正，支持动态传入方法返回	[number, () => [number]
isFixPositionRelativeTarget	是否相对跟踪对象进行位置修正， 设置为是时相机以跟踪对象为原点修正， 设置为否时相机自身为原点修正	boolean
enableGyro	是否启用陀螺仪	boolean
gyroX	是否应用陀螺仪 X 轴变化	boolean
gyroY	是否应用陀螺仪 Y 轴变化	boolean
gyroZ	是否应用陀螺仪 Z 轴变化	boolean
gyroAbsolute	陀螺仪变化时使用相对值还是绝对值， 使用绝对值时会以陀螺仪本身的方位为基准， 使用相对值时则以开启陀螺仪时的方位为基准	boolean
onControlStart	当控制器控制相机时允许外部自定义控制相机， 控制开始时触发	null ((followTar
onControlRender	当控制器控制相机时允许外部自定义控制相机， 控制进行时回调	null ((followTar
onControlEnd	当控制器控制相机时允许外部自定义控制相机， 控制结束时回调	null ((followTar

TS 类型

```
interface NavigateCameraOptions {  
  disabledAnimate: boolean;  
  fixedOrientation: boolean;  
  autoRestoreOrientation: boolean | number;  
  oppositeCamera: boolean;  
}
```

☰ SoonSpace.js 2.x

```
VISION: NAVIGATE_VISION_TYPE;  
orientationType: NAVIGATE_ORIENTATION_TYPE;  
distanceToTarget: number;  
rotationToTarget: Vector3;  
isFixRotationRelativeTarget: boolean;  
targetRotationFix: FixEuler;  
isFixPositionRelativeTarget: boolean;  
targetPositionFix: FixVector;  
enableGyro: boolean;  
gyroX: boolean;  
gyroY: boolean;  
gyroZ: boolean;  
gyroAbsolute: boolean;  
onControlStart: null | ControlCallback;  
onControlRender: null | ControlCallback;  
onControlEnd: null | ControlCallback;  
}
```

方法

active () {}

启用导航相机

resetOptions () {}

重置相机参数，初始化实例时，可以传入初始配置值

setOptions (options: Partial<NavigateCameraOptions>) {}: 设置导航相机配置

用此方法设置参数时，不会产生相机复原动画。

setCamera (camera: Camera) {}: 设置相机

当导航相机正在使用时，会立即切换到传入的相机，不需要再调用 **active** 方法启用相机

restoreOrientation() {}: 主动发起复原相机位置

根据 **disabledAnimate** 是否带有动画恢复

☰ SoonSpace.js 2.x

俯视图正交相机

导航相机库里提供基于导航相机封装的俯视图正交相机工具，用于实现展示俯视地图全览的相关需求

使用方法

```
import SoonSpace from "soonspacejs";
import { NavigateCamera, MapCamera } from "@soonspacejs/plugin-navigation";

const ssp = new SoonSpace({
  el: "#view",
  options: {},
  event: {},
});

const navigateCamera = new NavigateCamera(ssp);

const originCamera = navigateCamera.nativeCamera;

const mapCamera = new MapCamera(ssp);

const minimapCamera = mapCamera.nativeCamera;

function setMapCamera() {
  navigateCamera.setOptions({
    vision: NAVIGATE_VISION_TYPE.UP_VISION,
  });
  navigateCamera.setCamera(minimapCamera);
}
```

额外参数

- `MapCamera.zoom` 调整相机地图缩放大小

