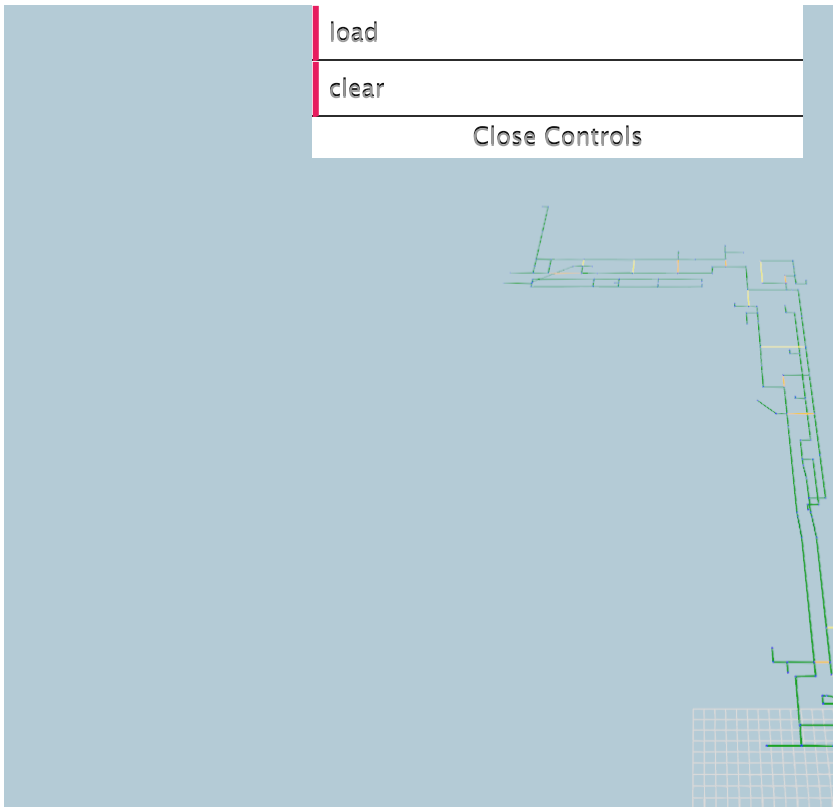
拓扑路径

样例：



createTopology

创建 `topology` 对象。

定义：

```ts
interface TopologyEffectInfo {
  //
  renderLink?: boolean;
  linkWidth?: number;
  linkColor?: IColor | IColor[];
  //
  renderNode?: boolean;
  nodeColor?: IColor;
  nodeRadius?: number;
```

```
  animation?: LinkInfo['animation'];
}

type TopologyType = 'line' | 'network';

interface TopologyNodeInfo {
  id: BaseObject3DInfo['id'];
  name?: BaseObject3DInfo['name'];
  position: Position;
  graphs?: TopologyNodeGraph[];
}

interface TopologyNodeGraph {
  targetNodeId: string;
  linkInfo: {
    id: string;
    name?: string;
  };
  passable: number;
}

interface TopologyInfo extends BaseObject3DInfo, TopologyEffectInfo {
  type: TopologyType;
  nodes: TopologyNodeInfo[];
}

function createTopology(topologyInfo: TopologyInfo): Topology;
```

用法：

```
                                                                                                                    js
const topology = ssp.createTopology({
  id: 'topology_1',
  name: 'topology_1_name',
  type: 'line',
  nodes: [
    {
      id: 'node1',
      position: { x: 0, y: 1, z: 0 },
    },
    {
      id: 'node2',
```

```
      {
        id: 'node3',
        position: { x: 100, y: 1, z: 100 },
      },
      {
        id: 'node4',
        position: { x: 100, y: 1, z: -100 },
      },
    ],
    renderNode: true,
  });
```

参数：

topologyInfo

- **描述:** 实例路径对象所需信息
- **类型:** TopologyInfo
- **必填:** ✓

TopologyInfo

属性	描述	类型	必填	默认值
id	唯一ID	string | number	✓	
name	名称	string	✗	
nodes	节点坐标集合	TopologyNodeInfo[]	✓	
type	路径类型	line | network	✓	
renderLink	是否渲染连接线	boolean	✗	true
linkWidth	线宽	number	✗	20
linkColor	连接线颜色	IColor | IColor[]	✗	0x00ff00
renderNode	是否渲染节点	boolean	✗	true
nodeColor	节点颜色	IColor	✗	0x0000ff

nodeRadius	节点半径	numbers	✗	10
imgUrl	非纯色线时使用的图片资源路径	string	✗	null
animation	非纯色线时的流动动画	boolean | AnimationOptions	✗	false
level	显示层级范围	Level	✗	{ max: null, min: null }
visible	是否可见	boolean	✗	true
position	位置坐标	Position	✗	{ x: 0, y: 0, z: 0 }
rotation	旋转弧度	Rotation	✗	{ x: 0, y: 0, z: 0 }
scale	缩放比例	Scale	✗	{ x: 1, y: 1, z: 1 }
userData	用户数据	any	✗	{}

`linkColor` 为数组时，有效长度是 4 个，分别对应 `passable` 的四个状态时路径颜色；为单个颜色时，表示所有路径颜色。

TopologyNodeInfo

属性	描述	类型	必填	默认值
id	节点唯一ID	string | number	✓	
name	节点名称	string	✗	
position	节点坐标	Position	✓	
graphs	网结构信息	TopologyNodeGraph[]	✗	

TopologyNodeGraph

属性	描述	类型	必填	默认值
targetNodeId	目标 node ID	string | number	✓	
linkInfo	路径信息	{ id: string, name?: string }	✓	
passable	路径通行许可	0 | 1 | 2 | 3	✓	

passable 的枚举含义分别为：双向通行（0） | 单向正向通行（1） | 单向反向通行（2） | 禁止通行（3）

> **注意**
>
> 此处的 passable 数据是配合 linkColor 使用，直接设置无法影响 getShortestPath 等方法的结果
>
> 需要动态设置链路的通行属性请使用 setTopologyPassable 方法

setTopologyPassable

样例

定义：

```ts
interface TopologyPassableInfo {
  sourceNodeId: BaseObject3DInfo['id'];
  targetNodeId: BaseObject3DInfo['id'];
  passable: number;
}

function setTopologyPassable(
  topology: Topology,
  info: TopologyPassableInfo[]
): void;
```

用法：

```js
ssp.setTopologyPassable(topology, [
  {
    sourceNodeId: '8NM2FFLB40ZD',
    targetNodeId: '8NM2Z1GHW1OK',
    /**
     * 禁止通行，当使用 getShortestPath 等方法时会避开此链路
     */
```

```
]);
```

参数：

topology

- **描述:** 拓扑路径对象
- **类型:** `Topology`
- **必填:** ✓

info

- **描述:** 最短路径信息
- **类型:** `TopologyPassableInfo[]`
- **必填:** ✓

属性	描述	类型	必填	默认值
sourceNodeId	原始节点id	string | number	✓	
targetNodeId	目标节点id	string | number	✓	
passable	路径通行许可	0 | 1 | 2 | 3	✓	

`passable` 的枚举含义分别为：双向通行（0） | 单向正向通行（1） | 单向反向通行（2） | 禁止通行（3）

> **提示**
>
> 如果需要获取拓扑路径中的 node 对象时，可临时绑定一个点击事件

```js
ssp.signals.click.add((event) => {
  const [intersect] = ssp.viewport.getIntersects(event, topology.nodes

  if (intersect) {
    console.log('node对象 id', intersect.object.sid);
```

getShortestPath

获取最短路径

定义：

```ts
interface ShortestPathInfo extends BaseObject3DInfo, TopologyEffectInfo {
  start: Position;
  end: Position;
}

function getShortestPath(
  topology: Topology,
  info: ShortestPathInfo
): Topology | null;
```

用法：

```js
const shortestTopology = ssp.getShortestPath(topologyFromOther, {
  start: { x: 0, y: 0, z: 0 },
  end: { x: 100, y: 0, z: 300 },
  id: 'shortestPath',
  linkColor: 'red',
  nodeColor: 'orange',
  imgUrl: '../../asstes/img/topology/arrow.png',
  animation: true,
});
```

参数：

topology

- **必填:** ✓

info

- **描述:** 最短路径信息
- **类型:** `ShortestPathInfo`
- **必填:** ✓

属性	描述	类型	必填	默认值
`start`	路径的起始点（世界位置）	`Position`	✓	
`end`	路径的结束点（世界位置）	`Position`	✓	

部分配置参考 TopologyInfo

getShortestPathByMultipleStartPoints

通过指定 多个起点 和 一个终点，并计算每个起点 与 终点 间的最短路径，然后再从这些最短路径中 找出最短的那条 作为最终的路径 并 返回。

定义:

```ts
interface ShortestPathByMultipleStartPoints
  extends BaseObject3DInfo,
    TopologyEffectInfo {
  start: Position[];
  end: Position;
}

function getShortestPathByMultipleStartPoints(
  topology: Topology,
  info: ShortestPathByMultipleStartPoints
): Topology | null;
```

用法:

```
    {
      start: [
        { x: 0, y: 0, z: 0 },
        { x: 20, y: 0, z: 0 },
      ],
      end: { x: 100, y: 0, z: 300 },
      id: 'shortestPath',
      linkColor: 'red',
      nodeColor: 'orange',
      imgUrl: '../../asstes/img/topology/arrow.png',
      animation: true,
    }
  );
```

参数：

topology

- **描述:** 拓扑路径对象
- **类型:** `Topology`
- **必填:** ✓

info

- **描述:** 多起点最短路径信息
- **类型:** `ShortestPathByMultipleStartPoints`
- **必填:** ✓

属性	描述	类型	必填	默认值
start	路径的起始点（世界位置）	Position[]	✓	
end	路径的结束点（世界位置）	Position	✓	

部分配置参考 TopologyInfo

getShortestPathByMultipleEndPoints

SoonSpace.js 2.x

定义：

```ts
interface ShortestPathByMultipleEndPoints
  extends BaseObject3DInfo,
    TopologyEffectInfo {
  start: Position;
  end: Position[];
}

function getShortestPathByMultipleEndPoints(
  topology: Topology,
  info: ShortestPathByMultipleEndPoints
): Topology | null;
```

用法：

```js
const shortestTopology = ssp.getShortestPathByMultipleEndPoints(
  topologyFromOther,
  {
    start: { x: 0, y: 0, z: 0 },
    end: [
      { x: 100, y: 0, z: 300 },
      { x: 200, y: 0, z: 400 },
    ],
    id: 'shortestPath',
    linkColor: 'red',
    nodeColor: 'orange',
    imgUrl: '../../asstes/img/topology/arrow.png',
    animation: true,
  }
);
```

参数：

topology

- **必填:** ✓

info

- **描述:** 多终点最短路径信息
- **类型:** `ShortestPathByMultipleEndPoints`
- **必填:** ✓

属性	描述	类型	必填	默认值
start	路径的起始点（世界位置）	Position	✓	
end	路径的结束点（世界位置）	Position[]	✓	

部分配置参考 TopologyInfo

getTopologyById 👎

通过 `id` 查找

定义:

```ts
function getTopologyById(id: TopologyInfo['id']): Topology | null;
```

用法:

```js
const topology = ssp.getTopologyById('xxx');
```

> **弃用警告**
>
> 请使用 `getObjectById` 替代

getTopologyByName 👎

通过 `name` 查找

定义:

```ts
function getTopologyByName(name: string): Topology[];
```

用法:

```js
const topologyList = ssp.getTopologyByName('xxx');
```

> **弃用警告**
>
> 请使用 `getObjectByName` 替代

getAllTopology

获取所有 `Topology` 对象

定义:

```ts
function getAllTopology(): Topology[];
```

用法:

getTopologyByUserDataProperty

通过 `userData` 属性查找

定义：

```ts
function getTopologyByUserDataProperty(
  propNameOrFindFunc: string | UserDataPropertyFindFunc,
  value?: any
): Topology[];
```

用法：

```js
const topologyList = ssp.getTopologyByUserDataProperty('propKey', 'propVal')
// or
const topologyList = ssp.getTopologyByUserDataProperty(item => item['itemProp
```

参数：

propNameOrFindFunc

- **描述:** `userData` 内属性名 或 `find` 函数
- **类型:** string | function
- **必填:** ✓

propValue

- **描述:** `userData` 内属性值。
- **类型:** any
- **必填:** ✗

```js
topology.userData = {
  people: {
    name: 'xiaoming',
    age: 18,
  },
};
const topologyList = ssp.getTopologyByUserDataProperty(
  (userData) => userData?.people?.name === 'xiaoming'
);
```

> **弃用警告**
>
> 请使用 `getObjectByUserDataProperty` 替代

removeTopologyById 👎

通过 `id` 移除

定义：

```ts
function removeTopologyById(id: TopologyInfo['id']): boolean;
```

用法：

```js
ssp.removeTopologyById('xxx');
```

> **弃用警告**
>
> 请使用 `removeObjectById` 替代

createTopologyToGroup

创建 `topology` 到一个组内。

定义：

```ts
function createTopologyToGroup(
  groupInfo: GroupInfo,
  topologyInfoList: TopologyInfo[]
): Group;
```

用法：

```js
ssp.createTopologyToGroup(
  // groupInfo
  {
    id: 'firstTopologyGroup',
    name: 'name_firstTopologyGroup',
    // ...
  },
  // topologyInfoList
  [topologyInfo1, topologyInfo2, topologyInfo3]
);
```

参数

groupInfo

- **描述:** 实例组对象所需信息
- **类型:** GroupInfo
- **必填:** ✓

topologyInfoList

- 必填: ✓

createTopologyFromGml

创建 Topology 组，从 gml 文件资源。

定义:

```ts
interface TopologyInfoForGml extends BaseObject3DInfo {
  url: string;
  id: BaseObject3DInfo['id'];
  name?: BaseObject3DInfo['name'];
  linkWidth?: number;
  linkColor?: IColor;
  renderNode?: boolean;
  nodeColor?: IColor;
}

function createTopologyFromGml(
  topologyInfo: TopologyInfoForGml
): Promise<Topology>;
```

用法:

```js
ssp
  .createTopologyFromGml({
    url: './tuobutujinzui.gml',
    id: 'gml_for_topology',
    name: 'gml_for_topology_name',
    linkWidth: 100,
    linkColor: 'blue',
    renderNode: true,
    nodeColor: 'green',
  })
  .then((topology) => {
```

参数

topologyInfo

- **描述:** `topologyInfo` 对象
- **类型:** TopologyInfoForGml
- **必填:** ✓

TopologyInfoForGml

属性	描述	类型	必填	默认值
`url`	`gml` 资源路径	`string`	✗	
`id`	路径对象唯一ID	`string`	✗	
`name`	路径对象名称	`string`	✗	
`linkWidth`	路径线宽	`number`	✗	`20`
`linkColor`	路径线颜色	`IColor`	✗	`0x00ff00`
`renderNode`	是否渲染路径节点	`boolean`	✗	`true`
`nodeColor`	节点颜色	`IColor`	✗	`0x0000ff`

createGroupForTopology 👎

为 `topology` 提前创建一个空组。

> **使用场景**
>
> 与 `createTopologyToGroup` 不同，有些时候可能你还没有具体的 `topologyInfo` 数据，但你想提前创建一个批量管理的空组，当有数据时再使用 addTopologyForGroup 插入。

定义：

```ts
function createGroupForTopology(groupInfo: GroupInfo): Group;
```

用法：

```js
ssp.createGroupForTopology({
  id: 'firstTopologyGroup',
  name: 'name_firstTopologyGroup',
  // ...
});
```

参数

groupInfo

- **描述:** 实例组对象所需信息
- **类型:** GroupInfo
- **必填:** ✓

> **弃用警告**
>
> 请使用 `createGroup` 替代

addTopologyForGroup

向一个已经存在的组内添加 `topology` 对象。

定义：

```
    topologyInfoList: TopologyInfo[]
  ): Group | null;
```

用法：

```js
ssp.addTopologyForGroup(
  // groupId
  'firstTopologyGroup',
  // topologyInfoList
  [topologyInfo4, topologyInfo5],
  // onProgress
  (progress) => console.log('进度信息: ', progress)
);
```

参数

groupId

- 描述: 组 `id`
- 类型: groupId['id']
- 必填: ✔

topologyInfoList

- 描述: `topologyInfo` 集合
- 类型: topologyinfo[]
- 必填: ✔

getTopologyGroupById 👎

通过 `id` 查找 `topology` 组

定义：

用法:

```js
const group = ssp.getTopologyGroupById('firstTopologyGroup');
```

> **弃用警告**
>
> 请使用 `getObjectById` 替代

getTopologyGroupByName 👎

通过 `name` 查找 `topology` 组

定义:

```ts
function getTopologyGroupByName(name: string): Group[];
```

用法:

```js
const groupList = ssp.getTopologyGroupByName('name_firstTopologyGroup');
```

> **弃用警告**
>
> 请使用 `getObjectByName` 替代

getAllTopologyGroup 👎

获取所有 `Topology` 对象组

定义：

```ts
function getAllTopologyGroup(): Group[];
```

用法：

```js
const allTopologyGroupList = ssp.getAllTopologyGroup();
```

> **弃用警告**
>
> 请使用 `getAllGroup` 替代

removeTopologyGroupById 👎

通过 `id` 移除 `topology` 组

定义：

```ts
function removeTopologyGroupById(id: GroupInfo['id']): boolean;
```

用法：

```js
const isRemoveSuccess = ssp.removeTopologyGroupById('firstTopologyGroup');
```

> **弃用警告**
>
> 请使用 `removeObjectById` 替代

clearTopology

清除当前场景内所有 `topology` 对象。

定义：

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

用法：

```js
ssp.clearTopology();
```

showAllTopology

显示当前场景内所有 `topology` 对象。

定义：

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

用法：

```js
ssp.showAllTopology();
```

hideAllTopology

定义：

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

用法：

```js
ssp.hideAllTopology();
```

← 空间画布对象 控制器（废弃） →