

通用模型

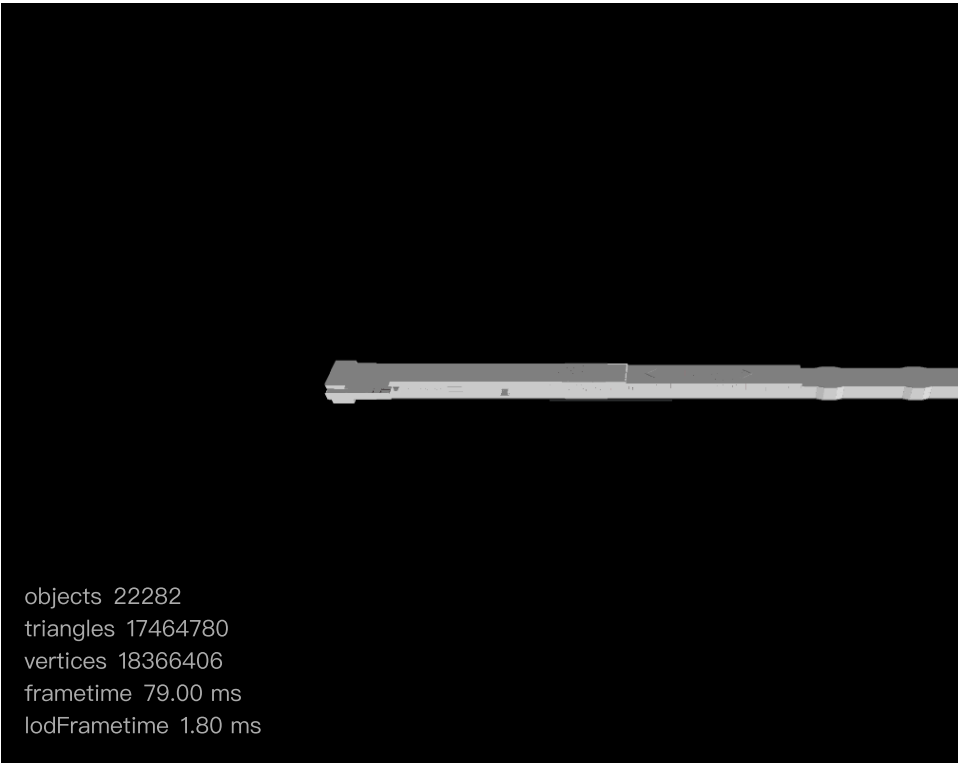
支持格式

支持的格式有 `glTF (glb)`、`fbx`

loadModel

加载 `model` 模型。

样例：



定义：

```
interface ModelInfo extends BaseObject3DInfo, ObjectEvents<Model> {  
  url: string;  
}
```

ts

```
}
```

```
function loadModel(modelInfo: ModelInfo): Promise<Model>;
```

用法:


ssp

js

```
.loadModel(  
  // modelInfo  
  {  
    id: 'xx',  
    name: 'xx',  
    url: 'xx/x.fbx',  
    level: {  
      max: 1000,  
      min: null,  
    },  
    position: { x: 0, y: 0, z: 0 },  
    rotation: { x: 0, y: 0, z: 0 },  
    scale: { x: 2, y: 2, z: 2 },  
    onClick(e) {  
      /**  
       * 对象的独立事件触发后，默认不传播（类似 DOM 的事件冒泡）到全局事件，  
       * 调用 eventPropagation 方法通知事件继续传播到全局。  
       *  
       * warn:  
       * 在 **非箭头函数** 中参数 e 与 this 的指向都是当前模型对象，  
       * 在 **箭头函数** 参数 e 依然是模型对象，但 this 指向会发生改变。  
       */  
      this.eventPropagation();  
  
      console.log('模型自身的点击事件触发', this);  
    },  
    onDbClick: (e) => {  
      /**  
       * 这里模拟在 **箭头函数** 中  
       */  
      e.eventPropagation();  
  
      console.log('模型自身的双击事件触发', e);  
    },  
    userData: {},  
  }  
)  
  .then((model) => console.log(model))  
  .catch((err) => console.error(err));
```

参数：

modelInfo

- 描述: 实例 `Model` 对象所需信息
- 类型: `ModelInfo`
- 必填: 

ModelInfo

属性	描述	类型	必填	默认值
id	唯一ID	string number		
name	名称	string		
url	资源路径	string		
level	显示层级范围	Level		{ max: null, min: null }
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		{}
onHover	鼠标悬浮事件	Function		null
onUnHover	鼠标悬浮后离开事件	Function		null
onClick	左键单击事件	Function		null
onDbClick	左键双击事件	Function		null
onRightClick	右键单击事件	Function		null
onLoad	加载完成事件	Function		null

getModelById

通过 `id` 查找

定义:

```
function getModelById(id: ModelInfo['id']): Model | null;
```

ts

用法:

```
const model = ssp.getModelById('xxx');
```

js

getModelByName

通过 `name` 查找

定义:

```
function getModelByName(name: string): Model[];
```

ts

用法:

```
const modelList = ssp.getModelByName('xxx');
```

js

getAllModel

获取所有 `Model` 对象

定义:

```
function getAllModel(): Model[];
```

ts

用法:

```
const allModelList = ssp.getAllModel();
```

js

getModelByUserDataProperty

通过 `userData` 属性查找

定义:

```
function getModelByUserDataProperty(  
  propNameOrFindFunc: string | UserDataPropertyFindFunc,  
  value?: any  
): Model[];
```

ts

用法:

```
const modelList = ssp.getModelByUserDataProperty('propKey', 'propVal')  
// or  
const modelList = ssp.getModelByUserDataProperty(item => item['itemPropKey'] === '')
```

js

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: ✗

find 函数使用场景

```
model.userData = {  
  people: {  
    name: 'xiaoming',  
    age: 18,  
  },  
};  
const modelList = ssp.getModelByUserDataProperty(  
  (userData) => userData?.people?.name === 'xiaoming'  
);
```

js

removeModelById

通过 `id` 移除

定义:

```
function removeModelById(id: ModelInfo['id']): boolean;
```

ts

用法:

```
ssp.removeModelById('xxx');
```

js

loadModelToGroup

加载 `model` 到一个组内。

定义:

```
function loadModelToGroup(  
  groupInfo: GroupInfo,
```

ts

```
modelInfoList: ModelInfo[]
): Promise<Group>;
```

用法:

```
ssp
  .loadModelToGroup(
    // groupInfo
    {
      id: 'firstModelGroup',
      name: 'name_firstModelGroup',
      // ...
    },
    // modelInfoList
    [modelInfo1, modelInfo2, modelInfo3]
  )
  .then((group) => console.log(group));
```

js

参数

groupInfo

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

modelInfoList

- 描述: `modelInfo` 集合
- 类型: **modelinfo[]**
- 必填: ✓

createGroupForModel

为 `model` 提前创建一个空组。

使用场景

与 `loadModelToGroup` 不同, 有些时候可能你还没有具体的 `modelInfo` 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 **addModelForGroup** 插入。

定义:

```
function createGroupForModel(groupInfo: GroupInfo): Group;
```

ts

用法:

```
ssp.createGroupForModel({  
  id: 'firstModelGroup',  
  name: 'name_firstModelGroup',  
  // ...  
});
```

js

参数

groupInfo

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

addModelForGroup

向一个已经存在的组内添加 `model` 对象。

定义:

```
function addModelForGroup(  
  groupId: GroupInfo['id'],  
  modelInfoList: ModelInfo[]  
): Promise<Group | null>;
```

ts

用法:

js

```
ssp
  .addModelForGroup(
    // groupId
    'firstModelGroup',
    // modelInfoList
    [modelInfo4, modelInfo5],
    // onProgress
    (progress) => console.log('进度信息: ', progress)
  )
  .then((group) => console.log(group));
```

参数

groupId

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

modelInfoList

- 描述: `modelInfo` 集合
- 类型: `modelinfo[]`
- 必填: ✓

getModelGroupById

通过 `id` 查找 `model` 组

定义:

ts

```
function getModelGroupById(id: GroupInfo['id']): Group | null;
```

用法:

js

```
const group = ssp.getModelGroupById('firstModelGroup');
```

getModelGroupByName

通过 `name` 查找 `model` 组

定义:

```
function getModelGroupByName(name: string): Group[];
```

ts

用法:

```
const groupList = ssp.getModelGroupByName('name_firstModelGroup');
```

js

removeModelGroupById

通过 `id` 移除 `model` 组

定义:

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

ts

用法:

```
const isRemoveSuccess = ssp.removeModelGroupById('firstModelGroup');
```

js

clearModel

清除当前场景内所有 `model` 对象。

定义:

```
function clearModel(): void;
```

ts

用法:

```
ssp.clearModel();
```

js

showAllModel

显示当前场景内所有 `model` 对象。

定义:

```
function showAllModel(): void;
```

ts

用法:

```
ssp.showAllModel();
```

js

hideAllModel

隐藏当前场景内所有 `model` 对象。

定义:

```
function hideAllModel(): void;
```

ts

用法:

```
ssp.hideAllModel();
```

js

playModelAnimation

播放模型动画。

定义:

```
interface ModelAnimationFindFunc {  
  (  
    animation: AnimationClip,  
    index: number,  
    animations: AnimationClip[]  
  ): boolean;  
}  
  
function playModelAnimation(  
  model: Model,  
  animation: number | AnimationClip | ModelAnimationFindFunc  
): AnimationAction | undefined;
```

ts

用法:

```
const model = ssp.getModelById('xxx');  
  
// number  
ssp.playModelAnimation(model, 0);  
// or AnimationClip  
ssp.playModelAnimation(model, model.animations[0]);  
// or ModelAnimationFindFunc  
ssp.playModelAnimation(model, (itemAnimation) => itemAnimation.name === 'run');
```

js

参数:

model

- 描述: 模型对象
- 类型: **Model**
- 必填: ✓

animation

- 描述: 动画信息
- 类型: number | **AnimationClip** [🔗](#) | ModelAnimationFindFunc
- 必填: ✓

stopModelAnimation

停止模型动画。

定义:

```
function stopModelAnimation(  
  model: Model,  
  animation: number | AnimationClip | ModelAnimationFindFunc  
): void;
```

ts

用法:

```
const model = ssp.getModelById('xxx');  
  
// number  
ssp.stopModelAnimation(model, 0);  
// or AnimationClip  
ssp.stopModelAnimation(model, model.animations[0]);  
// or ModelAnimationFindFunc  
ssp.stopModelAnimation(model, (itemAnimation) => itemAnimation.name === 'run');
```



js

参数:

model

- 描述: 模型对象
- 类型: **Model**
- 必填: 

animation

- 描述: 动画信息
- 类型: number | **AnimationClip** | ModelAnimationFindFunc
- 必填: 

[← Sbm 模型](#)

[Poi 对象 →](#)