plugin-heat-cloud

热力云

安装

```sh
npm install @soonspacejs/plugin-heat-cloud -S
# or
yarn add @soonspacejs/plugin-heat-cloud -S
```

使用方法

```ts
import SoonSpace from "soonspacejs";
import HeatCloudPlugin from "@soonspacejs/plugin-heat-cloud";

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

const heatCloudPlugin = ssp.registerPlugin(HeatCloudPlugin, "heatCloudPlugin"
```

属性

defaultGradientVolumeMaterialOptions

默认值

```ts
defaultGradientVolumeMaterialOptions: GradientVolumeMaterialOptions = {
  fit: VolumeFit.Raw,
  accFactor: 2,
  depthTest: false,
  side: BackSide,
  discardOut: false,
  // 数值的梯度映射纹理
  gradient: createLinearGradientTexture(this.defaultColorGradient),
};
```

使用

修改默认值

```ts
const heatCloudPlugin = ssp.registerPlugin(HeatCloudPlugin, "heatCloudPlugin"
heatCloudPlugin.defaultGradientVolumeMaterialOptions = {
  ...heatCloudPlugin.defaultGradientVolumeMaterialOptions,
  fit: VolumeFit.Fit,
  accFactor: 1,
};
```

defaultColorGradient

用于使用`createLinearGradientTexture`创建`defaultGradientVolumeMaterialOptions.gradient`,可在调用`createHeatCloud`、`createLineHeat`前进行修改

> **用于使用 createLinearGradientTexture 创建 defaultGradientVolumeMaterialOptions.gradient**
>
> ```ts
> defaultGradientVolumeMaterialOptions.gradient = createLinearGradientTe
>   defaultColorGradient
> ```

默认值

```ts
defaultColorGradient: ColorGradient = [
  [0, "rgba(0,255,0,0)"],
  [0.5, "rgba(64,255,255,0.5)"],
  [1, "rgba(255,64,255,1)"],
];
```

使用

修改默认值

```ts
const heatCloudPlugin = ssp.registerPlugin(HeatCloudPlugin, "heatCloudPlugin"
heatCloudPlugin.defaultColorGradient = [
  [0, "rgba(0,255,0,0)"],
  [0.5, "rgba(255,255,64,0.5)"],
  [1, "rgba(255,64,255,1)"],
];
```

方法

createHeatCloud

创建随机热力云

样例

SoonSpace.js 2.x

定义

```ts
createHeatCloud(points: HeatParticleVolumeFeaturePoint[], options?: CreateHea
```

```ts
type HeatParticleVolumeFeaturePoint = HeatParticleVolumeFeature & IVector3;
/**
 * 热力粒子体积特性
 */
interface HeatParticleVolumeFeature extends SphereParticleVolumeFeature {
  /**
   * 值累积函数
   * @remarks
   * 当某一个点处理多个热力球区域内时，需要考虑多个热力球在该点的累积效果，通过该函数来获取最
   * @defaultValue valuesAccumulate_Default
   */
  valuesAccumulate?: HeatValuesAccumulate;
}

interface SphereParticleVolumeFeature extends ParticleVolumeFeature {
  /**
   * 映射区间
   * @remarks
   * x 为最小值，y 为最大值
   *
```

```
  clim?: IVector2;
  /**
   * 值梯度函数
   */
  valueGradient?: GetGradientValue;
}
/**
 * 粒子体积特征
 */
interface ParticleVolumeFeature extends VolumeFeature {
  /**
   * 空心因子
   *
   * @defaultValue 0
   */
  hollow?: number;
}
/**
 * 体积特征
 */
interface VolumeFeature {
  /**
   * 半径
   *
   * @defaultValue 10
   */
  radius?: number;
  /**
   * 值
   *
   * @defaultValue 100
   */
  value?: number;
}
/**
 * 热力值累积函数
 * @remarks
 * 当某一个点处理多个热力球区域内时，需要考虑多个热力球在该点的累积效果，通过该函数来获取最终
 * @param values - 值的信息列表
 * @param clim - 默认的映射区间
 * @returns 累积后的最终值
 */
type HeatValuesAccumulate = (
  values: HeatAccumulateValue[],
```

```
type HeatAccumulateValue = Required<
  Omit<SphereParticleVolumeFeature, "valueGradient">
> &
  GradientParams;
/**
 * 值梯度函数
 * @remarks
 * 会通过该函数来获取在中心点及其半径范围内各点处的值
 * @param params - 梯度参数；包含了当前点处的相关信息
 * @returns 返回最终的值
 */
type GetGradientValue = (params: GradientParams) => number;
/**
 * 梯度函数的参数
 */
interface GradientParams {
  /**
   * 梯度的变化比率
   * @remarks
   * 一般是当前点与其所属中心点的距离和中心点半径的比率
   */
  ratio: number;
  /**
   * 当前中心点的半径
   */
  radius: number;
  /**
   * 当前中心点空心比率
   */
  hollow: number;
  /**
   * 根据空间比率计算的空心半径
   */
  hollowRadius: number;
  /**
   * 当前点与其所属中心点的距离
   */
  distance: number;
  /**
   * 值
   */
  value: number;
  /**
   * 值的范围
```

```
    * 当前点的坐标
    */
  point: Vector3;
}
```

```ts
type CreateHeatCloudOptions = HeatDataOptions & GradientOptions;
```

```ts
/**
 * 创建热力3D数据的选项
 */
interface HeatDataOptions {
  /**
   * 值
   *
   * @defaultValue 100
   */
  value?: number;
  /**
   * 半径
   *
   * @defaultValue 10
   */
  radius?: number;
  /**
   * 空心因子
   *
   * @defaultValue 0
   */
  hollow?: number;
  /**
   * 值累积函数
   * @remarks
   * 当某一个点处理多个热力球区域内时，需要考虑多个热力球在该点的累积效果，通过该函数来获取最
   * @defaultValue valuesAccumulate_Default
   */
  valuesAccumulate?: HeatValuesAccumulate;
  /**
   * 映射区间
   * @remarks
```

```
    */
  clim?: IVector2;
  /**
   * 值梯度函数
   */
  valueGradient?: GetGradientValue;
  /**
   * Data3D 的尺寸
   * @defaultValue 从原点到  points 的AABB包围盒最大点的空间尺寸
   */
  size?: IVector3 | null;
}
```

ts

```
type GradientOptions = GradientOptionsOptimizeOptions &
  CreateGradientData3DTextureOptions &
  GradientVolumeMaterialOptions;

/**
 * 优化选项
 */
interface GradientOptionsOptimizeOptions {
  /**
   * 指定优化后的梯度选项的所能达到的最大尺寸
   * @remarks
   * 会对原来的梯度选项进行等比缩放以达到该尺寸;
   *
   * 如果同时指定 maxSize 和 scale , 则优先使用 scale
   * @defaultValue 100
   */
  maxSize?: number;
  /**
   * 缩放系数
   * @remarks
   * 对原梯度选项进行缩放;
   *
   * 优先级高于 maxSize
   */
  scale?: number;
}
interface CreateGradientData3DTextureOptions {
  /**
```

```
    * 在转换时 IData3D 时，当遇到值为 undefined 或 null 的值时，就会使 voidValue 来替
    *
    * IData3D 中，可以通过 undefined 或 null 来表示空，但在将 IData3D 转为 Data3DTex
    *
    * @defaultValue 0
    */
  voidValue?: number;
  /**
    * 3D数据中包含的数值是否是 uint8 类型的
    * @defaultValue false
    */
  uint8?: boolean;
}
/**
 * 梯度体积材质选项
 */
interface GradientVolumeMaterialOptions
  extends VolumeMaterialOptions<GradientData3DTexture>,
    GradientData3DOptions {
  /**
    * 数值的梯度映射纹理
    */
  gradient?: GradientTextureOptions | null;
}

type GradientTextureOptions = Texture | string | ColorGradient;

/**
 * 颜色断点
 * @remarks
 * number 的范围是 0-1，表示断点的位置；
 *
 * string 表示颜色，支持 css 所支持的所有颜色格式；
 */
type ColorStop = [number, string];
/**
 * 颜色梯度，颜色断点列表
 * @remarks
 * 颜色断点可以不按顺序排列
 */
type ColorGradient = ColorStop[];
/**
 * 体积材质的选项
 */
```

```
 * 三维的纹理
 */
map?: GradientData3DTexture | null;
/**
 * 渲染体积材质的容器的最小点
 * @remarks
 * 容器的最小点和最大点一般是被设置成 geometry 的AABB包围盒；但也可以不是
 * 总之，体积材质的渲染依托于 geometry 的形状，它不会超出 geometry 的尺寸；
 * 但可以控制容器的最小点和最大点来控制体积材质的渲染尺寸；
 * @defaultValue {x:0,y:0,z:0}
 */
containerMin?: IVector3 | null;
/**
 * 渲染体积材质的容器的最大点
 * @remarks
 * 容器的最小点和最大点一般是被设置成 geometry 的AABB包围盒；但也可以不是
 * 总之，体积材质的渲染依托于 geometry 的形状，它不会超出 geometry 的尺寸；
 * 但可以控制容器的最小点和最大点来控制体积材质的渲染尺寸；
 * @defaultValue map尺寸中的最大点
 */
containerMax?: IVector3 | null;
/**
 * 体积材质在容器内的填充模式
 * @defaultValue VolumeFit.Fill
 */
fit?: VolumeFit | null;
/**
 * Mesh三角形的渲染面
 * @remarks
 * 可设置为前面、后面、或者两面都渲染
 * 如果只渲染前面，进入体积材质内部，体积材质就会消失
 * 如果只渲染后面，体积初遮挡，就会消失
 * 如果两面都渲染，则会隐藏看到容器的面
 */
side?: Side | undefined;
/**
 * 透明度
 * @remarks
 * 决定整体的透明度
 * @defaultValue 1
 */
opacity?: number | null;
/**
```

```
     * 如果最终颜色的透明度小于或等于 alphaRange.x ，则被认为是完全透明
     * 如果最终颜色的透明度在于或等于 alphaRange.y ，则被认为是完全不透明
     * 取值范围应当在 0 - 1
     * @defaultValue {x:0,y:0.95}
     */
    alphaRange?: IVector2 | null;
    /**
     * 是否开启雾化的效果
     * @remarks
     * 开启后的渲染效果会更像雾
     * @defaultValue true
     */
    atomize?: boolean | null;
    /**
     * 体积材质渲染的采样数
     * @remarks
     * 采样数越高，占用的GPU资源就越大；所以该数据设置的适宜最好；
     * @defaultValue 100
     */
    steps?: number | null;
    /**
     * 颜色累积系数
     * @remarks
     * 它是对体积材质进行积分时使用的系数；只在开启雾化效果{@link VolumeMaterial.atomize
     * 值越小，最终呈现出的效果就越雾化；
     * @defaultValue 1
     */
    accFactor?: number | null;
}
/**
 * 梯度 Data3D 选项
 */
interface GradientData3DOptions {
    /**
     * 数值的映射区间
     * @remarks
     * x 为最小值，y 为最大值
     *
     * @defaultValue {x:0,y:100}
     */
    range?: IVector2 | null;
    /**
     * 是否丢弃超出映射范围的数值
     * @remarks
```

```
    * 包含左右边界值
    * @defaultValue false
    */
  discardOut?: boolean | null;
  /**
    * 空值的范围
    * @remarks
    * 当数值在此范围中时，会被认为是空的值，渲染时会被丢弃
    *
    * 包含左右边界值
    *
    * @defaultValue {x:-100,y:-1}
    */
  voidRange?: IVector2 | null;
}

/**
 * 体积材质填充模式
 */
declare enum VolumeFit {
  /**
    * 填充
    * @remarks
    * 材质会充满容器
    */
  Fill = 0,
  /**
    * 对齐
    * @remarks
    * 材质会从容器的最小点开始绘制，但不会充满容器
    */
  Align = 1,
  /**
    * 原始
    * @remarks
    * 材质会从容器空间中的坐标原点开始绘制，也不会充满容器
    */
  Raw = 2,
}
```

ts

```
/**
 * 体积Mesh
```

```
 * 体积材质也可以用于普通的 Mesh；但如果使用 VolumeMesh ，则可以减少很多体积材质 和 geom
 */
declare class VolumeMesh extends Mesh {
  readonly isVolumeMesh = true;
  constructor(material: VolumeMaterial);
  protected _geometry: BufferGeometry;
  /**
   * 是否自动更新材质
   * @remarks
   * 当启动该选项后，在 VolumeMesh 监测到 geometry 变更时，会自动更新体积材质的相关选项，
   * @defaultValue true
   */
  autoUpdateMaterial: boolean;
  material: VolumeMaterial;
  protected _material: VolumeMaterial;
  /**
   * 是否自动更新几何体
   * @remarks
   * 当启动该选项后，在 VolumeMesh 监测到 material 变更时，会自动更新 geometry 的相关
   * @defaultValue true
   */
  autoUpdateGeometry: boolean;
  /**
   * 更新材质
   * @remarks
   * 更新体积材质的相关选项以使其匹配 geometry
   */
  updateMaterial(): false | undefined;
  /**
   * 更新几何体
   * @remarks
   * 更新 geometry 的相关选项以使其匹配体积材质
   */
  updateGeometry(): false | undefined;
  /**
   * 自动规范化
   * @remarks
   * 开启该选项，设置新的 geometry 时，对 geometry 自动执行规范化操作
   */
  autoNormalize: boolean;
  /**
   * 规范化
   * @remarks
```

```
  normalize(): void;
  /**
   * 将材质空间下的坐标转为 map 空间下的坐标
   * @param coord - 材质空间下的坐标
   * @returns
   */
  toMapPosition(coord: IVector3): Vector3;
  /**
   * 将世界坐标系下的深度转为 material 空间下的深度
   * @param axis - 材质空间下的坐标轴
   * @param depth - 深度
   * @returns
   */
  toMaterialDepth(axis: Axis, depth: number): number;
  /**
   * 将世界坐标系下的深度转为 map 空间下的深度
   * @param axis - map空间下的坐标轴
   * @param depth - 深度
   * @returns
   */
  toMapDepth(axis: Axis, depth: number): number;
  /**
   * 获取3D数据的切片
   * @remarks
   * 切片就是指定轴的指定位置的垂直截面;
   * 轴和深度都是局部坐标系下的
   * @param axis - map空间下的坐标轴
   * @param depth - 世界空间下在轴方向上的坐标值
   * @returns
   */
  getData3DSlice(axis: Axis, depth: number): IData2D<number> | null;
  /**
   * 获取3D数据中指定坐标处的数据项目
   * @param coord - 世界坐标系下的坐标
   * @returns
   */
  getItem(coord: IVector3): number[] | null;
}
/**
 * Data2D 类型
 */
interface IData2D<D = number> {
  data: ArrayLike<D>;
```

使用

```ts
const ssp = new SoonSpace({
  el: "#viewA",
  options: {
    showInfo: true,
    showGrid: false,
    background: {
      color: new Color(0x444444),
    },
  },
  events: {
    selectPosition: console.log,
    modelClick: console.log,
  },
});
const pointCount = 60;
const randomPoints = Array.from({ length: pointCount }, () => ({
  x: Math.random() * 90,
  y: Math.random() * 30,
  z: Math.random() * 60,
  value: Math.random() * 50,
  radius: Math.random() * 30,
}));
// const mesh = heatCloudPlugin.createHeatCloud(randomPoints) ;
const mesh = heatCloudPlugin.createHeatCloud(randomPoints, {
  range: { x: 0, y: 50 },
  voidRange: { x: -100, y: -50 },
  size: { x: 90, y: 30, z: 60 },
  steps: 100,
  opacity: 1,
  accFactor: 2,
  depthTest: false,
  // side: BackSide,
  side: FrontSide,

  discardOut: false,
```

参数

points

> **TIP**
>
> `points` 中与 `options` 相同的属性， `points` 优先级 **高于** `options`

- **描述:** 热力点
- **必填:** ✓
- **类型:** `HeatParticleVolumeFeaturePoint[]`

HeatParticleVolumeFeaturePoint

属性	描述	类型	必填	默认值
x	x	number	✓	
y	y	number	✓	
z	z	number	✓	
value	值	number	✗	100
radius	半径	number	✗	10
hollow	空心因子	number	✗	0
clim	映射区间	IVector2	✗	{x:0,y:100}
valueGradient	值梯度函数	GetGradientValue	✗	
valuesAccumulate	热力值累积函数	HeatValuesAccumulate	✗	线性累积函数

options

- **描述:** 配置
- **必填:** ✗
- **类型:** `CreateHeatCloudOptions`

value	值	number
radius	半径	number
hollow	空心因子	number
clim	映射区间	IVector2
valueGradient	值梯度函数	GetGradientValue
valuesAccumulate	热力值累积函数	HeatValuesAccumulate
maxSize	优化后的梯度选项的所能达到的最大尺寸	number
scale	缩放系数	number
voidValue	空值	number
uint8	是否是 uint8 类型	boolean
gradient	梯度映射纹理	GradientTextureOption
map	三维的纹理	GradientData3DTexture
containerMin	体积材质的容器的最小点	IVector3 | null
containerMax	体积材质的容器的最大点	IVector3 | null
fit	填充模式	VolumeFit
side	Mesh 三角形的渲染面	Side | undefined
opacity	透明度	number | null
alphaRange	有效透明度的取值范围	IVector2 | null
atomize	是否开启雾化的效果	boolean
steps	体积材质渲染的采样数	number | null
accFactor	颜色累积系数	number | null
range	数值的映射区间	IVector2 | null
discardOut	否丢弃超出映射范围的数值	boolean | null
voidRange	颜色累积系数	IVector2 | null

gradientTexture

```ts
const defaultColorGradient: ColorGradient = [
  [0, "rgba(0,255,0,0)"],
  [0.5, "rgba(64,255,255,0.5)"],
  [1, "rgba(255,64,255,1)"],
];
const gradientTexture = createLinearGradientTexture(tdefaultColorGradient);
```

createLineHeat

创建线状热力

样例

定义

```ts
createLineHeat(points: ParticleVolumeFeaturePoint[], options?: CreatLineHeatO
```

```ts
type CreatLineHeatOptions = LineDataOptions & GradientOptions;

interface LineDataOptions {
  /**
   * 值
   *
   * @defaultValue 100
   */
  value?: number;
  /**
   * 半径
   *
   * @defaultValue 10
   */
  radius?: number;
  /**
   * 空心因子
   *
   * @defaultValue 0
   */
  hollow?: number;

  /**
   * 映射区间
   * @remarks
   * x 为最小值，y 为最大值
   *
   * @defaultValue {x:0,y:100}
   */
  clim?: IVector2;
  /**
   * 值梯度函数
   */
  valueGradient?: GetGradientValue;
  /**
   * Data3D 的尺寸
   * @defaultValue 从原点到  points 的AABB包围盒最大点的空间尺寸
   */
  size?: IVector3 | null;
  /**
   * 半径梯度函数
   * @defaultValue radiusGradient_Default
```

```
    /**
     * 空心比梯度函数
     * @defaultValue hollowGradient_Default
     */
    hollowGradient?: GetLineGradientValue;
    /**
     * 轴线值梯度函数
     * @defaultValue lineValueGradient_Default
     */
    lineValueGradient?: GetLineGradientValue;
    /**
     * 值累积函数
     * @defaultValue numberValuesAccumulate_Default
     */
    valuesAccumulate?: NumberValuesAccumulate;
}

/**
 * 轴线值梯度函数
 * @remarks
 * 会通过该函数来获取从轴线的起始点到终点之间轴线上各点处的值
 * @param params - 梯度参数；包含了当前点处的相关信息
 * @returns 返回最终的值
 */
type GetLineGradientValue = (params: LineGradientParams) => number;
/**
 * 轴线梯度函数的参数
 */
interface LineGradientParams {
    /**
     * 梯度的变化比率
     * @remarks
     * 一般是当前轴线位置到轴线开始处距离 与 轴线的长度的比率
     */
    ratio: number;
    /**
     * 轴线的长度
     */
    length: number;
    /**
     * 当前点的坐标
     */
    point: Vector3;
    /**
```

```
    startRadius: number;
    /**
     * 轴线终点处的半径
     */
    endRadius: number;
    /**
     * 轴线终点处的半径相对起始点处的半径的增加量；即 `endRadius - startRadius`
     */
    addedRadius: number;
    /**
     * 选项中默认的半径
     */
    defaultRadius: number;
    /**
     * 当前中心点的半径
     * @remarks
     * 注意：在 `radiusGradient` 中，该值就是 `defaultRadius`；
     */
    radius: number;
    /**
     * 轴线起始点处的值
     */
    startValue: number;
    /**
     * 轴线终点处的值
     */
    endValue: number;
    /**
     * 轴线终点处的值相对起始点处的值的增加量；即 `endValue - startValue`
     */
    addedValue: number;
    /**
     * 选项中默认的值
     */
    defaultValue: number;
    /**
     * 轴线起始点处的空心比率
     */
    startHollow: number;
    /**
     * 轴线终点处的空心比率
     */
    endHollow: number;
    /**
```

```
  addedHollow: number;
  /**
   * 选项中默认的空心比率
   */
  defaultHollow: number;
}
/**
 * 数值累积函数
 * @remarks
 * 当某一个点处理多个区域数据内时，需要考虑多个数据区域在该点的累积效果，通过该函数来获取最终
 * @param values - 值的列表
 * @param clim - 默认的映射区间
 * @returns 累积后的最终值
 */
type NumberValuesAccumulate = (values: number[], clim: IVector2) => number;
```

使用

```ts
const ssp = new SoonSpace({
  el: "#viewA",
  options: {
    showInfo: true,
    showGrid: false,
    background: {
      color: new Color(0x444444),
    },
  },
  events: {
    selectPosition: console.log,
    modelClick: console.log,
  },
});

var linePoints = [
  { x: 50, y: 0, z: 50, value: 50, radius: 20 },
  { x: 50, y: 70, z: 50, value: 50, radius: 20 },
];
// const mesh = heatCloudPlugin.createLineHeat(linePoints);
const mesh = heatCloudPlugin.createLineHeat(linePoints, {
  range: { x: 0, y: 50 },
  voidRange: { x: -100, y: -50 },
  size: { x: 90, y: 30, z: 60 },
```

```
    depthTest: false,
    side: FrontSide,
    discardOut: false,
    fit: VolumeFit.Fill,
  });
```

参数

points

> **TIP**
>
> `points` 中与 `options` 相同的属性， `points` 优先级 **高于** `options`

- **描述:** 热力点
- **必填:** ✓
- **类型:** `ParticleVolumeFeaturePoint[]`

ParticleVolumeFeaturePoint

属性	描述	类型	必填	默认值
x	x	number	✓	
y	y	number	✓	
z	z	number	✓	
value	值	number	✗	100
hollow	空心因子	number	✗	0
clim	映射区间	IVector2	✗	{x:0,y:100}
valueGradient	值梯度函数	GetGradientValue	✗	

options

- **描述:** 配置
- **必填:** ✗

属性	描述	类型
value	值	number
radius	半径	number
hollow	空心因子	number
clim	映射区间	IVector2
valueGradient	值梯度函数	GetGradientValue
valuesAccumulate	热力值累积函数	HeatValuesAccumulate
radiusGradient	半径梯度函数	GetLineGradientValue
hollowGradient	空心比梯度函数	GetLineGradientValue
lineValueGradient	轴线值梯度函数	GetLineGradientValue
maxSize	优化后的梯度选项的所能达到的最大尺寸	number
scale	缩放系数	number
voidValue	空值	number
uint8	是否是 uint8 类型	boolean
gradient	梯度映射纹理	GradientTextureOptions
map	三维的纹理	GradientData3DTexture
containerMin	体积材质的容器的最小点	IVector3 | null
containerMax	体积材质的容器的最大点	IVector3 | null
fit	填充模式	VolumeFit
side	Mesh 三角形的渲染面	Side | undefined
opacity	透明度	number | null
alphaRange	有效透明度的取值范围	IVector2 | null
atomize	是否开启雾化的效果	boolean
steps	体积材质渲染的采样数	number | null

range	数值的映射区间	IVector2 | null
discardOut	否丢弃超出映射范围的数值	boolean | null
voidRange	颜色累积系数	IVector2 | null

gradientTexture

```ts
const defaultColorGradient: ColorGradient = [
  [0, "rgba(0,255,0,0)"],
  [0.5, "rgba(64,255,255,0.5)"],
  [1, "rgba(255,64,255,1)"],
];
const gradientTexture = createLinearGradientTexture(tdefaultColorGradient);
```

createImageExtrusion

创建图片挤压

样例

SoonSpace.js 2.x

定义

```ts
createImageExtrusion(imageUrl: string, options?: CreateImageExtrusionOptions
    depth?: number;
}): Promise<VolumeMesh>
```

```ts
type CreateImageExtrusionOptions = VolumeMaterialOptions<ImageData3DTexture>
  ExtrudeImageOptions;
interface VolumeMaterialOptions<GradientData3DTexture>
  extends Omit<ShaderMaterialParameters, "map" | "opacity"> {
  /**
   * 三维的纹理
   */
  map?: ImageData3DTexture | null;
  /**
   * 渲染体积材质的容器的最小点
   * @remarks
   * 容器的最小点和最大点一般是被设置成 geometry 的AABB包围盒；但也可以不是
   * 总之，体积材质的渲染依托于 geometry 的形状，它不会超出 geometry 的尺寸；
   * 但可以控制容器的最小点和最大点来控制体积材质的渲染尺寸；
   * @defaultValue {x:0,y:0,z:0}
   */
  containerMin?: IVector3 | null;
  /**
```

```
     * 容器的最小点和最大点一般是被设置成 geometry 的AABB包围盒，但也可以不是
     * 总之，体积材质的渲染依托于 geometry 的形状，它不会超出 geometry 的尺寸；
     * 但可以控制容器的最小点和最大点来控制体积材质的渲染尺寸；
     * @defaultValue map尺寸中的最大点
     */
    containerMax?: IVector3 | null;
    /**
     * 体积材质在容器内的填充模式
     * @defaultValue VolumeFit.Fill
     */
    fit?: VolumeFit | null;
    /**
     * Mesh三角形的渲染面
     * @remarks
     * 可设置为前面、后面、或者两面都渲染
     * 如果只渲染前面，进入体积材质内部，体积材质就会消失
     * 如果只渲染后面，体积初遮挡，就会消失
     * 如果两面都渲染，则会隐藏看到容器的面
     */
    side?: Side | undefined;
    /**
     * 透明度
     * @remarks
     * 决定整体的透明度
     * @defaultValue 1
     */
    opacity?: number | null;
    /**
     * 有效透明度的取值范围
     * @remarks
     * 如果最终颜色的透明度小于或等于 alphaRange.x ，则被认为是完全透明
     * 如果最终颜色的透明度在于或等于 alphaRange.y ，则被认为是完全不透明
     * 取值范围应当在 0 - 1
     * @defaultValue {x:0,y:0.95}
     */
    alphaRange?: IVector2 | null;
    /**
     * 是否开启雾化的效果
     * @remarks
     * 开启后的渲染效果会更像雾
     * @defaultValue true
     */
    atomize?: boolean | null;
    /**
```

SoonSpace.js 2.x

```
    * @defaultValue 100
    */
  steps?: number | null;
  /**
    * 颜色累积系数
    * @remarks
    * 它是对体积材质进行积分时使用的系数；只在开启雾化效果{@link VolumeMaterial.atomize
    * 值越小，最终呈现出的效果就越雾化；
    * @defaultValue 1
    */
  accFactor?: number | null;
}
/**
 * extrudeImage 的选项
 */
interface ExtrudeImageOptions {
  reverseY?: boolean | null;
  axis?: Axis | null;
  colorSpace?: PredefinedColorSpace | null;
}
type PredefinedColorSpace = "display-p3" | "srgb";
```

使用

```ts
const ssp = new SoonSpace({
  el: "#viewA",
  options: {
    showInfo: true,
    showGrid: false,
    background: {
      color: new Color(0x444444),
    },
  },
  events: {
    selectPosition: console.log,
    modelClick: console.log,
  },
});
const mesh = await heatCloudPlugin.createImageExtrusion(imgUrl, {
  side: BackSide,
  depthTest: false,
```

```
});
```

参数

options

- **描述:** 配置
- **必填:** ✗
- **类型:** `CreateImageExtrusionOptions & { depth?: number }`

属性	描述	类型	必填	默认值
map	三维的纹理	GradientData3DTexture	✗	
containerMin	体积材质的容器的最小点	IVector3 | null	✗	{x:0,y:0
containerMax	体积材质的容器的最大点	IVector3 | null	✗	{x:0,y:0
fit	填充模式	VolumeFit	✗	VolumeFi
side	Mesh 三角形的渲染面	Side | undefined	✗	BackSide
opacity	透明度	number | null	✗	1
alphaRange	有效透明度的取值范围	IVector2 | null	✗	{x:0,y:
atomize	是否开启雾化的效果	boolean	✗	true
steps	体积材质渲染的采样数	number | null	✗	100
accFactor	颜色累积系数	number | null	✗	1
range	数值的映射区间 l	IVector2 | null	✗	{x:0,y:1
discardOut	否丢弃超出映射范围的数值	boolean | null	✗	false
voidRange	颜色累积系数	IVector2 | null	✗	{x:-100,
depth	depth	number	✗	20

createSliceMesh

SoonSpace.js 2.x

定义

> `VolumeMaterial` 为 `VolumeMesh` 的材质

```ts
createSliceMesh(material: VolumeMaterial, options?: SliceMaterialOptions & Cr
```

```ts
/**
 * 切片材质的选项
 */
interface SliceMaterialOptions
  extends Omit<ShaderMaterialParameters, "opacity"> {
  /**
   * 三维的纹理
   */
  map?: Texture3D | null;
  /**
   * 透明度
   * @remarks
   * 决定整体的透明度
   * @defaultValue 1
   */
  opacity?: number | null;
  /**
   * 轴
   * @remarks
   * 切片就是垂直于该轴的截面
   *
   * @defaultValue Axis.z
   */
  axis?: Axis | null;
  /**
   * 轴上的坐标值
   * @remarks
   * 会在轴的该位置处获取切片
   * @defaultValue 0
   */
  depth?: number | null;
}

type CreateImageData3DTextureFromGradientOptions = {
```

```
     * @remarks
     * x 为最小值，y 为最大值
     *
     * @defaultValue {x:0,y:100}
     */
    range?: IVector2 | null;
    /**
     * 是否丢弃超出映射范围的数值
     * @remarks
     * 当数值超出映射区间 {@link GradientData3DOptions.range}  时，是否丢弃；
     *
     * 包含左右边界值
     * @defaultValue true
     */
    discardOut?: boolean | null;
    /**
     * 空值的范围
     * @remarks
     * 当数值在此范围中时，会被认为是空的值，渲染时会被丢弃
     *
     * 包含左右边界值
     *
     * @defaultValue {x:-100,y:-1}
     */
    voidRange?: IVector2 | null;

    voidColor?: [number, number, number, number];

    gradient: TexImageSource | ColorGradient;
};
type TexImageSource =
    | ImageBitmap
    | ImageData
    | HTMLImageElement
    | HTMLCanvasElement
    | HTMLVideoElement
    | OffscreenCanvas
    | VideoFrame;
```

ts

```
/**
 * 呈现3D图像切片的Mesh
 * @remarks
```

```
 */
declare class SliceMesh extends Mesh {
  readonly isSliceMesh = true;
  constructor(options: SliceMaterialOptions);
  protected _geometry: BufferGeometry;
  material: SliceMaterial;
  protected _material: SliceMaterial;
  /**
   * 是否自动更新几何体
   * @remarks
   * 当启动该选项后，在 SliceMesh 监测到切片尺寸相关的变更时，会自动更新 geometry 的尺寸
   * @defaultValue true
   */
  autoUpdateGeometry: boolean;
  /**
   * 更新几何体
   */
  updateGeometry(): false | undefined;
  /**
   * 三维的纹理
   */
  get map(): Texture3D | null;
  set map(value: Texture3D | null);
  /**
   * 轴
   * @remarks
   * 切片就是垂直于该轴的截面
   *
   * @defaultValue Axis.z
   */
  get axis(): Axis;
  set axis(value: Axis);
  /**
   * 轴上的坐标值
   * @remarks
   * 会在轴的该位置处获取切片
   * @defaultValue 0
   */
  get depth(): number;
  set depth(value: number);
  /**
   * 切片的尺寸
   * @remarks
   * 不同轴上的切片的尺寸一般不一样
```

```
type Texture3D = Data3DTexture | DataArrayTexture;
/**
 * 轴
 */
declare enum Axis {
  x = 0,
  y = 1,
  z = 2,
}
```

使用

```ts
const ssp = new SoonSpace({
  el: "#viewA",
  options: {
    showInfo: true,
    showGrid: false,
    background: {
      color: new Color(0x444444),
    },
  },
  events: {
    selectPosition: console.log,
    modelClick: console.log,
  },
});

var linePoints = [
  { x: 50, y: 0, z: 50, value: 50, radius: 20 },
  { x: 50, y: 70, z: 50, value: 50, radius: 20 },
];
// const mesh = heatCloudPlugin.createLineHeat(linePoints);
const mesh = heatCloudPlugin.createLineHeat(linePoints, {
  range: { x: 0, y: 50 },
  voidRange: { x: -100, y: -50 },
  size: { x: 90, y: 30, z: 60 },
  steps: 100,
  opacity: 1,
  accFactor: 2,
  depthTest: false,
  side: FrontSide,
```

```
});
const sliceMesh = heatCloudPlugin.createSliceMesh(mesh.material, {
  depth: 20,
  // opacity: 0.8,
});
sliceMesh.position.set(-500, -500, -500);
```

参数

options

- **描述:** 配置
- **必填:** ✗
- **类型:** `SliceMaterialOptions & CreateImageData3DTextureFromGradientOptions`

属性	描述	类型	必填	默认值
map	三维的纹理	GradientData3DTexture	✗	
opacity	透明度	number | null	✗	1
axis	切片截面垂直于该轴	Axis	✗	Axis.z
depth	轴上的坐标值	number | null	✗	0
range	数值的映射区间 l	IVector2 | null	✗	VolumeMate
discardOut	否丢弃超出映射范围的数值	boolean | null	✗	VolumeMate
voidRange	颜色累积系数	IVector2 | null	✗	VolumeMate
gradient	梯度映射纹理	TexImageSource | ColorGradient	✗	VolumeMate
voidColor	空值颜色	[number, number, number, number]	✗	

createImageSlice

创建切片 dom

SoonSpace.js 2.x

定义

> `VolumeMaterial` 为 `VolumeMesh` 的材质

```ts
createImageSlice(material: VolumeMaterial, options?: CreateImageData3DTexture
```

```ts
/**
 * 获取3D图像切片的工具
 * @remarks
 * 提供了用于呈现切片的 canvas 元素;
 * 生成切片图像的url;
 * 获取指定坐标的颜色等等;
 */
export declare class ImageData3DTextureSlice extends ImageData3DSlice {
  readonly isImageData3DTextureSlice = true;
  constructor(texture: IImageData3 | Texture3D);
  /**
   * 设置材质或 IImageData3 类数据
   */
  set texture(texture: IImageData3 | Texture3D);
}
```

使用

```ts
const ssp = new SoonSpace({
  el: "#viewA",
  options: {
    showInfo: true,
    showGrid: false,
    background: {
      color: new Color(0x444444),
    },
  },
  events: {
    selectPosition: console.log,
    modelClick: console.log,
  },
});
```

```
    { x: 50, y: 70, z: 50, value: 50, radius: 20 },
  ];
  // const mesh = heatCloudPlugin.createLineHeat(linePoints);
  const mesh = heatCloudPlugin.createLineHeat(linePoints, {
    range: { x: 0, y: 50 },
    voidRange: { x: -100, y: -50 },
    size: { x: 90, y: 30, z: 60 },
    steps: 100,
    opacity: 1,
    accFactor: 2,
    depthTest: false,
    side: FrontSide,
    discardOut: false,
    fit: VolumeFit.Fill,
  });
  const sliceMesh = heatCloudPlugin.createSliceMesh(mesh.material, {
    depth: 20,
    // opacity: 0.8,
  });
  sliceMesh.position.set(-500, -500, -500);
  const imageSlice = heatCloudPlugin.createImageSlice(mesh.material);
  //同步切片的深度
  imageSlice.depth = sliceMesh.depth;
  document.getElementById(domContainer).appendChild(imageSlice.canvas);
```

参数

options

- **描述:** 配置
- **必填:** ✗
- **类型:** `CreateImageData3DTextureFromGradientOptions`

属性	描述	类型	必填	默认值
range	数值的映射区间 l	IVector2 | null	✗	VolumeMaterial.ra
discardOut	否丢弃超出映射范围的数值	boolean | null	✗	VolumeMaterial.di
voidRange	颜色累积系数	IVector2 |	✗	VolumeMaterial.vo

		null		
gradient	梯度映射纹理	TexImageSource | ColorGradient	✗	VolumeMaterial.gr
voidColor	空值颜色	[number, number, number, number]	✗	

← plugin-heat-map plugin-drawing-shape →