gv-cesium-plugins

Cesium共性插件,基于Vue+Cesium+elementUI开发,此插件的目的是为了减少大家在项目实施过程中的重复性工作,使用过程中有任何问题或有新的需求都可以通过飞秋(192.168.23.104)联系我。

Install

共性插件被打包成npm镜像发布在公司内网的npm服务器上,你可以通过 `npm i gv-cesium-plugins` 安装它,但是在这之前你需要配置你的npm

内网npm服务器的地址为 `http://192.168.23.216:4873`

npm换源的方法:

1. 临时换源,只生效一次,缺点:每次安装共性插件前都要先执行该命令

```
npm --registry http://192.168.23.216:4873
```

2. 永久换源,缺点:如果你要安装其它npm包需要切换到互联网上的npm服务器,因为内网服务器目前没有这些包.

```
npm config set registry http://192.168.23.216:4873
```

3. nrm换源
   如果你的需要经常切换npm镜像,推荐使用nrm,若你的电脑可以访问互联网,你可以通过下面命令安装

```
npm i nrm -g
```

如果你不能访问互联网,可以在feiQ的共享文件中下载nrm.zip,里面有具体的使用方法.

- 添加本地npm镜像: `nrm add gvnpm http://192.168.23.216:4873`
- 查看可用的npm镜像: `nrm ls`
- 切换npm镜像: `npm use gvnpm`

将npm镜像切换为http://192.168.23.216:4873后,你就可以使用 `npm i gv-cesium-plugins` 安装共性插件了.

组件

1 基础标绘

包括点、线、面、文字和模型标绘，支持编辑和导入，导出。

在线示例

1.1 Usage

1. 按需引入

```
<template>
    <gv-drawPanel :viewer="viewer"></gv-drawPanel>
</template>
<script>
import {gvDrawPanel} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    data(){
        return{
            viewer:undefined
        }
    }
    components:{gvDrawPanel},
    mounted(){
        this.viewer=new Cesium.Viewer('mapId')
        //window.viewer=new Cesium.Viewer('mapId')
    }
}
</script>
```

2. 全局引入

```
//main.js
import Vue from 'vue'
import 'gv-cesium-plugins/dist/theme/default.css'
import gvplugin from 'gv-cesium-plugins'

Vue.use(gvplugin)


//you-component.vue
<template>
    <gv-drawPanel :viewer="viewer"></gv-drawPanel>
</template>
<script>

export default{
    data(){
        return{
            viewer:undefined
        }
    }
    mounted(){
        this.viewer=new Cesium.Viewer('mapId')
    }
}
</script>
```

> 如果Viewer对象没有存储在 `Vue data` 或 `window.viewer` 中，需要显式调用组件的init方法初始化组件,该方法适用于以下所有组件。

```
<template>
    <gv-drawPanel ref='draw'></gv-drawPanel>
</template>
<script>
import {gvDrawPanel} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvDrawPanel},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.draw.init(viewer)
    }
}
</script>
```

1.2 Config

参数	作用	默认值	是否必须	备注
viewer:Viewer	Cesium Viewer对象		否	如果没有传递该参数需要调用init文件初始化组件
extendMarkerImage:Array	扩展标记图片	[]	否	
extendMarkerModel:Array	扩展标记模型	[]	否	如果不定义，模型标记将不可用

- 示例

```
<template>
    <gv-drawPanel ref='draw'
    :extendMarkerImage='images'
    :extendMarkerModel='models'></gv-drawPanel>
</template>
<script>
import {gvDrawPanel} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    data(){
        return{
            images:[
                "./static/images/markers/1.png",
                "./static/images/markers/2.png",
                "./static/images/markers/3.png"
                ],
            models:[
                {
                    id: "model0", //唯一
                    name: "木塔", //显示的名称
                    url: "static/model/Wood_Tower.gltf",//模型url
                    thumb:'static/images/thumb.png'//模型缩略图
                },
                {
                    id: "model1",
                    name: "人",
                    url: "static/model/Cesium_Man.gltf"
                }
                ]
        }
    },
    components:{gvDrawPanel},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.draw.init(viewer)
    }
}
</script>
```

1.3 Methods

- getById(gvid) 根据id返回图形要素

- updateGraphic(id,info) 将图层信息同步到图层管理器
- attributeLabelMap(map) 设置属性字段别名,需要注意的是如果只给部分字段设置了别名,没有设置别名的字段将不会在属性表中显示
- importFromGeoJson(json) 从geojson字符串导入

1.4 Events

事件	说明	回调
deleteEvent	要素删除事件	参数为删除要素的id
locateEvent	要素定位事件	定位要素的id
editEvent	要素编辑事件	要素的id
renameEvent	要素重命名事件	两个参数,依次为要素id,要素更新前的名称
selectEvent	checkbox点击事件	两个参数,依次为要素id,checkbox状态
closeEvent	标绘面板关闭事件	无
addEvent	要素添加事件	要素id
updateEvent	要素更新事件	要素id
attributeUpdateEvent	属性表更新事件	要素id

> ps:
>
> 1.可以通过事件返回的id,调用getById方法获得相应的要素;
>
> 2.所有组件都有一个closeEvent事件,在点击关闭按钮时触发,后面不再单独说明;

1.5 自定义界面

> 仅v 0.1.8以上版本支持自定义界面

```
<gv-drawPanel>
  <template slot-scope="scope" slot="marker-main">
  <!--默认为两个输入框-->
  </template>
  <template slot="marker-aside">
  <!--默认为图标选择按钮-->
  </template>
  <template slot="marker-footer">
  <!--默认为确认和删除按钮-->
  </template>
</gv-drawPanel>
```

- 示例

```
<gv-drawPanel ref="marker" style="position:absolute; left:0px">
    <template slot-scope="scope" slot="marker-main">
      <!-- 这里随便定义，比如div,input,select等，样式要自己设置 -->
      <div class="main">
        <span class='row'>
          <span class="key">
            类型
          </span>
          <el-select v-model="markerType">
            <el-option value="医院"></el-option>
          </el-select>
        </span>
        <span class="row">
          <span class="key">名称</span>
          <el-input size='mini' v-model='scope.marker.markName'></el-input>
        </span>
        <span class='row'>
          <span class="key">描述</span>
          <el-input size='mini' v-model='scope.marker.description'></el-input>
        </span>
      </div>
    </template>
  </gv-drawPanel>
```

2.地图量算

包括贴地距离、空间距离、面积和高度量算。

在线示例

2.1 Usage

```
<template>
    <gv-measurePanel ref='measure'></gv-measurePanel>
</template>
<script>
import {gvMeasurePanel} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvMeasurePanel},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.measure.init(viewer)
    }
}
</script>
```

2.2 Config

参数	作用	默认值	是否必须	备注
viewer:Viewer	Cesium Viewer对象		否	如果没有传递该参数需要调用init文件初始化组件

3. 导航工具

包括指北针、比例尺、2/3切换和缩放。

在线示例

3.1 Usage

```
<template>
    <gv-navigation ref='nav'></gv-navigation>
</template>
<script>
import {gvNavigation} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvNavigation},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.nav.init(viewer)
    }
}
</script>
```

3.2 Config

参数	作用	默认值	备注
viewer:Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件
zoom:Bool	是否显示缩放按钮	true	
reset:Bool	是否显示复位按钮	true	
sceneMode:Bool	是否显示2/3D切换按钮	true	
scale:Bool	是否显示比例尺	true	
compass:Bool	是否显示罗盘	true	
defaultResetView:Cartesian3|Cartographic|Rectangle	默认位置		

4. 信息栏

包括：

- 3个位置参数（经度、纬度、高度）
- 3个相机参数（heading、pitch、roll）
- 2个性能参数（帧率、延迟）

4.1 Usage

```
<template>
    <gv-viewerInfo ref='info'></gv-viewerInfo>
</template>
<script>
import {gvViewerInfo} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvViewerInfo},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.info.init(viewer)
    }
}
</script>
```

4.2 Config

默认情况下，组件在开发环境下显示所有参数，在生产环境下显示经纬度、视高3个参数，其它的参数我称之为调试参数。

参数	作用	默认值	备注
viewer:Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件

参数	作用	默认值	备注
debugger|Bool	是否显示调试参数		
moreParams|Array	自定义要显示的调试参数	["fps", "ms", "heading", "pitch", "roll"]	可选值见默认值

5.底图选择器

底图选择器将底图分为影像地图、矢量地图、道路注记、地形高程四种。

在线示例

5.1 Usage

```
<template>
    <gv-baseLayerPicker ref='lp'></gv-baseLayerPicker>
</template>
<script>
import {gvBaseLayerPicker} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvBaseLayerPicker},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.lp.init(viewer)
    }
}
</script>
```

5.2 Config

参数	作用	默认值	备注
viewer:Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件
presetLayers|Array	预设底图		

如何自定义底图?

```
//一个有效的Cesium ImageryProvider或TerrainProvider
const provider=...
const layer={
    id:'mylayer',//id
    name:'My Layer',//name
    title:'My custom layer',//title
    icon:'static/image/xx.png',//icon
    provider:provider//provider
}
//然后把layer放在presetLayers参数中传过去就行了
```

6. 卫星仿真

支持任意卫星的轨道模拟。

在线示例

6.1 Usage

```
<template>
    <gv-obritPredictor ref='sate'></gv-obritPredictor>
</template>
<script>
import {gvObritPredictor} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvObritPredictor},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.sate.init(viewer)
    }
}
</script>
```

6.2 Config

参数	作用	默认值	是否必须	备注
viewer:Viewer	Cesium Viewer对象		否	如果没有传递该参数需要调用init文件初始化组件
preset|Array	预设要显示的卫星	['gaofen']	否	模糊匹配，不区分大小写
tle|String	tle文件路径		是	
currentPosition|Bool	是否显示星云，即卫星当前位置	true	否	
dispose|Bool	组件销毁时是否移除数据	true	否	

7. 专题图制作

在线示例

- 说明

使用该功能需要在初始化Viewer时加入以下参数。

```
const opt={
    ...
    contextOptions: {
        //cesium状态下允许canvas转图片convertToImage
        webgl: {
          preserveDrawingBuffer: true
        }
      }
}
const viewer=new Cesium.Viewer(opt)
```

7.1 Usage

```
<template>
    <gv-mapEditor ref='map'></gv-mapEditor>
</template>
<script>
import {gvMapEditor} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvMapEditor},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.map.init(viewer)
    }
}
</script>
```

7.2 Config

参数	作用	默认值	是否必须	备注
viewer|Viewer	Cesium Viewer对象		否	如果没有传递该参数需要调用init文件初始化组件

8.天气模拟

包括太阳、月亮、雨、雪、雾、云、光等。

8.1 Usage

```
<template>
    <gv-weatherPanel ref='lp'></gv-weatherPanel>
</template>
<script>
import {gvWeatherPanel} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvWeatherPanel},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.lp.init(viewer)
    }
}
</script>
```

8.2 Config

参数名称	作用	默认值	备注
viewer|Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件

functions|Object	需要显示哪些天气特效工具开关	{
		lighting:true,
		sun:true,
		moon:true,
		snow:true,rain:true,cloud:true,
		fog:true,
		groundAtmosphere:true
		}
			值为true表示显示该工具，
			如 sun:true 表示界面显示太阳特效开关，
			如 sun:false 表示界面不显示太阳

positions|Array	雨、雪、云添加的位置		雨、雪、云使用时此为必须参数

9. 场景设置

包括亮度、景深、阴影、泛光、耀斑

9.1 Usage

```
<template>
    <gv-sceneSetting ref='setting'></gv-sceneSetting>
</template>
<script>
import {gvSceneSetting} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvSceneSetting},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.setting.init(viewer)
    }
}
</script>
```

9.2 Config

参数名称	作用	默认值	备注
viewer|Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件

10. 鹰眼

鹰眼功能,双图联动,小地图基于leaflet实现.

在线示例

10.1 Usage

```
<template>
    <gv-eagleEye ref='eye'></gv-eagleEye>
</template>
<script>
import {gvEagleEye} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvEagleEye},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.eye.init(viewer)
    }
}
</script>
```

10.2 Config

参数名称	作用	默认值	备注
viewer|Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件
status|Bool	鹰眼默认状态	false	true表示展开
eagleMap|String	小地图所要显示的地图url	osm地图	必须是tms服务

如果鹰眼要显示其它服务（**WMS,WTMS**）的地图，请调用**setLayer**方法设置

11. 时间轴

时间序列影像对比(卷帘).

在线示例

11.1 Usage

```
<template>
    <gv-timeLine ref='tl'></gv-timeLine>
</template>
<script>
import {gvTimeLine} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvTimeLine},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.tl.init(viewer)
    }
}
</script>
```

11.2 Config

参数名称	作用	默认值	备注
viewer|Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件
config|Object	时间轴相关配置		详情见示例
timelineData|Object	时间轴数据		

时间轴的配置我觉得还是比较复杂的，因为它是基于开源js时间轴Timelinejs封装的，所有它的所有配置
其实是Timelinejs的配置。

- 示例

```
<template>
    <gv-timeLine ref='tl' :timelineData='timelineData' :config="config"></gv-timeLine>
</template>
<script>
import {gvTimeLine} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    data(){
        return{
            timelineData:{events:[]}
            config:{
        checkResize: true,
        //buildItemContent定义了时间轴显示的内容
        //如下，定义了一个img标签，用于显示影像的缩略图
        //另外定义了一个checkbox，用于表示影像的状态
        //其中evt是timelineData中值
        //这里定义了checkbox点击后执行的函数layerAction
        //你可以在这个函数里定义你要做的内容
        buildItemContent: (evt) => {
          return `<img src='${evt.thumb}' title='${evt.startDate}' onerror="this.src='new.png'"/>
                    <input onClick="layerAction(this,'${evt.url}',
                    '${evt.type}','${evt.geom}','${evt.startDate}')" id='${evt.id}' type='checkbox'/><span>${evt.startDate}
                    </span>`;
        }
      },
        }
    }
    components:{gvTimeLine},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.tl.init(viewer)
        window.layerAction=function(url,type,geom,start){
            //做你要做的事
            //比如加载影像
            //比如定位到影像所在的位置
        }
    }
}
</script>
```

12. 卫星过顶分析

分析指定区域某时间段内飞过的卫星,不过虑侧摆等复杂参数.

在线示例

12.1 Usage

```
<template>
    <gv-transit ref='transit'></gv-transit>
</template>
<script>
import {gvTransit} from 'gv-cesium-plugins'
import 'gv-cesium-plugins/dist/theme/default.css'
export default{
    components:{gvTransit},
    mounted(){
        const viewer=new Cesium.Viewer('mapId');
        this.$refs.transit.init(viewer)
    }
}
</script>
```

12.2 Config

参数名称	作用	默认值	备注
viewer|Viewer	Cesium Viewer对象		如果没有传递该参数需要调用init文件初始化组件
tle|string	tle文件路径		

接口

除了以上组件外, `gv-cesium-plugins` 还定义了一些接口，这些接口定义在GV对象下，使用方法如下：

```
import 'gv-cesium-plugins/dist/extension.umd.min'
```

导入上面的扩展之后会自动将这个接口注册到window,你可以通过window.GV.methodName使用它们。

1. 热力图

在线示例

```
new GV.HeatMap(viewer,options)
```

参数	描述	必须	备注
viewer	Viewer对象	是	
options.geojson	通过geojson对象创建热图	否	
options.url	通过geojson文件创建热力图	否	
options.value	各点位的权重值	否	

方法

1. `zoomTo()`

- 作用:定位到热力图位置

2. `remove()`

- 作用：清除热力图

2. 近景天空盒

在线示例

```
new GV.SkyBoxOnGround(options);
```

参数	描述	必须	默认值	备注
options.sources	定义天空盒的6张图片	是		

属性/方法

> SkyBoxOnGround继承了Cesium.SkyBox的所有属性和方法

1. `static DUSH`

- 预设天空盒

示例

```
const options={}
const viewer=new Cesium.Viewer('mapId',options);
//大气层会遮挡天空盒
viewer.scene.skyAtmosphere.show=false
viewer.scene.skyBox = new GV.SkyBoxOnGround({
      sources: GV.SkyBoxOnGround.DUSH
});
//DUSH为预设天空盒，SkyBoxOnGround的定义和Cesium.SkyBox相同
viewer.scene.skyBox = new GV.SkyBoxOnGround({
      sources: {
    positiveX : 'skybox_px.png',
    negativeX : 'skybox_nx.png',
    positiveY : 'skybox_py.png',
    negativeY : 'skybox_ny.png',
    positiveZ : 'skybox_pz.png',
    negativeZ : 'skybox_nz.png'
  }
});
```

3. 坐标转换工具

方法

1. `static toPixel(position,viewer)`

- 转屏幕坐标

参数	类型	描述	必须	默认值	备注
position	Carteisan3|Cartographic		是		
viewer	Viewer		是		

2. `static toDegrees(position,viewer)`

- 转经纬度

参数	类型	描述	必须	默认值	备注
position	Carteisan3|Cartesian2		是		
viewer	Viewer		是		

3. `static toDegrees(position,viewer)`

- 转经纬度(弧度)

参数	类型	描述	必须	默认值	备注
position	Carteisan3|Cartesian2		是		
viewer	Viewer		是		

4. `static toCartesian(position,viewer)`

- 转笛卡坐标

参数	类型	描述	必须	默认值	备注
position	Cartesian2		是		
viewer	Viewer		是		

示例

```
//屏幕坐标转变经纬度
const pixel=new Cartesian2(0,0);
GV.toDegrees(pixel,viewer)
GV.toRadians(pixel,viewer)
//笛卡尔坐标转经纬度
const cartesian=new Cartesian3(0,0,0)
GV.toDegrees(cartesian,viewer)
GV.toRadians(cartesian,viewer)
//经纬度坐标转屏幕坐标
const cartographic=new Cartographic(0,0,0)
GV.toPixel(cartographic,viewer)
//笛卡尔坐标转屏幕坐标
GV.toPixel(cartesian,viewer)
```

4.可拖拽的窗口

使窗口可以通过鼠标拖动,该元素的position必须为position或fixed

```
/**
 * 通过鼠标托动div。
 * @param {string} container 要拖动的div的id
 * @param {string} target 触发拖动事件的div的id，一般指标题栏，如果不设置将使用container
 */
GV.moveDiv(container,target)
```

参数	描述	必须	备注
container	要拖动的div的id	是	
target	触发拖动事件的div的id	否	一般指标题栏，如果不设置将使用container

5. 鼠标跟随信息

```
new GV.CursorTip(text,id,viewer)
```

参数	描述	必须	备注
text	要显示的内容	是	可以包含HTML元素
id	元素id	否	
viewer	Viewer	否	如果不定义将无法自动跟随鼠标移动,需要通过updatePosition方法控制

方法

1. `updateText(text)`

- 作用：更新显示内容

```
tip.updateText('hello')
```

2. `updatePosition(pixel)`

- 作用：更新元素位置

```
hander.setInputAction(e=>{
    tip.updatePosition(e.endPosition)
},MOUSE_MOVE)
```

属性

1. `visible`

- 获得/设置元素可见状态

```
tip.visible=false
```

6. 保存当前屏幕

```
GV.saveCurViewerImage(viewer,'file.png')
```

7. 3dtileset模型加载

在线示例

```
GV.loadTileset(viewer,url,options={},height=undefined,position=undefined,degub=false)
```

参数	描述	必须	默认值	备注
viewer	Viewer	是		
url	模型url	是		
options	描述模型的参数	否		和Cesium3dTileset一致
height	高度调整	否		
position	位置调整	否		
debug	打开调试模型	否	false	用于微调模型位置,调试模式下可以通过ASDQWEXZ按键控制模型

示例

```
const viewer=new Cesium.Viewer('map')
//最简单的调用
GV.loadTileset(viewer,'static/model/tileset.json')
//调整参数
GV.loadTileset(viewer,'static/model/tileset.json',{
    maximumScreenSpaceError:5
})
//调整高度
GV.loadTileset(viewer,'static/model/tileset.json',{},0)
//调整位置
const pos=Cesium.Cartesian3.fromDegrees(110,40)
GV.loadTileset(viewer,'static/model/tileset.json',{},undefined,pos)
```

8. 移除 Cesium logo

```
GV.removeLogo(nweLogo=false)
```

参数	描述	默认值	备注
nweLogo	是否添加新logo	false	如果该值为true，在移除默认logo的同时会添加一个新的logo， 如果要自定义logo图片，可以传递一个url

9.交互绘图(线,多边形)

这是从CesiumDrawViewer衍生出来的接口,作用是在三维场景通过鼠标添加一个polyline或polygon.

```
new GV.GraphicManager(viewer)
```

参数	描述	必须	备注

参数	描述	必须	备注
viewer	Viewer	是	

方法

1. `createPolyline(options=CesiumPolyline.defaultStyle)`

- 作用:创建线段,调用该方法后会自动注册鼠标监听事件,然后利用鼠标交互添加线段

参数	描述	默认值	备注
options	描述线段的参数	CesiumPolyline.defaultStyle	同Cesium.PolylineGraphic

2. `createPolygon(options=CesiumPolygon.defaultStyle)`

- 作用:创建多边形,调用该方法后会自动注册鼠标监听事件,然后利用鼠标交互添加多边形

参数	描述	默认值	备注
options	定义一个多边形要素	CesiumPolygon.defaultStyle	和Cesium.PolygonGraphics相同

3. `cancel()`

- 作用:取消添加要素,如果执行了createPolyline或createPolygon又不想画了,用它取消

4. `edit(id)`

- 作用:编辑要素

参数	描述	必须	默认值	备注
id	要素id	是		

5. `destroy()`

- 作用:销毁对象

10 创建三维场景（地球）

封装了创建地球时的一些默认设置,当然你可以通过重新设置来覆盖它们.

```
GV.createEarth(container,options)
```

参数	描述	必须	默认值	备注
container	容器id	是		
options	同Cesium.Viewer	否		

11 面积计算

贴地面积计算

```
GV.surfaceArea(cartesianArr)
```

参数	类型	描述	单位
cartesianArr	Array[Cartesian3]	多边形顶点	平方米

12 距离计算

```
GV.surfaceDiatance(cartesianArr)
```

- 作用:计算贴地距离

参数	类型	描述	单位
cartesianArr	Array[Cartesian3]	线段顶点	米

```
GV.spaceDiatance(cartesianArr)
```

- 作用:计算空间距离

参数	类型	描述	单位
cartesianArr	Array[Cartesian3]	线段顶点	米

13.将当前屏幕保存为图片

```
GV.saveCurViewerImage(viewer,filename)
```

参数	描述	必须	类型
viewer	Viewer	是	Cesium.Viewer
filename	文件名	是	string

14. 挖方分析

在线示例

ps:该方法目前尚不完善,有两个问题:一是只能画凸多边形,二是只能逆时针画

```
new GV.ClipAnalysis(viewer,options={})
```

参数	描述	必须	默认值	备注
viewer	Viewer对象	是		
options.fillMaterial	填充的材质	否	Color.WHITE.withAlpha(0.5)	
options.edgeColor	挖方边界颜色	否	Color.WHITE	
options.edgeWidth	挖方边界线的宽度	否	0	

属性

1. `show`

- 获得或设置当前可见性

方法

1. `do()`

- 开始分析

参数	描述	必须	备注
positions	挖方顶点坐标	否	如果为空将通过鼠标交互绘制

2. `clear()`

- 清除结果

3. `destroy()`

- 销毁对象

15.剖面分析

在线示例

```
new GV.ProfileAnalisys(viewer,options);
```

参数	描述	必须	默认值	备注
viewer	Viewer对象	是		
options.terrain	用于计算高程的地形数据	是	viewer.terrainProvider	
options.step	两点之间插值的个数,如果太多会影响计算速度,如果太小会影响精度			

方法

1. `do()`

- 开始分析

2. `clear()`

- 清除分析结果

3. `destroy()`

- 销毁对象

16 水文(淹没)分析

在线示例

```
new GV.HydrologyAnalysis(viewer,options)
```

参数	描述	必须	默认值	备注
viewer	Viewer对象	是		
options.step	淹没速度	否	1	m/s
options.maxHeight	最大淹没高度	是		

方法

1. `do()`

- 作用:开始分析
- 参数:同构造函数

2. `remove()`

- 作用:清除分析结果

3. `destroy()`

属性

1. `time`

- 淹没开始经过的时间
  2. `height`
- 当前淹没高度

事件

1. `hydrology-done`

- 作用:达到最大淹没高度时触发

17. 视域分析

在线示例

```
new GV.ViewshedAnalysis(viewer,options)
```

参数	描述	必须	默认值
viewer	Viewer对象	是	
options.observer	相机所在位置	是	
options.viewPosition	视点位置	否	options.observer

属性

1. `viewPosition`

- 视线最远点所在的位置

2. `direction`

- 视线方向

3. `observe`

- 观察都位置,即相机原点

4. `readonly frustum`

- 视锥primitive

5. `far`

- 远裁剪面

6. `near`

- 近裁剪面

7. `aspectRatio`

- 裁剪面的宽高比

8. `readonly shadowMap`

- 可视范围渲染的primitive

方法

1. `destroy()`

18. 日期格式化

就是将时间处理成 `YYYY-mm-dd HH:MM:SS` 的形式,一般时间用于显示时都需要处理成这种格式.

ps:和其它接口不同的是,如果你引用了共性接口文件,这个函数将会被注册到Date的prorotype上.而不是GV对象上.

```
const now =new Date("2020-4-22 9:1:3")
now.format('YYYY-mm-dd HH:MM:SS')
//2020-04-22 09:01:03
now.format('YYYY-mm-dd')
//2020-04-22
now.format('YYYY-mm-dd HH:MM:SS')
//2020-04-22 09:01:03
now.format('HH:MM:SS')
//09:01:03
now.format('YYYY-mm-dd H:M:S')
//2020-04-22 9:1:3
```

19.圆形扩散

在线示例

```
new GV.CircleScan(viewer,center,maxRadius,color,duration)
```

参数	描述	必须	默认值
viewer	Viewer对象	是	
center	中心位置	是	
radius	扫描半径	是	
color	颜色	否	Color.RED
duration	一个周期所需要的时间,单位毫秒	否	4000

方法

1. `add()`
   - 将图形添加到场景中
2. `remove()`
   - 从场景中移除
3. `destroy()`
   - 销毁对象,释放资源

20.雷达扫描

在线示例

```
new GV.RadarScan(viewer,center,radius,color,duration)
```

参数	描述	必须	默认值
viewer	Viewer对象	是	
center	中心位置	是	
maxRadius	最大扩散半径	是	
color	颜色	否	Color.RED

参数	描述	必须	默认值
duration	一个周期所需要的时间,单位毫秒	否	4000

方法

1. `add()`

- 将图形添加到场景中

2. `remove()`

- 从场景中移除

3. `destroy()`

- 销毁对象,释放资源

21.流动线(纹理)

在线示例

```
new GV.PolylineTrailLinkMaterialProperty(color,duration)
```

参数	描述	必须	默认值
color	颜色	是	
duration	一个周期所需要的时间,单位毫秒	是	

22.Echarts叠加

在线示例

```
new GV.EchartsLayer(viewer,options)
```

参数	描述	必须	备注
viewer	Viewer对象	是	
options.animation	是否显示动画	是	必须为false
options.GVMap	自定义地图相关描述	是	一般定义为一个空对象
options.series	同Echarts的series定义	是	coordinateSystem必须为GVMap

方法

1. `update(options,merge)`

- 更新数据

参数	描述	必须	默认值
options	同构造函数中的options	是	
merge	是否保留原来的数据	否	false

2. `destroy()`

- 销毁对象,释放内存

属性

1. `visible`

- 设置地图可见性

23.MapV叠加

在线示例

```
new GV.MapVLayer(viewer,dataset,options,container)
```

ps:和官方定义方法完全相同,请参数官网

24 Primitive箭头线

它画的是一条以某个点为中心,延Z轴延伸固定长度得到的直线,该方法开法的目的是为了画坐标轴,目前不清楚是否会有别的用途

```
new GV.ArrowPolyline(options)
```

参数	描述	默认值
options.position	指线的中心点位置	
options.color	颜色	Color.RED
options.width	线宽	3
options.headWidth	箭头最宽值	options.width*2
options.length	线的长度	300
options.headLength	箭头长度	10
options.id	id	random

25.模型剖切

在线示例

```
new GV.ModelClipPlane(viewer,model,options)
```

参数	描述	必须	默认值
viewer	Viewer对象	是	
model	Cesium.Model或Cesium.Cesium3DTileset对象	是	
options.type	剖切类型	否	
options.distance	剖切距离	否	
options.showAxis	是否显示剖切坐标轴	否	false
options.axis	定义坐标轴	否	
options.axis.length	坐标轴长度	否	
options.axis.width	坐标轴半径	否	
options.axis.headLength	箭头长度	否	
options.axis.headWidth	箭头半径	否	

属性

1. `static Type`

- `GV.ModelClipPlane.Z` 延Z轴剖切
- `GV.ModelClipPlane.ZR` 延Z轴反向剖切
- `GV.ModelClipPlane.X` 延X轴剖切
- `GV.ModelClipPlane.XR` 延X轴反向剖切
- `GV.ModelClipPlane.Y` 延Y轴剖切
- `GV.ModelClipPlane.YR` 延Y轴反向剖切

2. `distance`

- 返回或设置剖切距离

3. `showAxis`

- 返回或设置坐标轴可见性

4. `type`

- 显示或设置剖切类型

ps:有效值为Type的属性

方法

1. `createPlane(type,distance,clear)`

参数	描述	必须	默认值	积液
type	剖切类型	否	this.type	如果构造函数中未定义type,此处必须传
distance	剖切遗弃	否	this.diatance	distance,此处必须传
clear	是否清除上一个剖切面	否	true	

2. `clear()`

- 清除剖切效果

3. `destroy()`

- 销毁对象,释放资源

26.开场动画

在线示例

```
GV.flyTo(viewer,options)
```

参数	描述	默认值
viewer	Viewer对象	
options.destination	目的地	
options.orientation	方向	
options.duration	飞行持续的时间	

27.动态水面

在线示例

```
new GV.WaterFace(viewer,options)
```

参数	描述	默认值
viewer	Viewer对象	
options.image	作为水面纹理的图片	
options.alpha	水面透明度	
options.potitions	多边形顶点	
options.extrudedHeight	多边形拉升高度	

方法

1. `add()`
   - 将水面添加到场景
2. `update(positions,extrudedHeight)`
   - 更新水面范围
3. `zoomTo()`
   - 定位到水面所在位置
4. `destroy()`
   - 消除水面,销毁对象
5. `static waterFaceApperace(image)`
   - 创建水面Appearance

28.风场

在线示例

```
new GV.WindField(viewer,options)
```

28.加载Gif图片

```
GV.loadGif(url,imgs)
```

参数	描述	默认值
url	图片url	
imgs	gif图片的每一帧	[]

29.拉框选择

在线示例

```
new GV.RectangleSelector(option)
```

参数	描述	默认值	备注
option.left	用于捕获画面的有效范围	0	

参数	描述	默认值	备注
option.top	用于捕获画面的有效范围	0	
option.width	用于捕获画面的有效范围	body的宽	
option.height	用于捕获画面的有效范围	body的高	
buttonMode	是否添加确定/取消按钮	false	
option.mask	选择时是否添加掩膜		

方法

1. `on()`

- 开始监听鼠标事件绘图

2. `destroy()`

- 销毁对象

事件

1. `selector-submit`

- 绘图结果提交时触发

2. `selector-cancel`

- 绘图结果取消时触发

30.判断点是否在球的背面

```
GV.pointVisibilityOnEarth(point,viewer)
```

参数	描述	默认值
point	Cartesian坐标	
viewer	Viewer对象	

更新日志

v0.1.4

- 更新时间:2020.4.17
- 更新内容:

1. 规范了组件名称,所有共性插件以gv-component的形式命名.
2. 添加了卫星过顶分析组件gv-transit.
3. 鹰眼(gv-eagleEye)添加了从小地图到大地图的联动,可以通过在小地图点击或拉模框定位.
4. 对代码做了静态检查.

v0.1.5

- 更新时间:2020.4.24
- 更新内容:基础标绘组件添加了属性表,并暴露了相应的事件,使得利用该组件添加的要素可以保存的数据库(包括几何信息和属性信息))

v0.1.6

- 更新时间:2020.4.24
- 更新内容:基础标绘组件添加了检索功能

v0.1.7

- 更新时间:2020.4.26
- 更新内容:

1. 添加了版本信息
2. 添加了拉框选择

v0.1.8

- 更新时间:2020.7.1
- 更新内容:
  标绘插件支持自定义界面