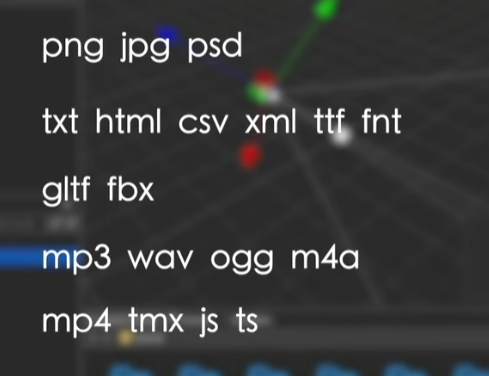
支持格式



调整摄像机】 ctrl+shift+F

吸附】按住V可以选中对应顶点 进行移动吸附

## 装饰器：

*const* { ccclass, property,executeInEditMode } = \_decorator;

\_decorator

不运行条件下执行组件

@executeInEditMode(true)

挂在的组件显示的名:

@ccclass('Launch')

公开组件到节点上

  @property

    public foo = 10;

    @property(Label)

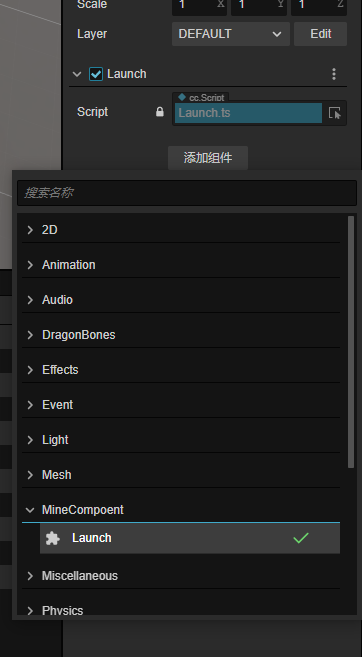
    public label: Label;

强依赖组件

@requireComponent(MeshRenderer)//强依赖组件

挂在组件显示的分类

@menu("MineCompoent/Launch")



## Tween

*var* targetTween: Tween<*any*> = tween(uiOpacity).to(1, { opacity: 2 }).delay(*delayTime*).start().call(*callBack*);

1. Tween类需要跟类型
2. 返回对应的Tween<T>
3. .start().delay()这些函数会先放在一个队列，然后在渲染层执行，

    export *enum* TweenEase{

        "linear", "smooth", "fade", "constant", "quadIn", "quadOut", "quadInOut", "quadOutIn", "cubicIn", "cubicOut", "cubicInOut", "cubicOutIn", "quartIn", "quartOut", "quartInOut", "quartOutIn", "quintIn", "quintOut", "quintInOut", "quintOutIn", "sineIn", "sineOut", "sineInOut", "sineOutIn", "expoIn", "expoOut", "expoInOut", "expoOutIn", "circIn", "circOut", "circInOut", "circOutIn", "elasticIn", "elasticOut", "elasticInOut", "elasticOutIn", "backIn", "backOut", "backInOut", "backOutIn", "bounceIn", "bounceOut", "bounceInOut", "bounceOutIn"

    }

*var* targetTween: Tween<*any*> = tween(uiOpacity).delay(*delayTime*).to(*showTime*, { opacity: 0 }, { easing: *tweenEasing* }).call(*callBack*).start();

Tween 的调用顺序 delay – to -call - start

## 生命周期

onLoad是代码层 相当于unity的awake

start 是渲染层

## JSon

Json转换对象内不能添加字段 不然会无效