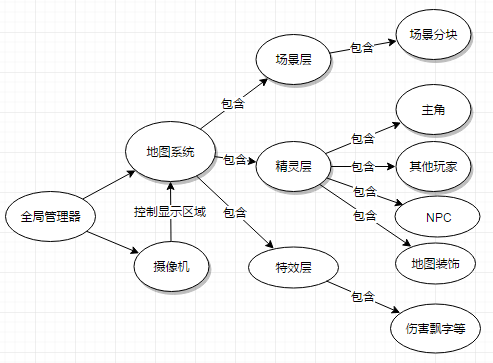
H5游戏地图系统设计

**一、概述**

地图系统作为游戏的核心模块之一，承载着场景、角色、装饰与技能特效的显示，摄像机的运动等。他们之间的关系如下：



全局管理器持有摄像机与地图的引用。

摄像机对地图的显示区域进行控制。

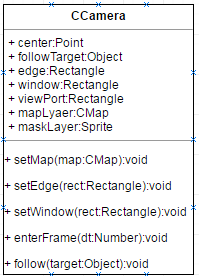
地图包含场景层、精灵层、特效层。

场景层处理场景分块的加载与卸载。

精灵层是主角，其他玩家，NPC，装饰物的容器。

特效层用于显示伤害飘字等特效。

**二、摄像机**



在这里，摄像机设计为一个容器，把要显示的对象（CMap）作为摄像机的子对象

maskLayer对超出viewPort的部分进行遮罩。

viewPort为视口，是最终显示的地图区域。

center为viewPort的中心。

followTarget为摄像机跟随的对象，一般为myself。

edge为地图的边界，摄像机不能超出边界显示。

window为摄像机窗格，当followTarget在window内移动时，不会导致viewPort的变化。

mapLayer为要显示的对象

maskLayer为遮罩

setMap(map:CMap):void添加显示的地图。

setEdge(rect:Rectangle):void设置地图的边缘

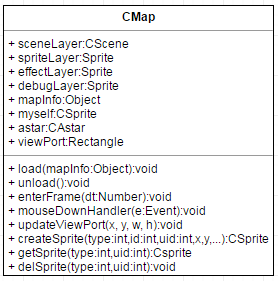
setWidow(rect:Rectangle):void设置摄像机窗格

follor(target:Object):void设置摄像机跟随的对象

enterFrame(dt:Number):void为帧刷新函数，处理摄像机的运动。当跟随的对象运动时，做出相应的运动。一般采用**线性插值**的方法对摄像机的运动进行平滑处理，以达到较好的显示效果。

额外的，CCamera需要处理resize事件，做出相应的更新。

**三、地图CMap**



地图包含场景层sceneLayer、精灵层spriteLayer、特效层effectlayer。

还可以包含一个debugLayer，用于显示阻挡格子等调试用途。

通过createSprite添加精灵，delSprite删除精灵，getSprite获取精灵对象

load加载地图，按需派发加载进度的事件等。在加载完成后创建myself。

unload卸载地图。

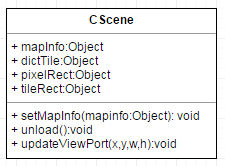
enterFrame处理所有Sprite的更新，对显示对象是否在viewPort内进行相应的处理。隔一定时间对Sprite进行深度排序。

mouseDownHandler处理鼠标的点击事件，判断鼠标是否点中NPC，玩家还是地表。可以派发相应的事件，由关心地图点击的控制器去处理。

updateViewPort更新视口的范围。

注意：少了事件点的设计，应该在加载地图的时候，存储事件点的信息，以备检查使用。

**四、场景CScene**



场景类负责场景分块的加载与卸载。

dictTile存储场景分块的Sprite。

pixelRect 当前显示的像素区域

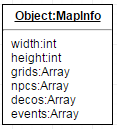
tileRect 当前显示的分块区域

在CMap的updateViewPort中调用CScene的updateViewPort。

遍历tileRect对分块进行添加或移除，可以对矩阵由内到外进行回形遍历，以获得较好的加载体验，应该会比排序消耗要低。

这里为了减少CPU负担，并没有设置马赛克图片。如果有需要，则在setMapInfo的时候，加载一个缩略图，在updateViewPort的时候，如果清晰的图片没有加载完成，则先显示一个马赛克图片，通过drawTextrue实现。

**五、地图数据MapInfo**



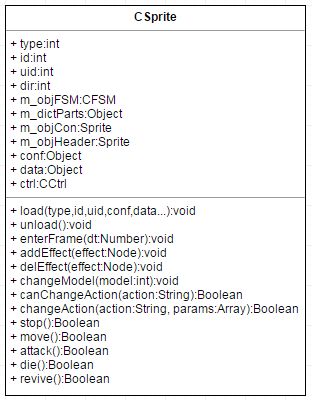
地图数据应该包括，像素宽高，格子阻挡数据。

npc位置数据，装饰物位置数据，事件点数据等。

可能还应该包括出生点的信息。

**六、精灵CSprite**

精灵类用于显示玩家角色，其他角色，以及NPC。



type类型

id 配置表ID

uid

dir 方向，4方向为0-3,8方向为0-7

m\_objFSM 有限状态机

m\_dictParts 部件，如身体、武器、翅膀、坐骑

m\_objCon 部件的容器

m\_objHeader 头顶控件

conf 配置数据

data 服务器数据

ctrl 控制器，如控制行走路径

load加载与初始化

unload卸载回收

enterFrame 更新显示

addEffect 添加特效，如受击特效

delEffect 删除特效

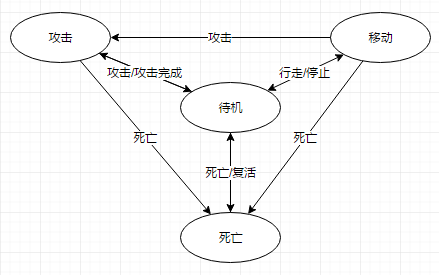
changeModel 更换模型

canChangeAction 能否切换动作

changeAction 切换动作

stop,move,attack,die,revive...切换动作的方便接口

角色的状态设计：

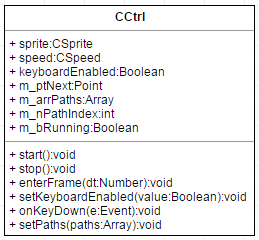


极简版：只有4个状态，待机、移动、攻击、死亡

互相转化的动作(事件)如图

可根据需求增加状态。

**七、精灵控制CCtrl**



目前主要用于精灵移动的控制，可以通过setKeyboardEnabled开启或关闭键盘的控制。

sprite 被控制的精灵

speed 角色移动的速度，具有整体速度、X轴分速度、Y轴分速度以及速度的方向。

keyboardEnabled 指示是否开启键盘控制，一般只有主角才能开启键盘控制

m\_ptNext 将要移动到的位置

m\_arrPaths 移动路径

m\_nPathIndex 当前处理的路径的节点

当m\_arrPaths为空的时候，且keyboardEnabled为true时处理键盘的按方向移动、要考虑阻挡的问题。

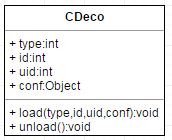
start,stop 开始，停止控制

enterFrame 由CSprite调用，帧处理

onKeyDown 处理键盘按下与弹起事件，可以考虑再多个onKeyUp

setPaths 由服务器广播的其他玩家的移动路径，或者设置A\*寻路的结果路径。

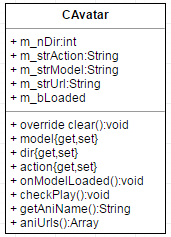
**八、装饰CDeco**

****

装饰为最简单的只支持单张图片。如后期不能满足要求，则可以改为序列帧动画。

CDeco继承自laya.display.Sprite，通过texture属性设置贴图

**九、CAvatar——CSprite的显示部件**



CAvatar是CSprite用于显示的部件。

CAvatar继承自laya.display.Animation，为序列帧动画。

一个动作是由model,action,dir共同决定的

应该有配置文件确定每一个model，对应有哪些action，有哪些dir，每个action有几帧。

clear清理以便复用对象

model,dir,action分别设置属性

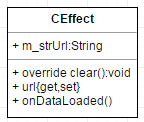
onModelLoaded在设置对应的属性后，加载资源完成后判断如何播放。

checkPlay 判断属性修改后，该如何播放。

getAniName 返回model\_action\_dir组成的key，用于Animation.creatFrames作为cacheName

aniUrls 返回某个动作的所有帧的URL，当没有cache的时候，第一次构建播放的序列使用。

**十、CEffect无方向特效**



无方向特效继承自laya.display.Animation。

通过直接设置url进行播放。

**十一、技能特效**

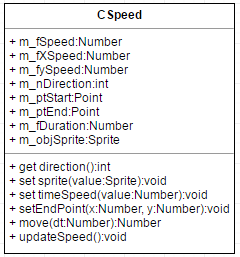
技能特效包括：攻击特效、被击特效以及远程攻击的法球特效。

一般攻击特效可能有方向，可以使用CAvatar去播放。

被击特效无方向，使用CEffect去播放。

远程攻击的法球无方向，但要求旋转，可以使用CEffect设置旋转角度播放。

**十二、速度CSpeed**



速度为CSprite移动服务，基于时间去计算，而不是基于帧。

设置sprite，基础速度timeSpeed（像素/秒），设置终点，计算出移动的方向（八方向），以及单位速度（像素/毫秒）便于计算。

调用move实现对sprite的移动，移动基于sprite的当前位置（可能存在外部改变了位置的情况会导致BUG）。

当move传入的时间大于移动到终点需要的时间时，会返回剩余的时间，以便做折线运动，针对帧率不稳的情况以获得比较精确的结果。

**十三、关于预加载**

在H5平台，基于流量与性能的考虑，尽量不使用预加载的功能。

预加载不应该作为游戏的必须组件而存在，应该是作为PC平台的一个加分项。

预加载的逻辑独立设计，尽量由数据获取需要加载的资源，加入一个预加载的队列，以最低优先级进行加载。