

地图模块优化

轰

September 5, 2017

1 目标

- 减少内存使用用时加载，用后销毁
- 不加载未使用的地图资源

2 方案

2.1 地图加载

- 维护一定量的地图块，人物在移动时，通过设置地图块的坐标，反复利用地图块。
- 当前地图的地图块在使用过，当前不再使用后则设置有效期，过了有效期删除之。再次使用时刷新有效期。
- 切换地图时，清除上一地图的所有地图块。

2.2 地图块

- 地图块大小256x256
- 手机屏幕大小1366?x768

由于

$$768 = 3 \times 256 \quad 1366 \div 256 = 5.336$$

所以需要维护一个4x7 的地图矩阵。如下图：

1	2	3	4

如果地图块不再这28个地图块中，地图块则不显示，即`visible = false`.

```
if (mapBlock4.x + 256 >= 768) { //
    mapBlock1.x = 768;
    let resource = mapBlock1.resource;
    mapBlock1.resource = "next_image_resource";

    //
    ResourceManager.setValidity(resource);
}
```

3 实现

3.1 有效期管理

准备回收的地图块，再次使用的使用从`resources`中删除

```
class ResourceManager {
    public resources = {};
    public timeRange = 25;
    public timer: egret.Timer = new egret.Timer(1000, -1);

    public setValidity(resourceName: string) {
        this.resources[resourceName] = this.timeRange;
    }

    public update() {
```

```

        for (let key in this.resources) {
            this.resources[key] --;
            if (value <= 0) {
                this.destroy(key);
            }
        }
    }

    public destroy(resourceName: string) {

    }

}

```

3.2 地图切换

$$Map_a \rightarrow Map_b$$

```

init_map(mapId: number) {
    // 初始化地图，此时上一个地图里面的资源会进入resources
    // 然后再调用destroyMap
}

destroyMap(mapA) {
    for (let resourceName of mapA) {
        if (resources[resourceName]) {
            destroy(resourceName);
            resources[resourceName] = null;
            delete resources[resourceName];
        }
    }
}

```

3.3 地图初始化

```

for (let i = 0; i < 7; ++i) {
    for (let j = 0; j < 4; ++j) {
        init_map_block(i, j);
    }
}

init_map_block(i, j) {
    let image = getImage(i, j);
}

```

```

        image.source = getResource(i, j); //若地图块不存在返回""
    }

```

3.4 新进入一个地图的初始化

- 地图的初始化位置为initialX initialY
- 地图大小为width height
- 地图块row col

> Input: initialX initialY width height row col

```

for (let i = 0; i < row; ++i) {
    for (let j = 0; j < col; ++j) {
        if (visibleTest(i, j)) {
            let image = getAvailabelImage();
            image.x = 256 * j;
            image.y = 256 * i;
            image.visible = false;
            image.source = getResource(i, j);
            usingBlocks.push(image);
            this.addChild(bitmap);
        }
    }
}

public visibleTest(i, j): boolean {
    let x = 256 * j + mapImage.x;
    let y = 256 * i + mapImage.y;
    let stageHeight = CommonUtils.getStageHeight();
    let stageWidth = CommonUtils.getStageWidth();
    if (x > -256 && x < stageWidth + 256 &&
        y > -256 && y < stageHeight + 256) {
        return true;
    } else {
        return false;
    }
}

```

$x, y = \text{MapBlock's Location}$

$x = 256 \times n$, 其中 n 是正整数。
 $y = 256 \times m$, 其中 m 是正整数。

$$MapBlocks(4 \times 7) \begin{cases} using & \text{正在使用的地图块} \\ available & \text{可用的地图块} \end{cases}$$