Cocos2dx 实现橡皮擦效果的实现

DionysosLai([906391500@qq.com](mailto:906391500@qq.com)) 2014/8/25

之前项目在做一个绘本游戏，要求实现擦除效果，具体效果可以参考绘本《我是一只暴龙》，当时由于项目比较紧，是直接拿网上代码来用（感谢仁兄Zrong的入门之引，具体博文，详见地址， <http://zengrong.net/post/2067.htm> ）。当时，没有对其做一些具体优化工作，一些原理，也是似懂非懂。今天，在工作之余，重写了代码，并从始至末将知识点理清楚，务必要求自己能够搞清楚整个工作流程。

**橡皮擦具体功能要求：**

1. 实现擦除效果：具体要求是点击位置，拖动轨迹路上，均可以擦除。在快速拖动过程中，不能出现断层和锯齿现象。
2. 擦除的形状，最好可以自定义。默认可以提供正方形、圆形两种，最好能提供自定义图片形状。
3. 判断图片是否擦除完毕。
4. 如果擦除形状过小，那么难免在擦除过程中，会遗留一些细小的、可能难以注意的残留点。在擦除过程中，要求可以自动擦除这些残留点。

**功能分析：**

1. **擦除效果实现**：
2. 所谓“擦除”，就是将要擦除的图片RGB和alpha值，全部去掉。可以通过两张图片的混合实现。这里简单介绍OpenGL中的混合原理。

OpenGL中的混合，就是将原来的原色和将要画上去的颜色，经过“一些处理”，得到一种新的颜色，然后再次将得到的新颜色画到画布上。这里，我们将要画上去的颜色，称为“源颜色”，把原来的颜色称为“目标颜色”。

上文中的“一些处理”，实际是将源颜色和目标颜色各自取出，乘以一个因数（这里，对应的因数，我们称之为“源因子”和“目标因子”），然后二者相加（当然也可以不是相加，可以是相减、或者取二者最大值等等，新版的OpenGl可以设置运算方式），这样既可以得到一个新的颜色值。我们假设源颜色的四个分量（指红色，绿色，蓝色，alpha值）是(Rs, Gs, Bs, As)，目标颜色的四个分量是(Rd, Gd, Bd, Ad)，又设源因子为(Sr, Sg, Sb, Sa)，目标因子为(Dr, Dg, Db, Da)。则混合产生的新颜色可以表示为：



当然，如果某个分量，超过了最大值，会自动截取的。

源因子和目标因子是可以通过glBlendFunc函数来进行设置的。 glBlendFunc有两个参数，前者表示源因子，后者表示目标因子。这两个参数可以是多种值，下面介绍比较常用的几种。

GL\_ZERO： 表示使用0.0作为因子，实际上相当于不使用这种颜色参与混合运算。

GL\_ONE： 表示使用1.0作为因子，实际上相当于完全的使用了这种颜色参与混合运算。

GL\_SRC\_ALPHA：表示使用源颜色的alpha值来作为因子。

GL\_DST\_ALPHA：表示使用目标颜色的alpha值来作为因子。

GL\_ONE\_MINUS\_SRC\_ALPHA：表示用1.0减去源颜色的alpha值来作为因子。GL\_ONE\_MINUS\_DST\_ALPHA：表示用1.0减去目标颜色的alpha值来作为因子。

利用OpenGl原理，如果我们将源颜色的颜色值设置为0， 并源因子和目标因子分别设置为GL\_OEN，GL\_ZERO，则新颜色具体值如下所示：

a.wmf

注：这里的Rs、Gs、Bs、As均为0。

因此可以很方便的实现的擦除效果了。其详细代码如下所示：

m\_pEraser->setPosition(point);

ccBlendFunc blendFunc = { GL\_ONE, GL\_ZERO }; ///< 设置混合模式, 源---1， 目标---0

m\_pEraser->setBlendFunc(blendFunc);

m\_pRTex->begin();

m\_pEraser->visit();

m\_pRTex->end();

如果是自定义的形状（这里我们的讨论的自定义形状，是图片提供的形状，而不是自己画出来的-----因为自己画出来的，跟前面没有区别）。这里对图片有比较特殊的要求，即要求图片中间形状是镂空的，外部的alpha通道必须为255。如下图所示：

PIC_GJY_副本.png

(\*^\_\_^\*) 嘻嘻……，这里是一张动物图片（这次是做有关动物绘本游戏），在其轮廓内部是镂空的，外部只要alpha最大即可。然后我们将源因子和目标因子分别设置为GL\_ONE\_MINUS\_SRC\_ALPHA、GL\_SRC\_ALPHA。

则新颜色如下表示:

在外部区域：GL\_ONE\_MINUS\_SRC\_ALPHA = 0; GL\_SRC\_ALPHA = 1。则新颜色值如下所示：



还是原来的值。

在内部区域：GL\_ONE\_MINUS\_SRC\_ALPHA = 1; GL\_SRC\_ALPHA = 0。则新颜色值如下所示：

a.wmf

可以看出，值全部为0。

具体代码如下所示：

CCSprite\* drawSprite = CCSprite::createWithTexture(m\_drawTextture);

drawSprite->setPosition(point);

ccBlendFunc blendFunc = { GL\_ONE\_MINUS\_SRC\_ALPHA, GL\_SRC\_ALPHA }; ///< 设置混合模式, 源---1-alpha， 目标---alpha

drawSprite->setBlendFunc(blendFunc);

m\_pRTex->begin();

drawSprite->visit();

m\_pRTex->end();

1. 利用动态纹理，实现纹理的变化。

使用擦除效果，纹理必然是发生动态变化的。这里采用CCRenderTexture实现动态纹理改变。对于CCRenderTexture的具体使用方法，可见引擎里描述语言：

To render things into it, simply construct a render target, call begin on it, call visit on any cocos scenes or objects to render them, and call end.

其实，就是下面一段话：

1. 创建一个新的CCRenderTexture. 这里，你可以指定将要创建的纹理的宽度和高度。.
2. 调用 CCRenderTexture:begin. 这个方法会启动OpenGL，并且接下来，任何绘图的命令都会渲染到CCRenderTexture里面去，而不是画到屏幕上。
3. 绘制纹理. 你可以使用原始的OpenGL调用来绘图，或者你也可以使用cocos2d对象里面已经定义好的visit方法。（这个visit方法就会调用一些opengl命令来绘制cocos2d对象）
4. 调用 CCRenderTexture:end. 这个方法会渲染纹理，并且会关闭渲染至CCRenderTexture的通道。
5. 从生成的纹理中创建一个sprite. 你现在可以用CCRenderTexture的sprite.texture属性来轻松创建新的精灵了。

## 这里，引用的是子龙山人博文《[（译）如何使用CCRenderTexture来创建动态纹理](http://www.cnblogs.com/zilongshanren/archive/2011/07/01/2095479.html)》，具体博文，详见地址：<http://www.cnblogs.com/andyque/archive/2011/07/01/2095479.html>。博文中，详细介绍了CCRenderTexture的动态纹理创建方法。下面给出如何将一个精灵纹理添加进CCRenderTexture中：

CCSprite\* sprite = CCSprite::create(pszFileName);

spriteSize = sprite->getContentSize();

/// 将精灵加入纹理后，其中心点坐标应该设置在（0,0）处， 这是由于纹理的中心点在（0,0），当然，可以通过设置其偏移坐标实现；

sprite->setAnchorPoint(ccp(0.f, 0.f));

// sprite->setPosition(ccp(spriteSize.width/2.f, spriteSize.height/2.f));

m\_pRTex = CCRenderTexture::create(spriteSize.width, spriteSize.height);

m\_pRTex->setPosition(CCPointZero);

this->addChild(m\_pRTex);

m\_pRTex->begin();

sprite->visit();

## m\_pRTex->end();

1. 避免出现断层和锯齿现象

之所以出现断层和锯齿的原因，是由于在快速擦除过程中，系统接收到的第一个点位置和第二个点位置，可能有很大的位移偏差。如果只是简单的处理这两个点的信息，显而亦然中间很多点就会缺失，而不画。因此就出现了断层和锯齿的现象。

因此，我们只要简单的判断两点之间的距离是否超过一定程度，就在二者间再次处理。至于二者间，要抽取多少点进行处理，就要看其距离的长度了。这边，我简单的判断距离超过1，就要处理（显然这样做，会比较准确，但消耗性能大，可以适当更改）。下面给出具体代码：

void EraserSprite::ccTouchMoved( cocos2d::CCTouch \*pTouch, cocos2d::CCEvent \*pEvent )

{

if (m\_bEraser)

{

CCPoint point = pTouch->getLocation();

CCPoint normal = ccpNormalize(point-m\_touchPoint);

/// 处理一次移动过多，造成中间有遗漏，或者锯齿现象；

while(1)

{

if (ccpDistance(point, m\_touchPoint) < 1.f)

{

/\* m\_pEraser->setPosition(-this->getPosition() + point + spriteSize/2.f);\*/

eraseByBlend(-this->getPosition() + point + spriteSize/2.f);

break;

}

m\_touchPoint = m\_touchPoint + normal\*1.f;

/\* m\_pEraser->setPosition(-this->getPosition() + m\_touchPoint + spriteSize/2.f);\*/

eraseByBlend(-this->getPosition() + m\_touchPoint + spriteSize/2.f);

}

m\_touchPoint = point;

}

}

1. **擦除形状**

对于擦除形状，其实上文已经提到了。这里简单提一下。如果是采用点或者圆形，可以使用自定义画节点，即CCDrawNode实现。对于CCDrawNode的扩展使用，也可以用到CCClippingNode中，即实现自定义裁剪模板。下面给出正方形和圆形擦除形状代码：

正方形形状：

m\_pEraser = CCDrawNode::create();

float width = 10.f;

m\_pEraser->drawDot(CCPointZero, width, ccc4f(0,0,0,0));

圆形形状：

m\_pEraser = CCDrawNode::create();

/// 绘制圆形区域

float fRadius = 30.0f; ///< 圆的半径

const int nCount = 100; ///< 用正100边型来模拟园

const float coef = 2.0f \* (float)M\_PI/nCount; ///< 计算每两个相邻顶点与中心的夹角

static CCPoint circle[nCount]; ///< 顶点数组

for(unsigned int i = 0;i <nCount; i++) {

float rads = i\*coef; ///< 弧度

circle[i].x = fRadius \* cosf(rads); ///< 对应顶点的x

circle[i].y = fRadius \* sinf(rads); ///< 对应顶点的y

}

m\_pEraser->drawPolygon(circle, nCount, ccc4f(0, 0, 0, 0), 0, ccc4f(0, 0, 0, 0));//绘制这个多边形!

对于自定义的图片形状，其实就是一个精灵对象而已。

1. **判断图片是否擦除完毕**

判断是否擦除完毕，基本思路就是对纹理像素值逐点判断，当所有像素值均为0时，则代表图片已经擦除完毕了。

首先，获取纹理的图片信息。关键函数是newCCImage。具体代码如下：

CCImage\* image = new CCImage();

image = m\_pRTex->newCCImage(true);

这里要注意一点就是，最后要手动删除image。

其次，获取各个位置的像素值。代码如下所示：

unsigned char \*pixel = data\_ + (x + y \* image->getWidth()) \* m;

// You can see/change pixels' RGBA value(0-255) here !

unsigned int r = (unsigned int)\*pixel;

unsigned int g = (unsigned int)\*(pixel + 1);

unsigned int b = (unsigned int)\*(pixel + 2) ;

unsigned int a = (unsigned int)\*(pixel + 3);

其中，x、y代表位置。

完整代码如下所示：

bool EraserSprite::getEraserOk()

{

m\_bEraserOk = false;

CCImage\* image = new CCImage();

image = m\_pRTex->newCCImage(true);

int m = 3;

if (image->hasAlpha())

{

m = 4;

}

unsigned char \*data\_= image->getData();

int x = 0, y = 0;

/// 这里要一点，即Opengl下，其中心点坐标在左上角

for (x = 0; x < spriteSize.width; ++x)

{

for (y = 0 ; y < spriteSize.height; ++y)

{

unsigned char \*pixel = data\_ + (x + y \* image->getWidth()) \* m;

// You can see/change pixels' RGBA value(0-255) here !

unsigned int r = (unsigned int)\*pixel;

unsigned int g = (unsigned int)\*(pixel + 1);

unsigned int b = (unsigned int)\*(pixel + 2) ;

unsigned int a = (unsigned int)\*(pixel + 3);

if (r != 0 && g != 0 && b != 0 && a != 0)

{

m\_bEraserOk = false;

break;

}

}

if (spriteSize.height != y)

{

break;

}

}

if (x == spriteSize.width && y == spriteSize.height)

{

m\_bEraserOk = true;

}

delete image;

return this->m\_bEraserOk;

}

这里，参考了文章：《[Getting and setting the RGB / RGBA value of a pixel in a CCSprite (cocos2d-x)](http://stackoverflow.com/questions/9665700/getting-and-setting-the-rgb-rgba-value-of-a-pixel-in-a-ccsprite-cocos2d-x)》，详细地址：<http://stackoverflow.com/questions/9665700/getting-and-setting-the-rgb-rgba-value-of-a-pixel-in-a-ccsprite-cocos2d-x>

好囧啊，这个部分，花了我整整一个上午时间，没想到就这样的过去，一点都没有前面高大善的赶脚。

1. **残留点清除问题**

**对于这个问题，还没有很好的思路**