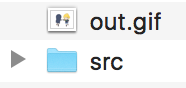
Project 3文档

13302010013 周信安

1. GIF解码工具使用方法

首先，用Eclipse导入src文件夹下的所有java源代码文件。然后，把你想要解码的GIF图片文件放在你的Eclipse项目文件夹的根目录下。比如，如果你想要解码Question 1中生成的GIF文件out.gif，你的Eclipse项目文件夹应该是这样的：



接着，在Eclipse中运行Main.java，程序会提示你输入想解码的GIF的文件名。此时，输入out.gif并按回车键。稍等片刻后，项目文件夹的根目录中将会出现Frame0-Frame15.bmp。你可以查看这些bmp图像。在控制台中将会打印GIF文件的各种信息，你可以有选择地查看。

1. GIF解码工具工作原理

共有四个java文件。Main.java读取GIF文件名并调用GIFDecoder.java对GIF文件进行解码。GIFDecoder.java负责解析GIF文件结构，将每一帧的图片信息以FrameImage对象存储，并调用LZWDecoder.java对GIF数据块部分进行解码。LZWDecoder.java中是LZW算法的核心部分，比较复杂，包含一些位运算，比如从字节流中取码。FrameImage.java是FrameImage类，存储GIF每一帧的图像的长宽和颜色表。

1. MATLAB在生成你所得出的GIF时，哪里可以优化，从而节约更多的空间？

首先，有一些帧自带local color table（局部颜色表），虽然可以使GIF图片颜色更真实，但这是比较消耗空间的，如果可能，尽量使用全局颜色表，可以节约一些空间。

第二，LZW编码的精髓在于，待编码字符串（实际上是字节流中存在的不定长的码串，但由于字节流操作与算法无关，故还是称作“字符串”、“字符”以体现算法的思想）中出现的重复越多，编码效果越好。那么怎么才能使待编码字符串中出现重复的概率变高？一个可行的办法是压缩颜色表，让整个图像用更少种类的颜色来表示。这样的话，本来是几种相近的颜色，经过压缩后就变成一种颜色。这个颜色出现的概率即为原来几种颜色出现的概率的和。也就是说，原来字符串中代表几种颜色的若干种字符被同一种颜色的字符替换了。这样的话，待编码字符串中出现重复的概率就会显著提高，对压缩很有好处。但是相应的后果是GIF的颜色没有以前那么丰富了。

第三，颜色表的空间是存在浪费的。比如，在demo.gif中，全局颜色表从下标205到下标255的颜色都是#000000，这浪费了一些空间。每帧图像的局部颜色表也存在浪费，如果用局部颜色表的帧图像很多，那么就是十分显著的浪费。但是颜色表的颜色种类数用二进制进行表示的话，比如256色，就是8位。要是减少一点颜色的种类，用7位即128色的话，颜色种类数又会下降一半，所以有点难取舍。可以考虑压缩一下颜色表，就像第二点中讲的那样。

第四，每一帧图像在其image descriptor中都有其长宽，而这其实是不必要的，因为在本次PJ中每一帧图像都有同样的长宽，可以考虑在GIF文件中设置一个位，如果置为1，则代表每一帧的图像都采用逻辑屏幕标识符中的长宽，这样也可以减少一点空间使用。

第五，注释块等拓展块如果对GIF的显示没有帮助，那么可以在生成GIF的时候完全丢弃，可以节省很多空间。

以上纯手打心得，希望多多给分，谢谢！