实验报告

实验题目:基于八叉树颜色删减实验

姓名:刘潇远　学号:161220083　邮箱:liuxy@smail.nju.edu.cn

【实验目的及要求】

实现真彩色到 256 色的颜色转换算法

（1）提供的代码：

main.cpp：提供了主控函数 main，八叉树类 octTree 和八叉树节点结构 octNode。

（2）代码的编译：

由于需要使用 bmp 的信息头和文件头结构，该结构在 windows.h 文件中，建议使用 VisualStudio 系统编译，也可以自己定义相关变量在其它编译器里编译。

在 VS 里编译过程：打开 VS，创建新的空项目，添加 main.cpp，编译即可，运行命令：程序名　输入文件名　输出文件名

本程序只是简单的 demo，未考虑太多的正确性检查，故输入文件应为 24 位的真彩色 bmp 格式文件。

（3）代码简介：

1、打开输入图像，读取文件头部和信息头部并填充输出图像的

文件头部和信息头部；

2、依次读取输入图像的每一像素的颜色并插入八叉树中；

3、生成 256 色的调色板，并写出到文件中；

4、依次读取输入图像的每一个像素并在调色板中查找出与之最相近的颜色索引值。

5、输出 256 色图像

（4）实验要求：

代码魔改：将 octTree 类从 main.cpp 中分离出来，写入 octTree.cpp 和 octTree.h 文件中。

1、完成 octTree 类的成员函数 insertColor，该函数的功能是往颜色八叉树中添加一个颜色；

insertColor 成员函数根据八叉树原理（见下方【实验原理】），将 RGB 三个分量按位切割后拼接成 3bit 子树索引。这里需要注意的是，**靠近根的的节点应当取 RGB 分量的高位**，比如对于 R1000111，G0010101，R11100111，八叉树根节点子节点索引应为 101(RGB 分量取最高位后拼接)。如果靠近根取的是低位拼接，那么如果该节点后面被合并，高位信息就会被丢失（也不能叫丢失，而是和其他合并节点取平均），颜色会严重失真。

考虑到在插入时合并可能会造成后续出现的频繁色被添加到发生合并的分支导致颜色失真，因此将**颜色删减放到生成调色板步骤中进行**。

2、完成 octTree 类的成员函数 generatePalette，该函数的功能是从颜色八叉树生成 256 个调色板颜色；

在所有颜色插入完毕后，需要生成调色板颜色，而在生成调色板颜色前，首先应当检查总颜色数，即 octTree 类的 colors 属性值是否大于 maxColors，如果大于，则需要进行颜色删减，即八叉树分支合并。合并完成后，使用深度遍历方法遍历八叉树，找到所有叶节点，将其总 RGB 量除以其代表的像素数，将结果赋值给 pal 数组 RGBQUAD 结构体的相应属性即可，rgbReserved 属性为保留属性，我的处理方法是置零。下述颜色删减过程：

颜色删减使用 while 循环，在颜色数溢出，即 colors>maxColors 时，进行颜色删减。颜色删减遍历八叉树，**寻找代表颜色数量最少、深度最深的子树进行合并**。合并的子树必须满足的条件是有两个及以上子节点，这是基本条件，首先如果该子树中的某个子孙节点也有两个及以上子节点，那么原先的子树就不会被合并。在八叉树中，满足基本条件的子树有很多，我们按照以下顺序进行筛选：1、**先选深度更深的节点，深度相同选代表像素数最少的点**。这个顺序是有**两点考量**：首先，在构建二叉树时，越深的节点，对应的是颜色的 RGB 三个分量中低位，将低位进行平均，得出的结果与原颜色差的绝对值小于更高位平均结果与更高位差的绝对值，简单点说，就是合并更深层节点，对颜色的影响更小。第二点考量是代表的像素数越少，合并完了以后影响的像素点就越少，删色就越不容易被发现。除此之外，人眼对颜色的敏感程度并不是均匀分布，而是对绿色（我记得好像是绿色）

更敏感一些，所以按理说应当也把颜色考虑进去。

颜色删减的原理如上所述，问题出在了，如何寻找候选删除子树上，最开始我使用的是**广度优先遍历**，这意味着，程序要从八叉树根节点开始，一层一层进行递归，慢的一批！后来我在 **octTree 类中增加了一个二维动态数组（vector 实现），用以记录每层节点，这样每次寻找候选子树时，直接从最底层开始寻找，找到就直接结束寻找，**这样一定可以加快速度！但是这样实现后，由于底层节点数量十分庞大（有一张测试图片达到 12 万），程序运行速度依然不理想，我又想到，上一次考察过的节点，如果**不含候选子树的话**，为何不**直接从数组中删去**呢，这样子后面进行遍历时就不需要重复进行无用功了。这样，程序的运行时间从分钟级降低到了秒级。

3、完成函数 selectClosestColor，该函数的功能是从调色板中选出与给定颜色最接近的颜色；

一共就 256 个颜色，挨个比较毫无压力，**对于给定颜色，比较它与调色板中颜色三个分量的曼哈顿距离**，取曼哈顿距离最低的即可。考虑到人眼的敏感程度，曼哈顿距离相等时，取绿色分量差距最低的。

至此，程序完成

（5）实验提交：

提交本实验报告和源文件，均以学号为文件名。

实验报告：161220083.pdf、161220083.docx

源文件： 1、所有全都放在一个文件中：161220083.cpp

2、将 octTree 放到单独的文件后的程序代码：

main.cpp+octree.h+octree.cpp

检查作业时直接编译运行 161220083.cpp 即可

测试图片：1.bmp、2.bmp、3.bmp、4.bmp、5.bmp

【实验原理】（简要说明本实验项目所涉及的理论知识）

在数据结构中，有一种数据结构叫做二叉树，类比可以得知八叉树的构造：每个非叶节点最多有八个子节点。在 24 位 RGB 真彩色图片中这里有前提，RGB 三个分量在计算机中都是由 8 比特来表示的。现在把 RGB 当做一个整体来看待，为了方便理解，现在假设有一个矩阵，列分别是由颜色分量中的各位来表示的，于是这个矩阵就有了 3 列，分别是 RGB。接着是行，那就是由 RGB 中处于相同位序的 01 组成的了。于是每一行由 01 所代表的二进制数字可以转换成一个十进制数字，并且这种转换是一一对应的。即，这个十进制数字可以唯一地转换成对应行的二进制数字，而那一行的二进制数字也只能转换成这个十进制数字。于是这个十进制数字和那一行二进制数字就具备互相代表的能力。

由上述可知，对于 24 位真彩色图片中的每个颜色，根据其 RGB 分量从上到下就可以被分割成由 8 个十进制数字表示的 8 个层次了，每个层次对应 000~111。RGB 分量有着固定长度，能拆成一个含 8 个 3bit 的序列，序列中每个值范围相同，从 RGB 高位到低位可以被看成是一条路径，3bit 共 8 种取值，所以可以作为八叉树，那么这条路径就是八叉树中的一个分支，其中每个数字就代表了当前结点的以该数

字作为标识的子结点了，简而言之，该数字就是当前结点的某个子结点的索引。

【实验结果】（比较转换结果，原有转换值与自己方法的比较，分析结果的好坏）

下图原图：

下图采取“在插入节点时进行节点合并”策略的结果：可见衣服颜色已经失真

下图采取“插入节点时不进行节点合并，而是在生成调色板前才进行节点合并，不考虑节点深度，而是只选代表颜色数少的节点进行合并”的结果：可见衣服颜色失真情况有所缓解，但是人物头发又出现了失真

下图严格按照实验报告中所述原理和实现，效果最好，且运行速度较快

【实验小结】（对本次实验的心得体会、思考和建议）

本次实验写了好几天，我十分感谢实现周边代码的老师/助教，把我从调试 bmp 文件格式读写的深渊中解救出来，而专注于八叉树的实现上。可以从前面我的代码实现部分叙述中得知，我从最初的颜色失真、运行速度漫长到现在结果还算不错、运行速度也能让人接受，中间确实花费了不少心思，虽然原理较为简单，但是真正到实现还是有一部分难度的，比起其他课程的编程作业啥都不给只提需求，导致花费很多时间在周边组建和文件输入输出的调试，多媒体实验编程作业让我专注思考核心实现，编程体验极为良好!!!