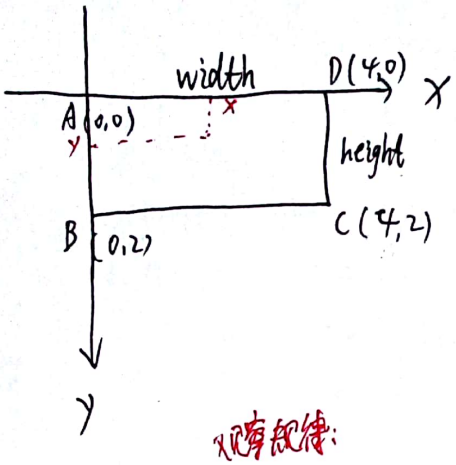
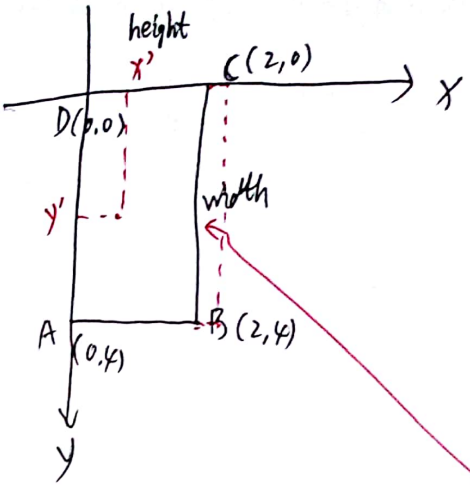
上下翻转及左右翻转是直接提取像素法，操作简单，故不赘述（根据代码即可看懂）

灰度化是指针法（也可用内存法），只是在原本R、G、B分量上乘以一个系数，不难理解

实验报告可以结合以上内容：并分析三种方法的优劣



左转90°



观察规律：

$$\begin{cases} x' = y \\ y' = width - x - 1 \end{cases}$$

若只考虑坐标关系

则 $\begin{cases} x' = y \\ y' = width - x \end{cases}$

但是因为要用数组存储，故要减1

思路：新建一个空的 bitmap，宽为原图的高（即宽 = height），高为 width.（若宽（即height）不被4整除，则加到4的）

对于补的这一部分像素不管，存储只考虑 height × width 部分（注意，每个点都包含三个分量）

建一个 byte 数组，大小为 width × (height) × 3　补了之后的宽度记为 ~~newheight~~ rotatedheight

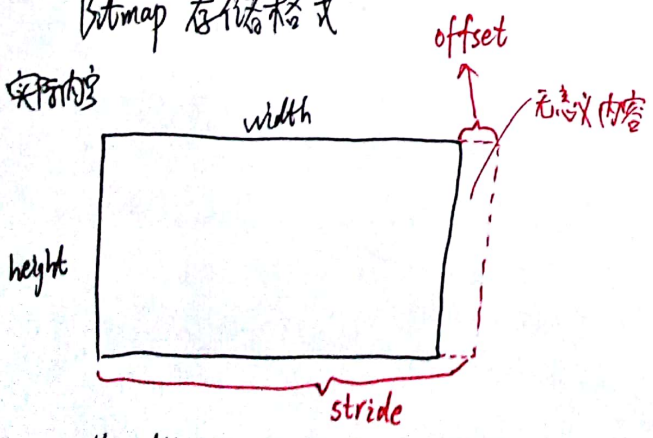
将上图所有数据点按从上至下、从左至右存到一维数组中

分析可知数组的索引 i 与 x' 和 y' 的关系为：

$i = (rotatedheight * y' + x') * 3$

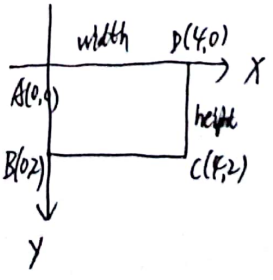
然后一个数据点存三个像素的分量，故 $p += 3$

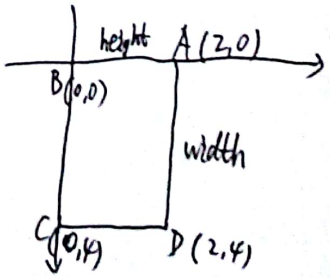
Bitmap 存储格式



若 width % 4 != 0，则补到 width % 4 == 0

右转同理.





$$\begin{cases} x' = height - y - 1 \\ y' = x \end{cases}$$

（因为考虑到数组存储，故减1）

只考虑坐标关系

$$\begin{cases} x' = height - y \\ y' = x \end{cases}$$

$i = 3 * (rotatedheight * y' + x')$