HW1

1 Exercise

* 1. 1. How many bit planes are there for this image?

Eight.

2. Which panel is the most significant one?

The plane eight is the most visually significant one.

3. How many bytes are required for storing this image?

2048\*2048 = 4194304 bytes are needed.

* 1. The 4-path between p and q does not exist because the numbers 4-neighboring q are not in V.

The length of the shortest 8-path between p and q is 4.

The length of the shortest m-path between p and q is 5.

* 1. The first picture: A∩B∩C.

The second picture:(A∩B)∪(B∩C)∪(A∩C).

The third picture:( A∩C∩)∪(B∩∩).

1. Programming
   1. I make use of the openVC library to load the picture and store it.







This is my original picture----81.png.



2.2 Scaling

192\*128: 

96\*64:

48\*32:

24\*16:

300\*200:

400\*300:

500\*200:

Description:

I use the scale between the widths of source picture and destination picture to calculate to calculate the coordinates of a specific pixel of source picture.

The coordinates of a specific pixel of source picture and the coordinates of a pixel corresponding to it of destination picture satisfy following rule:

srcX=dstX\* (srcWidth/dstWidth) , srcY = dstY \* (srcHeight/dstHeight).

After knowing the relation between the coordinates, we then give a pixel of destination picture a value most nearby in the source picture.

This method can easily scale the source picture into a picture in a given size. But it will bring us a problem-----serious distortion. When decreasing the size, the destination picture will become will lose many details. When increasing the size, the destination picture will have many mosaics.

This problem results from the one-to-one relate between the pixel values of the source picture and the destination picture. In fact, the value of a pixel in the destination picture is not only corresponding to one point in the source picture but is related to the pixels neighboring that points. We should take them into consideration. A better method is to use the bilinear transformation to scale the picture.

This is the picture after using bilinear transformation:

96\*64:

48\*32:

24\*16:

12\*8:

* 1. Quantization

128:

32:

8:

4:

2:

Description:

I calculate the gray values as the given number and store them in a vector. Since the order of these values are increasing, we can know which interval a gray value of a specific pixel lying in. If a gray value lying in the smaller half-interval, we can give the lower bound value of the interval to the destination pixel corresponding to the source pixel. And we can give the upper bound value of the interval to the destination pixel corresponding to the source pixel when the gray value lying in the larger half-interval. After traversing the whole picture, we can get a picture only having gray values as the give number. We should note that the gray value cannot be larger than 255.