

2

2.1 BMP

2.1.1 BMP

BMP BMP 4 (bitmap-file header) (bitmap-information header) (color table)

1. (bitmap-file header)

bfType	2	BMP 0x42,0x4D ASCII "B""M"
bfSize	4	BMP
bfReserved1	2	
bfReserved2	2	

UltraEdit BMP

ImageProcess_Image001

Image not found or type unknown

BMP 32 0x42,0x4D

4 0x36,0xF9,0x15,0x00

BMP

0x36,0xF9,0x15,0x00 0x0015F936()

ImageProcess_Image002

Image not found or type unknown

4 0x00

4 + + 0x36

2. (bitmap-information header)

biSize	4	
biWidth	4	
biHeight	4	
biPlanes	2	1
biBitCount	2	1 1 4 8 16 24 32 24
biCompression	4	
biSizeImage	4	
biXPelsPerMeter	4	/
biYPelsPerMeter	4	/
biClrUsed	4	
biClrImportant	4	0

ImageProcess_Image003

Image not found or type unknown

0E-11 00000028h = 40, 40

12-15 00000320h = 800 800

16-19 00000258h = 600 600 X Y

1A-1B 0001h, 1

1C-1D 0018h = 24, 24 24

ImageProcess_Image004

Image not found or type unknown

1E-21 00000000h BI_RGB

22-25 00000000h BI_RGB 0

26-29 00000000h

2A-2D 00000000h

2E-31 00000000h 24 0

32-35 00000000h 24 0

3. (color palette)

24

4.

24 BGR

2.1.2 BMP RGB LCD

BMP LCD

1. LCD RGB 2 RGB565 3 RGB888 24 BMP BG

2. LCD LCD BMP

2.1 24 BMP RGB

```
2. 1
1. ☒ *****
2. ☐ *      IsBmp
3. ☐ *      BMP
```

```

4.  *      ptFileMap -
5.  *
6.  *      0 - BMP , -1 - BMP
7.  * *****/
8.  int IsBmp(FILE **ppFp, const char *strFileName)
9.  {
10.     char strCheckHeader[2];
11.     *ppFp= fopen(strFileName, "rb+");
12.     if (*ppFp== NULL) {
13.         return -1;
14.     }
15.     if (fread(strCheckHeader, 1, 2, *ppFp) != 2)
16.         return -1;
17.
18.     if (strCheckHeader[0] != 0x42 || strCheckHeader[1] != 0x4d)
19.         return -1;
20.     else
21.         return 0;
22. }
23.
24.
25.
26. /*****
27.  *      MapFile
28.  *      mmap      ,
29.  *      PT_PictureData ptData
30.  *      ptData->iFileSize :
31.  *      ptData->pucFileData :
32.  *      0      -
33.  * *****/
34.  int MapFile(PT_PictureData ptData)
35.  {
36.     int iFd;
37.     struct stat tStat;
38.
39.     /*      */
40.     iFd = fileno(ptData->ptFp);
41.     fstat(iFd, &tStat);
42.     ptData->iFileSize= tStat.st_size;

```

```

43. □   pData->pucFileData= (unsigned char *)mmap(NULL , tStat.st_size, PROT_READ |
PROT_WRITE, MAP_SHARED, iFd, 0);
44. □   if (pData->pucFileData == (unsigned char *)-1)
45. □   {
46. □       printf("mmap error!\n");
47. □       return -1;
48. □   }
49. □   return 0;
50. □}
51. □
52. □/*****
53. □ *      DecodeBmp2Rgb
54. □ *      BMP      rgb
55. □ *      strFileName -
56. □ *                  pData -
57. □ *      0      -      -
58. □ *                  -1      -      BMP
59. □ *                  -2      -      bpp
60. □ *                  -3      -
61. □ *****/
62. □static int DecodeBmp2Rgb(const char *strFileName, PT_PictureData pData) {
63. □   int x,y;
64. □   int iPos = 0;
65. □   int iLineWidthAlign;
66. □   BITMAPFILEHEADER *ptBITMAPFILEHEADER;
67. □   BITMAPINFOHEADER *ptBITMAPINFOHEADER;
68. □   unsigned char *aFileHead;
69. □   unsigned char *pucSrc;
70. □   unsigned char *pucDest;
71. □   int iLineBytes;
72. □
73. □   /*      BMP      */
74. □   if (IsBmp(&pData->ptFp, strFileName))
75. □       return -1;
76. □
77. □   /* BMP      */
78. □   MapFile(pData);
79. □
80. □

```

```

81. □   aFileHead = pData->pucFileData;
82. □
83. □   ptBITMAPFILEHEADER = (BITMAPFILEHEADER *)aFileHead;
84. □   ptBITMAPINFOHEADER = (BITMAPINFOHEADER *)(aFileHead + sizeof(BITMAPFILEHEADER));
85. □   /*          */
86. □   pData->iWidth = ptBITMAPINFOHEADER->biWidth;
87. □   pData->iHeight = ptBITMAPINFOHEADER->biHeight;
88. □   pData->iBpp = ptBITMAPINFOHEADER->biBitCount;
89. □       iLineBytes    = pData->iWidth*pData->iBpp/8; //
90. □   pData->iBmpDataSize= pData->iHeight * iLineBytes; //  BMP
91. □   /*    24bpp    */
92. □   if (pData->iBpp != 24)
93. □   {
94. □       printf("iBMPBpp = %d\n", pData->iBpp);
95. □       printf("sizeof(BITMAPFILEHEADER) = %d\n", sizeof(BITMAPFILEHEADER));
96. □       return -2;
97. □   }
98. □
99. □   /*          */
100. □   pData->pucBmpData = malloc(pData->iBmpDataSize);
101. □   pData->pucRgbData = malloc(pData->iBmpDataSize);
102. □
103. □   if (NULL == pData->pucBmpData || NULL == pData->pucRgbData)
104. □       return -2;
105. □
106. □   /*  bmp          24bpp BMP   BGR   */
107. □   pucDest = pData->pucBmpData;
108. □   iLineWidthAlign = (iLineBytes + 3) & ~0x3;    /*  4   */
109. □   pucSrc = aFileHead + ptBITMAPFILEHEADER->bfOffBits;
110. □
111. □   pucSrc = pucSrc + (pData->iHeight - 1) * iLineWidthAlign;
112. □
113. □   /*  bmp          */
114. □   for (y = 0; y < pData->iHeight; y++)
115. □   {
116. □       memcpy(pucDest, pucSrc, pData->iWidth*3);
117. □       pucSrc -= iLineWidthAlign;
118. □       pucDest += iLineBytes;
119. □   }

```

```

120. □
121. □
122. □ /*      BGR      RGB      */
123. □ for (y = 0; y < pData->iHeight; y++){
124. □     for(x = 0; x<pData->iWidth*3; x+=3){
125. □         pData->pucRgbData[ iPos++] = pData->pucBmpData[ y*pData->iWidth*3+x+2];
126. □         pData->pucRgbData[ iPos++] = pData->pucBmpData[ y*pData->iWidth*3+x+1];
127. □         pData->pucRgbData[ iPos++] = pData->pucBmpData[ y*pData->iWidth*3+x+0];
128. □     }
129. □ }
130. □
131. □ return 0;
132. □
133. □}

```

2.2 JPEG

2.2.1 JPEG libjpeg

JPEG .jpg JPEG BMP JPEG BMP JPEG JPEG
 BMP JPEG Linux jpeg
 libjpeg jpeg libjpeg X86 ARM libjpeg
 libjpeg libjpeg libjpeg

1.

```

tar xzf libjpeg-turbo-1.2.1.tar.gz
cd libjpeg-turbo-1.2.1/

```

2.

```

tar xzf libjpeg-turbo-1.2.1.tar.gz
./configure --prefix=/work/projects/libjpeg-turbo-1.2.1/tmp/ --host=arm-linux
make
make install

```

3.

```
cd /work/projects/libjpeg-turbo-1.2.1/tmp/include
cp * /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/usr/include
cd /work/projects/libjpeg-turbo-1.2.1/tmp/lib
cp *.so* -d /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib
```

4.

```
cd /work/projects/libjpeg-turbo-1.2.1/tmp/lib
cp *.so* /work/nfs_root/fs_mini_mdev_new/lib/ -d
```

2.2.2 libjpeg

libjpeg libjpeg.txt example.c libjpeg

1. jpeg_compress_struct

```
cinfo.err = jpeg_std_error(&jerr);
jpeg_create_decompress(&cinfo);
```

2.

```
jpeg_stdio_src(&cinfo, infile);
```

1 1 jpeg_compress_struct

2 JPEG

3. jpg

```
jpeg_read_header(&cinfo, TRUE);
```

cinfo image_width image_height

cinfo scale_num scale_denom

4.

```
jpeg_start_decompress(&cinfo);
```

cinfo

cinfo

5.


```
jpeg_read_scanlines(&cinfo, buffer, 1);
```

RGB buffer 3

6.

```
jpeg_finish_decompress(&cinfo);
```

7. jpeg_compress_struct

```
jpeg_destroy_decompress(&cinfo);
```

2.2.3 libjpeg JPEG RGB LCD

JPEG RGB

2. 2

```
1. 1/*****
2. 2 *        IsJpg
3. 3 *        Jpg
4. 4 *        ptData -
5. 5                strFileName -
6. 6 *        0 -    JPG    -    JPG
7. 7 *****/
8. 8 static int IsJpg(PT_PictureData ptData, const char *strFileName)
9. 9 {
10. 10    int iRet;
11. 11
12. 12    jpeg_stdio_src(&ptData->tInfo, ptData->ptFp);
13. 13
14. 14    /* jpeg_read_header jpeg */
15. 15    iRet = jpeg_read_header(&ptData->tInfo, TRUE);
16. 16
17. 17        return (iRet == JPEG_HEADER_OK);
18. 18 }
19. 19
20. 20 1/*****
21. 21 *        DecodeJpg2Rgb
```

```

22. □ *      JPG      RGB888
23. □ *      ptData -
24. □ *
25. □ *      PT_PictureData->pucRgbData -   rgb
26. □ *      0 -      -
27. □ *****/
28. □ static int DecodeJpg2Rgb(const char *strFileName, PT_PictureData ptData){
29. □     int iRowSize;
30. □     unsigned char *pucbuffer;
31. □     unsigned char *pucHelp; //
32. □
33. □     /* 1.      jpeg_compress_struct      */
34. □     ptData->tInfo.err = jpeg_std_error(&ptData->tJerr);
35. □     jpeg_create_decompress(&ptData->tInfo);
36. □
37. □
38. □     /* 2.      */
39. □     if ((ptData->ptFp= fopen(strFileName, "rb")) == NULL) {
40. □         fprintf(stderr, "can't open %s\n", strFileName);
41. □         return -1;
42. □     }
43. □
44. □     /* 3.  jpg      JPEG      */
45. □     if (!IsJpg(ptData, strFileName)) {
46. □         printf("file is not jpg ... \n");
47. □         return -1;
48. □     }
49. □
50. □
51. □
52. □     /*      */
53. □     ptData->tInfo.scale_num = 1;
54. □     ptData->tInfo.scale_denom = 1;
55. □     /* 4.      jpeg_start_decompress */
56. □     jpeg_start_decompress(&ptData->tInfo);
57. □
58. □
59. □     /*      tInfo      */
60. □     ptData->iWidth= ptData->tInfo.output_width;

```

```

61. □   ptData->iHeight = ptData->tInfo.output_height;
62. □   ptData->iBpp = ptData->tInfo.output_components*8;
63. □   /*           */
64. □   iRowSize = ptData->iWidth * ptData->tInfo.output_components;
65. □   pucbuffer = malloc( iRowSize);
66. □   ptData->iRgbSize= iRowSize * ptData->iHeight;
67. □   ptData->pucRgbData = malloc(ptData->iRgbSize);
68. □
69. □   /* pucHelp ptData->pucRgbData */
70. □   pucHelp = ptData->pucRgbData;
71. □   /* 5.   jpeg_read_scanlines */
72. □   while (ptData->tInfo.output_scanline < ptData->tInfo.output_height)
73. □   {
74. □       /* jpeg_read_scanlines pucbuffer */
75. □       jpeg_read_scanlines(&ptData->tInfo, &pucbuffer, 1);
76. □       /*           */
77. □       memcpy( pucHelp, pucbuffer, iRowSize);
78. □       pucHelp += iRowSize;
79. □   }
80. □   free( pucbuffer);
81. □   /* 6.   */
82. □   jpeg_finish_decompress( &ptData->tInfo);
83. □   /* 7.   jpeg_compress_struct */
84. □   jpeg_destroy_decompress( &ptData->tInfo);
85. □   return 0;
86. □ }

```

2.3 PNG

2.3.1 PNG libpng

JPEG

PNG

JPEG

PNG

LZ77

PNG

libpng

libpng

<http://www.libpng.org/pub/png/libpng.html>

libpng

libpng

libpng

1.

```
tar xzf libpng-1.6.37.tar.gz
cd libpng-1.6.37/
```

2.

```
./configure --prefix=/work/projects/libpng-1.6.37/tmp/ --host=arm-linux
make
make install
```

3.

```
cd /work/projects/libpng-1.6.37/tmp/include
cp * /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/usr/include
cd /work/projects/libpng-1.6.37/tmp/lib
cp *.so* -d /usr/local/arm/4.3.2/arm-none-linux-gnueabi/libc/armv4t/lib
```

4.

```
cd /work/projects/libpng-1.6.37/tmp/lib
cp *.so* /work/nfs_root/fs_mini_mdev_new/lib/ -d
```

2.3.2 libpng

libpng libpng-manual.txt example.c libjpeg

1. libpng png_ptr info_ptr

A. png_ptr = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL, NULL);
 2 3 4 NULL

B. info_ptr = png_create_info_struct(png_ptr);

2.

setjmp(png_jmpbuf(png_ptr));
libpng

png_create_read_struct

3.

png_init_io(png_ptr, fp);
 1 1 png_ptr 2 PNG

4. PNG

A.

png_read_png(png_ptr, info_ptr, png_transforms, png_voidp_NULL);
 info_ptr png_transforms

libpng

B.

```
png_get_image_width png_get_image_height png_get_color_type      png
```

5. info_ptr

PNG

A.

```
png_read_image(png_ptr, row_pointers);
```

```
1 1  png_ptr 2
```

B.

```
row_pointers = png_get_rows(png_ptr, info_ptr);
```

```
1 2 1  png_ptr, info_ptr
```

```
1 1  png_ptr 2
```

6.

```
png_destroy_read_struct(&png_ptr, &info_ptr, 0);
```

2.3.3 libpng png rgb LCD

2.3

```
1. /******
2.  *      Isn't Png
3.  *      PNG
4.  *      ppFp -
5.  *      strFileName -
6.  *      0 - PNG - PNG
7.  * *****/
8. int Isn't Png(FILE **ppFp, const char *strFileName)
9. {
10.     char strCheckHeader[8];
11.     *ppFp= fopen(strFileName, "rb");
12.     if (*ppFp== NULL) {
13.         return -1;
14.     }
15.     /* PNG 8 png_sig_cmp PNG */
16.     if (fread(strCheckHeader, 1, 8, *ppFp) != 8)
17.         return -1;
18.     return png_sig_cmp(strCheckHeader, 0, 8);
19. }
20.
21. */
```

```

22. /******
23.  *      DecodePng2Rgb
24.  *      PNG      RGB888
25.  *      pData -
26.  *                      strFileName -
27.  *      PT_PictureData->pucRgbData -   rgb
28.  *      0 -      -
29.  * *****/
30. static int DecodePng2Rgb(const char *strFileName, PT_PictureData pData)
31. {
32.     int i, j;
33.     int iPos = 0;
34.     png_bytepp pucPngData;
35.     /* 0.      PNG      */
36.     if (IsnotPng(&pData->ptFp, strFileName)) {
37.         printf("file is not png ... \n");
38.         return -1;
39.     }
40.
41.     /* 1.      libpng      png_ptr info_ptr */
42.     pData->ptPngStrPoint = png_create_read_struct(PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
43.     pData->ptPngInfoPoint= png_create_info_struct(pData->ptPngStrPoint);
44.
45.     /* 2.      */
46.     setjmp(png_jmpbuf( pData->ptPngStrPoint));
47.     rewind(pData->ptFp); // fseek( fp, 0, SEEK_SET);
48.
49.     /* 3.      */
50.     png_init_io(pData->ptPngStrPoint, pData->ptFp);
51.
52.     /* 4.  PNG
53.      *      PNG_TRANSFORM_EXPAND
54.      *      PNG      BGR888 ABGR888 */
55.     png_read_png(pData->ptPngStrPoint, pData->ptPngInfoPoint, PNG_TRANSFORM_EXPAND,
0);
56.     pData->iChannels      = png_get_channels(pData->ptPngStrPoint, pData-
>ptPngInfoPoint);
57.     pData->iWidth      = png_get_image_width(pData->ptPngStrPoint, pData-

```

```

>ptPngInfoPoint);
58. □   ptData->iHeight  = png_get_image_height(ptData->ptPngStrPoint, ptData-
>ptPngInfoPoint);
59. □
60. □
61. □   /* 5. info_ptr          */
62. □   pucPngData = png_get_rows(ptData->ptPngStrPoint, ptData->ptPngInfoPoint);
//       png_get_rowbytes();
63. □   if (ptData->iChannels == 4) { //    24    32
64. □       ptData->iRawSize= ptData->iWidth * ptData->iHeight*4; //
65. □       ptData->pucRawData= (unsigned char*)malloc(ptData->iRawSize);
66. □       if (NULL == ptData->pucRawData) {
67. □           printf("malloc rgba faile ...\n");
68. □           png_destroy_read_struct(&ptData->ptPngStrPoint, &ptData->ptPngInfoPoint, 0);
69. □           fclose(ptData->ptFp);
70. □           return -1;
71. □       }
72. □       /* pucPngData      RGBA
73. □          *      ABGR */
74. □       for (i = 0; i < ptData->iHeight; i++)
75. □           for (j = 0; j < ptData->iWidth * 4; j += 4) {
76. □               ptData->pucRawData[iPos++] = pucPngData[i][j + 3];
77. □               ptData->pucRawData[iPos++] = pucPngData[i][j + 2];
78. □               ptData->pucRawData[iPos++] = pucPngData[i][j + 1];
79. □               ptData->pucRawData[iPos++] = pucPngData[i][j + 0];
80. □           }
81. □
82. □       /*      RGBA  RGB888 */
83. □       if(RgbaToRgb(ptData)!=0)
84. □           return -1;
85. □
86. □   }
87. □   else if (ptData->iChannels == 3 ) { //        24    32
88. □       ptData->iRgbSize= ptData->iWidth * ptData->iHeight*3; //
89. □       ptData->pucRgbData = (unsigned char*)malloc(ptData->iRgbSize);
90. □       if (NULL == ptData->pucRgbData) {
91. □           printf("malloc rgba faile ...\n");
92. □           png_destroy_read_struct(&ptData->ptPngStrPoint, &ptData->ptPngInfoPoint, 0);
93. □           fclose(ptData->ptFp);

```

```

94. □         return -1;
95. □     }
96. □     /* pucPngData      RGB
97. □         *      BGR */
98. □     for (i = 0; i < ptData->iHeight; i++) {
99. □         for (j = 0; j < ptData->iWidth*3; j += 3) {
100. □             ptData->pucRgbData[iPos++] = pucPngData[i][j+2];
101. □             ptData->pucRgbData[iPos++] = pucPngData[i][j+1];
102. □             ptData->pucRgbData[iPos++] = pucPngData[i][j+0];
103. □         }
104. □     }
105. □     ptData->iBpp = 24; //      RGB888
106. □ }
107. □ else return -1;
108. □
109. □
110. □ /* 6:      */
111. □ png_destroy_read_struct(&ptData->ptPngStrPoint, &ptData->ptPngInfoPoint, 0);
112. □ fclose(ptData->ptFp);
113. □
114. □
115. □ return 0;
116. □ }

```

2.4

2.4.1

2.4.1.1

"lantianyu520" " "

	(0,0)	(0,0),(0,100),(200,0),(200,100)		
(40,50)	x	40/200=0.2	y	50/100=0.5
			Dx,Dy	Dx/400
(Sx,Sy)		(Dx,Dy)	Sw,Sh	Dw,Dh

$$Sx/Dx = Sw/Dw \quad Sy/Dy = Sh/Dh$$

$$Sx = Dx * Sw/Dw \quad Sy = Dy * Sh/Dh$$

2.4.1.2

2. 4

```

1. 1/*****
2. 2. *      PicZoom
3. 3. *
4. 4. *      ,      free
5. 5. *      "      "      "lantianyu520"      "      "
6. 6. *      ptPicData -
7. 7. *      fSize      -
8. 8. *      ptPicData->pucZoomData,
9. 9. *      0 - ,      -
10. 10. * *****/
11. 11. int PicZoom(PT_PictureData ptPicData, float fSize)
12. 12. {
13. 13.     ptPicData->iZoomWidth = ptPicData->iWidth * fSize;
14. 14.     ptPicData->iZoomHeight= ptPicData->iHeight* fSize;
15. 15.     unsigned long* pdwSrcXTable;
16. 16.     unsigned long x;
17. 17.     unsigned long y;
18. 18.     unsigned long dwSrcY;
19. 19.     unsigned char *pucDest;
20. 20.     unsigned char *pucSrc;
21. 21.     unsigned long dwPixelBytes = ptPicData->iBpp/8;
22. 22.     ptPicData->pucZoomData= malloc(sizeof(unsigned char) * ptPicData->iZoomWidth*ptPicData-
23. 23.     >iZoomHeight*ptPicData->iBpp/8);
24. 24.     pdwSrcXTable = malloc(sizeof(unsigned long) * ptPicData->iZoomWidth);
25. 25.     if ( NULL == pdwSrcXTable){

```

```

25. □      printf("malloc error!\n");
26. □      return -1;
27. □  }
28. □
29. □  /*      for      Sx = Dx * Sw/Dw Sy = Dy * Sh/Dh*/
30. □  for (x = 0; x < ptPicData->iZoomWidth; x++){//      pdwSrcXTable
31. □      /*      for      x
32. □      * pdwSrcXTable[x]      Sx,
33. □      * x      Dx,
34. □      * ptPicData->iWidth      Sw
35. □      * ptPicData->iZoomWidth      Dw*/
36. □      pdwSrcXTable[x]=(x*ptPicData->iWidth/ptPicData->iZoomWidth);
37. □  }
38. □
39. □  for (y = 0; y < ptPicData->iZoomHeight; y++){
40. □  /*      2      y
41. □      * dwSrcY      Sy,
42. □      * y      Dy,
43. □      * ptPicData->iHeight      Sh
44. □      * ptPicData->iZoomHeight      Dh*/
45. □      dwSrcY = (y * ptPicData->iHeight / ptPicData->iZoomHeight);
46. □  /*      RGB      */
47. □      pucDest = ptPicData->pucZoomData + y*ptPicData->iZoomWidth*3;
48. □      pucSrc = ptPicData->pucRgbData + dwSrcY*ptPicData->iWidth*3;
49. □
50. □  /*      */
51. □      for (x = 0; x < ptPicData->iZoomWidth; x++){
52. □          memcpy(pucDest+x*dwPixelBytes, pucSrc+pdwSrcXTable[x]*dwPixelBytes,
dwPixelBytes);
53. □      }
54. □  }
55. □
56. □  free(pdwSrcXTable);
57. □  return 0;
58. □

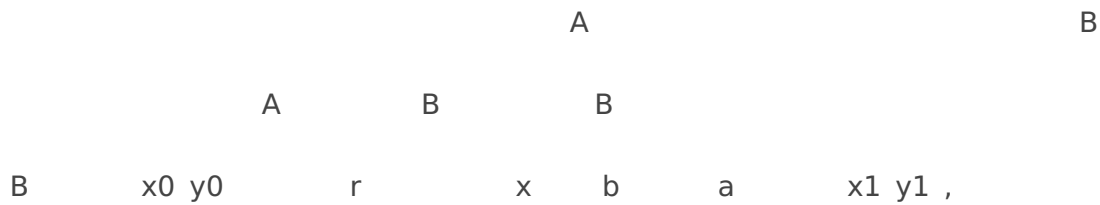
```

2.4.2

2.4.2.1

“ “ “ “

1.



ImageProcess_Image005

Image not found or type unknown

$x_0 = r \cos b$ $y_0 = r \sin b$

$x_1 = r \cos(b-a) = r \cos b \cos a + r \sin b \sin a = x_0 \cos a + y_0 \sin a$

$y_1 = r \sin(b-a) = r \sin b \cos a - r \cos b \sin a = -x_0 \sin a + y_0 \cos a$

RGB A B x1,y1 A

2.4.2.2

2.5

```
1. #define PI 3.1415926535
2. //
3. #define RADIAN(angle) ((angle)*PI/180.0)
4.
5.
6.
7.
8.
9. typedef struct ConcernCoor {
10.     int iLTx; // left top x
11.     int iLTy; //left top y
```

```

12. □ int iLBx; //left bottom x
13. □ int iLBy; //left bottom y
14. □ int iRTx; //right top x
15. □ int iRTy; //right top y
16. □ int iRBx; // right bottom x
17. □ int iRBy; // right bottom y
18. □ T_ConcernCoor, *PT_ConcernCoor;
19. □
20. □
21. □ /*****
22. □ *      max
23. □ *
24. □ *      x y  int
25. □ *
26. □ *      x y
27. □ *****/
28. □ static int max(int x,int y){
29. □     return x>y?x:y;
30. □
31. □ /*****
32. □ *      PicRotate
33. □ *
34. □ *      ,      free
35. □ *      "      "      "      "
36. □ *      ptPicData -
37. □ *      fAngle      -      0<=angle<=360
38. □ *      ptPicData->pucRotateData,      rgb
39. □ *      0 -      ,      -
40. □ *****/
41. □ int PicRotate(PT_PictureData ptPicData,float fAngle)
42. □ {
43. □     int i ,j;
44. □     T_ConcernCoor tConCor,tRonCor;
45. □     //
46. □     //int iSrcLineSize = bitCount * srcW / 8;
47. □     int iSrcLineSize = ptPicData->iBpp* ptPicData->iZoomWidth / 8;
48. □     int iDesLineSize;
49. □     int iX; //      x
50. □     int iY; //      y
51. □
52. □     /*      A      B      ,

```

```

53. □      *
54. □      * tConCor      B
55. □      * tRonCor      B      */
56. □      tConCor.iLTx = -ptPicData->iZoomWidth/2; tConCor.iLTy = ptPicData->iZoomHeight/2;
57. □      tConCor.iRTx = ptPicData->iZoomWidth/2; tConCor.iRTy = ptPicData->iZoomHeight/2;
58. □      tConCor.iLBx = -ptPicData->iZoomWidth/2; tConCor.iLBy = -ptPicData->iZoomHeight/2;
59. □      tConCor.iRBx = ptPicData->iZoomWidth/2; tConCor.iRBy = -ptPicData->iZoomHeight/2;
60. □
61. □
62. □      /*      B      */
63. □      double sina = sin(RADIAN(fAngle));
64. □      double cosa = cos(RADIAN(fAngle));
65. □      tRonCor.iLTx = tConCor.iLTx * cosa + tConCor.iLTy * sina;
66. □      tRonCor.iLTy = -tConCor.iLTx * sina + tConCor.iLTy * cosa;
67. □      tRonCor.iRTx = tConCor.iRTx * cosa + tConCor.iRTy * sina;
68. □      tRonCor.iRTy = -tConCor.iRTx * sina + tConCor.iRTy * cosa;
69. □      tRonCor.iLBx = tConCor.iLBx * cosa + tConCor.iLBy * sina;
70. □      tRonCor.iLBy = -tConCor.iLBx * sina + tConCor.iLBy * cosa;
71. □      tRonCor.iRBx = tConCor.iRBx * cosa + tConCor.iRBy * sina;
72. □      tRonCor.iRBy = -tConCor.iRBx * sina + tConCor.iRBy * cosa;
73. □
74. □
75. □      /*      */
76. □      ptPicData->iRotateWidth = max(abs( tRonCor.iRBx - tRonCor.iLTx), abs( tRonCor.iRTx -
77. □      ptPicData->iRotateHeight = max(abs( tRonCor.iRBy - tRonCor.iLTy), abs( tRonCor.iRTy -
78. □      tRonCor.iLBy));
79. □      /*      3      */
80. □      iDesLineSize = ((ptPicData->iRotateWidth* ptPicData->iBpp+ 23) / 24) * 3 ;
81. □      /*      */
82. □      ptPicData->pucRotateData = malloc(iDesLineSize * ptPicData->iRotateHeight);
83. □      if(NULL == ptPicData->pucRotateData){
84. □          printf("malloc error\n");
85. □          return -1;
86. □      }
87. □
88. □      /*      *
89. □      * i, j      B x1, y1*/
90. □      for (i = 0; i < ptPicData->iRotateHeight; i++){
91. □          for (j = 0; j < ptPicData->iRotateWidth; j++){

```

```

92. □          /*      B  x,y1          iX,iY,      x0,y0 */
93. □          iX = (j - ptPicData->iRotateWidth / 2)*cos(RADIAN(360 - fAngle)) + (-i +
ptPicData->iRotateHeight / 2)*sin(RADIAN(360 - fAngle));
94. □          iY = -(j - ptPicData->iRotateWidth / 2)*sin(RADIAN(360 - fAngle)) + (-i +
ptPicData->iRotateHeight / 2)*cos(RADIAN(360 - fAngle));
95. □          /*          */
96. □          if (iX > ptPicData->iZoomWidth / 2 || iX < -ptPicData->iZoomWidth / 2 || iY >
ptPicData->iZoomHeight / 2 || iY < -ptPicData->iZoomHeight / 2){
97. □              continue;
98. □          }
99. □          /*      B  x0,y0          A      */
100. □          int iXN = iX + ptPicData->iZoomWidth / 2;
101. □          int iYN = abs(iY - ptPicData->iZoomHeight / 2);
102. □          /*      */
103. □          memcpy(&ptPicData->pucRotateData[i * iDesLineSize + j * 3], &ptPicData-
>pucZoomData[iYN * iSrcLineSize + iXN * 3], 3);
104. □          }
105. □      }
106. □      return 0;
107. □

```

Revision #1

Created 2 March 2022 10:23:10 by

Updated 2 March 2022 10:23:20 by