C++ i/o (output) and image "matrix" (1D array)

Friday, October 29, 2010
8:51 AM

```
#ifndef uchar
typedef unsigned char uchar;
#endif
```

} on top of main.cpp or on top of pixel.h

with

```
drgb_t   color;
irgb_t   icolor;
```

or later

```
rgb_t<double> color;
rgb_t<uchar> color;
```

uchar in #include <sys/types.h>

```
std::cout << "P6" << w << "␣";
std::cout << h << "␣" << 255 << std::endl;

for (y = ...)
    for (x = ...)
        :
        ray_trace(... color ...);

        color = 255.0 * color;

for(i=0; i<3; i++)
    icolor[i] =
static_cast<uchar>(color[i]);
```

0.0 < double rgb < 1.0 effectively clamp()

255.0 * color: for unsigned uchar representation, 8 bits per channel

255 0 0

not a sign bit

1111 1111 0000 0000 0000 0000 base 2

0xff) 00 0 0 base16 (hex)

pixel

[3];

(400)

pixel

rgb_t<uchar> pix[3];

pix[0]='r'
pix[1]='g'
pix[2]='b'

0000 0000 = 0
1111 1111 = 0xff 255

base 2 base16 base 10

0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f

10

$10^1 + 10^0$

r	g	b
1111 1111	0000 0000	0000 0000
255	0	0

pix[0] = (0xff << 16) | (0x00 << 8) | (0x00)

```
                    0000 0000    bitwise
     0000 0000 0000 0000 0000      or
1111 1111 0000 0000 0000 0000
-----------------------------------------
1111 1111 0000 0000 0000 0000 0
```

//outputs 3 character r g b
for(i=0; i<3; i++)
    std::cout.put(ichar[i]);

Friday, October 29, 2010
9:25 AM

Another method for image assembly:

unsigned char * img = NULL;

(just a 1D array)

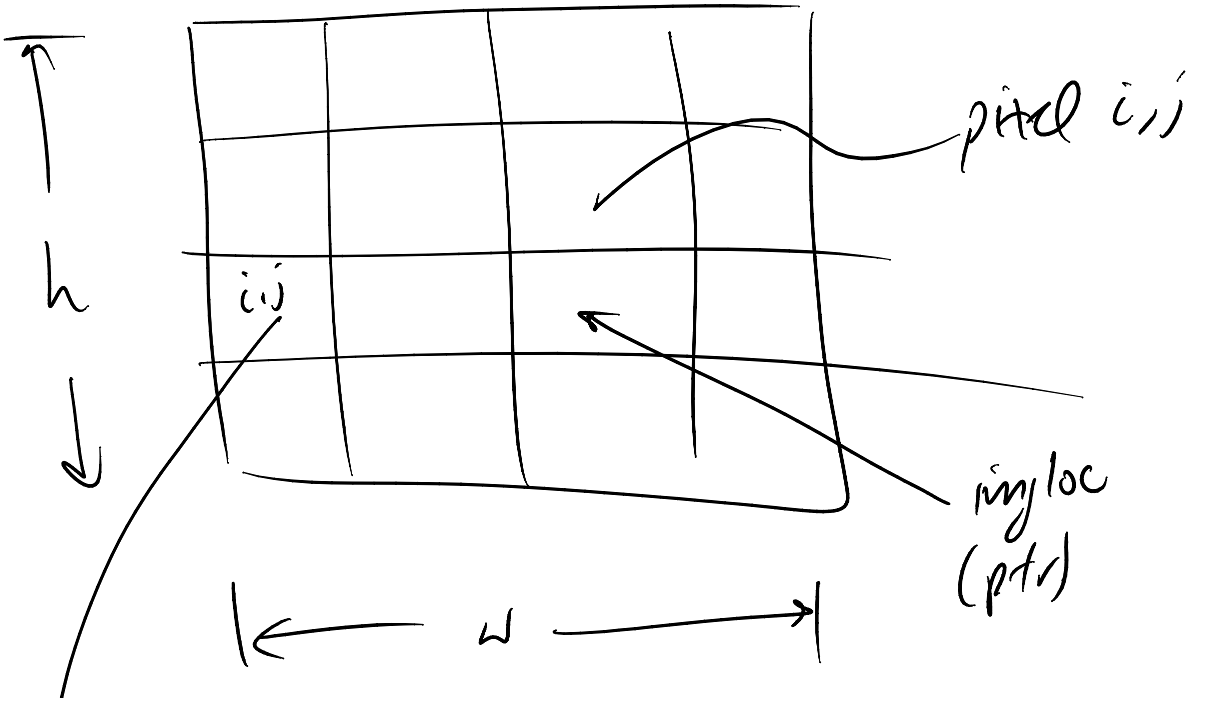
I can use 2D array

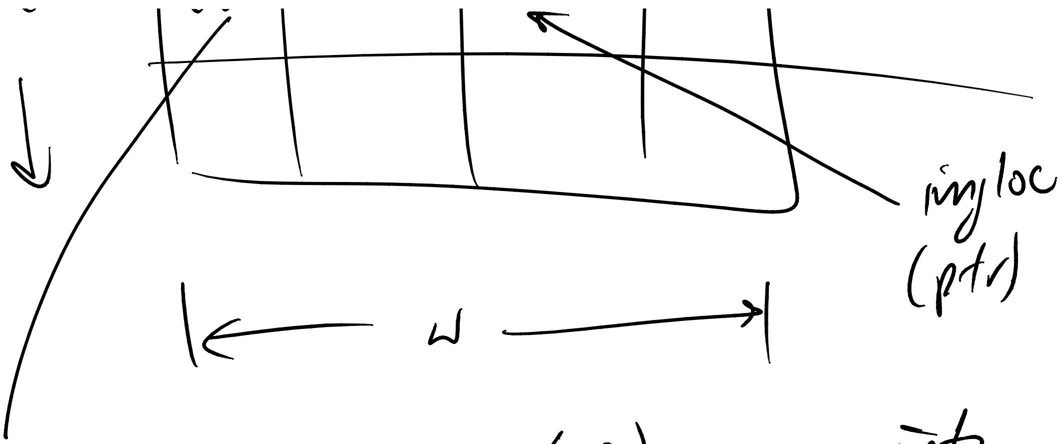
unsigned char img[w][h]

unsigned char *imgloc;

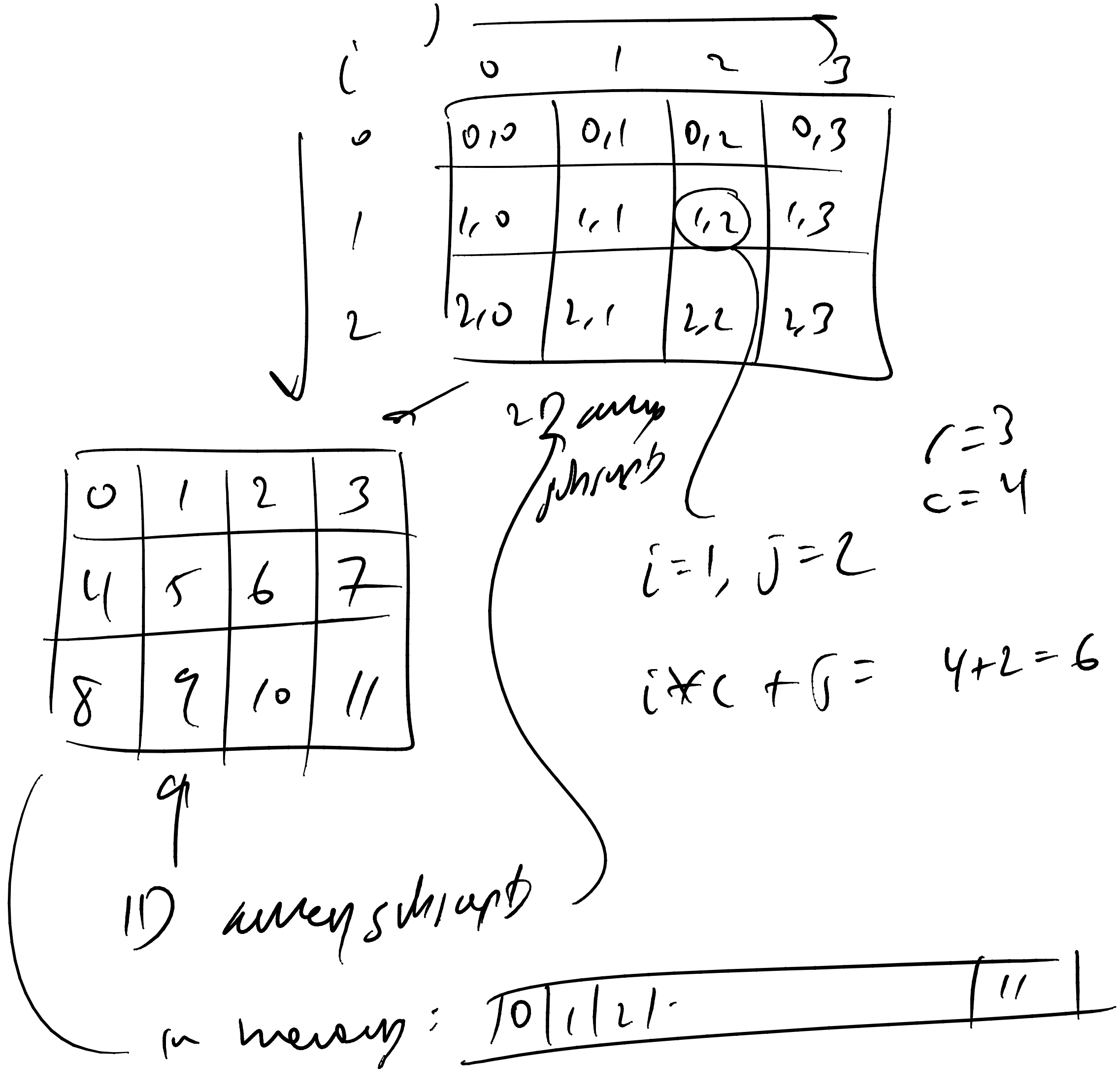
ptr to image location

Conceptually, you have a 2D matrix





given coordinates i,j (2D), access into a 1D array is via: $i * c + j$



$$\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 0,0 & 0,1 & 0,2 & 0,3 \\ 1 & 1,0 & 1,1 & 1,2 & 1,3 \\ 2 & 2,0 & 2,1 & 2,2 & 2,3 \end{array}$$

0	1	2	3
4	5	6	7
8	9	10	11

$r = 3$
$c = 4$

$i = 1, j = 2$

$i * c + j = 4 + 2 = 6$

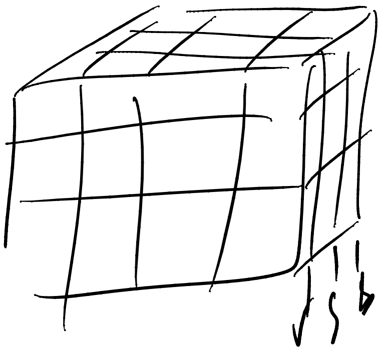
in memory: | 0 | 1 | 2 | · … | 11 |

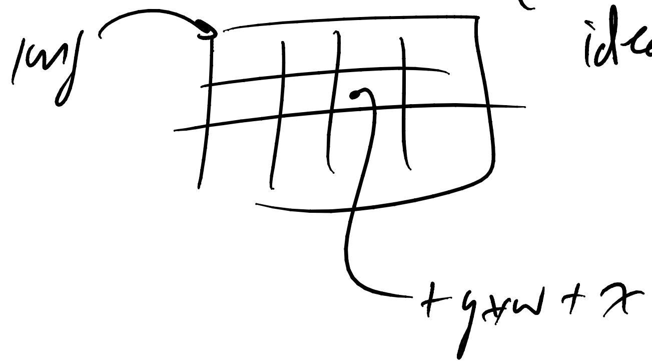
for (y = --)

```
for(x= - - - )
    :
        ray_trace( - - - color - - - )
        icolor = color * 255
        imgloc = img + y*w + x
```

↑ ptr to (0,0)

ptr offset (i*c+j idea)





```
for (i=0; i<3; i++)
    imgloc[i] = color[i]
```

↑ cast

```
std::cout.write((char*) imgloc, 3);
                (const char*)
```