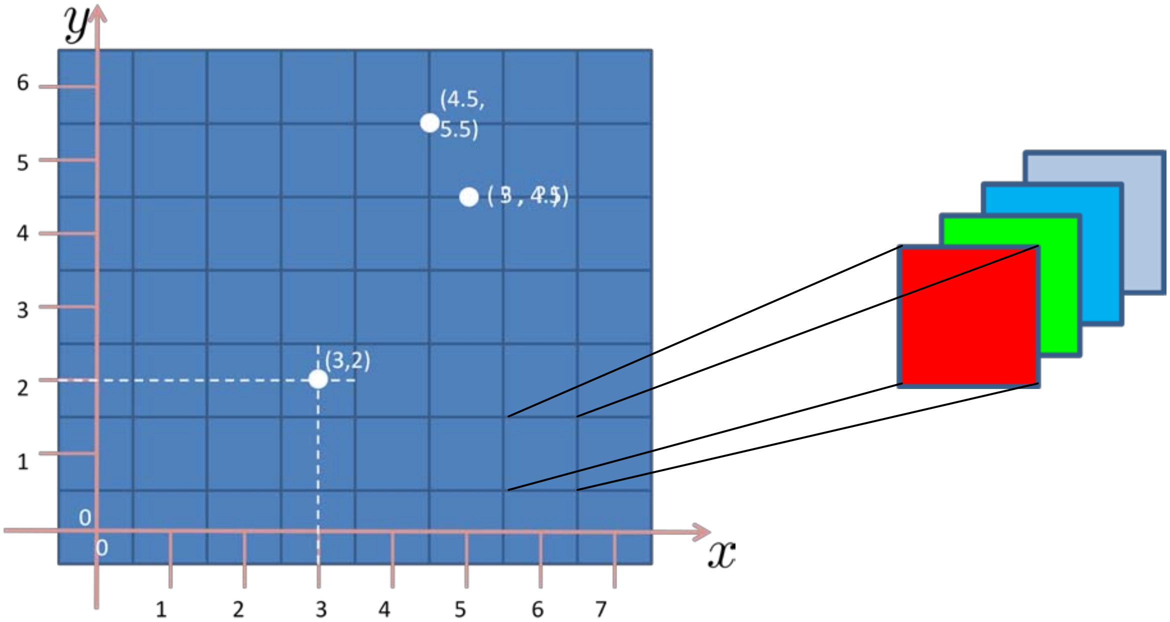
Frame buffer

A Frame buffer is shown in Figure below, the lower-left pixel is pixel (0, 0), corresponding to the window coordinates of the lower-left corner of the 1 × 1 region occupied by this pixel. In general, pixel (*x, y*) fills the region bounded by *x* on the left, *x*+1 on the right, *y* on the bottom, and *y*+1 on the top.



In the code you have been provided, this frame buffer is represented as a 2D array and the pointer to this 2D array is passed as "**`FrameBuffer *buffer`**" to functions drawline, drawcircle etc.

Using this pointer you can change the value of any pixel in the frame buffer using pointer arithmetic. The syntax for C with pointers is:

- `*array means to dereference the contents of array.`
- array[i] `means element number i, 0-based, of array which is translated into` *(array + i)
- `array + i` is the memory location of the (i+1)th element of `array`
- `*(array + i)` takes that memory address and dereferences it to access the value.

Since we are dealing with a color image, each pixel represents 4 values one each for red, green and blue channel and a alpha value (transparency). The integer range of these three color channels is 0-255 each.

Example 1: How to change the color of pixel location (x,y) = (4,3) to blue

```
int row = 3, col = 4*4;
buffer->data[row*buffer->width*4 + col+0] = 0;   //RED
buffer->data[row*buffer->width*4 + col+1] = 0;   //GREEN
buffer->data[row*buffer->width*4 + col+2] = 255; //BLUE
buffer->data[row*buffer->width*4 + col+3] = 0;   //ALPHA
```

The multiplication by 4 indicates that to move from one pixel to another I have to move 4 cells in the array.

Example 2: How to change the color of pixel location (x,y) = (3,5) to brown

```
int row = 5, col = 3*4;
buffer->data[row*buffer->width*4 + col+0] = 128; //RED
buffer->data[row*buffer->width*4 + col+1] = 64;  //GREEN
buffer->data[row*buffer->width*4 + col+2] = 64;   //BLUE
buffer->data[row*buffer->width*4 + col+3] = 0;   //ALPHA
```

For more information about numerical representation of RGB colors see link below:
http://en.wikipedia.org/wiki/RGB#Numeric_representations