From Data to Pixels

Day 4, Markku Reunanen

Direct pixel access

Let's leave the comfort of ready-made graphics functions, such as *rect()*, *line()* and so on

Much faster for certain things, needed for image processing in any case

The *pixels[]* array contains screen pixels

Call *loadPixels()* first and *updatePixels()* when all done

Don't use built-in graphics functions between the two

Direct pixel access

Any modifications to the *pixel[]* array are reflected on the screen

How to get a pixel at a given location (x,y)?

No safety networks here: go outside the array and the sketch will crash – extra care and checks needed

PImages work the same way, there is *.pixels[]*

Processing pixel format

The easy way is to use the #RRGGBB hexadecimal notation

Let's see the binary/hex/decimal trainer

Pixels are internally 32-bit (4 byte) numbers consisting of:

Alpha (8 bits), Red (8 bits), Green (8 bits), Blue (8 bits)

Each component can be in the range 0..255 (0..0xFF in hex)

Binary and hex

Binary	Hex	Binary	Hex
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Logical operators

AND: turn bits into zeroes / leave certain bits only

- In Processing marked as &

OR: turn bits into ones / combine multiple values

- In Processing marked as |

XOR: change bit status

- In Processing marked as ^

NOT: invert bits, in Processing marked as ~

Shifting

Moving bits left or right

Shift left in Processing: <<

00110100 shifted left becomes 01101000

Shift right in Processing >>

00110100 shifted right becomes 00011010

Exctracting A, R, G and B

Alpha is at the top bits, shift right by 24 and then AND with 0xFF

Red is next, shift right by 16 and then AND with 0xFF

Green is next, shift right by 8 and then AND with 0xFF

Blue doesn't need shifting, just AND with 0xFF

When rebuilding a pixel out of individual values, shift to the opposite direction and combine using OR