Realtime Graphics

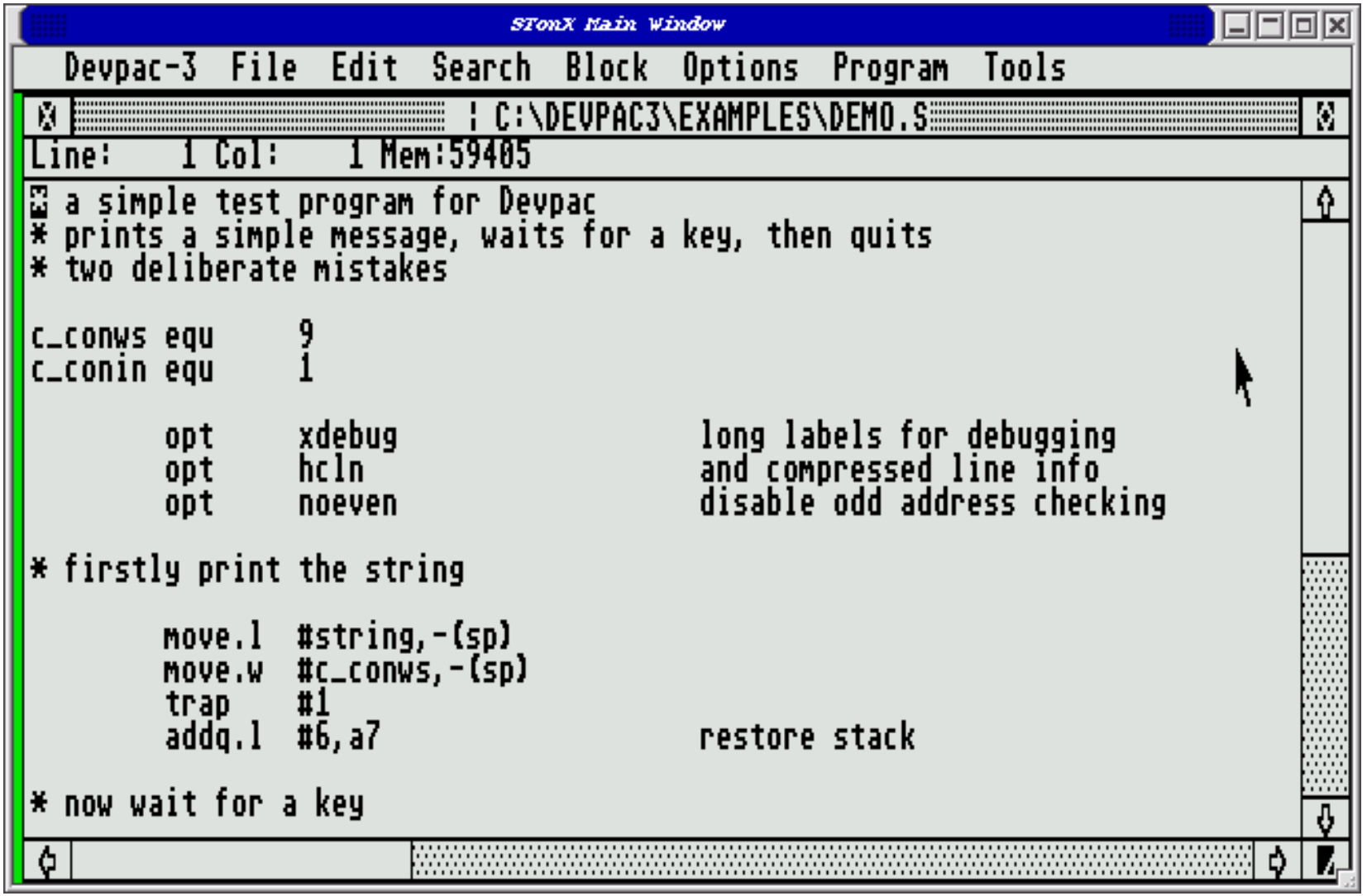
Guest lecture in “Datorgrafik med Interaktion”

by Mikael Kalms

Me.

My formative years

My first assembler



Oups.

Back then, this was cool:

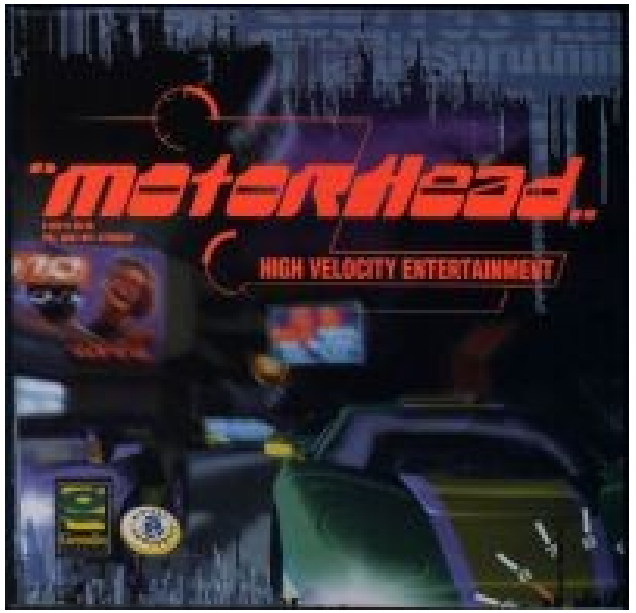
Back then, this was fresh:

My machine of choice since 1994

+

50**M**Hz CPU
64**M**B Main Memory
2**M**B Video Memory
Typical display resolution: 320x256 pixels, 256 simultaneous colours
No 3d hardware whatsoever

I've worked on games.



I've worked on games.

I've worked on games.

I've worked on games.

But, back on topic.

Characteristics of realtime graphics

- 10-100 images/second generated (so: high throughput)
- If it's interactive, then latency is also important
- Minimum performance is important; it is often better to have consistent performance, than high peak performance

My approach to realtime graphics programming

50% play around

50% directed work

My choice of language: C++

Why?

- “the most advanced assembler on the market”
- Available on nearly all machines
- Can talk directly to the OS
- Control over memory usage
- Control over CPU usage
- No garbage collection

You can use other languages too, but then you are limiting yourself in some ways.

Things to come

Before the break, we will discuss...

Our test harness.

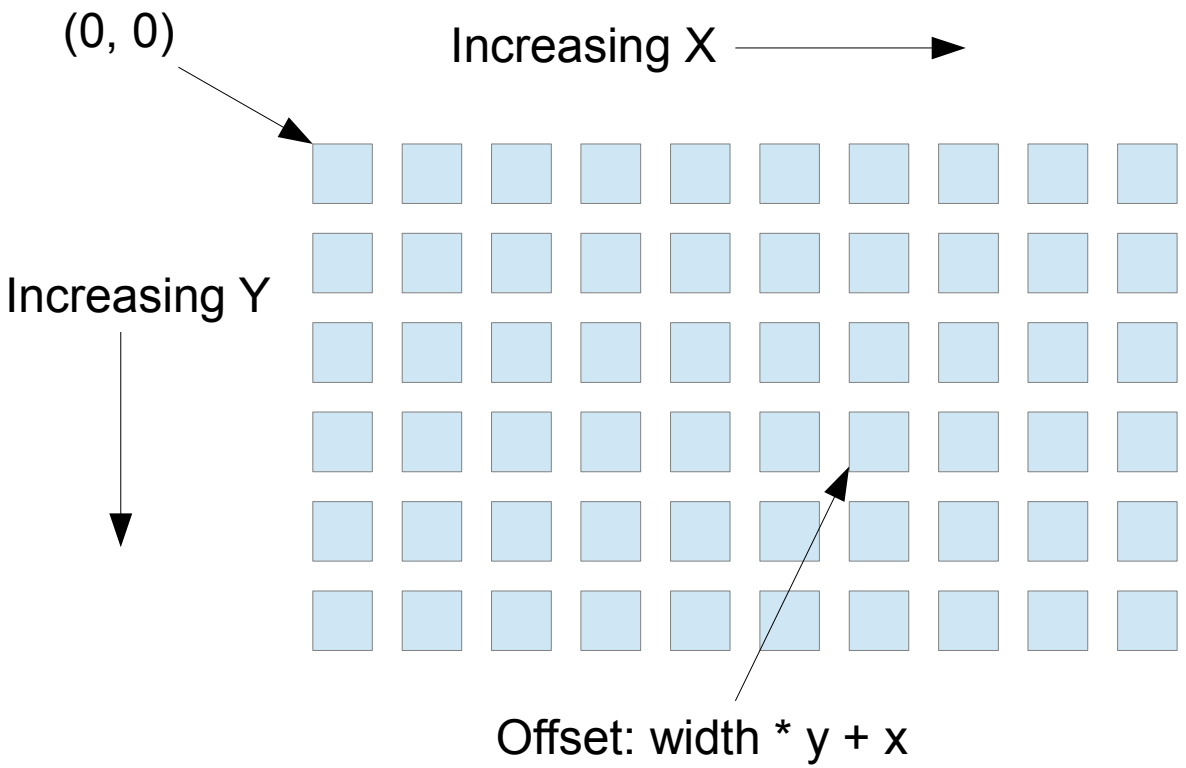
Image processing.

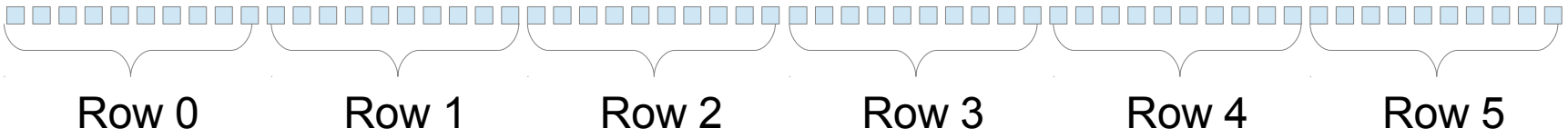
Image bending.

A. Our test harness

- Blank canvas
- What's an Image?
- What's a FloatRGBColor?
- How do RGB colors work?
- How do I draw a pixel?

Image coordinate system



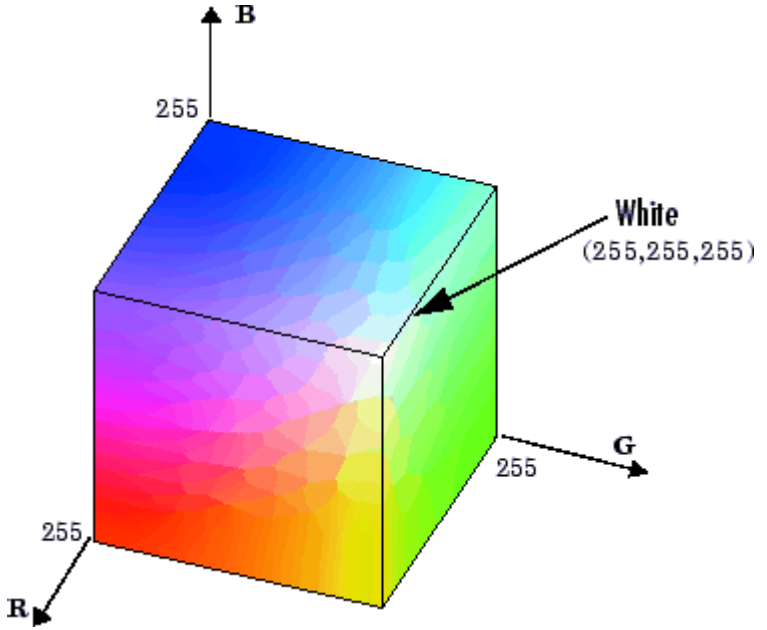


What's a an RGB color?

A triplet of color intensities (Red / Green / Blue)

Can be seen as “a vector in the RGB color space”

Or, a position in the “color cube”

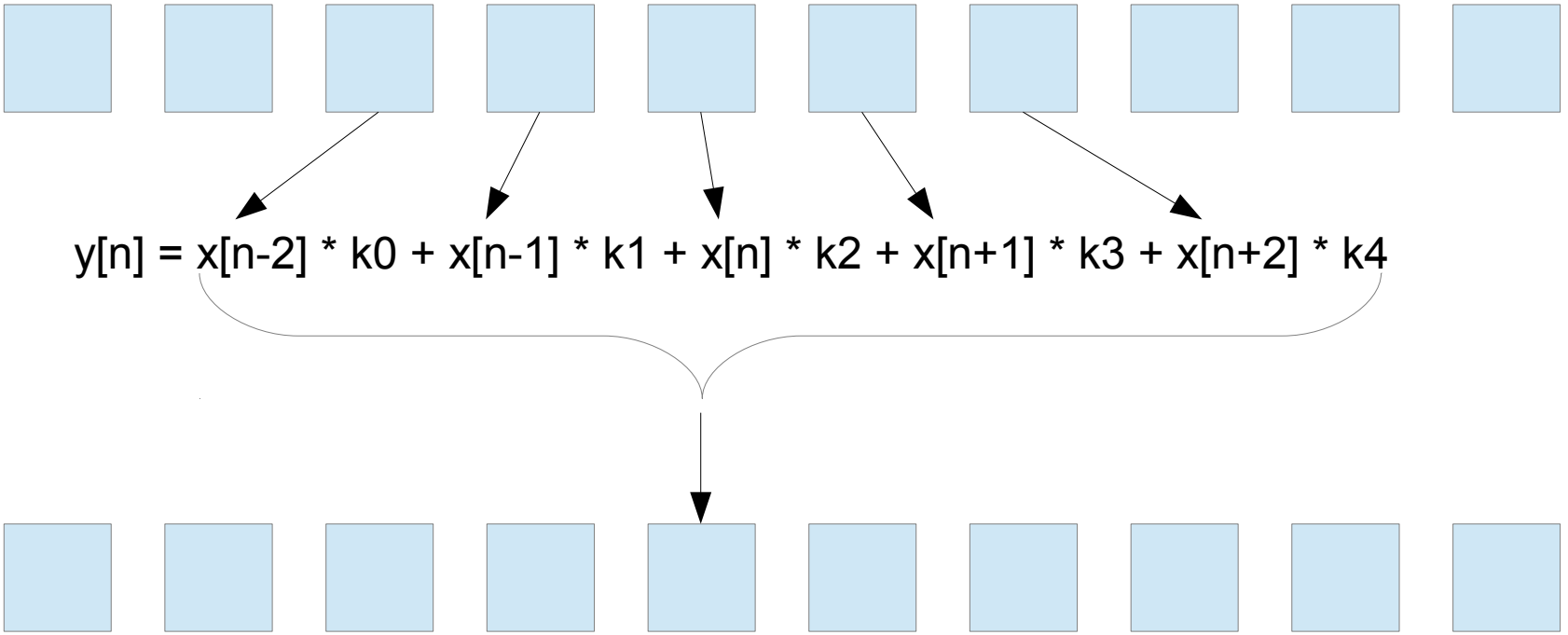


B. Image processing

- Displaying an image.
- Computations on color values.
- Computations on color components.
- FIR low pass filter.
- IIR low pass filter.

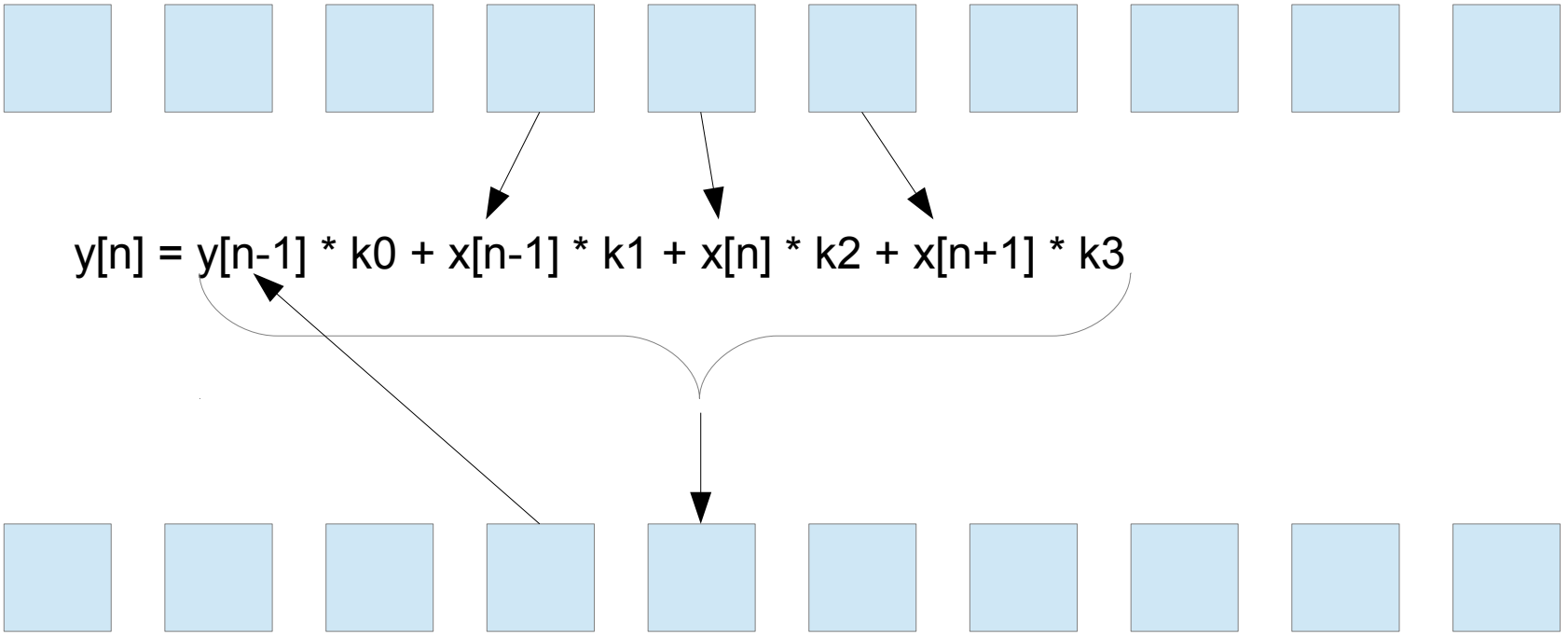
FIR Filter

Output pixel = weighted combination of some neighbouring input pixels



IIR Filter

Output pixel = weighted combination of some neighbouring input pixels AND some neighbouring input pixels



C. Image bending

- How do we scroll an image?
- Per-line distortion.
- Image zooming (map source → dest)
- Image zooming (map dest → source)

Conclusions from image zoomer

Mapping dest → source is usually better when doing image manipulation!

Relax.

It's break time.

Regroup in 15.

Things to come

Rendering filled shapes...

… using per-pixel test

… using scanconversion

… using spantables

And then, filling the shapes with interesting bits

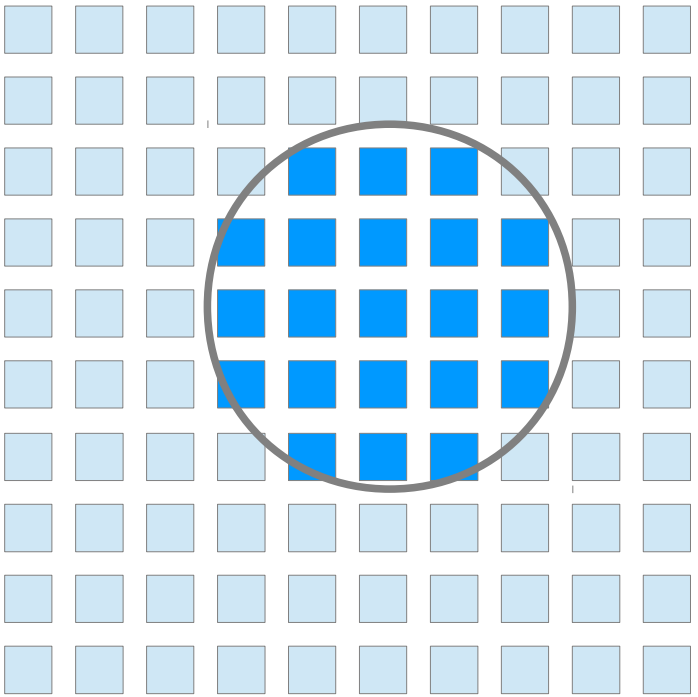
Finally, let's do something interesting with our tools.

E. Per-pixel test

- How do we draw a circle?
- Drawing a halfspace.
- Drawing a triangle.

Circle equation

$$x^2 + y^2 <= r^2$$



Example:

$(x-x0)^2 + (y–y0)^2 <= r^2$

with:

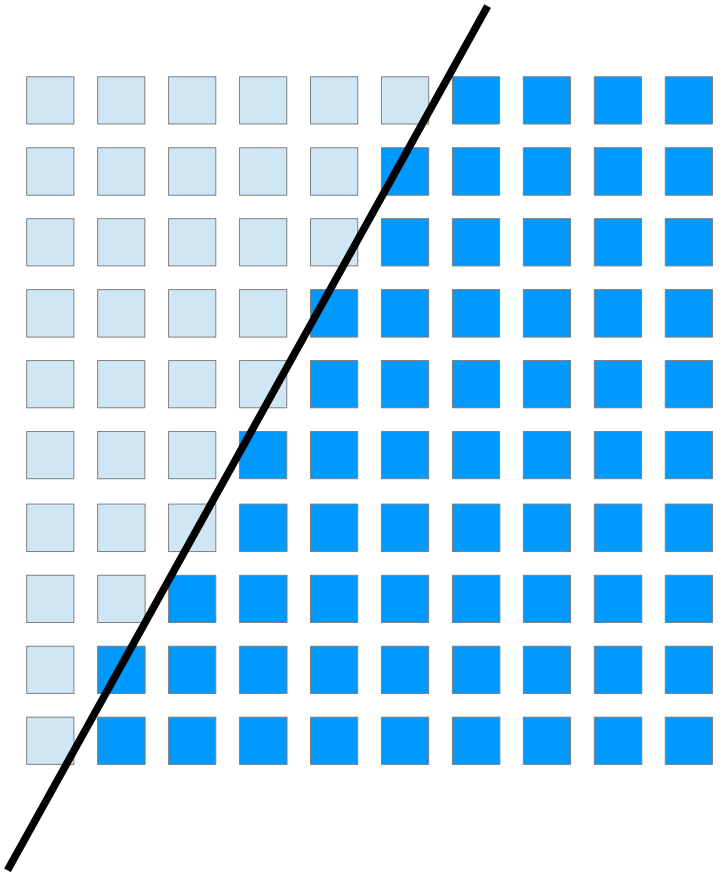
x0 = 5

y0 = 4

r = 2

Halfspace equation

$$Ax + By + C >= 0$$



Example:

$Ax + By + C >= 0$

with:

$A = -2$

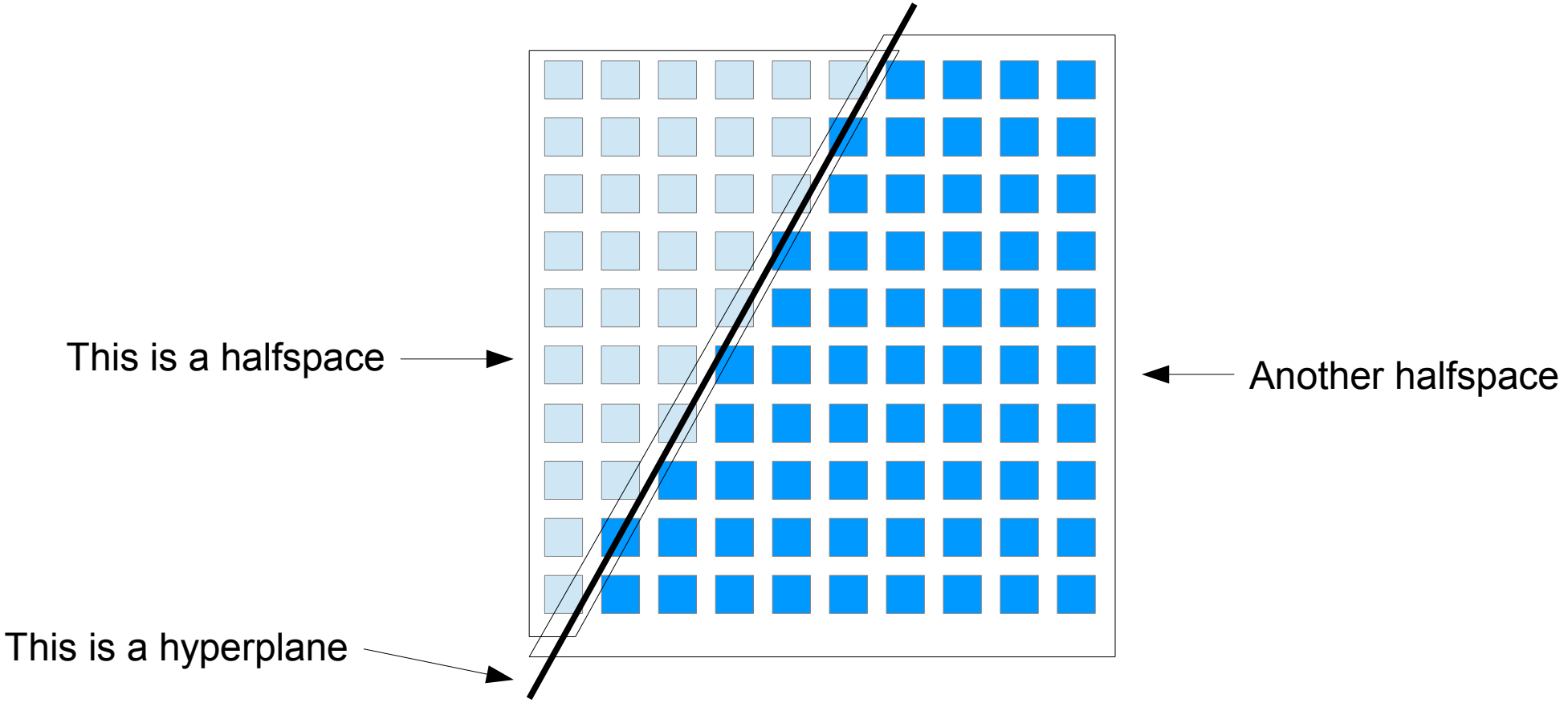
$B = -1$

$C = 10$

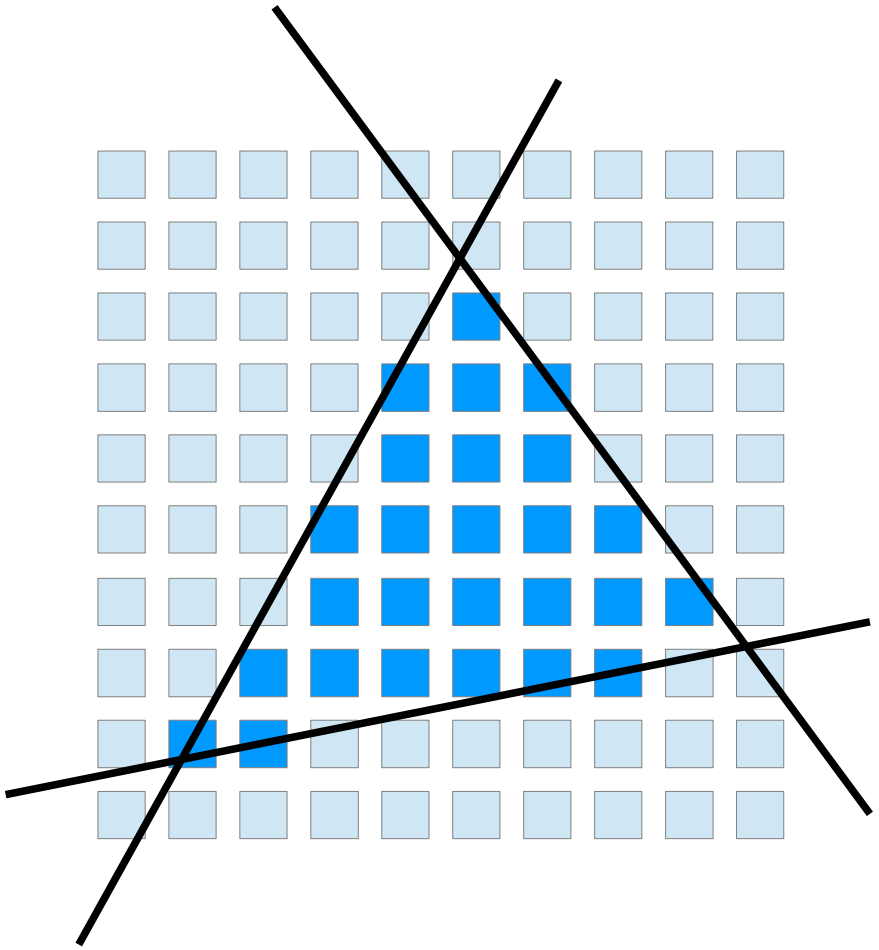
Halfspace vs hyperplane

Hyperplane = point (1D), line (2D), plane (3D)

A hyperplane divides space into two halfspaces



Triangle equation set



Set up one halfspace equation per edge.

If halfspace1 && halfspace2 && halfspace3 holds, then (x,y) is inside the triangle.

F. Scanconversion / spantables

- Why scanconversion?
- Draw circle using scanconversion.
- Draw triangle using scanconversion.
- Draw triangle using spantables.

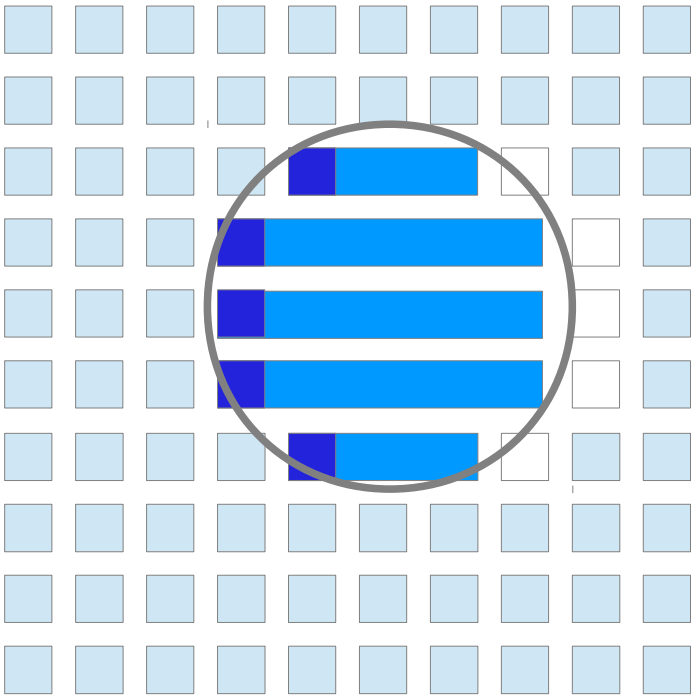
Why scanconversion?

Take advantage of correlation between neighboring pixels → performance

Also – it expands your mind

Circle scanconversion

$$(x-x0) = \pm(r^2 - (y-y0)^2)^{1/2}$$



Example:

$x = x0 \pm(r^2 - (y-y0)^2)^{1/2}$

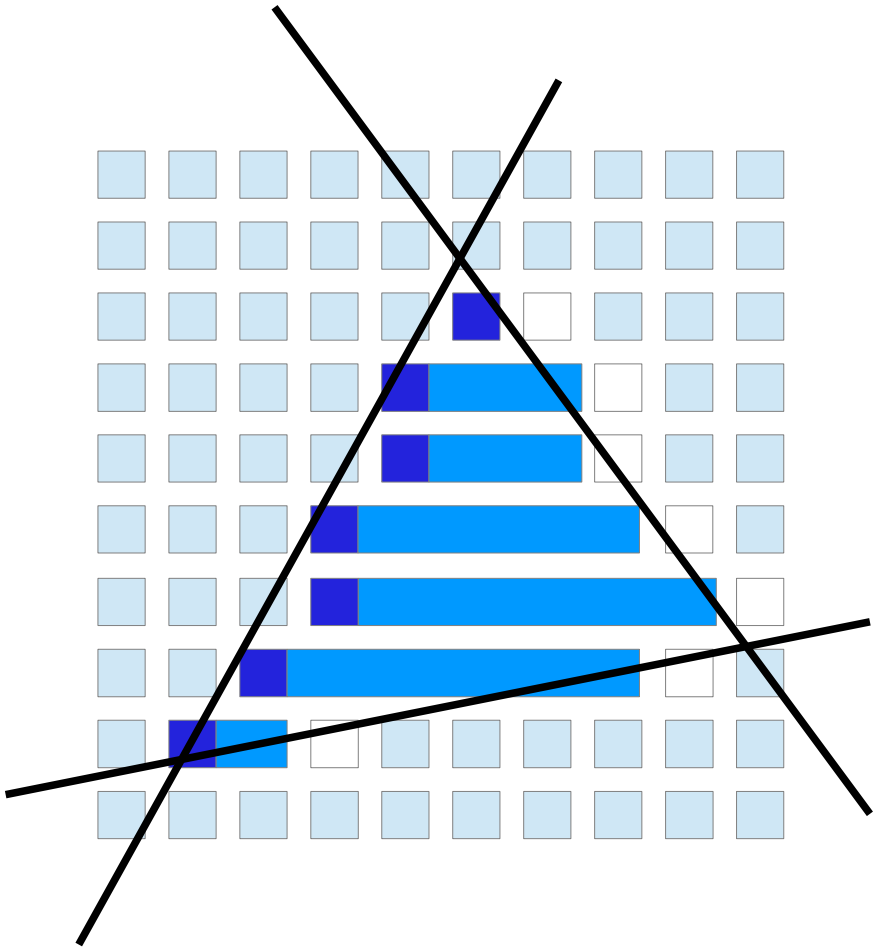
with:

x0 = 5

y0 = 4

r = 2

Triangle scanconversion



G. Shading our triangles

- Gouraud shaded triangle
- Texturemapped triangle

Gouraud shading

Specify colour values at each vertex in triangle

Linearly interpolate colour values within triangle

Linear gradients!

Texture mapping

Specify texture coordinates at each vertex in triangle

Linearly interpolate texture coordinates within triangle

Linear gradients!

At each pixel, lookup color from texture (at the current texture coordinate)

H. Using our texturemapper

- Zoomrotator.
- Image distorter.
- Tunnel.

The end.

Questions?

Reach me at: mikael@kalms.org