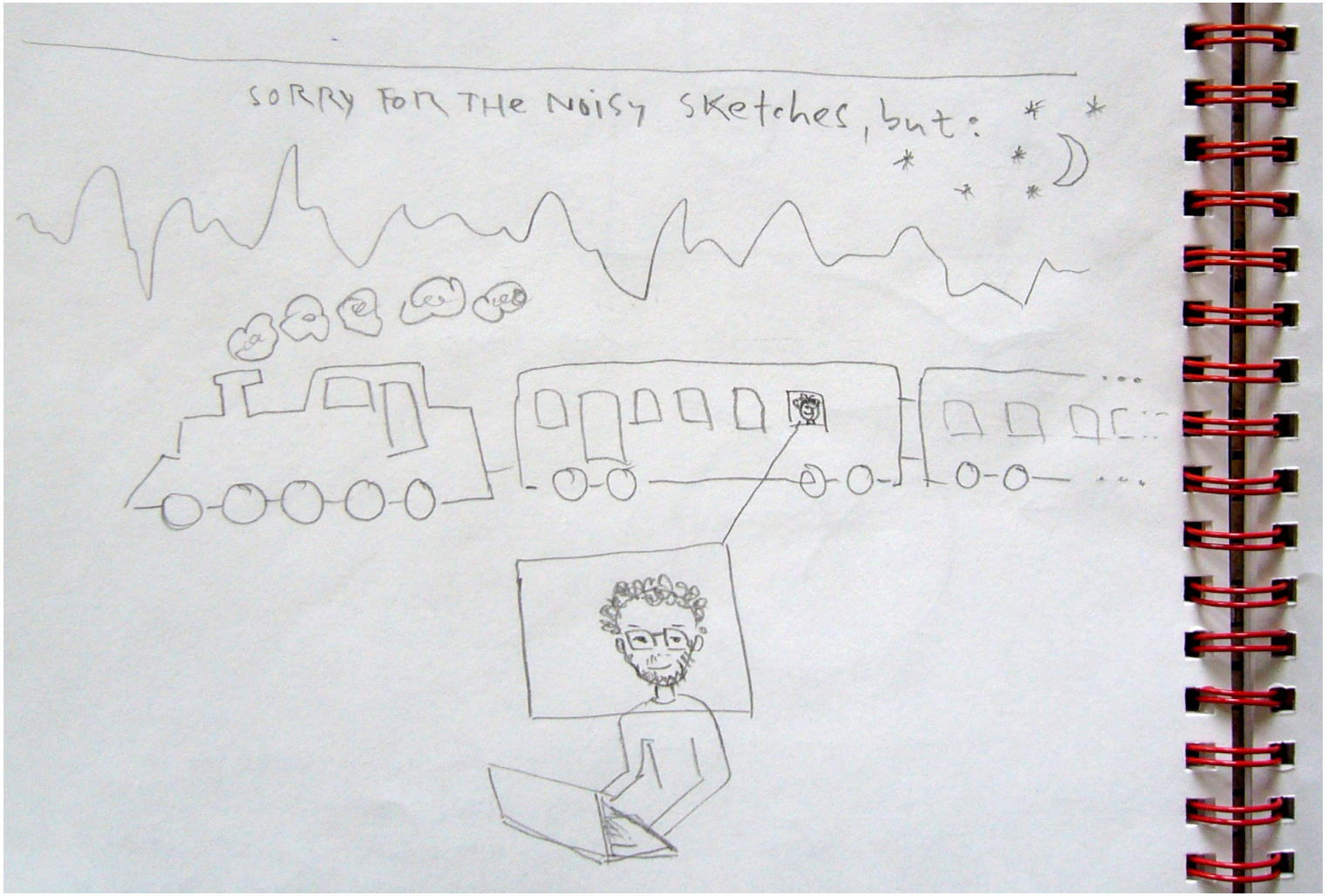
Shader Applications

Michael Mehling / michel
Sebastian Gregor / gregsn

Node08

April, 11th 2008

Sorry for the noise, but...



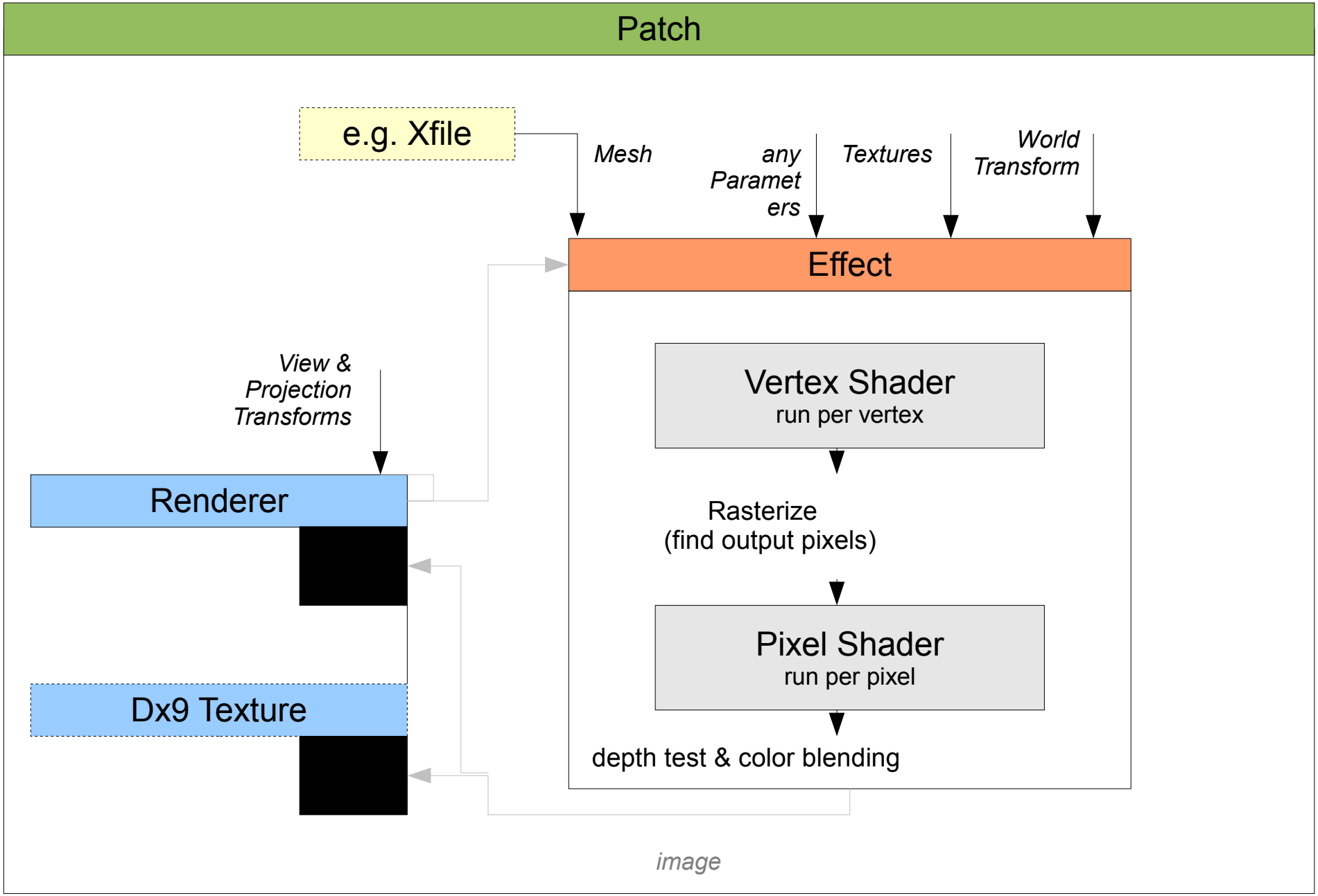
Overview

- Introduction / Repetition
  - Workflow
  - Framework & Pipeline
  - Spaces & Transformations
  - Texture Formats
- Selected Applications
  - „Canvas“ based
  - Geometry based
- Daily Tools

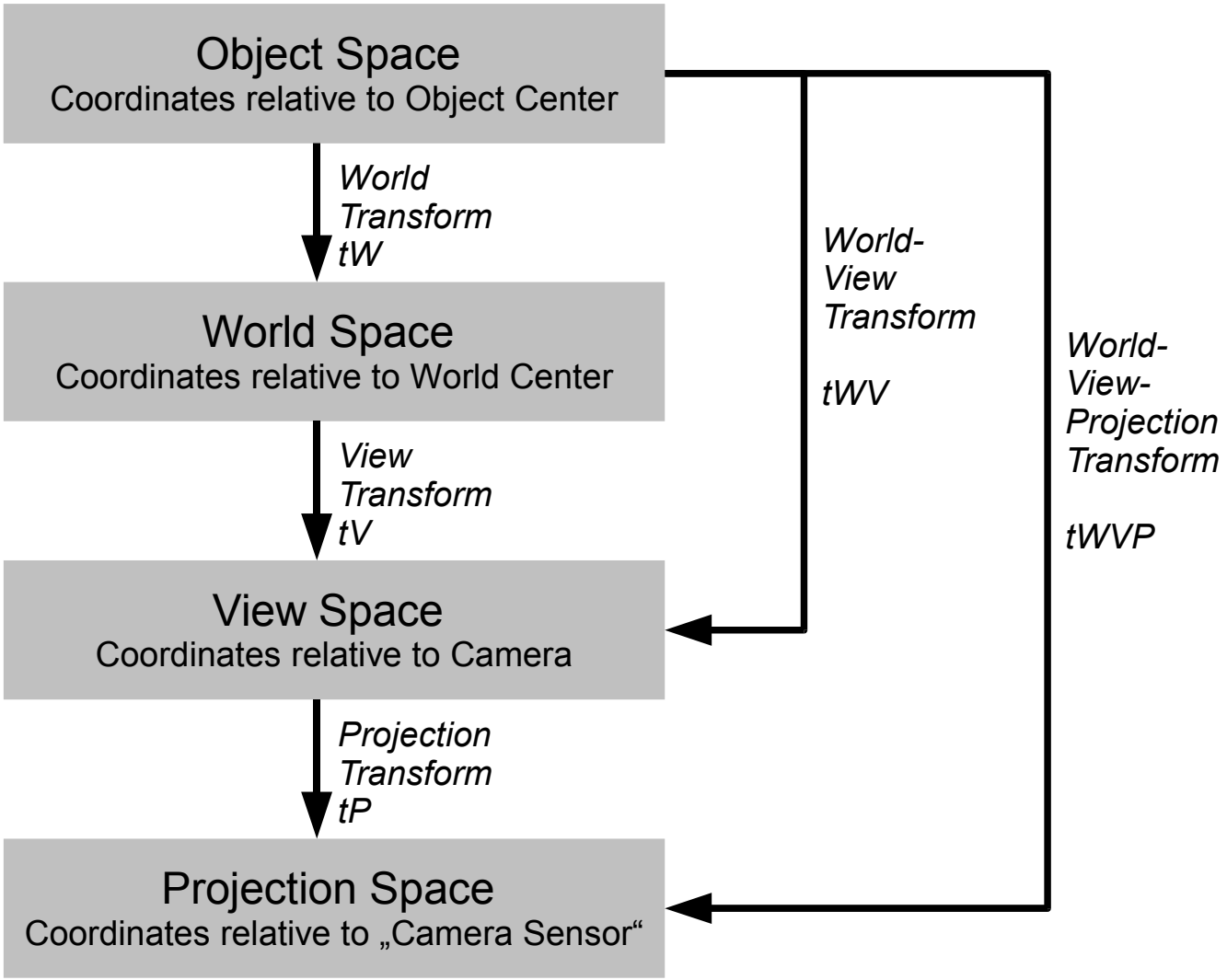
Workflow

- Pencil & Paper
  - What do i want to achieve?
  - What do i need? (a good article, some math...)
- Patch (CPU) & Shader (GPU)
  what has to be done where?
- Modularity
- Interfaces
  - of Patches
  - of Shaders

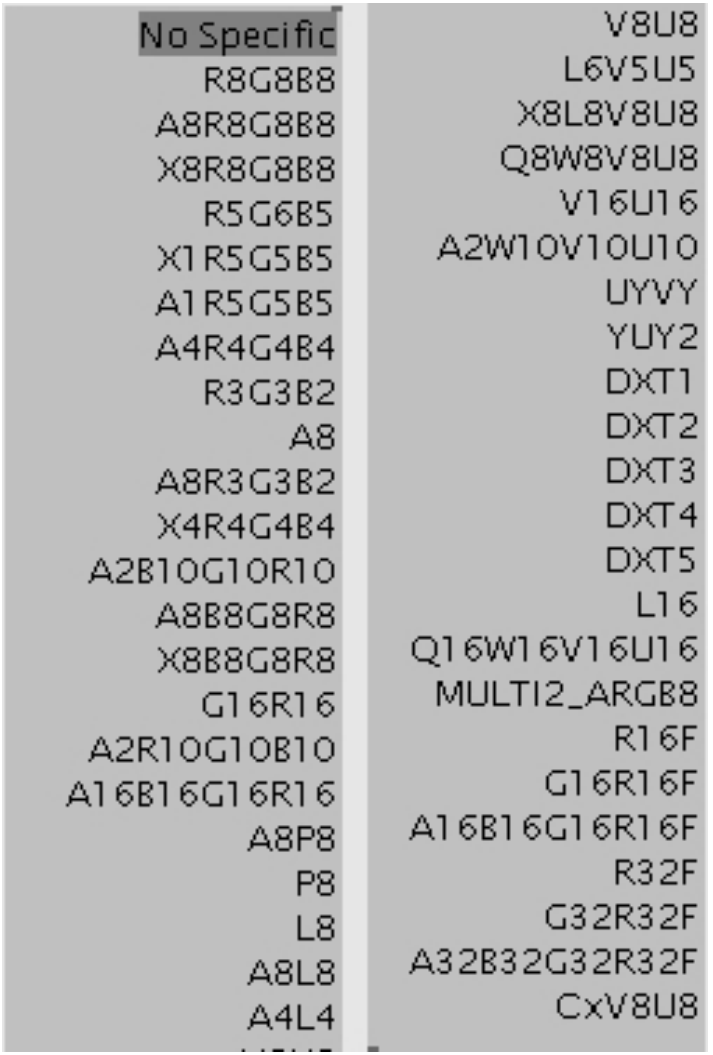
Framework / Pipeline



Spaces & Transformations



Texture Formats



Channels

R = Red

B = Blue

G = Green

A = Alpha

X = Not Used

L = Luminosity

U = ?

V = ?

W = ?

Q = ?

P = ?

Type

F = Float *(7.2340)*

„No F“ = values can only range from 0..1 !

The number means how many bits

Specials

DXT1 - DXT4 = some compressed texture format

... = ?

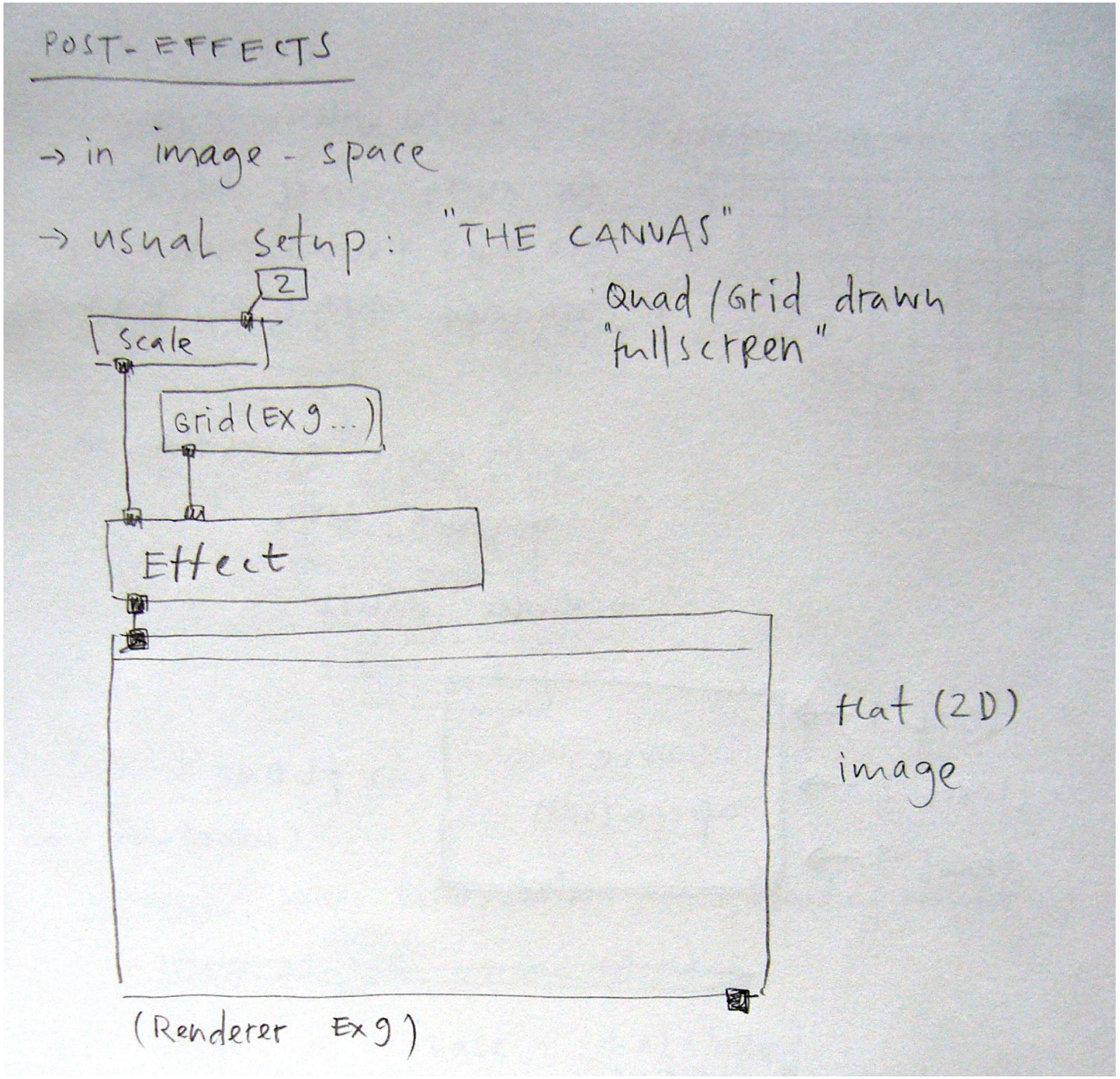
examples:

- *G16F = 16 bit for Green FLOAT! (- unlimited to +unlimited)*
- *R8 = 8 bits for Red*
- *A2B10G10R10 =*
  - *A2: upper most 2 bits for Alpha Channel (0, 0.333, 0.666, or 1)*
  - *B10: next ten bits also for 0..1*
  - *Next ten bits go to G ...*

„Canvas“ Based Applications

- Image Post Processing
  - Color Transforms
  - Blending
  - Halftoning
  - Ascii Shader
- Feedback based
  - Wave Shader
- Metaballs

„Canvas“ based



Pixelshader Tricks

When thinking of pixel shader programs we can think of a program acting on one pixel only, which is much easier. All pixels behave in the same way...

In a „Canvas“ based shader we use a Fullscreen Quad (a quad scaled to cover the whole screen / scale = 2)

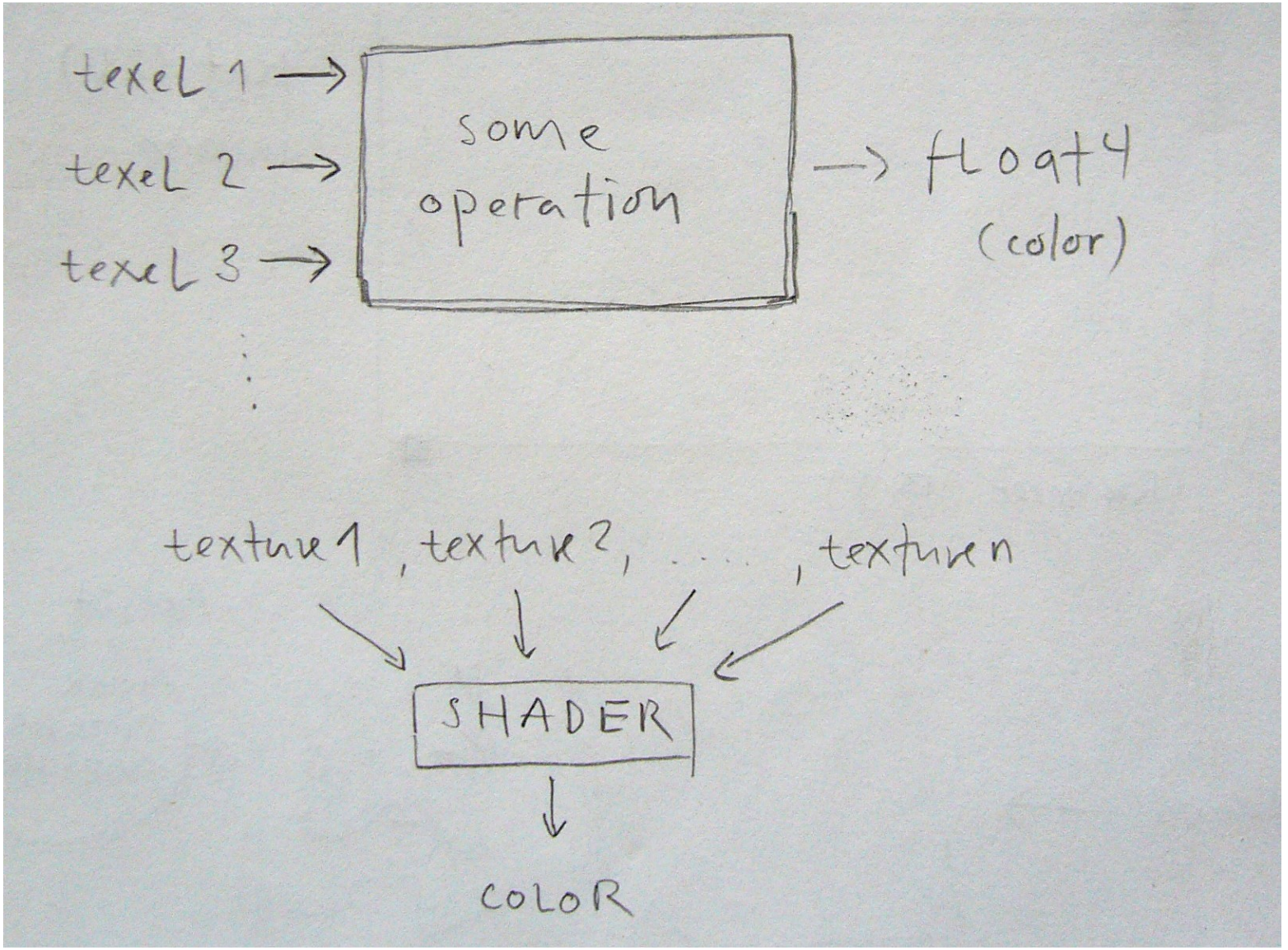
To access neighbouring pixels we need to calculate the texture coordinate offset.

- PixelOffsetX = 1 / TextureWidth
- PixelOffsetY = 1 / TextureHeight

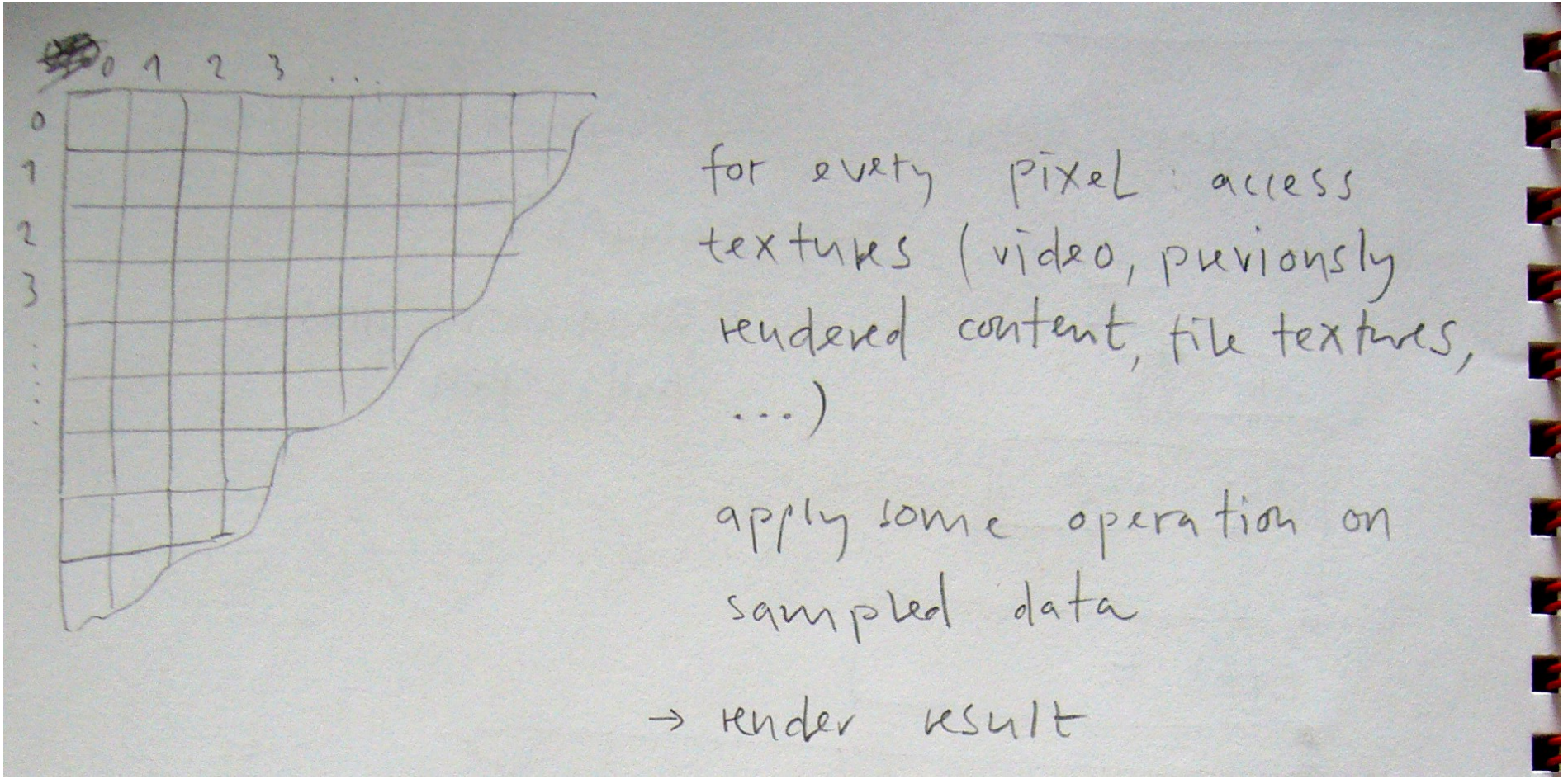
The Framework patch should support the effect with TextureWidth & TextureHeight info.

For that it is convenient to use a „Info (EX9.Texture)“ node.

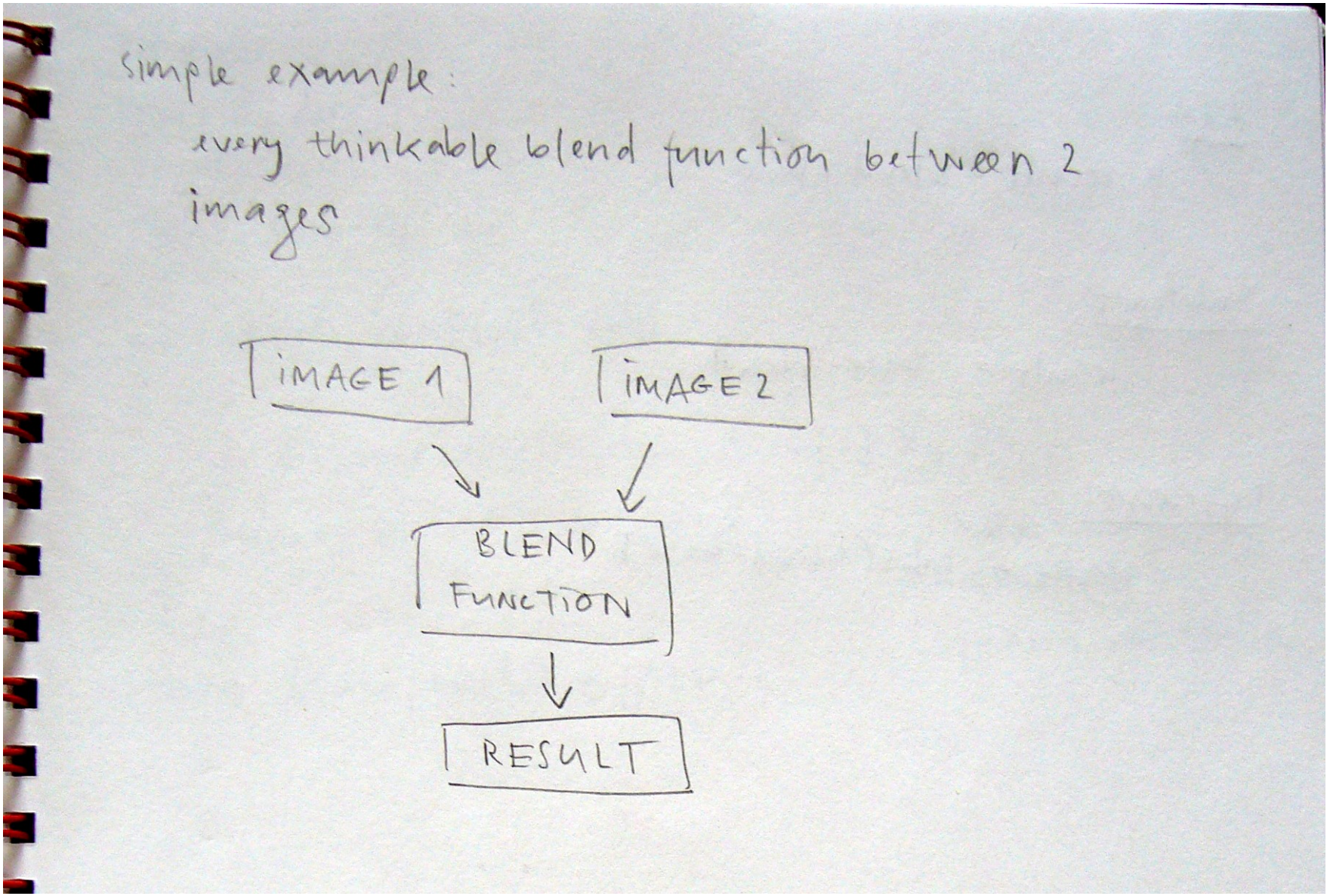
„canvas“ based



„canvas“ based



Blending


simple example:
every thinkable blend function between 2 images
IMAGE 1
IMAGE 2
BLEND FUNCTION
RESULT

Blend

some examples:

base: sampled colour from base texture

blend:sampled colour from texture to blend to

result: resulting colour

Normal:
result = a x base + (1-a) x blend

Average:
result = (base + blend) / 2

Multiply:
result = base x blend

Add:
result = base + blend

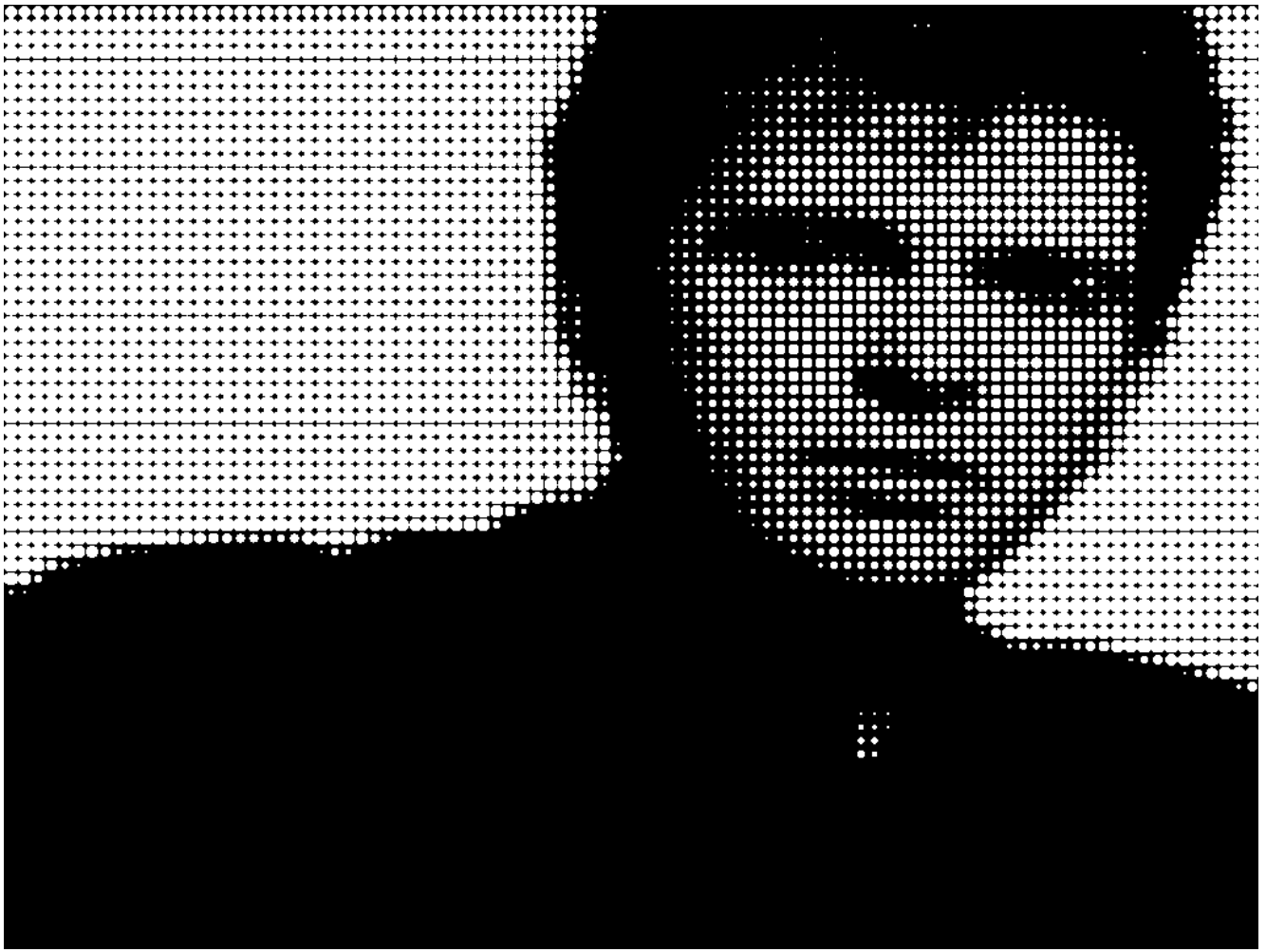
Subtract:
result = base – blend

Difference:
result = abs(blend - base)

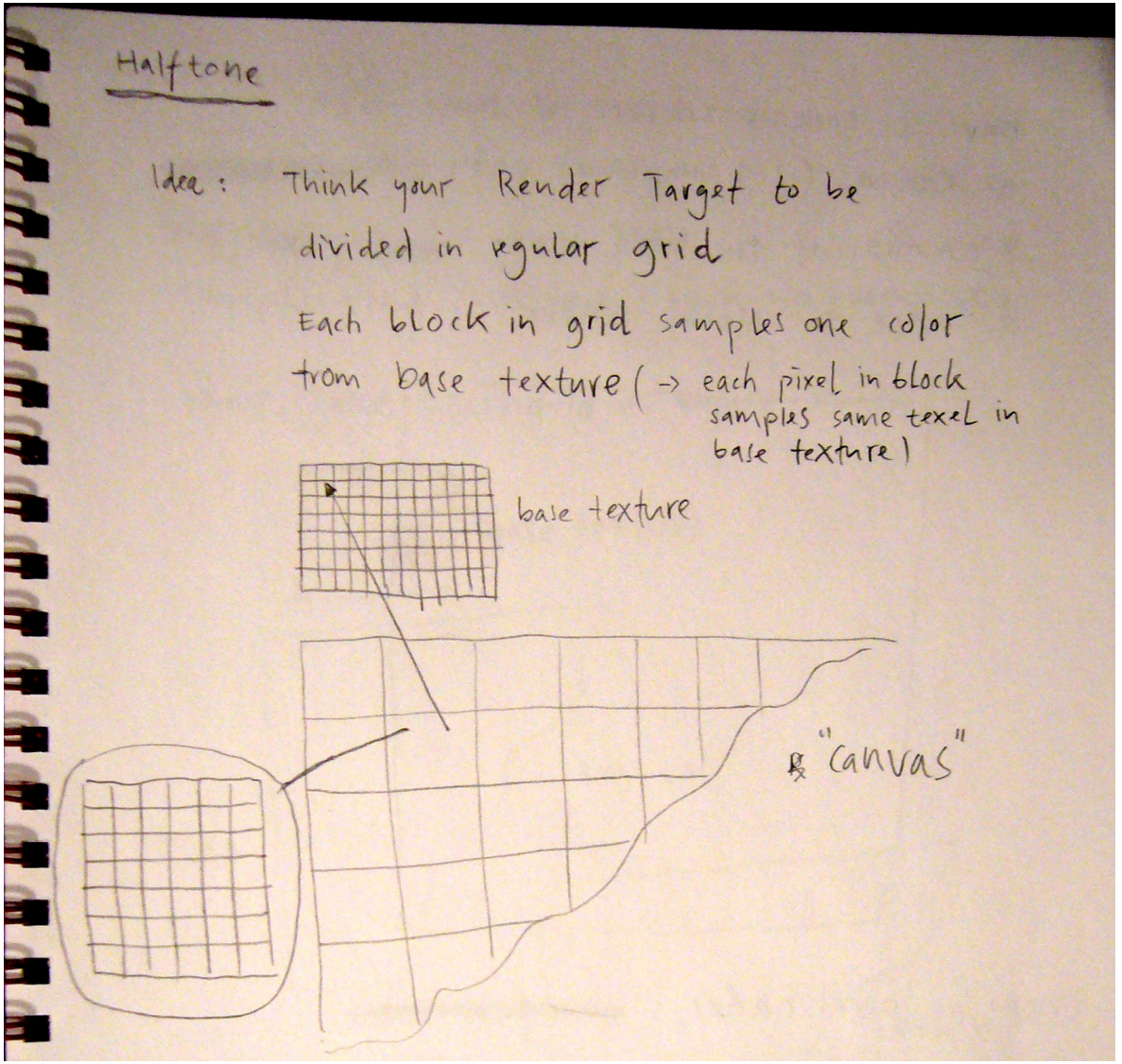
Blend

imagemanipulations/blend.v4p

Halftoning



Halftoning



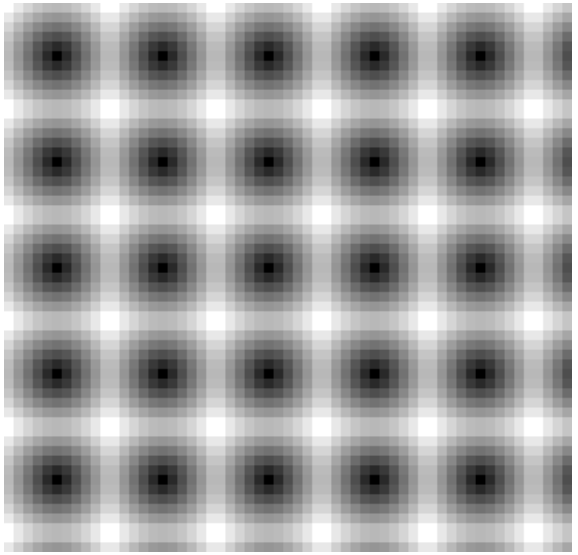
canvas size must then be a multitude of the base texture

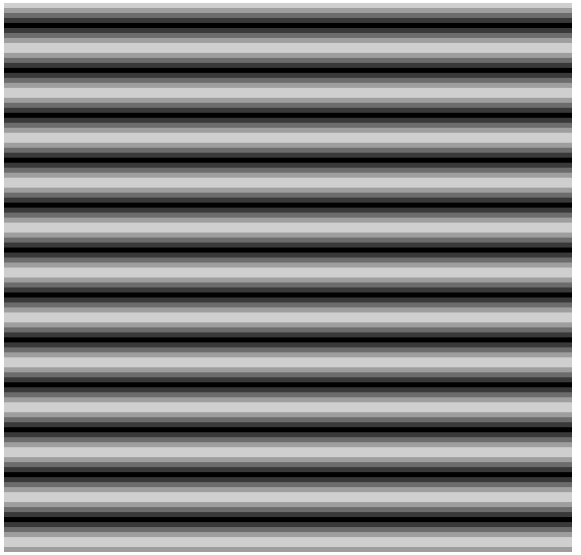
e.g.

base texture	128 x 96
block size	8 x 8
>> canvas	1024 x 768

Halftoning

have a look up texture of same size as canvas (1:1 sampling) with some information of the grid, e.g. each pixel's distance to block's center





now in pixel shader:
implement some compare function, to compare sampled color from base texture to grid structure

>> e.g. color brightness / density

Halftoning

halftoning/01_halftone_dots.v4p

halftoning/02_halftone_dots02.v4p

halftoning/03_halftone_lines.v4p

halftoning/04_halftone_lines_blurred.v4p

halftoning/halftone_dots_blurred.v4p

Ascii Shader

could we do that in a shader?

Ascii Shader

halftone shader with more complicated compare / look up / sampling

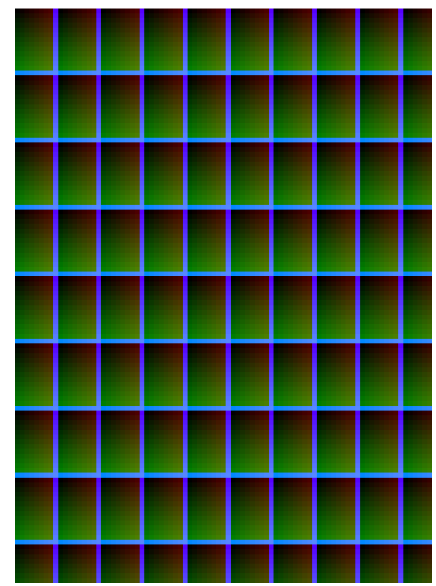
again, same scenario as before

base texture,

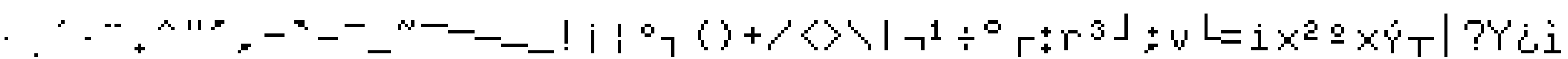
canvas

position look up table (!!)

ascii table



position relative to grid blocks origins



ascii characters sorted due to characters density (number of „black“ pixels) / block size

Ascii Shader

for every pixel:

• sample block's color from base texture

•calc. Ascii – char due to brightness of base texture texel

•get pixel's position within pixel's block

•knowing where Ascii char in Ascii table texture is + offset of pixel's position in it's block
>> sample texel from Ascii table

Ascii Shader

ascii/image2ascii.v4p

Pixelshader Tricks

When thinking of pixel shader programs we can think of a program acting on one pixel only, which is much easier. All pixels behave in the same way...

In a „Canvas“ based shader we use a Fullscreen Quad (a quad scaled to cover the whole screen / scale = 2)

When feeding back the texture output back into the shader we now can access the last pixel at a certain postion by sampling the texture at the current texture coordinate.

To access neighbouring pixels we need to calculate the texture coordinate offset.

- PixelOffsetX = 1 / TextureWidth
- PixelOffsetY = 1 / TextureHeight

The Framework patch should support the effect with TextureWidth & TextureHeight info.

For that it is convenient to use a „Info (EX9.Texture)“ node.

Wave Simulation

Goal: Render a Height Map of the wave

Each pixel represents the height of the wave at one position of the texture

The height at one pixel position is calculated depending upon

- The last height at that position
- The last height of neighbouring pixels
- The last velocity at that position

The maths is not the topic right now. It is now more about the framework. (Please look it up in the vvvviki)

Wave Simulation

Let's name

- the height map of the last frame wave1
- the height map of the frame before the last frame wave2

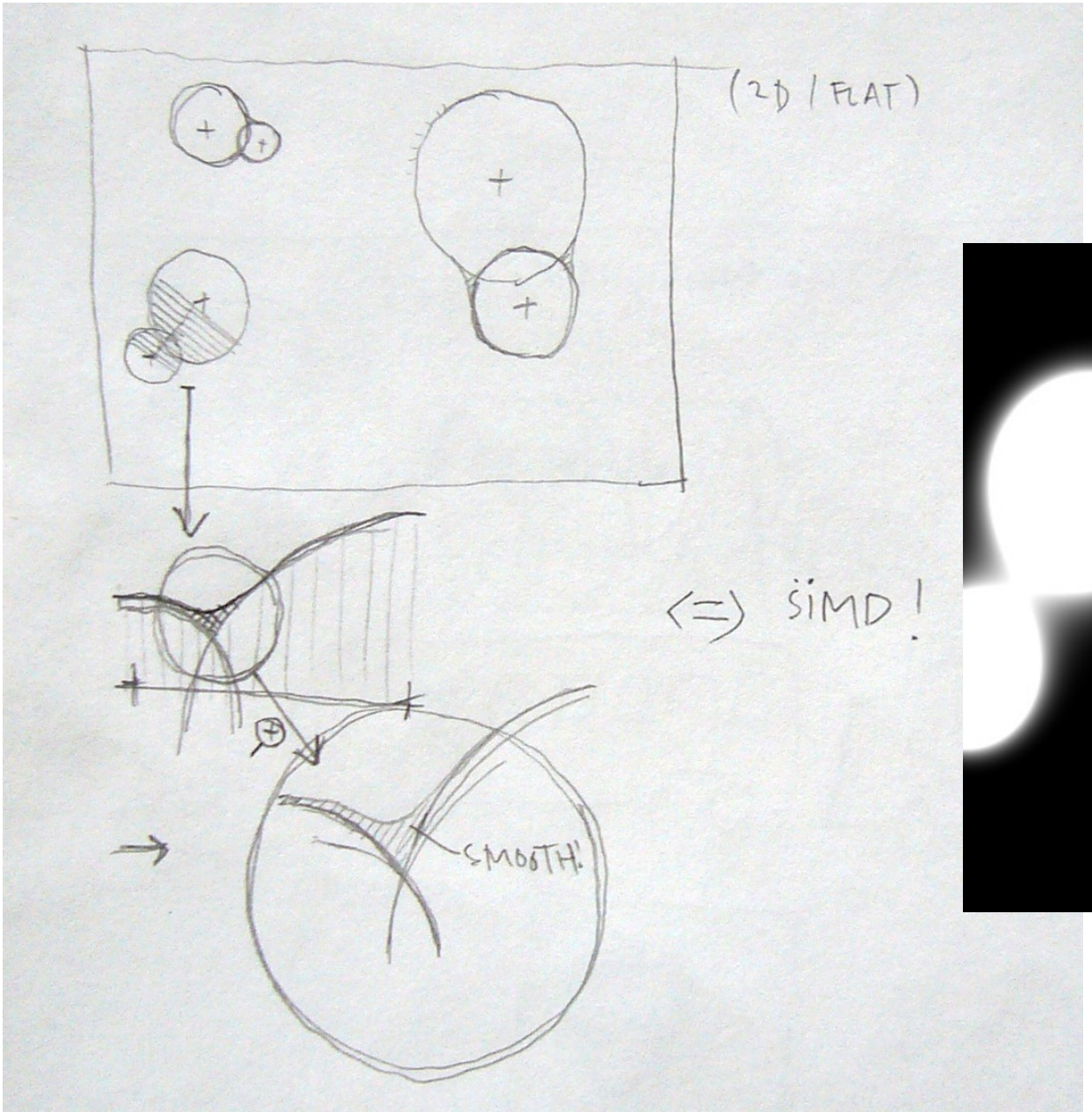
The last height of the wave at a certain position
= sample wave1 at that pixel / texture coordinate.

The last height in neighbouring pixels
= sample wave1 at the left, right, top and bottom pixels.

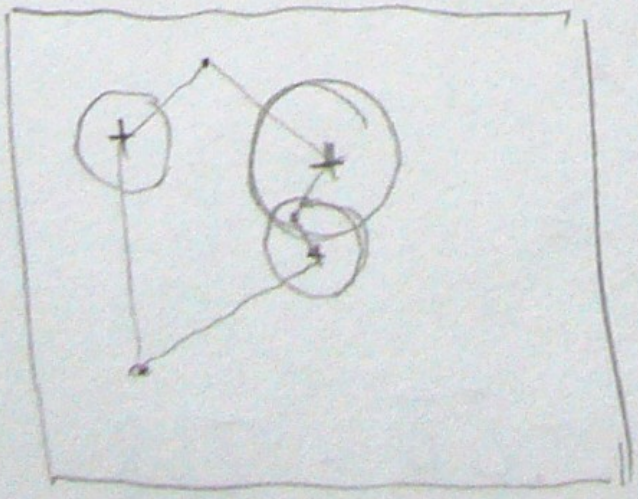
The last velocity at that position
= wave1 – wave2 at that position,
representing the change in height at that pixel.

So we really need to store two Height Maps to get that velocity also.

Metaballs



Metaballs



calc Distance to Blobs for every Pixel
(same a number of instructions used for every pixel, parallel processing heavily exploited)

Metaballs

metaballs/metaballs01.v4p

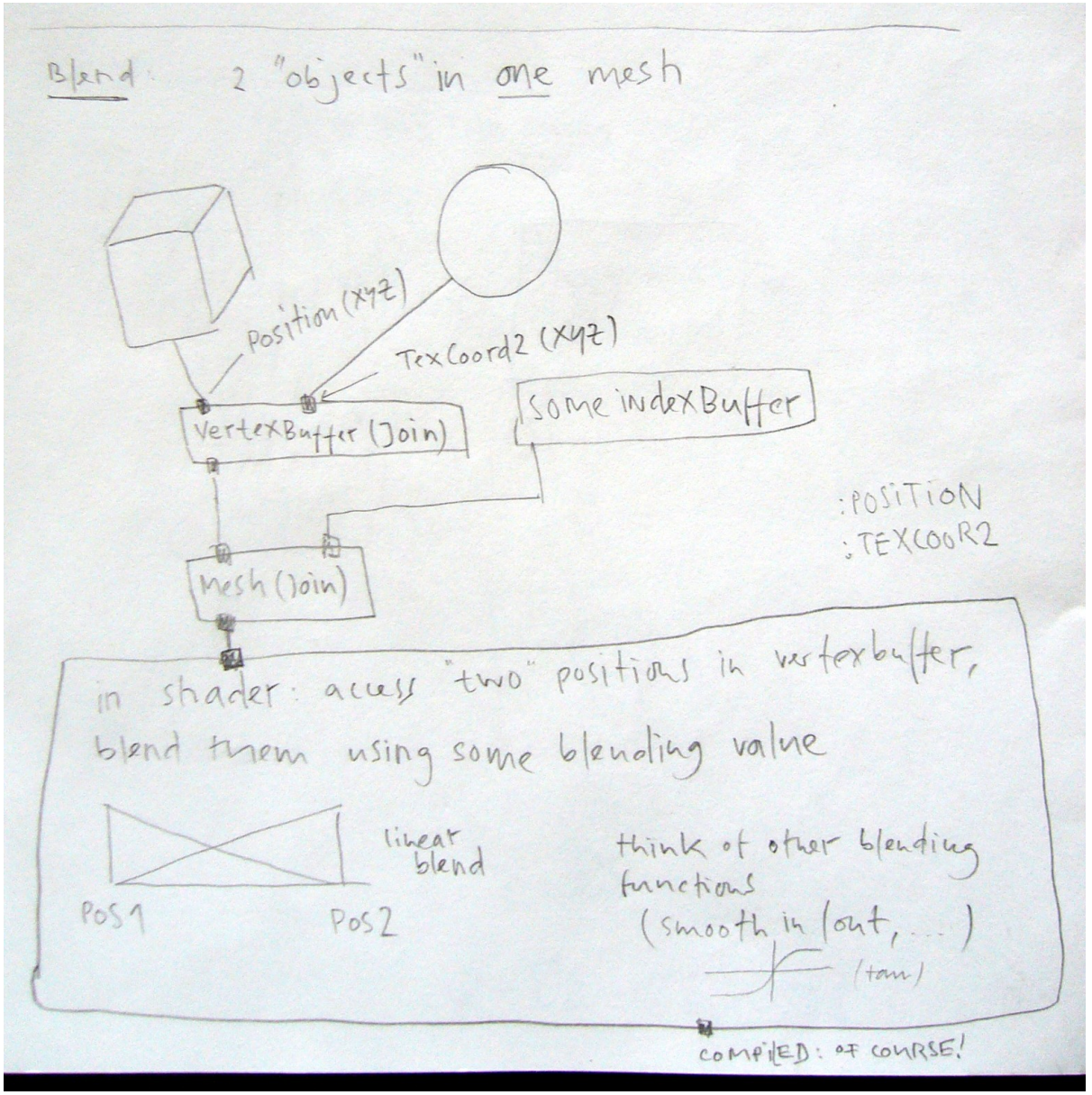
metaballs/metaballs02.v4p

metaballs/metaballs03.v4p

Geometry based Applications

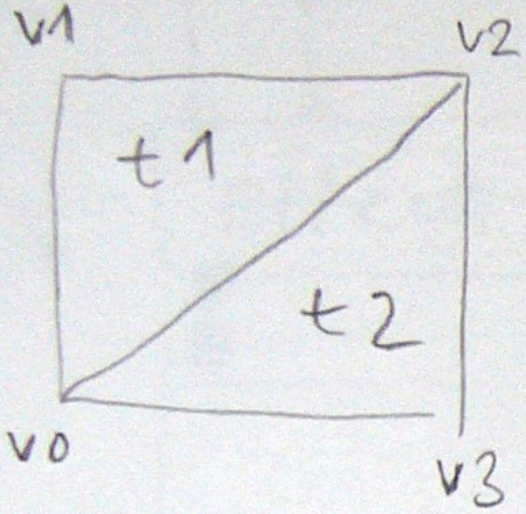
- Distortion
- Morphing

Geometry/Morphing



Geometry/Morphing

Recall: Vertices / Indices...



t1: v0, v1, v2
t2: v0, v2, v3

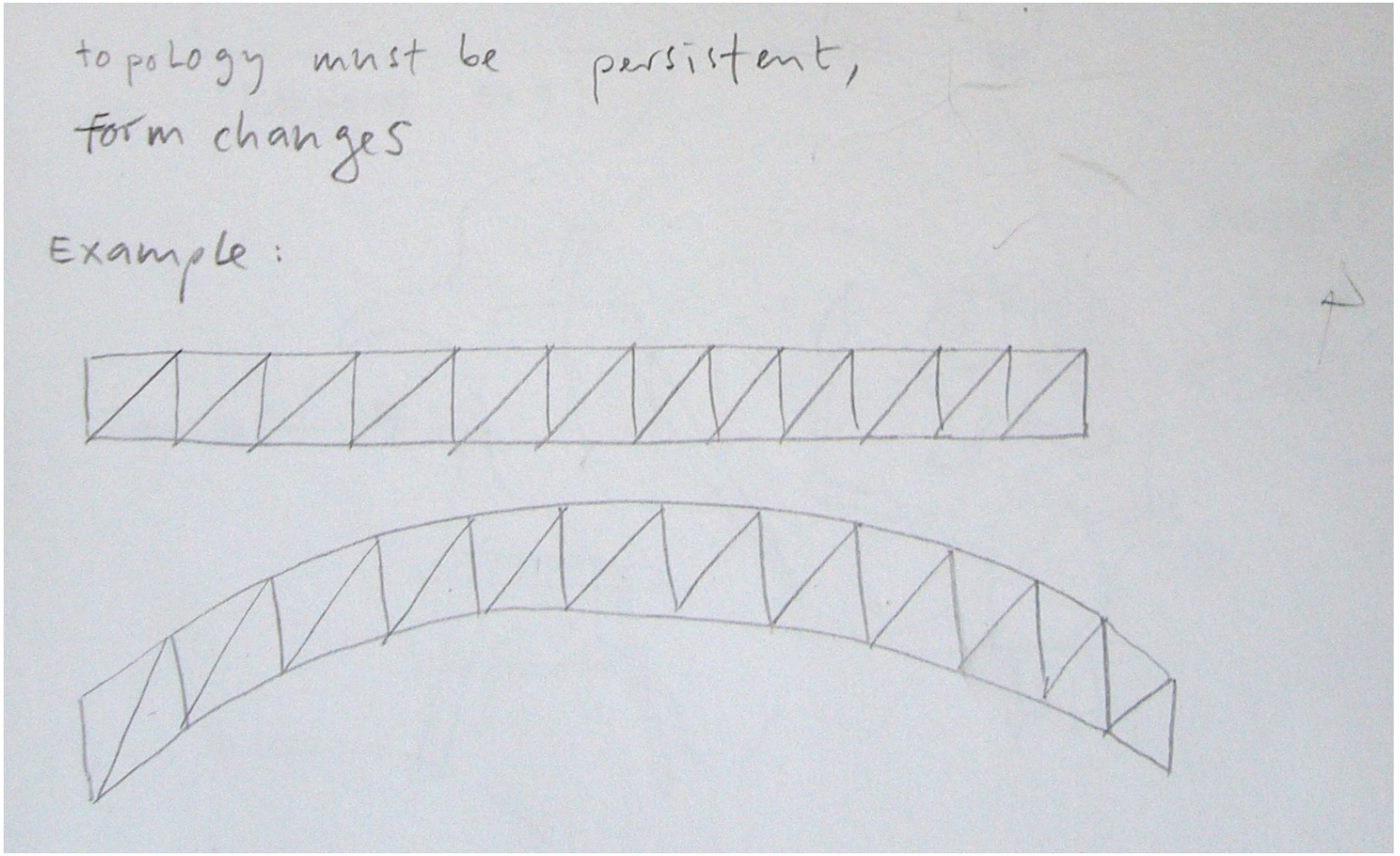
→ Vertexbuffer

v0 (x,y,z)
v1 "
v2 "
v3 "
⋮

IndexBuffer

v0
v1
v2
} triangle 1
v0
v2
v3
} triangle 2
⋮

Geometry/Morph



requies careful creation oft he geometry, two vertexbuffers need to be of the same size, must both work with the same indexbuffer

Geometry/Morph

morp/01_blendGeometry.v4p

morp/02_morphCubeSphere.v4p

morp/03_morphPoints.v4p

morp/04_morphWave.v4p

morp/05_morphGridSphere.v4p

Bonus Bloom

bloom/BloomDemo.v4p

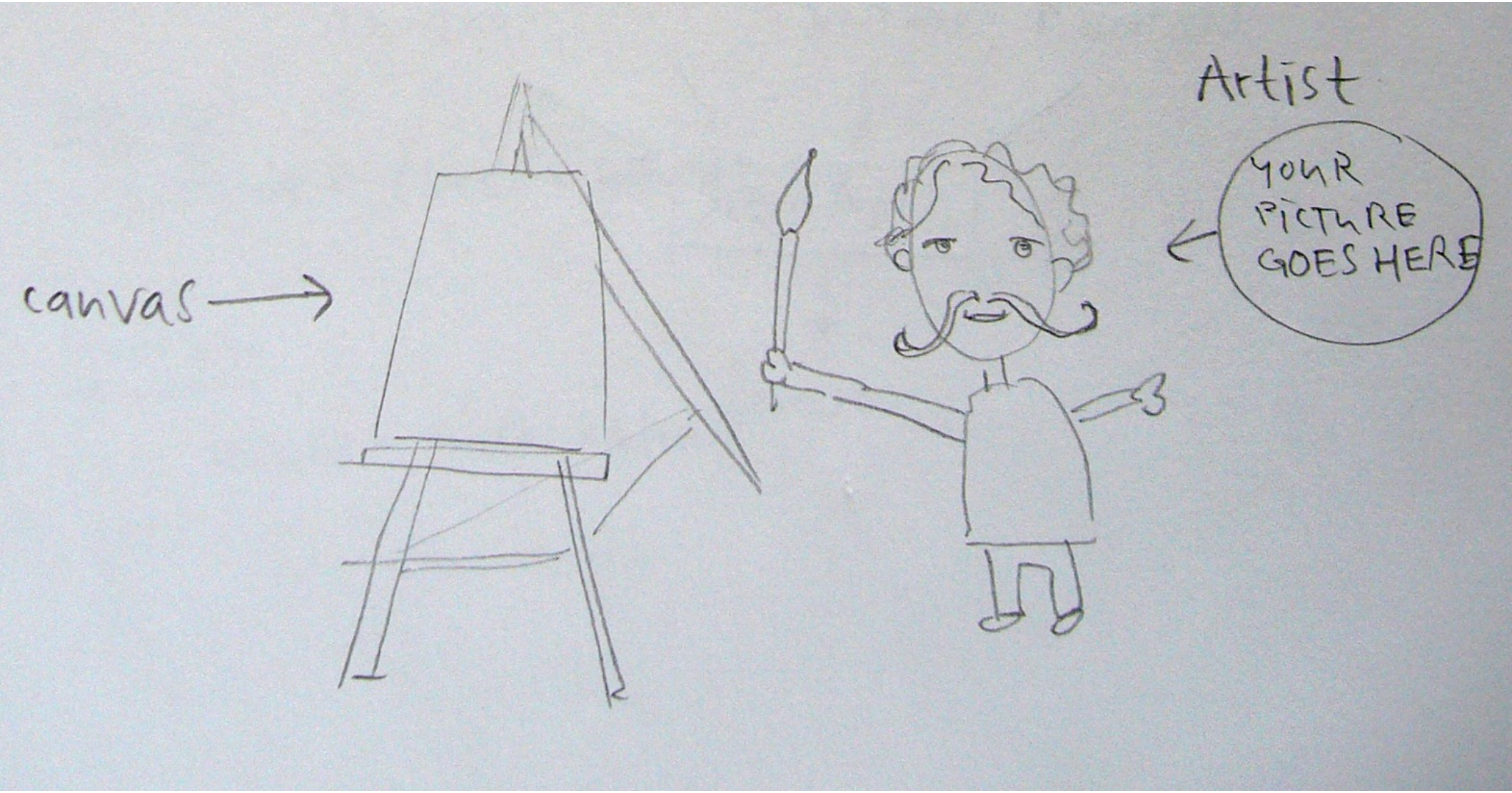
Daily Tools

- VVVViki

  http://www.vvvv.org/tiki-index.php?page=Effects+Framework#Effect_Files
  http://www.vvvv.org/tiki-index.php?page=node08.workshop.ShaderProgramming
  http://www.vvvv.org/tiki-index.php?page=node08.workshop.ShaderApplications

- Pencil & Paper
- A little bit of Vector Math / Linear Algebra
- Book: Real Time Rendering, Tomas Akenine-Möller, Eric Haines
- WWW: Paul Bourke

Remember the Canvas...



& the artist is you!