Game Technology

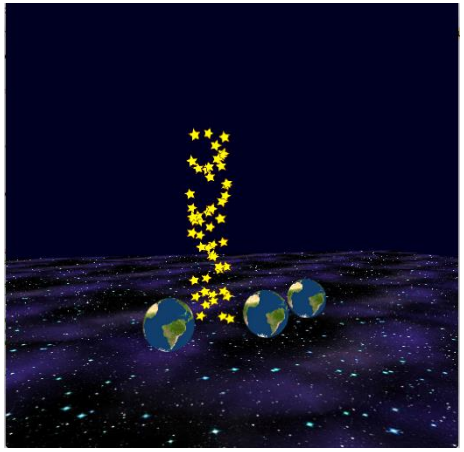
Lecture 10 – 09.01.2018
Large Game Worlds

TECHNISCHE UNIVERSITÄT DARMSTADT

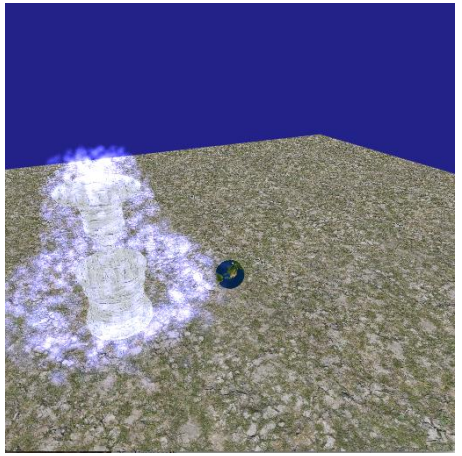
Dipl-Inf. Robert Konrad
Polona Caserman, M.Sc.

Prof. Dr.-Ing. Ralf Steinmetz
KOM - Multimedia Communications Lab

Best of …

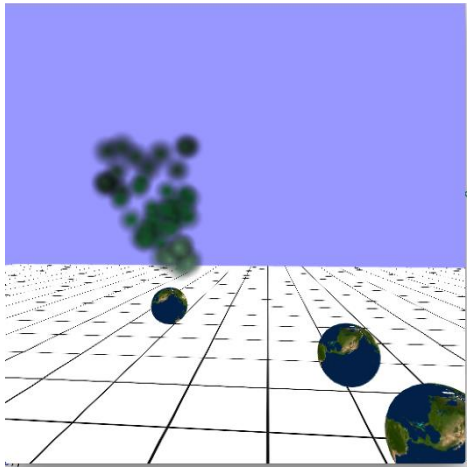


FliegendesSpaghettimonster

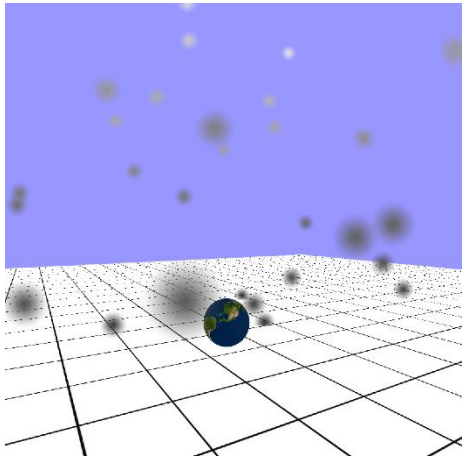


K^3

pathToExam



teamnameunknown




STEALTH-COM

Today‘s Games

TECHNISCHE UNIVERSITÄT DARMSTADT

Typical hardware requirements

- 8 GiB RAM
- 2 GiB Video-RAM
- 50 GiB on disk

All SNES games ever (including all language versions)

- ~3000 games
- ~4.5 GiB

Today‘s Data

TECHNISCHE UNIVERSITÄT DARMSTADT

One uncompressed texture

- 4096 x 4096 x 4 Bytes = 67108864 Bytes = 64 MiB
- 2 GiB / 64 MiB = 32
- Physically based rendering – typically 4 textures

Killzone 4 CPU data

TECHNISCHE UNIVERSITÄT DARMSTADT

Sound	553 MB
Havok Scratch	350 MB
Game Heap	318 MB
Various Assets, Entities, etc.	143 MB
Animation	75 MB
Executable + Stack	74 MB
LUA Script	6 MB
Particle Buffer	6 MB
AI Data	6 MB
Physics Meshes	5 MB
Total	**1,536 MB**

Killzone 4 GPU data

TECHNISCHE UNIVERSITÄT DARMSTADT

Non-Steaming Textures	1,321 MB
Render Targets	800 MB
Streaming Pool (1.6 GB of streaming data)	572 MB
Meshes	315 MB
CUE Heap (49x)	32 MB
ES-GS Buffer	16 MB
GS-VS Buffer	16 MB
Total	**3,072 MB**

PNG and JPEG

TECHNISCHE UNIVERSITÄT DARMSTADT

PNG

- Lossless
- Compression highly dependent on image content

JPEG

- Lossy
- Generally strong compression

Both

- Slow decompression
  - Can slow down loading times
- Not possible to access a single pixel while compressed
  - Not usable for image computations aka not usable as a texture format

Texture Compression

TECHNISCHE UNIVERSITÄT DARMSTADT

Many different formats

- S3TC, PVRTC, ASTC,…
- Has to be supported by GPU and Graphics API
- Of course much of it is patented and hard to standardize

Design goals

- High compression
- Low visual degradation
- Efficient single pixel access
  - → What we need during fragment shader execution
  - Constant size of a pixel or a pixel block

Strategies

TECHNISCHE UNIVERSITÄT DARMSTADT

2K Texture

- Uncompressed: 4 byte * 2048 * 2048 = 16.77 MB
- DXT1: 2048 * 2048 / 16 * 8 bytes = 2.1 MB
- PNG: ~6 MB (Depends on content and compression details)
- JPEG: ~1 MB (Depends on content and compression details)

Save compressed for GPU (e.g. DXT)

- Quick file load direct into memory
- Fast, simple
- Requires much space

Save in complex format, convert to compressed while loading

- Smaller file sizes (e.g. mobile)
- Longer loading times

Advanced Strategies

TECHNISCHE UNIVERSITÄT DARMSTADT

Save GPU format in lightly compressed format

- lz4, snappy,…
- Speeds up loading times (saved load time > decompression time)

Very advanced strategies

- Save in a compression format optimized for compressing gpu formats
  - crunch
- Uncompress on GPU
  - „GPU-decodable supercompressed textures“

Possible compression strategies

TECHNISCHE UNIVERSITÄT DARMSTADT

Less than 8 bits per color might be ok

The eye‘s color resolution is less then its intensity resolution

Neighboring pixels likely have similar colors

S3TC

TECHNISCHE UNIVERSITÄT DARMSTADT

Originally developed by S3 Graphics for the Savage 3D graphics accelerator

Also known as DXTn/DXTC (DirectX names)

De-facto standard for OpenGL implementations

Series of 5 algorithms that handle compression differently

- Mainly depending on the way alpha is treated

Block-based compression algorithm

- Input always 4x4 pixel block
- Output 64 or 128 bits

DXT1

TECHNISCHE UNIVERSITÄT DARMSTADT

Input: 4x4 block

Output

- c0: Color encoded with r=5,g=6,b=5 bits (16 bits)
- c1: Color encoded with r=5,g=6,b=5 bits (16 bits)
- 4x4 lookup table with 2 bits per pixel (32 bits)

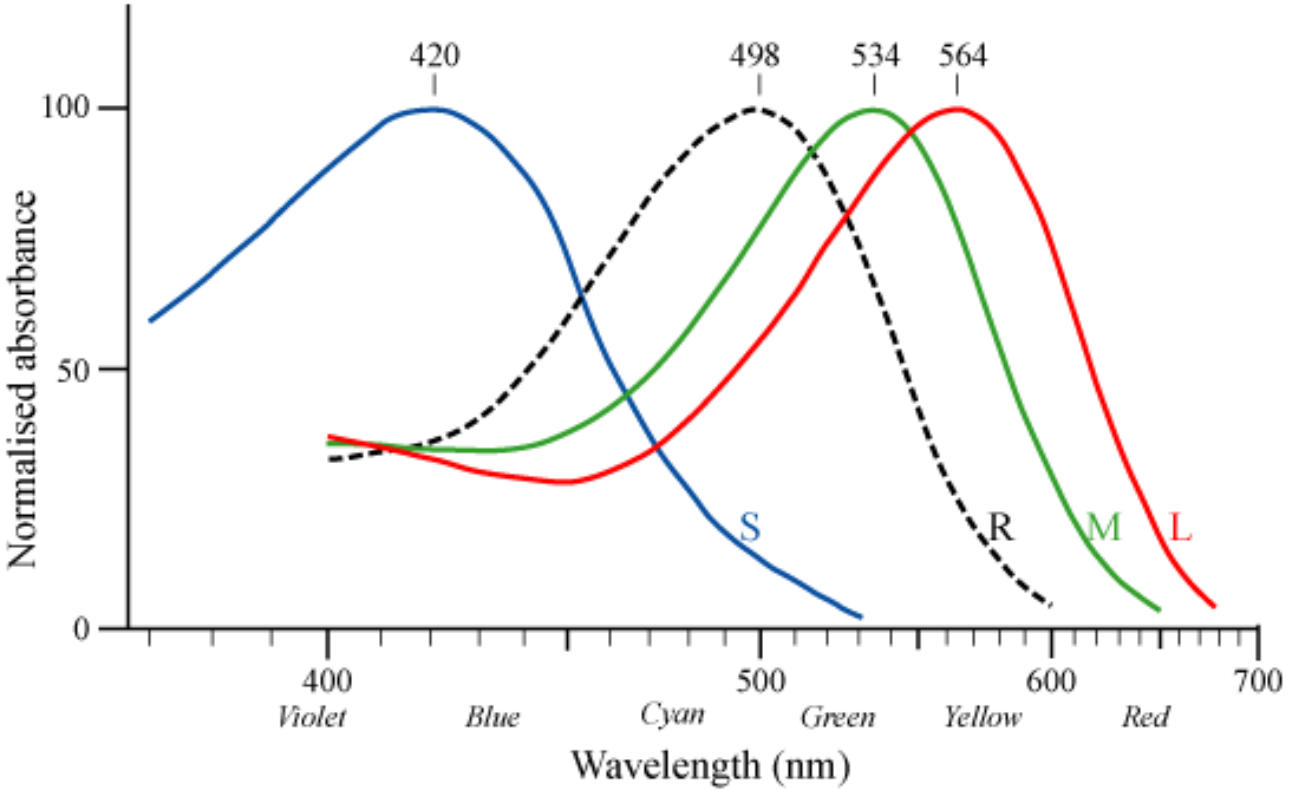
Intermediate values

- if (c0 > c1)
  - c2 = 2/3 c0 + 1/3 c1
  - c3 = 1/3 c0 + 2/3 c2
- if (c0 <= c1)
  - c2 = 1/2 c0 + 1/2 c1
  - c3 = transparent black

Why green?

Dotted line: Absorption of cones, Colored lines: Absorption of rods

Overlap in green area → human eyes can better differentiate variations of green than other colors (555/15bits would not align)



https://commons.wikimedia.org/wiki/File:Cone-response.png

DXT1 example

TECHNISCHE UNIVERSITÄT DARMSTADT

Input block

4x4, 32 bits per pixel

DXT1 example

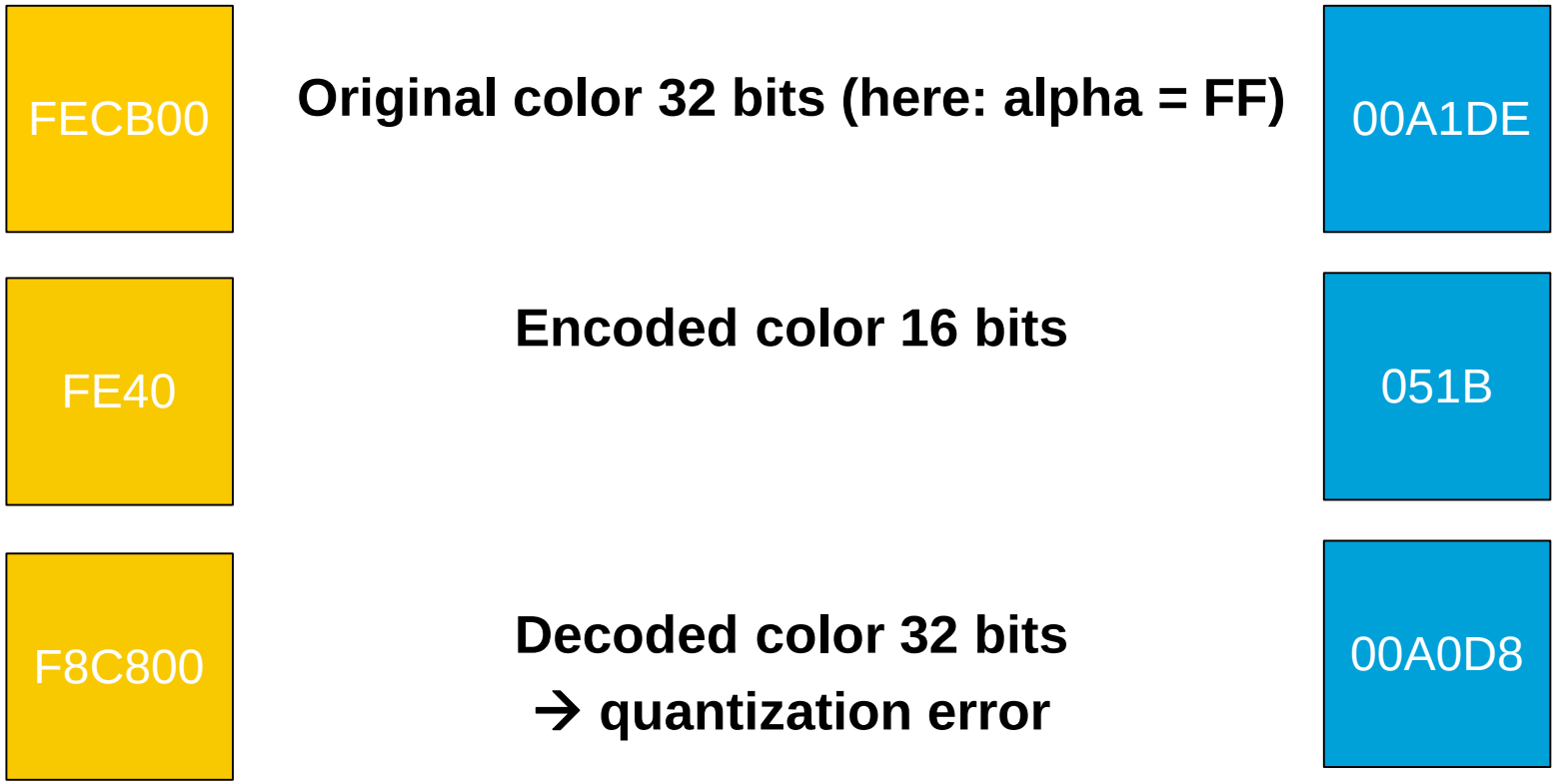
TECHNISCHE UNIVERSITÄT DARMSTADT

Choose two colors

Here: Max distance

Encode using R5G6B5

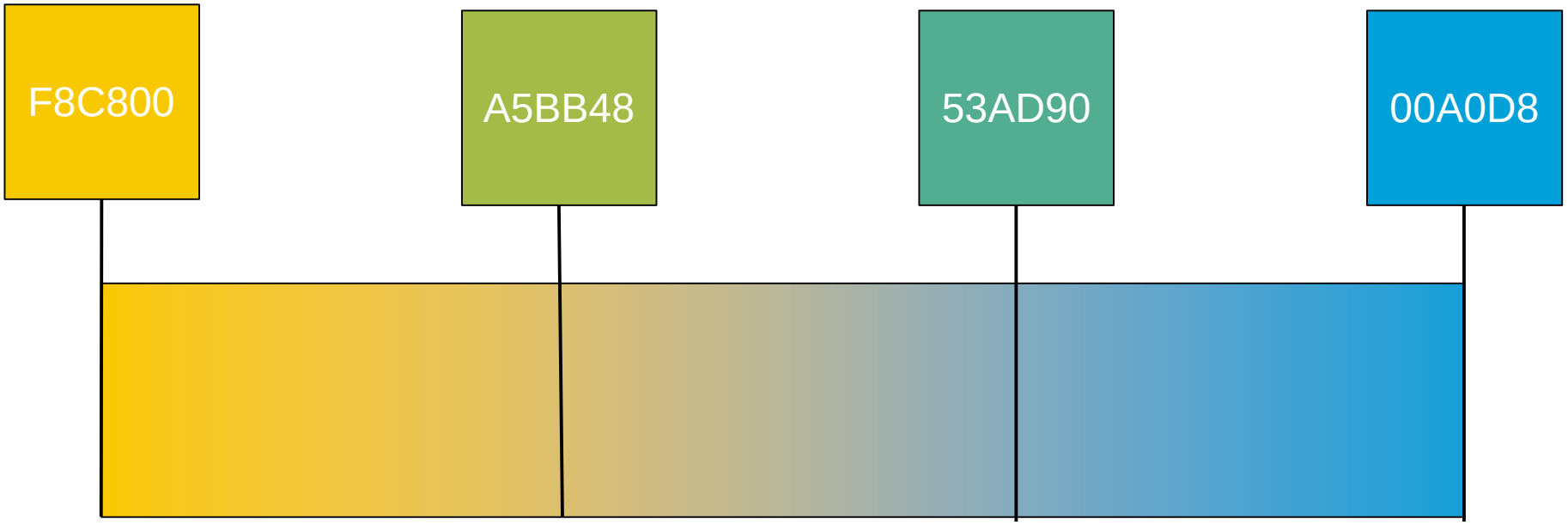
TECHNISCHE UNIVERSITÄT DARMSTADT



Build the palette

TECHNISCHE UNIVERSITÄT DARMSTADT

Choose c2 and c3 to lie at 1/3 and 2/3 between c0 and c1



Choosing endpoints

TECHNISCHE UNIVERSITÄT DARMSTADT

Determines the quality of the result

Can apply several strategies

- Local: Only within our block
- Global: Optimize over the image

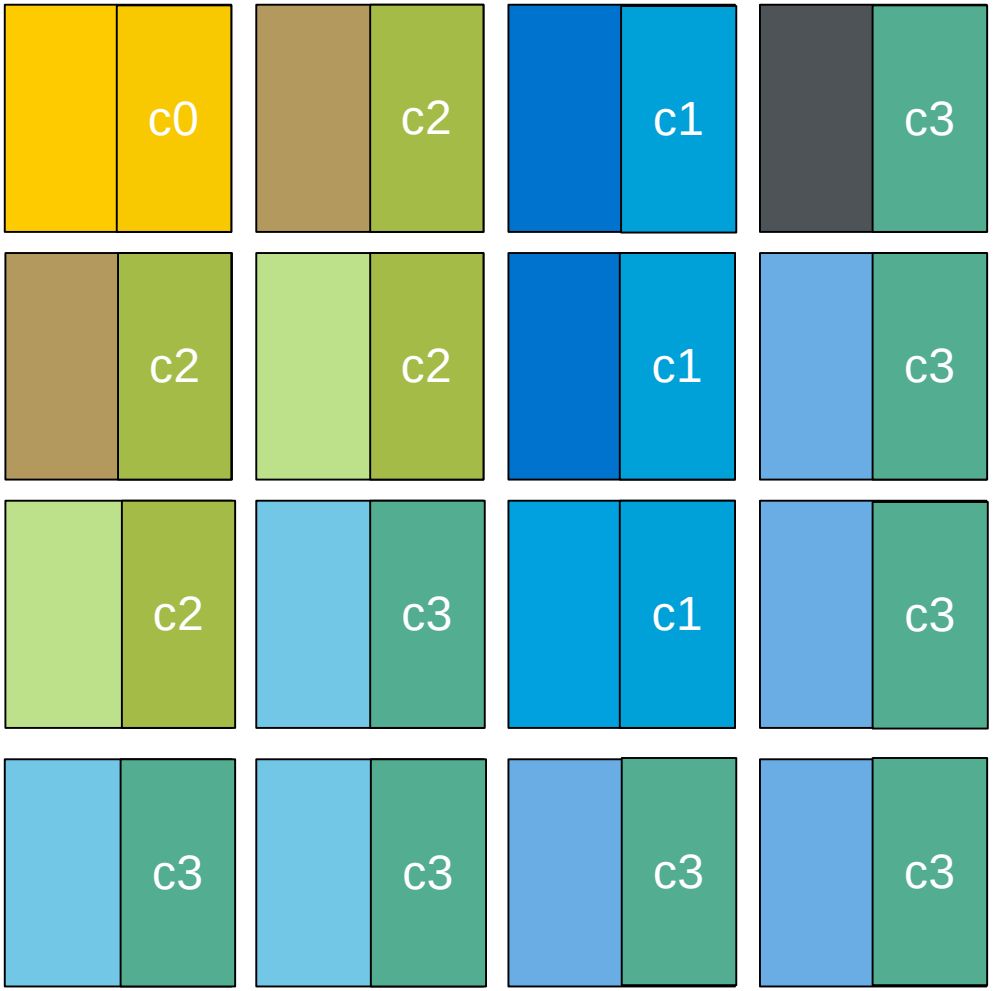
Principal Component Analysis

"Bounding Box", choose minimum and maximum values along 3 axes

Find the closest colors

TECHNISCHE UNIVERSITÄT DARMSTADT

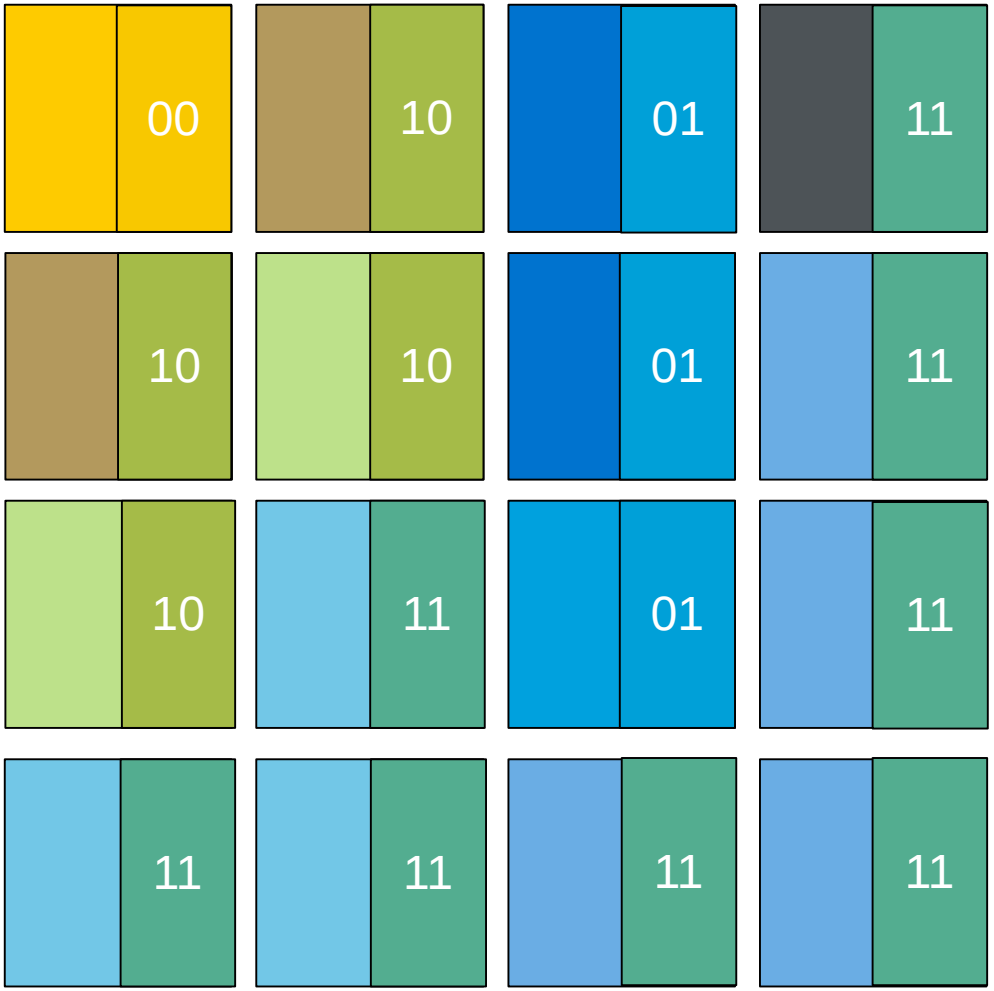
Compare to colors in the block to c0 to c3



Find the closest colors

TECHNISCHE UNIVERSITÄT DARMSTADT

Color index can be encoded in 2 bits



Compressed block

TECHNISCHE UNIVERSITÄT DARMSTADT

c0 = 0xFE40; // 565 – 16 bits

c1 = 0x051B; // 565 – 16 bits

LookupTable =

{00, 10, 01, 11, 10, 10, 01, 11, 10, 11, 01, 11}; // 16x2 bits = 32 bits

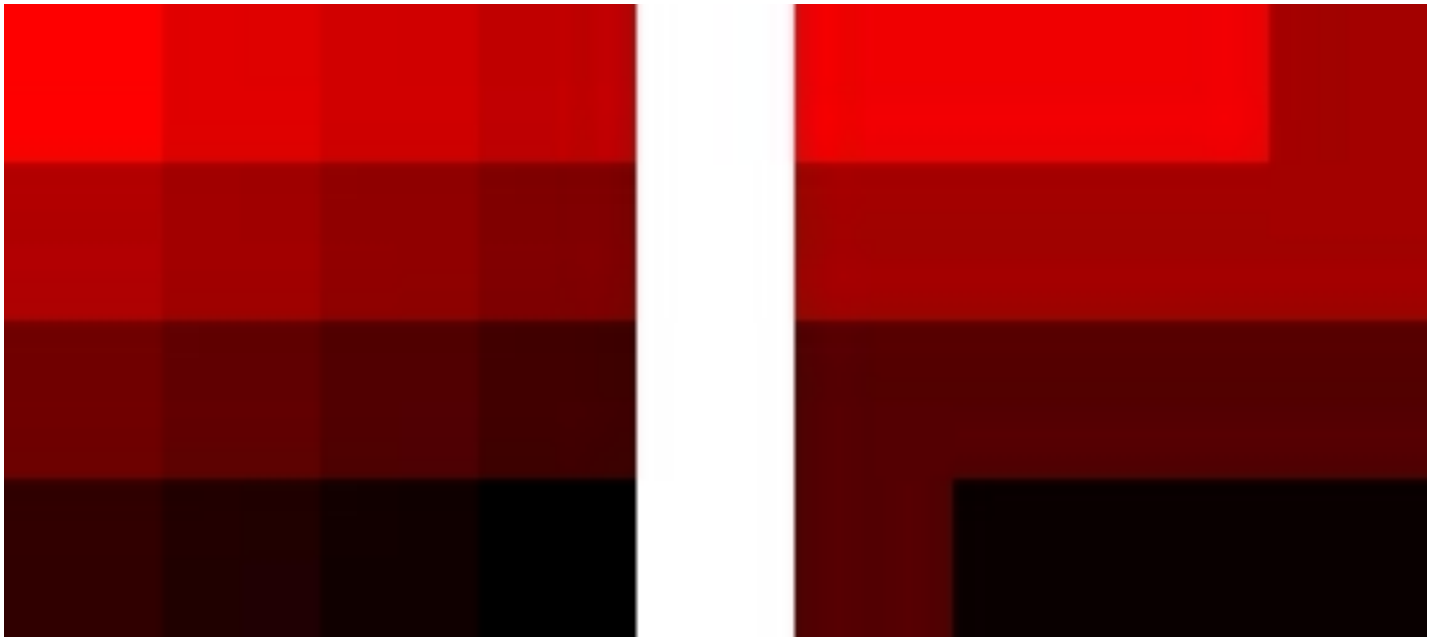
Comparison

DXT1 Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

Well suited for similar gradients

→ We only lose accuracy due to quantization

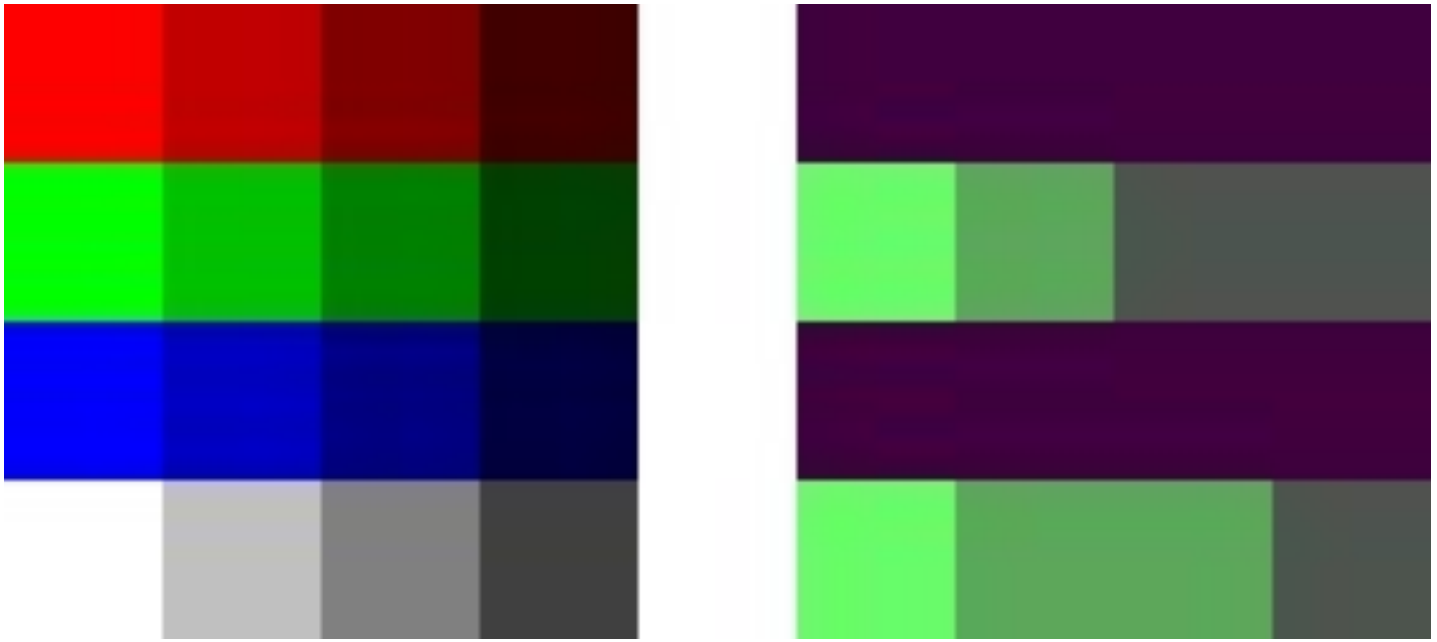


DXT1 Examples

TECHNISCHE UNIVERSITÄT DARMSTADT

Worst case: Colors not on a gradient

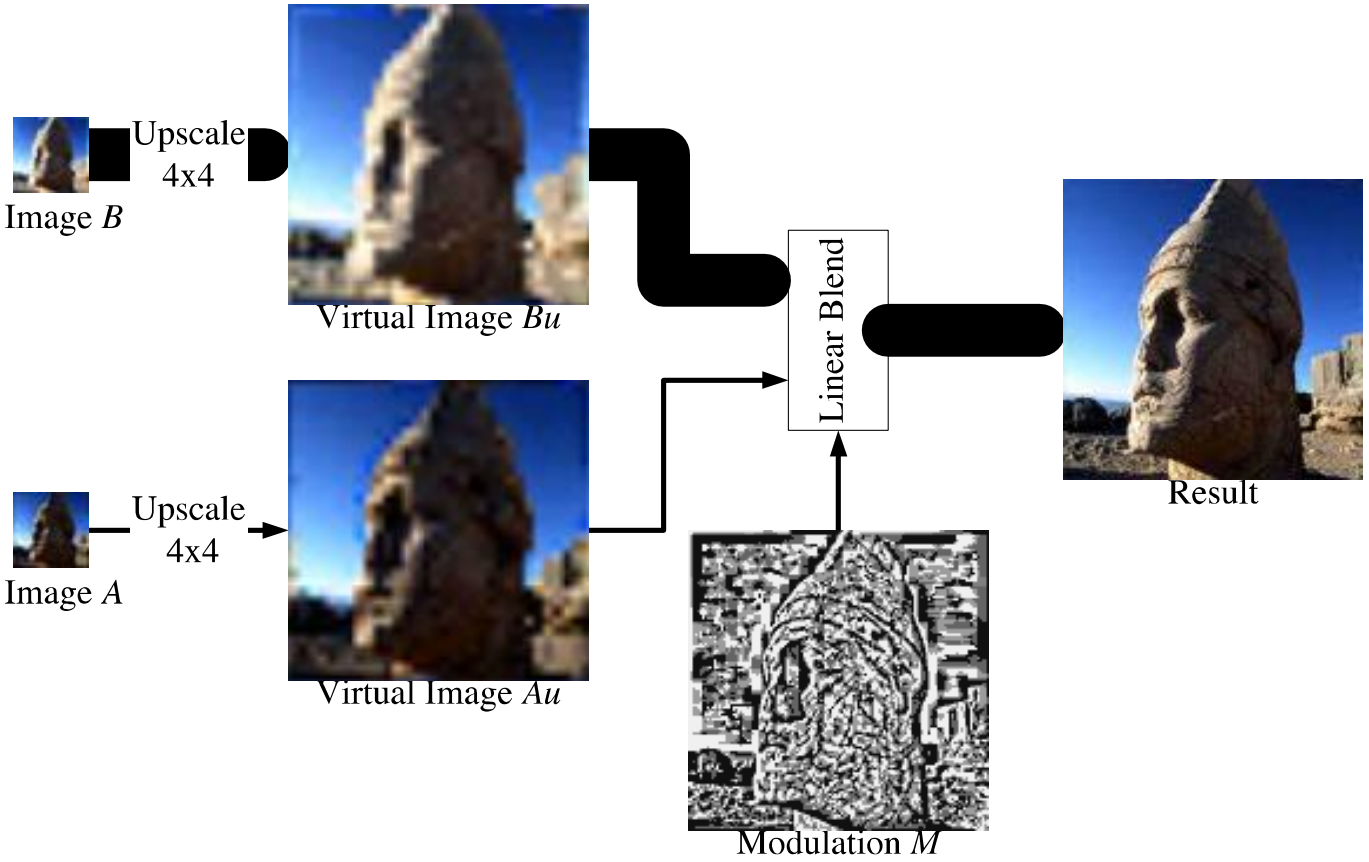
→ We can‘t preserve all colors



TECHNISCHE UNIVERSITÄT DARMSTADT

PVRTC

- PowerVR Texture Compression



Normal Maps, Masks, ...

TECHNISCHE UNIVERSITÄT DARMSTADT

Compression for images might not be optimal for other textures

- But it might just work
- Swizzling channels can help
- No Alpha used for normal maps
- Some algorithms encode alpha better than other values
  - Move one channel to alpha

3Dc

- $x^2+y^2+z^2=1$
  - $z^2=1-x^2-y^2$
  - One value can be omitted
- Can save normals unnormalized, recover later
- Plus block compression

Manual Compression

Let the artists do the job

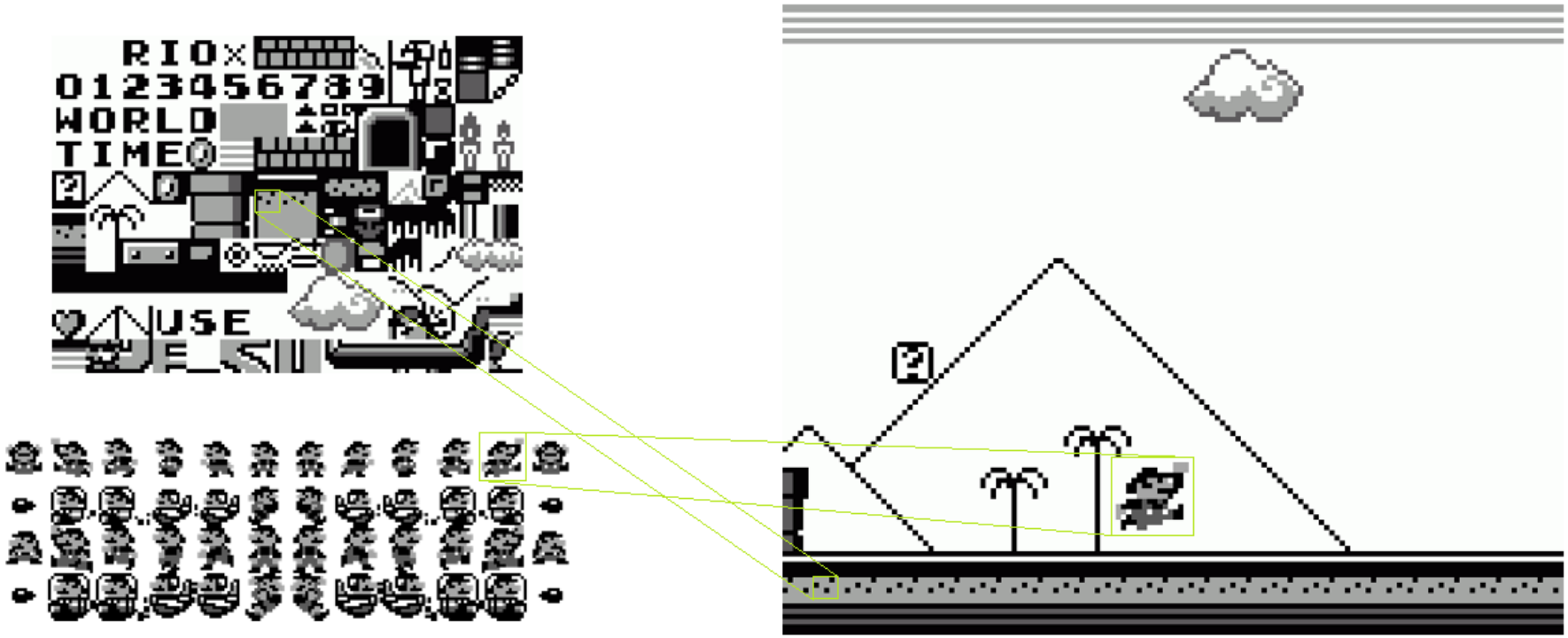
Repeat images over and over

- Nobody might notice it when you do it cleverly

Manual Compression

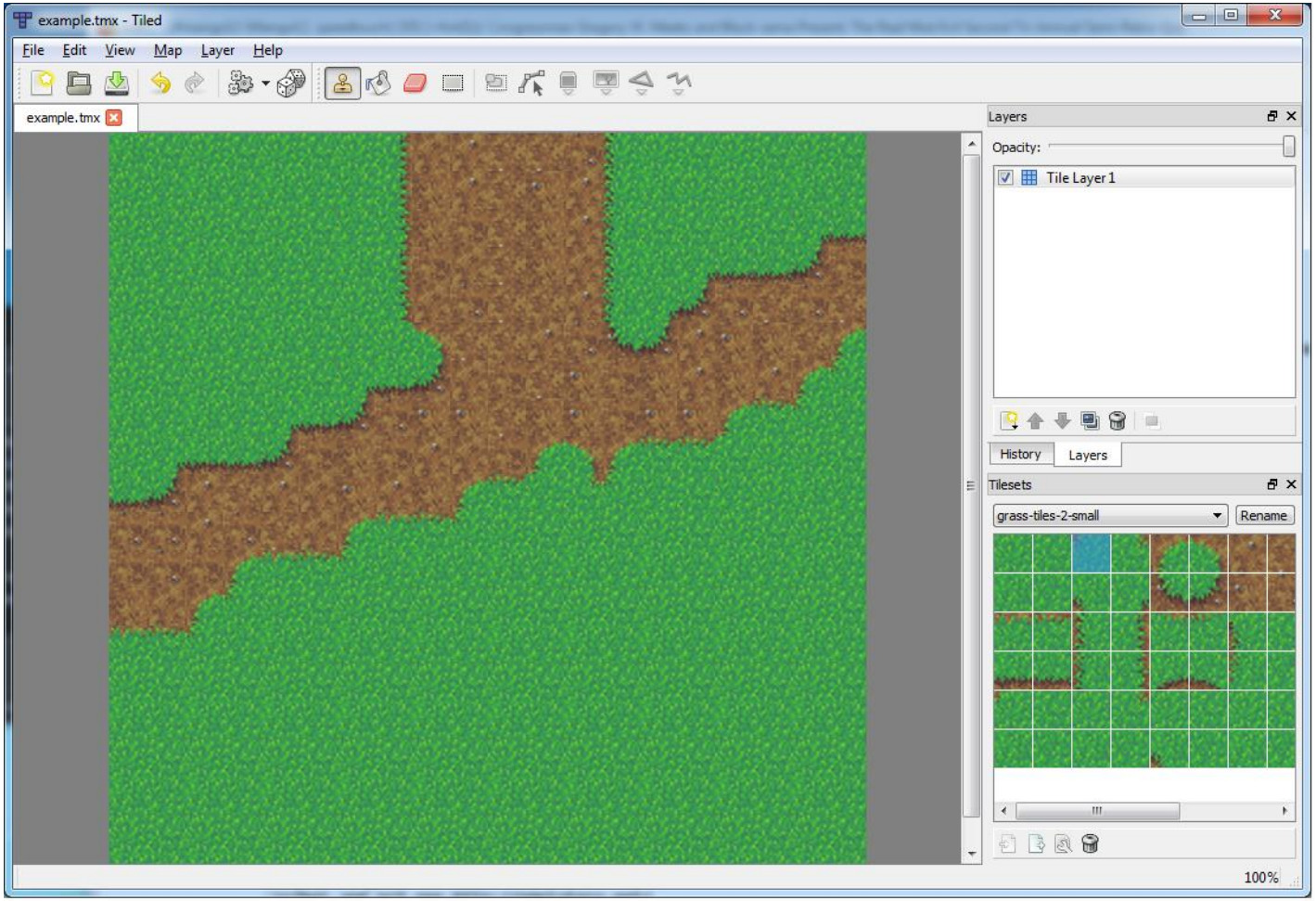
Uncharted 3, 2011

Tilemaps/Tilesets



Tile Editors

TECHNISCHE UNIVERSITÄT DARMSTADT



http://www.mapeditor.org/

Pitfall: The Mayan Adventure (1994)

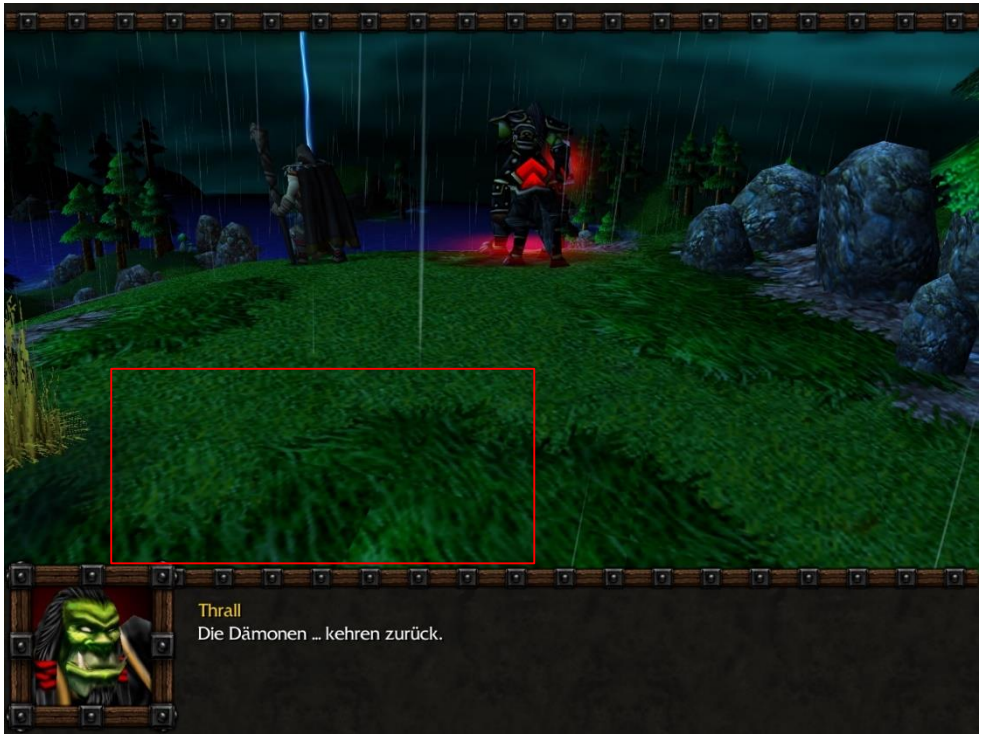
Warcraft 3 (2002)

Tile textures in 3D

TECHNISCHE UNIVERSITÄT DARMSTADT

Bilinear Filtering

- Would have to use texels from two tiles at tile boundaries
- Complicated
- Expensive, Rarely used



Multitexturing

Multitexturing

TECHNISCHE UNIVERSITÄT DARMSTADT

Multitexturing

Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

Performance

- More textures, less performance
- Precalculating which polys actually use more textures can help

Needs good tool support

- Scary communication with artists

Streaming

TECHNISCHE UNIVERSITÄT DARMSTADT

Coarse Streaming

- Load and replace complete assets

Fine Grained Streaming

- Load and show/play a single asset bit by bit

Coarse Streaming

TECHNISCHE UNIVERSITÄT DARMSTADT

Similar to level of detail systems

- Load big textures for near objects
- Kick out big textures for far away objects
- Maybe blend texture changes in and out

http://www.opensg.org/htdocs/doc-1.8/lod.png

Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

Disks are slow and unreliable

- No timing guarantees at all
- Load textures in a second thread, always have an emergency strategy ready (keep super low resolution textures of everything in RAM)

Changing textures at runtime from a helper thread is problematic

- Driver might decide to convert the texture
- Easier on consoles
- Easier with Direct3D 11/12, Vulkan,…

Fine grained texture streaming

Rage (2011)

MegaTextures

TECHNISCHE UNIVERSITÄT DARMSTADT

Really huge textures

- Rage supports textures of up to 128000×128000
  - That‘s ~60 GiB

Compression

- Texture is highly compressed on disk
  - Using lossy JPEG like compression

One texture for everything

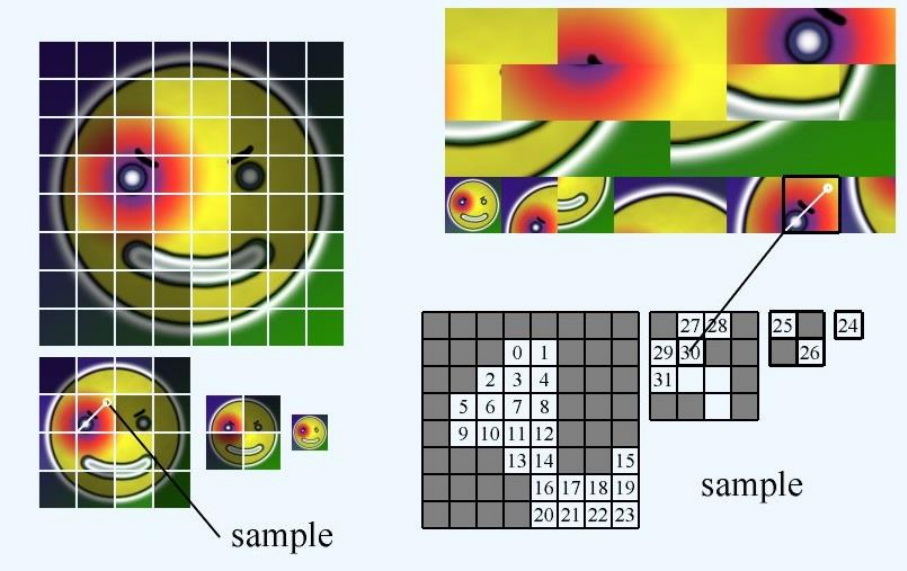
- Complete world in one texture
- No restrictions for artists
  - But toolsets provide classical multitexturing tricks
  - Artists don‘t manually paint 128000x128000 pixels

MegaTextures Implementation

TECHNISCHE UNIVERSITÄT DARMSTADT

Similar to virtual memory

- Application (= pixel shader) believes that there is a huge, continuous area of memory (= texture) it can work on
- Operating system (= texture manager) provides required memory pages by mapping them

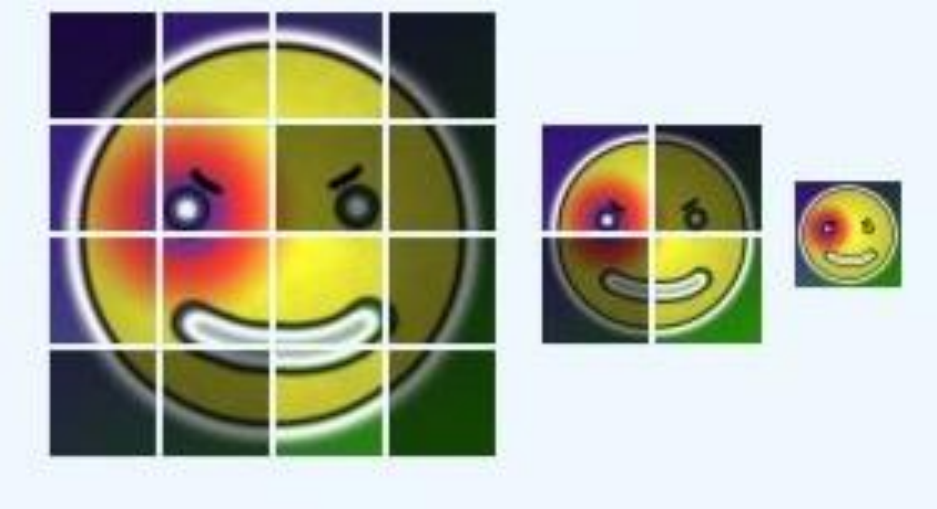


For details, see GDC Talk by Sean Barret: https://www.youtube.com/watch?v=MejJL87yNgI

Level of Detail

TECHNISCHE UNIVERSITÄT DARMSTADT

- Similar to mip maps
- We provide different resolutions of the MegaTexture
- Smaller resolution version should encompass everything we need to sample



MegaTextures

TECHNISCHE UNIVERSITÄT DARMSTADT

Geometry is split up in tiles

- Engine determines screen size of visible tiles
- Loads texture parts in varying sizes to optimize current view

MegaTextures

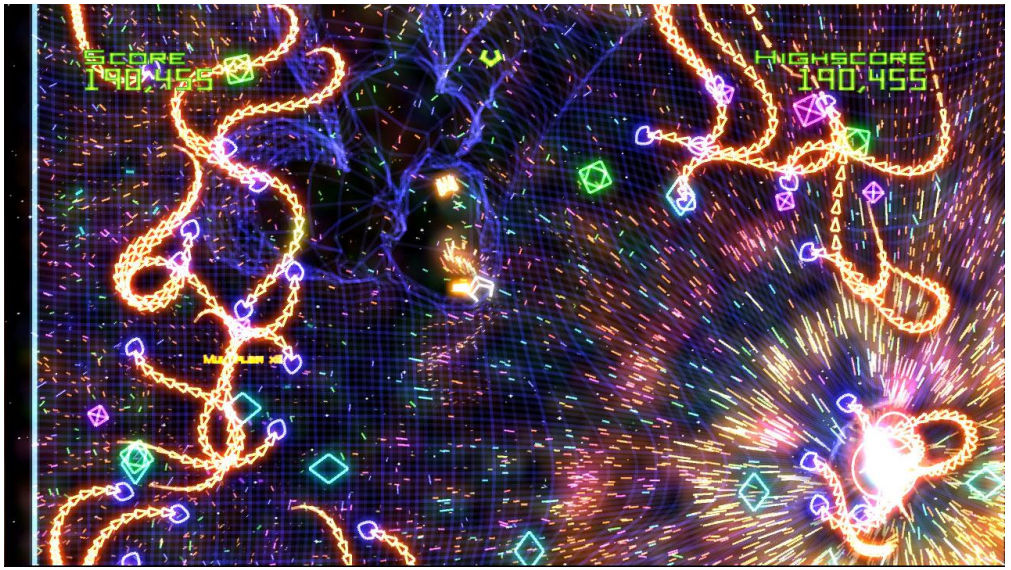
https://www.youtube.com/watch?v=YNilHiBpVic

Geometry

TECHNISCHE UNIVERSITÄT DARMSTADT

Geometry compression

- Not widely used
- No hardware support
- Animation data mostly small (thanks to skeletal animations)



Geometry Wars, 2003

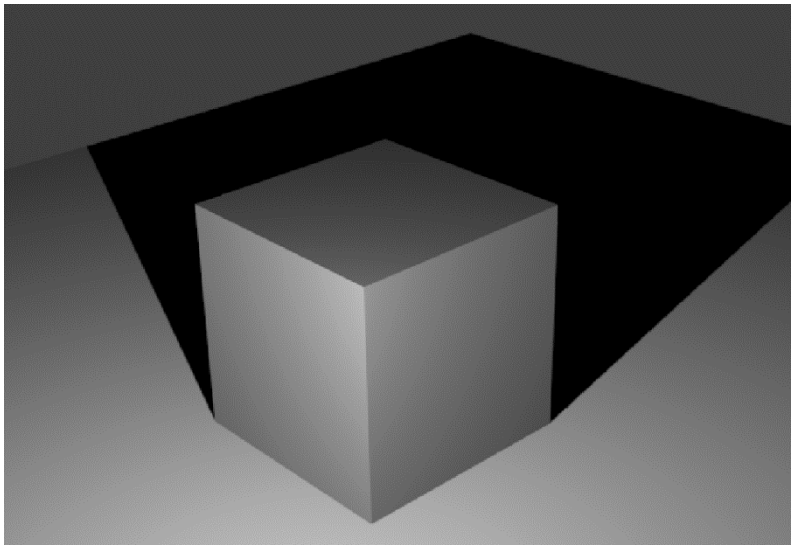
Normal Maps

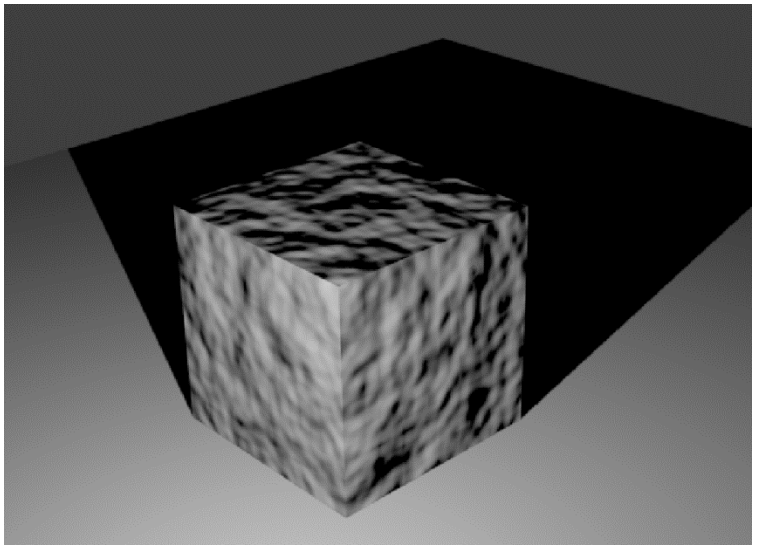
TECHNISCHE UNIVERSITÄT DARMSTADT

Remove super detailed geometry

Replace with normal maps

- Which is a form of compression by itself
- Plus normals can be compressed further



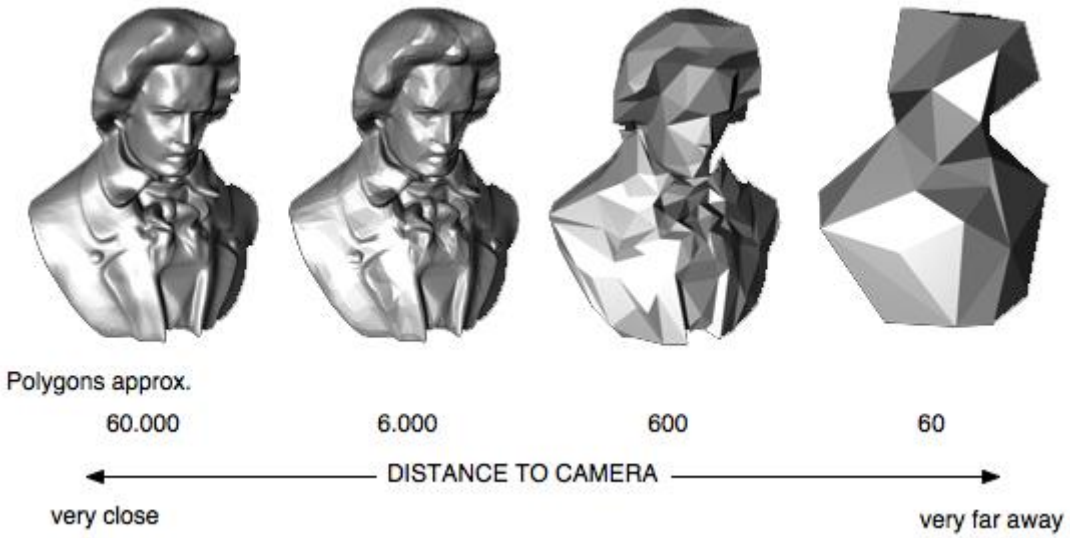


Coarse Geometry Streaming

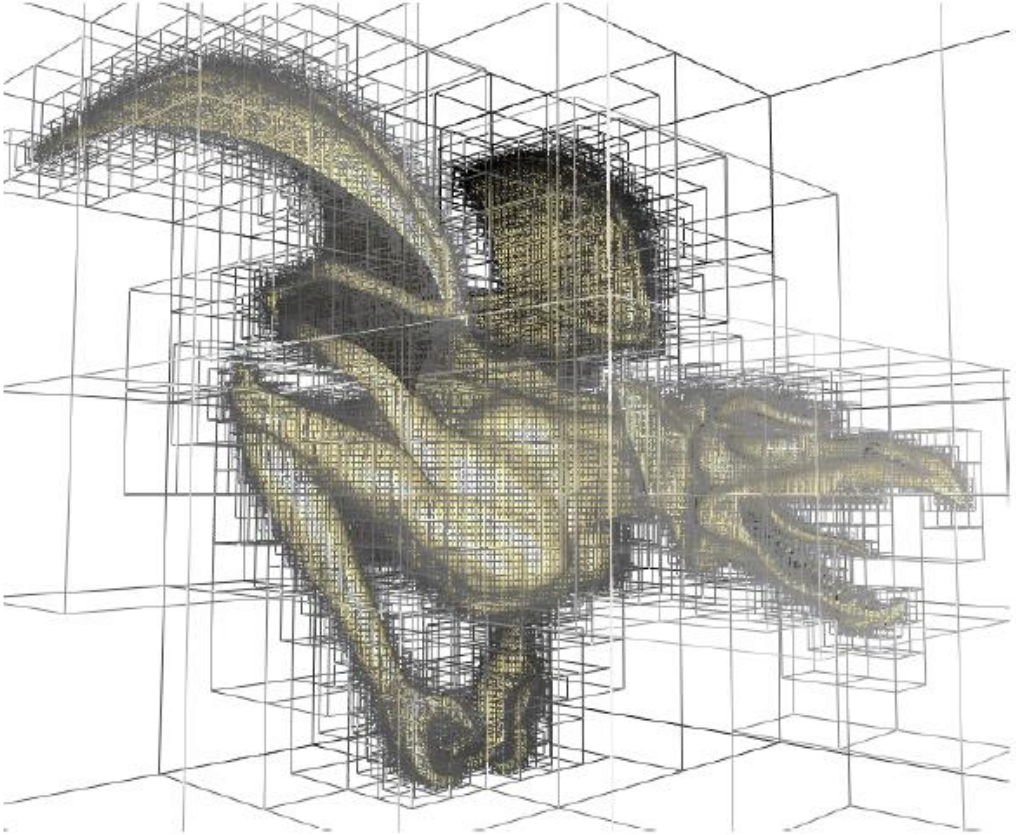
TECHNISCHE UNIVERSITÄT DARMSTADT

Same strategies as for textures

- Could be directly plugged into a level of detail system



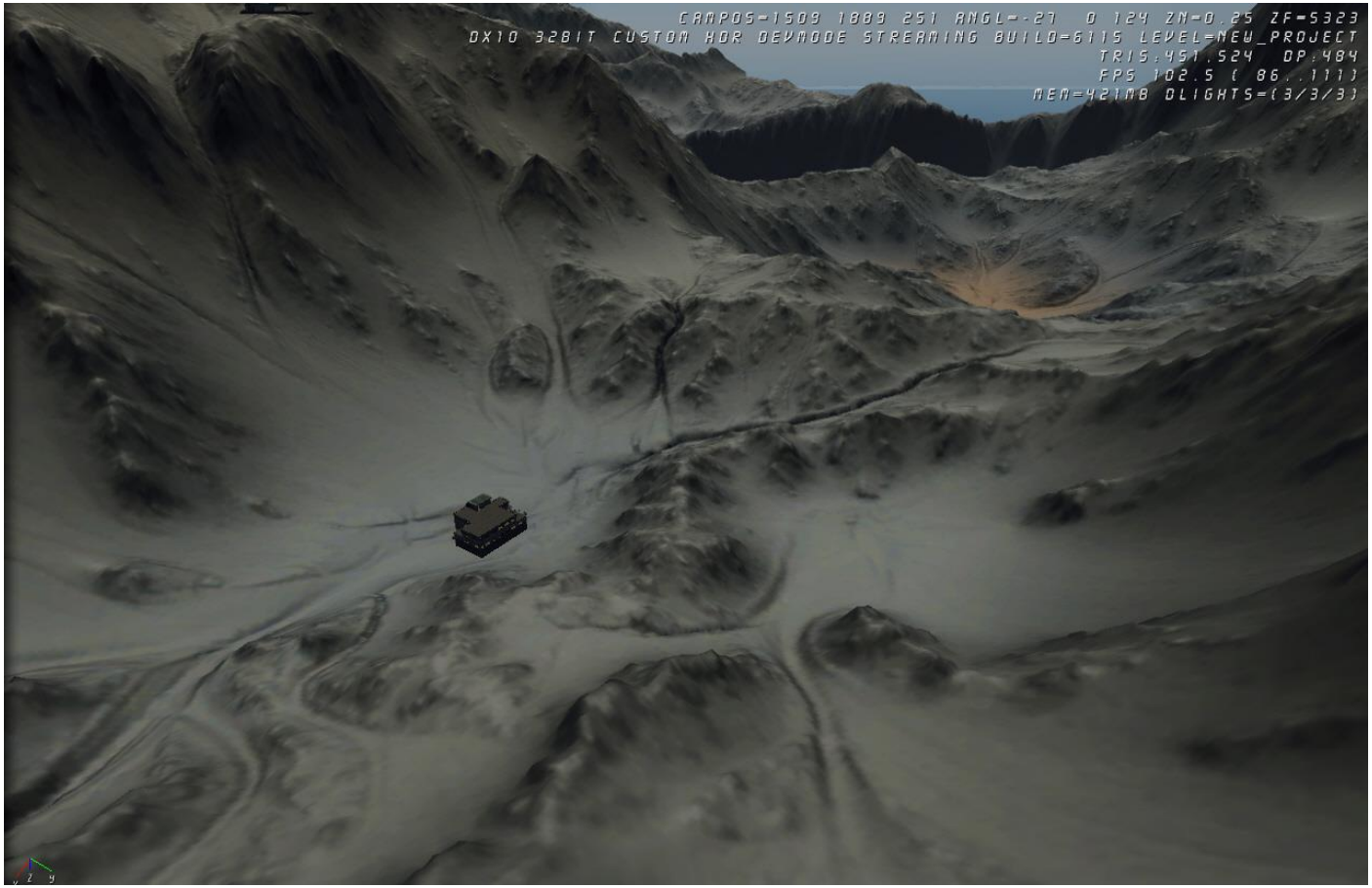
Fine-grained geometry streaming



Height Maps

TECHNISCHE UNIVERSITÄT DARMSTADT

Just Y instead of X/Y/Z



Level Streaming

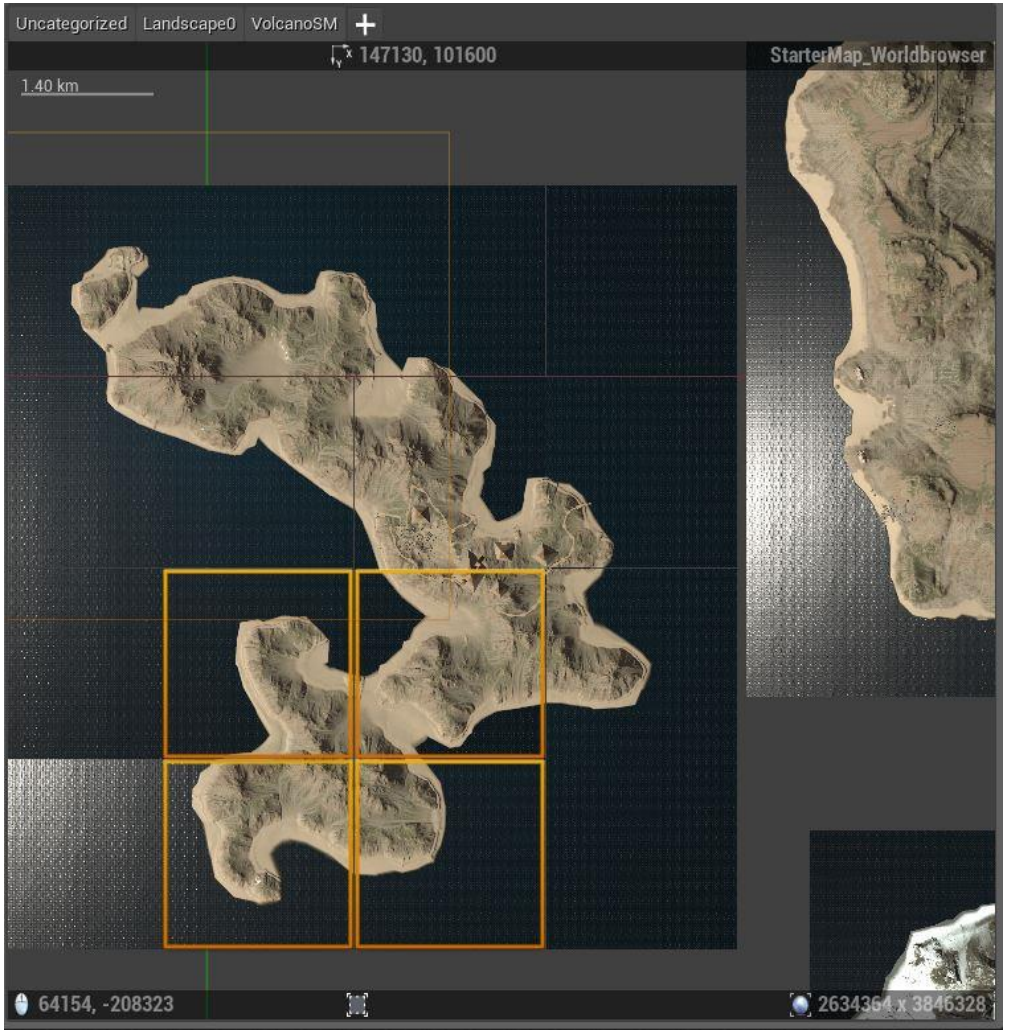
TECHNISCHE UNIVERSITÄT DARMSTADT

Included in current game engines

Very coarse geometry streaming

Need to watch out for objects and data

- Pathfinding
- AIs
- …



Sound

TECHNISCHE UNIVERSITÄT DARMSTADT

mp3 and similar compressed formats

- Nothing special – at least not anymore

Coarse streaming for sound effects

- Easy
  - Sound effects are short
  - Sound effects don’t stay on screen
  - Sound effects can stay in CPU RAM

Fine grained streaming for music and maybe speech

- Even mp3 players do it

Really Big Worlds

TECHNISCHE UNIVERSITÄT DARMSTADT

32 bit floats

- "total precision is 24 bits (equivalent to $\log_{10}(2^{24}) \approx 7.225$ decimal digits)"
  - Can be a little tight for big worlds

Use 64 bit floats for positions

- Hard to integrate 32 bit physics engines

Split and Shift the world

- Split the world
- Shift the closest parts to a position nearer at the camera
  - Physics work better with smaller coordinates

Procedural Worlds

TECHNISCHE UNIVERSITÄT DARMSTADT

Elite

- 1984
- 8 galaxies with 256 planets each
- Generated galaxies, planets including names and properties
- BBC Micro: max. 128 KB Memory, Elite was 52 KB of disk space

Minecraft

- Official Release 2011
- Generates terrain including placement of settlements, resources, ... Procedurally
- Sold for 2.5 billion USD to Microsoft in 2014

Vegetation

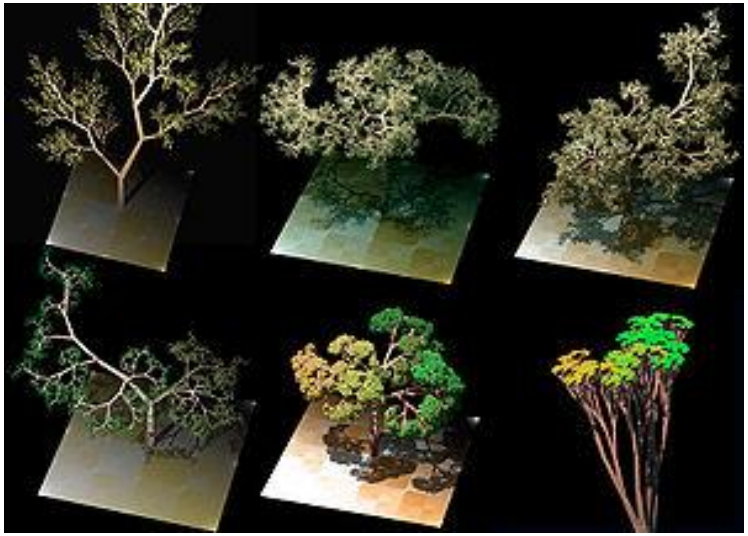
TECHNISCHE UNIVERSITÄT DARMSTADT

One of the oldest fields of procedural content generation in computer graphics

Lots of info available at:
http://algorithmicbotany.org

Well suited for generation

- Based on natural processes
- Complexity makes the shapes look realistic
- Can be found by examining how nature handles growth



Principles

Structure

- Many PCG algorithms can create instances of classes of objects
- One type of house, tree, clothing, ...
- Recognizable structure in each instance
- Structured way of deriving an instance

Randomness

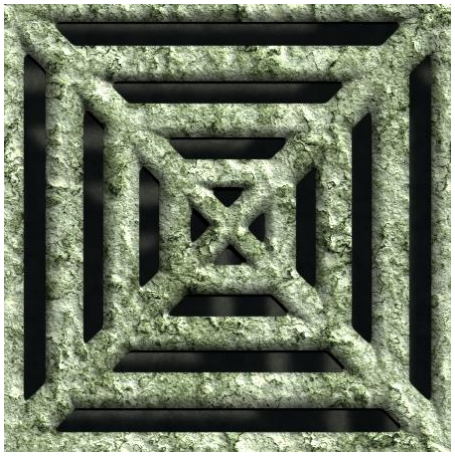
- Not a defining characteristic of PCG
- But often a central component

Texture Generation

TECHNISCHE UNIVERSITÄT DARMSTADT

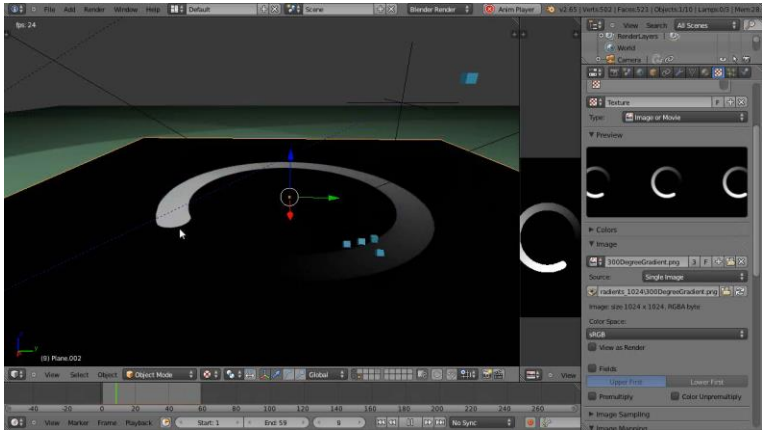
Generate

- Texture
- Normal Map
- Specular Map
- ...



Can be used in different systems

- Textures for objects
- Height maps
- Controlling flow or emission of particles



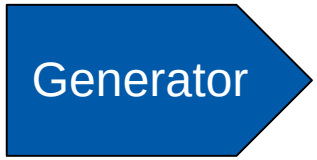
https://www.youtube.com/watch?v=UZGoht2vkzU

Texture Generation

TECHNISCHE UNIVERSITÄT DARMSTADT

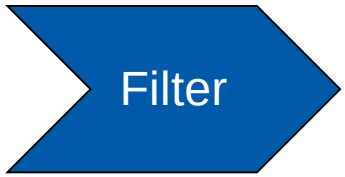
Basic Generators & Image inputs

- Provide basic shapes and patterns
- Can insert randomness into the process
- Also image inputs to use in further steps



Filters

- Change the look of the input texture
- Enhance, blur, filter, ...
- Carry out mathematical operations



Combinations

- Combine different textures

Texture Generation Node Networks

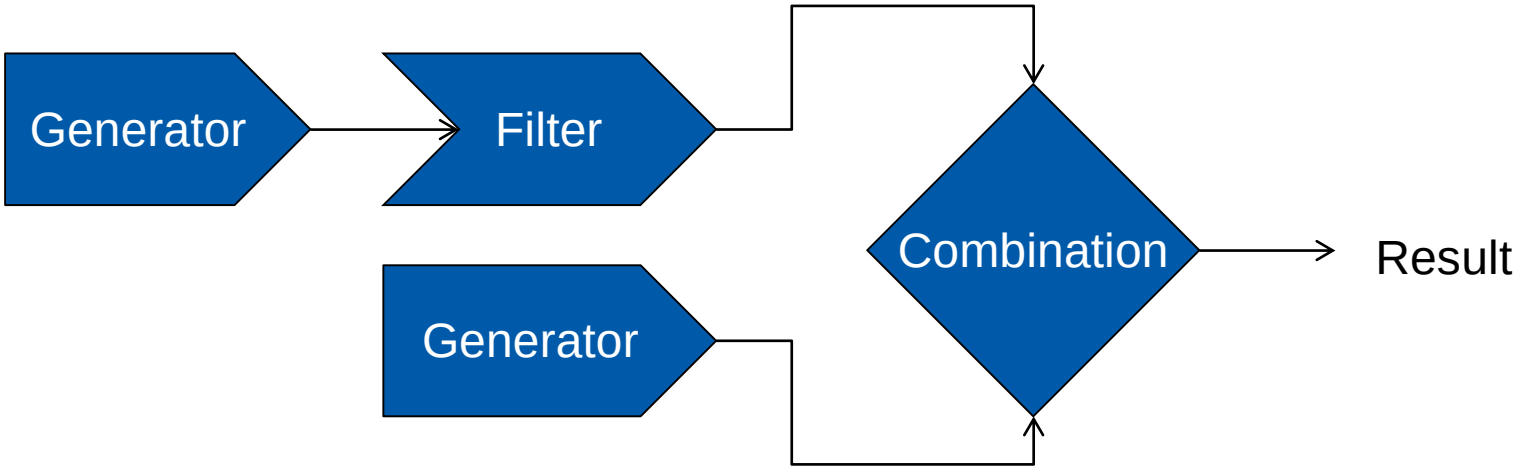
TECHNISCHE UNIVERSITÄT DARMSTADT

Combine different algorithms

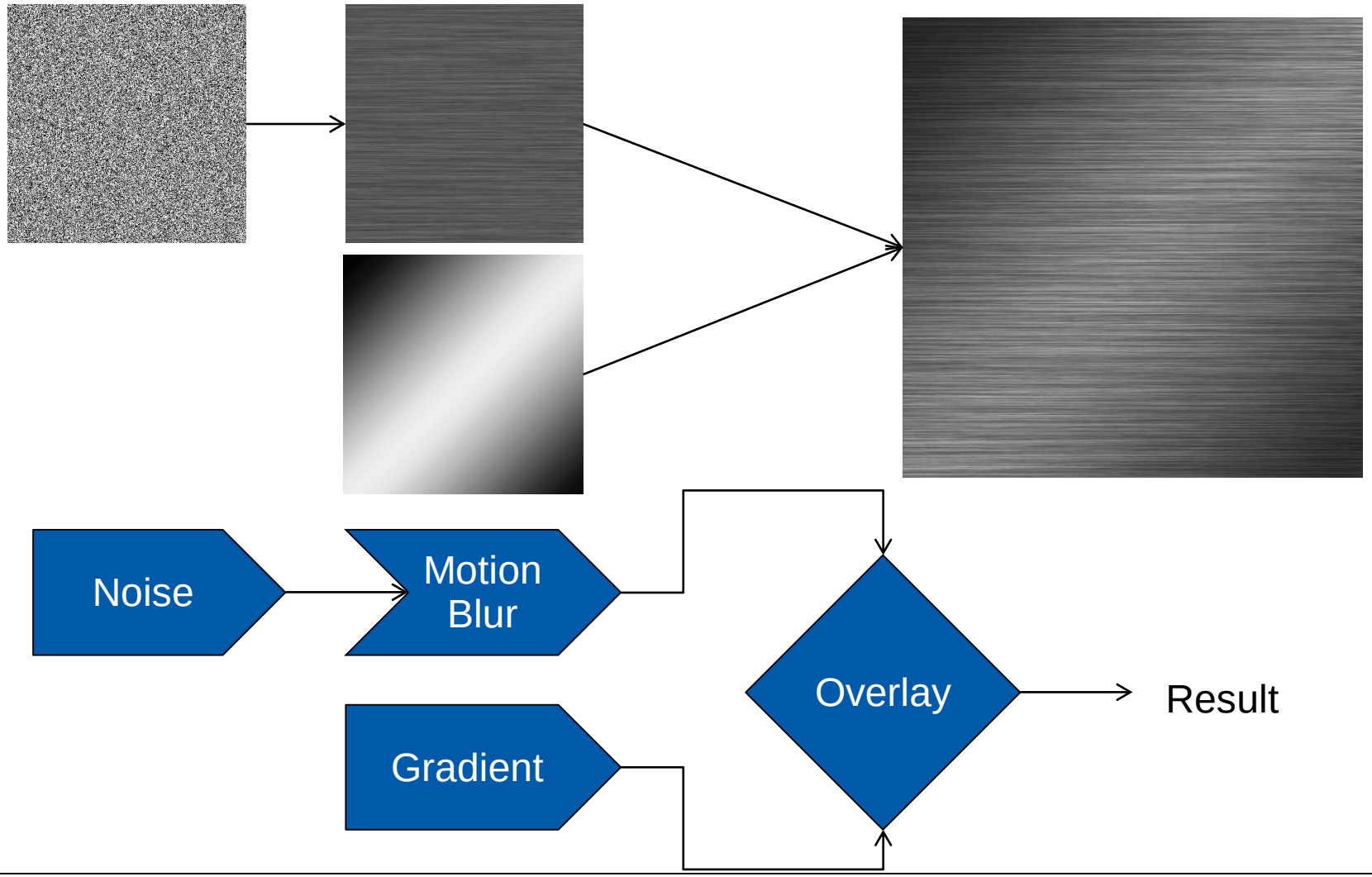
Basic Generators have only texture output(s)

Filters and Combiners have

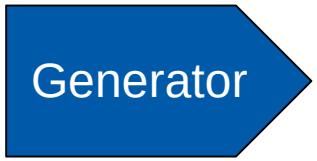
- One or more texture inputs
- One or more texture outputs



Example of networks - Metal



Basic Generators



TECHNISCHE UNIVERSITÄT DARMSTADT

Random

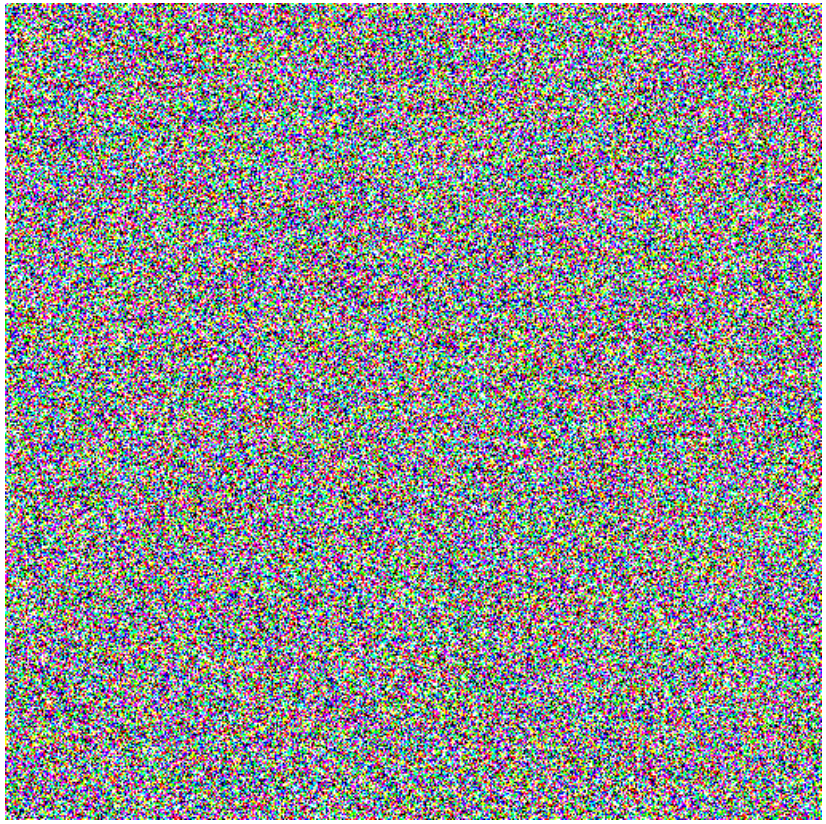
- **All colors**
- Grayscale

Patterns

- Grids
- Dots/Spheres
- Jittered patterns
- Voronoi Diagram

Random Noise

- Perlin Noise



Basic Generators

Generator

TECHNISCHE UNIVERSITÄT DARMSTADT

Random

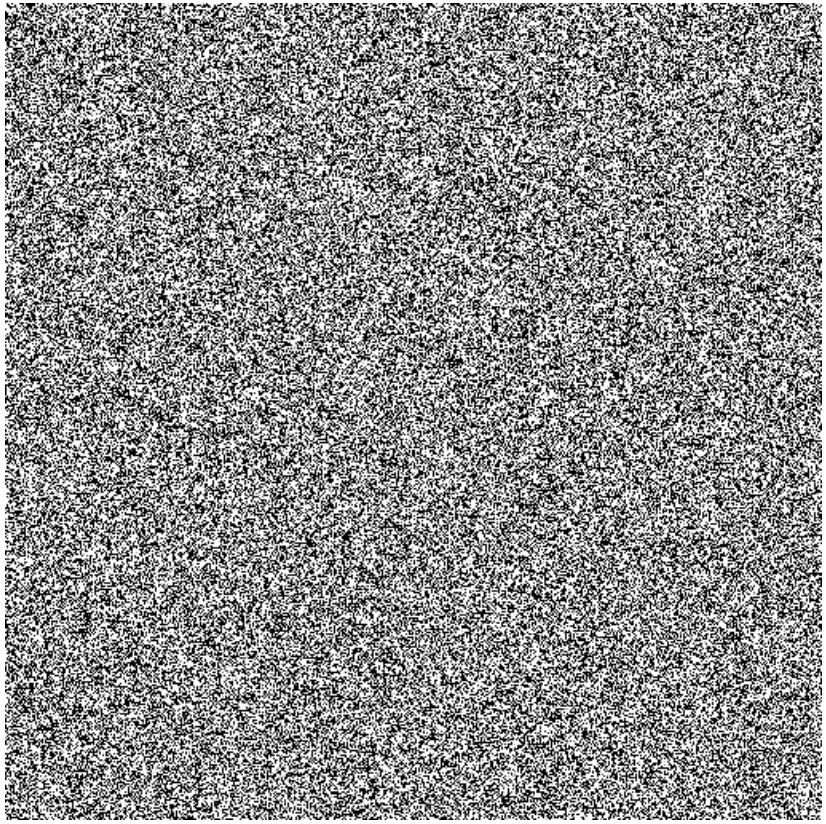
- All colors
- **Grayscale**

Patterns

- Grids
- Dots/Spheres
- Jittered patterns
- Voronoi Diagram

Random Noise

- Perlin Noise



Basic Generators

Generator

TECHNISCHE UNIVERSITÄT DARMSTADT

Random

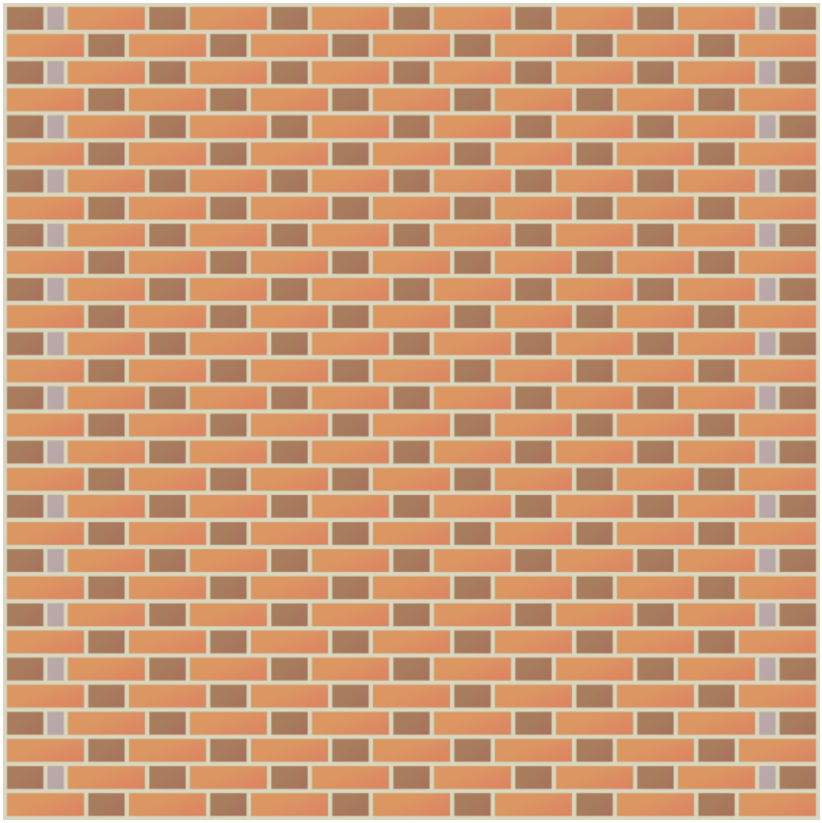
- All colors
- Grayscale

Patterns

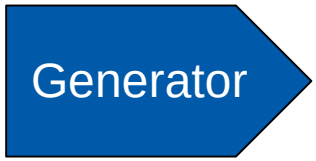
- **Grids**
- Dots/Spheres
- Jittered patterns
- Voronoi Diagram

Random Noise

- Perlin Noise



Basic Generators



TECHNISCHE UNIVERSITÄT DARMSTADT

Random

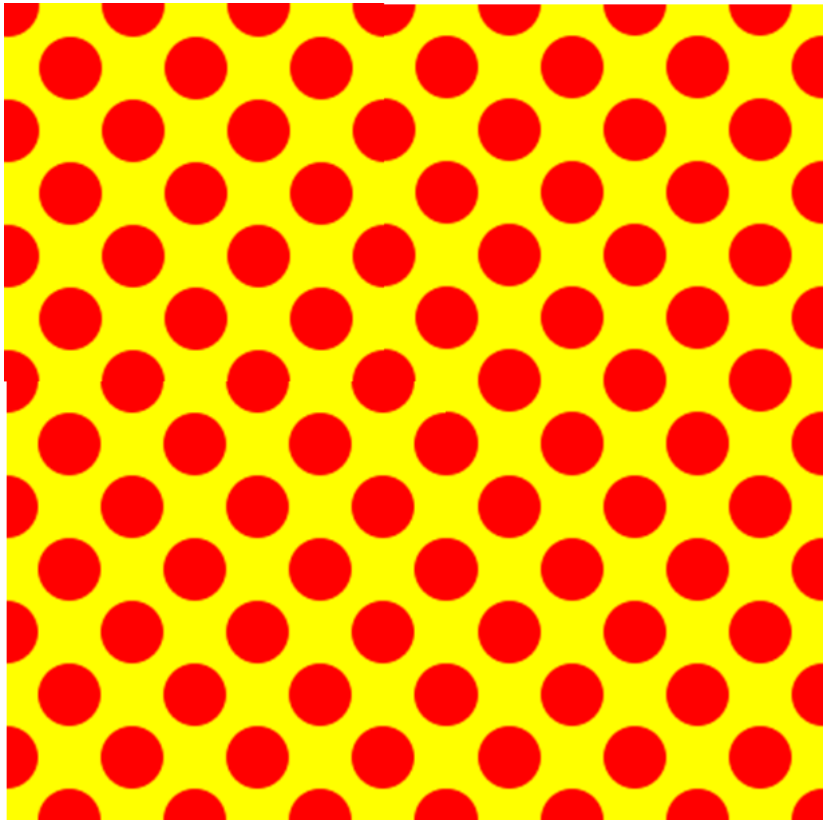
- All colors
- Grayscale

Patterns

- Grids
- **Dots/Spheres**
- Jittered patterns
- Voronoi Diagram

Random Noise

- Perlin Noise



Basic Generators

Generator

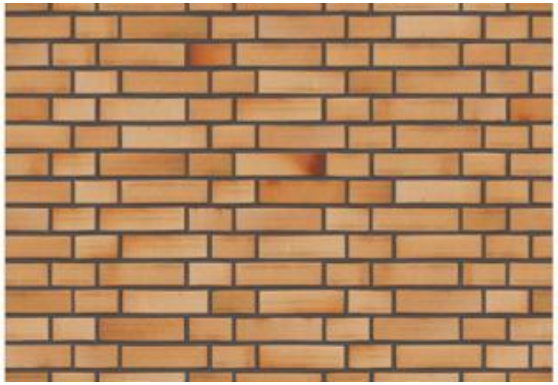
TECHNISCHE UNIVERSITÄT DARMSTADT

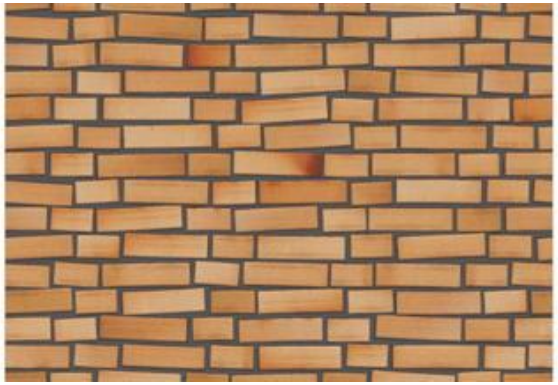
Random

- All colors
- Grayscale

Patterns

- Grids
- Dots/Spheres
- **Jittered patterns**
- Voronoi Diagram





Random Noise

- Perlin Noise

Image Source: http://www.tri-nitro.com/project_mt_1_key_features.php

Basic Generators

Generator

TECHNISCHE UNIVERSITÄT DARMSTADT

Random

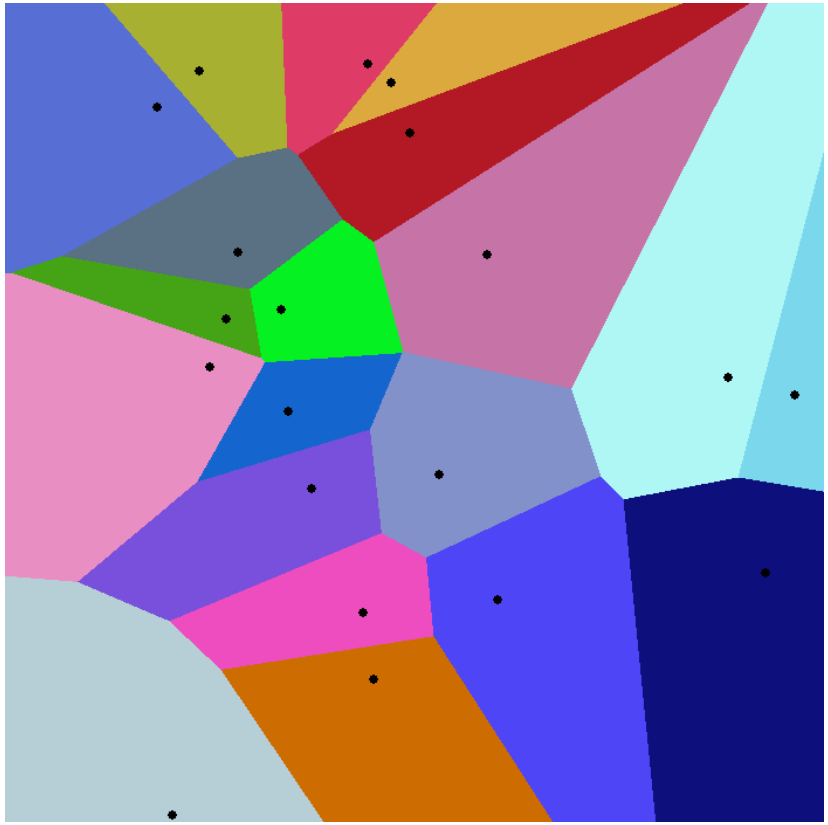
- All colors
- Grayscale

Patterns

- Grids
- Dots/Spheres
- Jittered patterns
- **Voronoi Diagram**

Random Noise

- Perlin Noise



Voronoi Diagram

TECHNISCHE UNIVERSITÄT DARMSTADT

Take a set of points C1 to Cn, „sites“

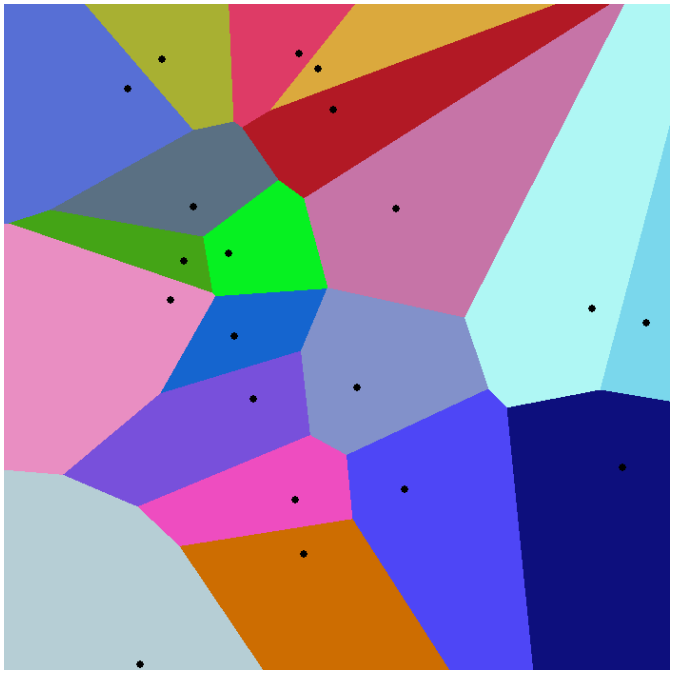
Every point Ci defines a cell such that for each point P in the cell, no other point in C lies closer to P than Ci

Regular points lead to regular patterns

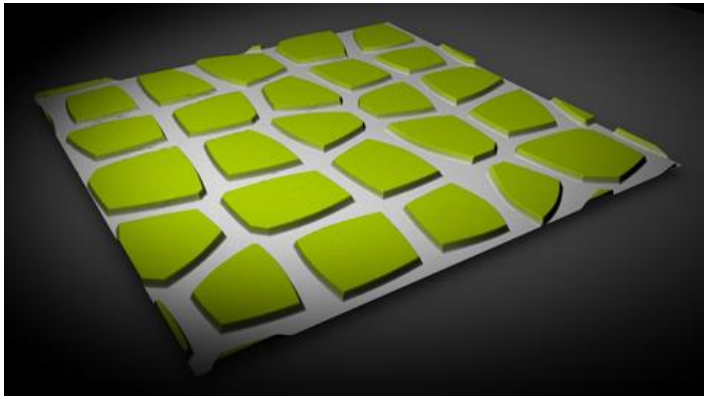
Random points lead to irregular patterns

- Reptile skin
- Parcels of land
- ...

(Dual to Delaunay Triangulation)

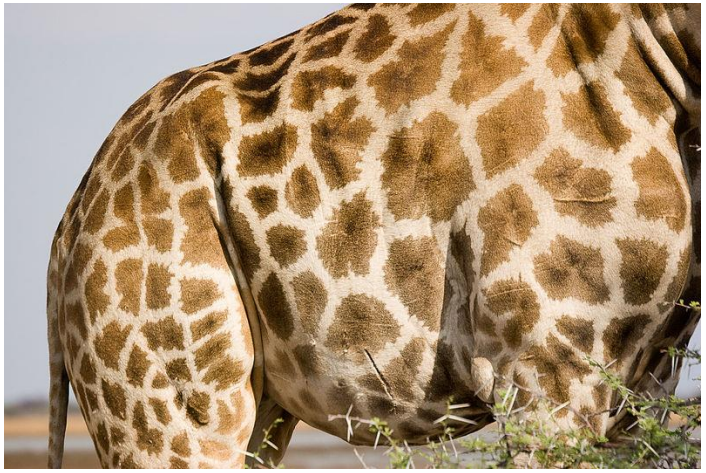


Voronoi Diagram – Texture examples

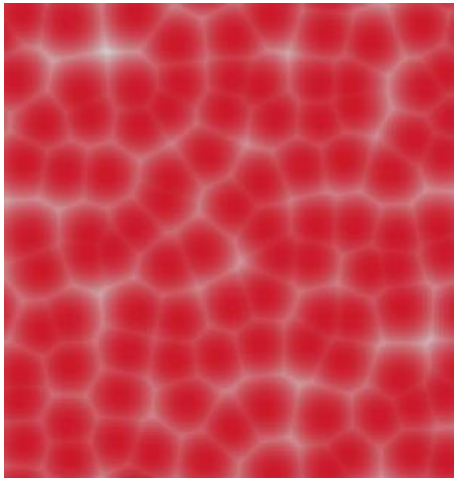


Cobblestone

Dry dirt



Giraffe skin



Blood cells

Basic Generators

Generator

TECHNISCHE UNIVERSITÄT DARMSTADT

Random

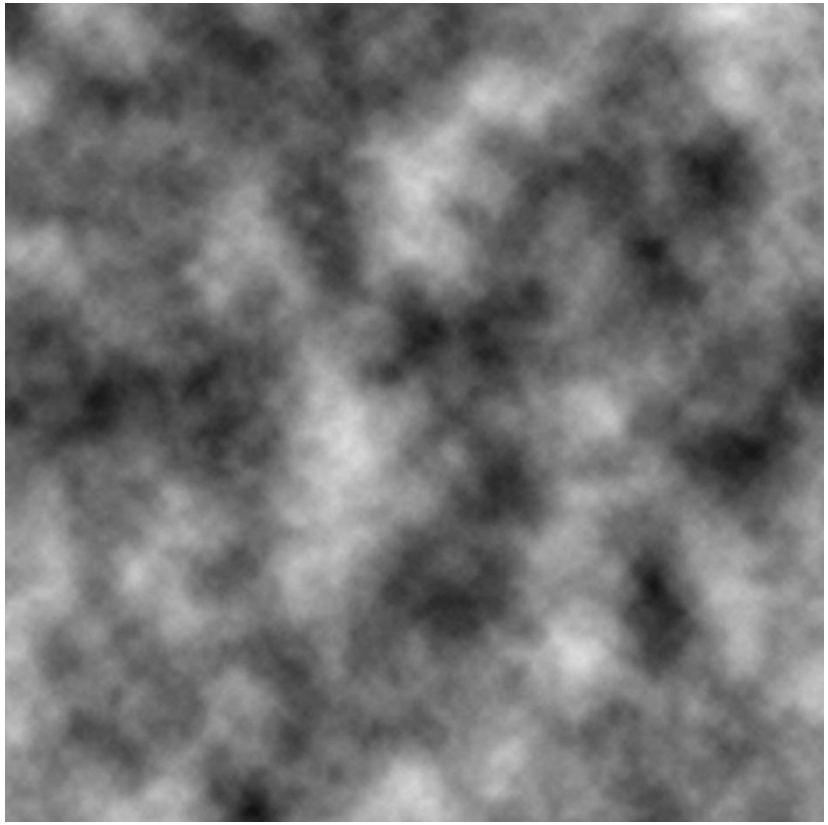
- All colors
- Grayscale

Patterns

- Grids
- Dots/Spheres
- Jittered patterns
- Voronoi Diagram

Random Noise

- **Perlin Noise**

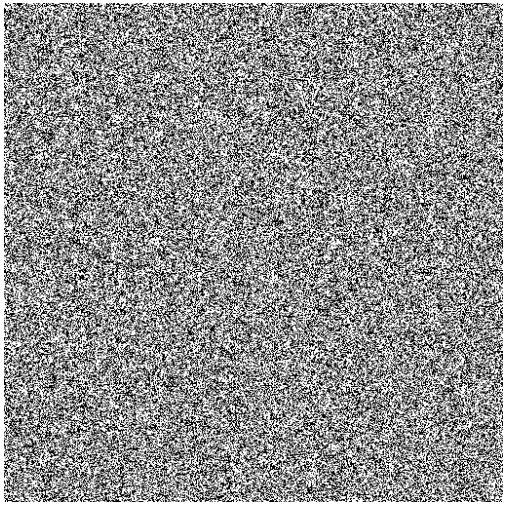


Perlin Noise / Simplex Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

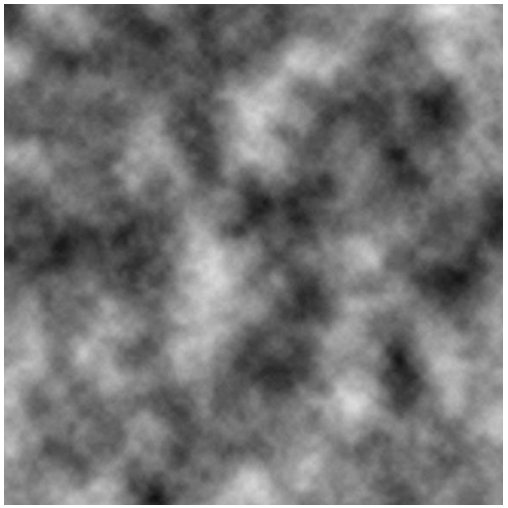
Random pixels

- No continuity
- If we interpreted it as a 2-dimensional function (heightmap), it would not work

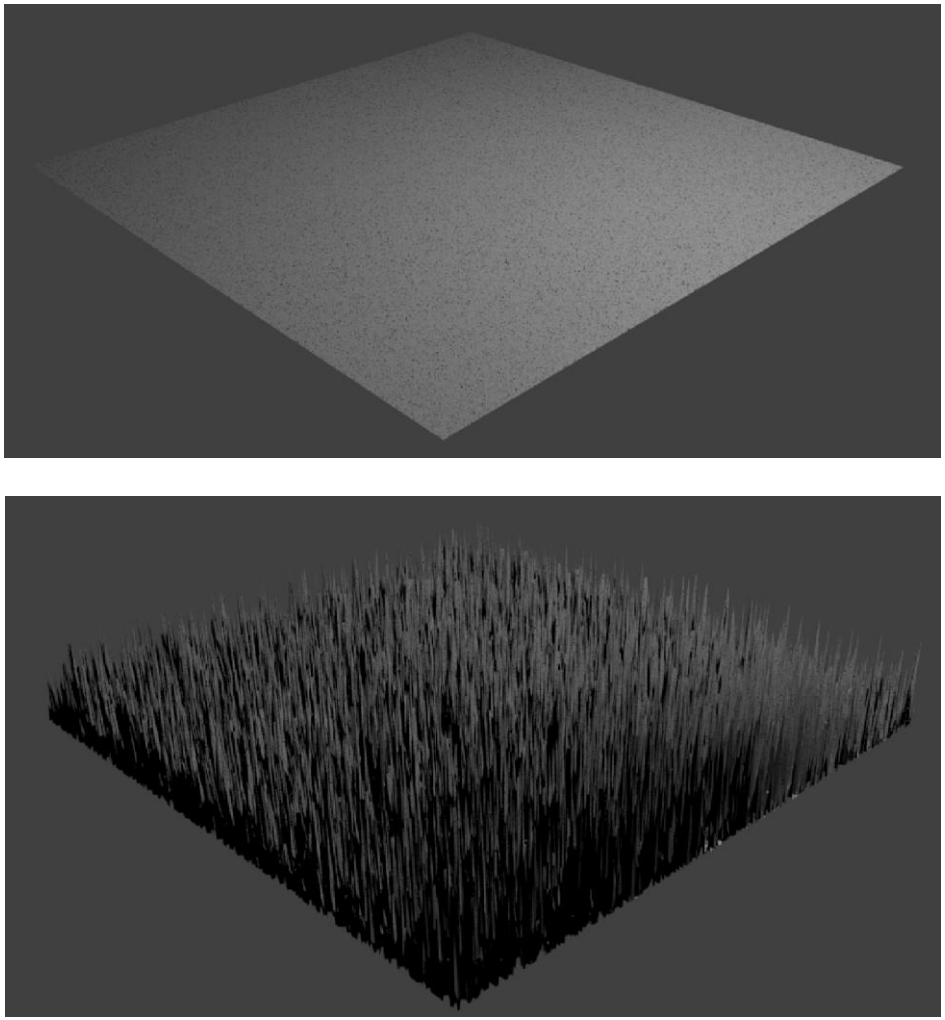


Semi-Random Noise

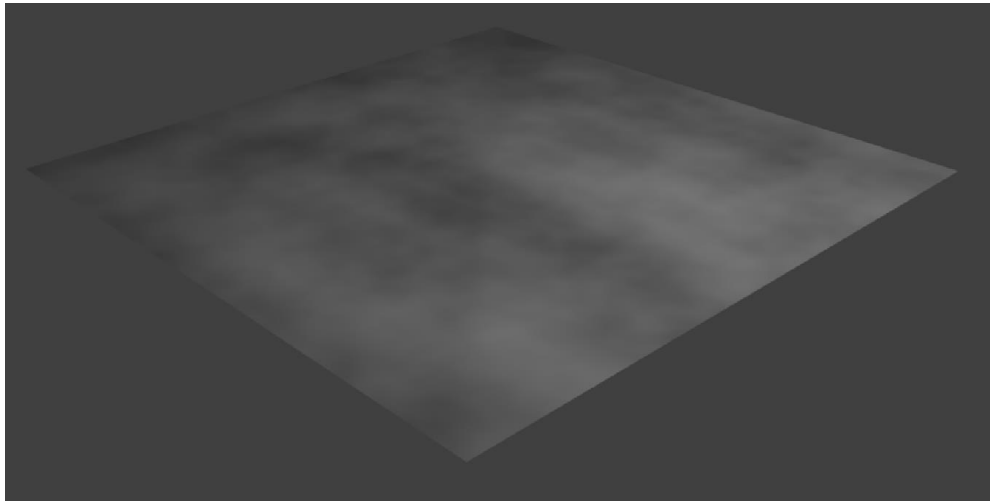
- Cloud-like look
- Continuous
- Works well as a heightmap

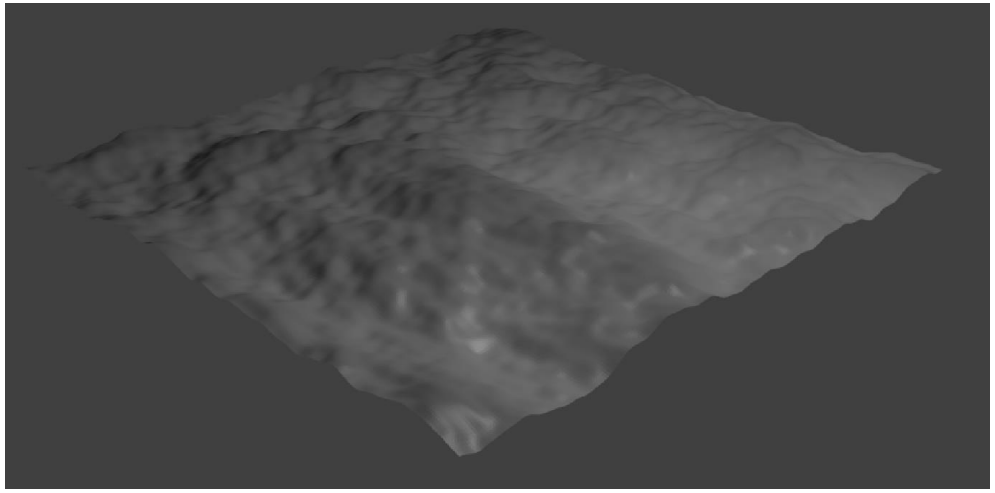


Purely random noise



Continuous noise





Perlin Noise, Simplex Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

Perlin Noise

- Developed by Ken Perlin
  - Invented while working on „Tron“ in 1982
  - Won an Oscar in 1997
- Omnipresent noise generation function

Simplex Noise

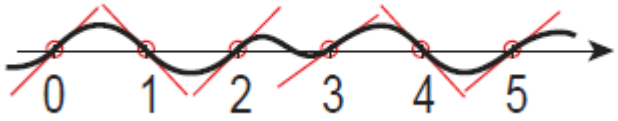
- Suggested by Perlin in 2001 as a succesor to the previous noise function
- Better properties
- Scales better to higher dimensions

Perlin Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

Gradient-based noise

- Determine for each integer value
  - Function value 0
  - Pseudo-random gradient

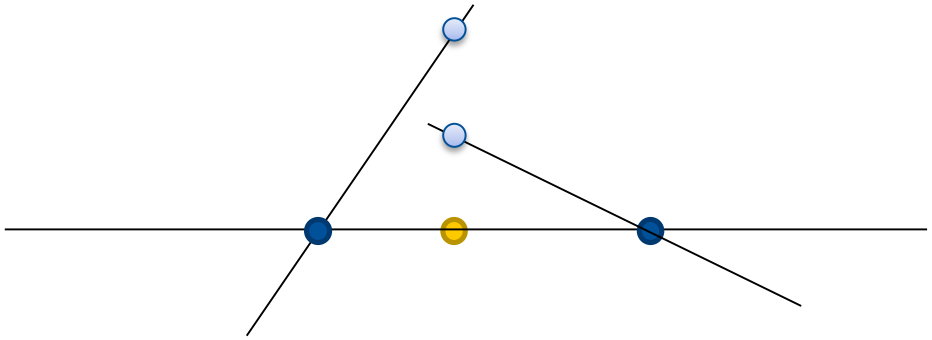


Perlin Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

For a given point x (2D), the result is computed by blending

- The value of the previous gradient extrapolated to point x
- The value of the next gradient extrapolated to point x

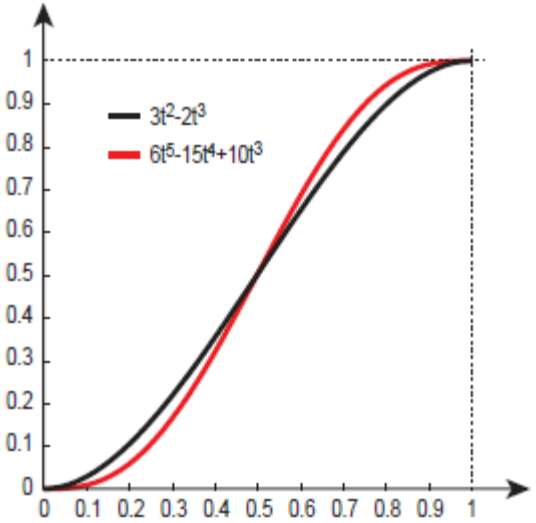


Perlin Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

Blending function

- Originally f(t) = 3t^2 – 2t^3
- Later f(t) = 6t^5 – 15t^4 + 10t^3



Purpose

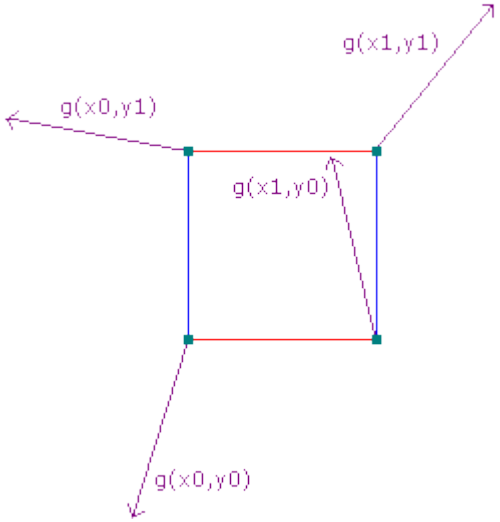
- This way, the noise is also continuous at the integer positions

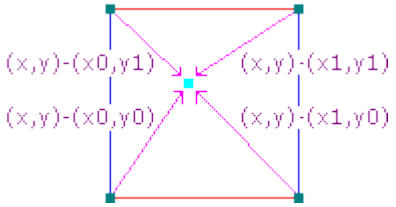
Gradients, Computation

TECHNISCHE UNIVERSITÄT DARMSTADT

Use the dot product to calculate the contribution of a gradient to the sample

- Gradients are defined at the grid points
- Use vectors from grid points pointing to (x, y)





Interpolate in x-direction (2 rows)

Interpolate in y-direction

Perlin Noise – Dot product

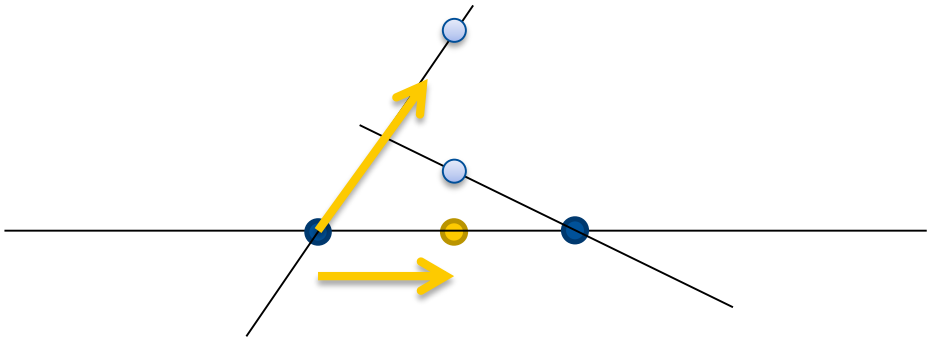
TECHNISCHE UNIVERSITÄT DARMSTADT

Can be seen better in 1D

Gradient is the slope of the function

Vector towards the evaluated point is the x-Value

In this case, the dot product becomes slope * x



Using Perlin Noise

TECHNISCHE UNIVERSITÄT DARMSTADT

Normalize the noise

- Divide x by width and y by height

Frequency

- Noise = perlin(xnormalized * frequency, ynormalized * frequency)

Amplitude

- Noise = perlin(x, y) * amplitude

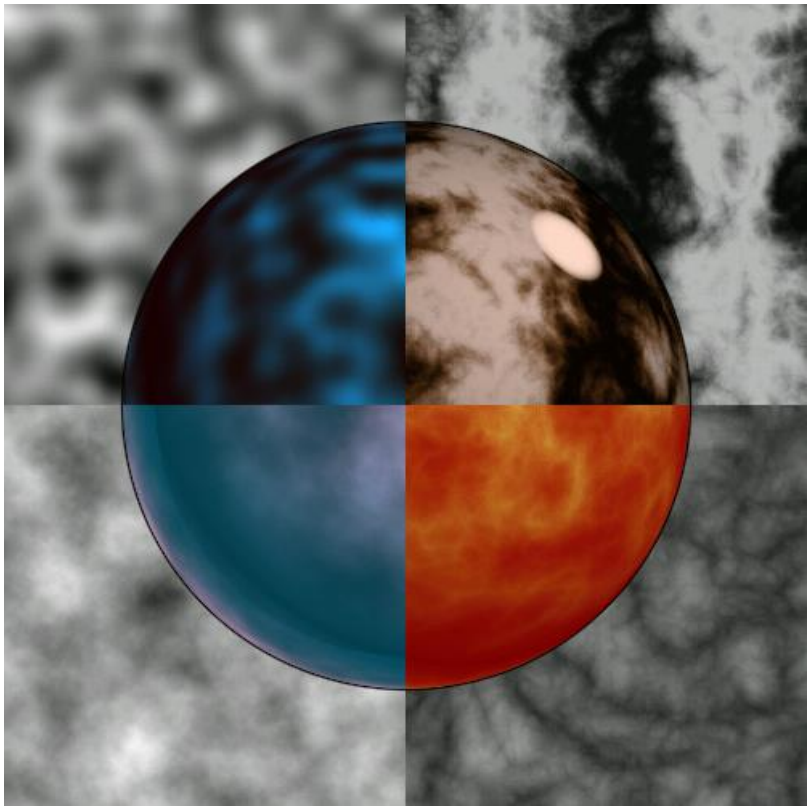
Bring into range [0, 1]

- Noise is in [-1, 1]
- → Add 1, Divide by 2

Using Perlin Noise

noise

sin(x + |noise(**p**)| + ½ |noise(2**p**)| + ...)



noise(**p**) + ½ noise(2**p**) + ¼ noise(4**p**) ...

|noise(**p**)| + ½ |noise(2**p**)| + ¼ |noise(4**p**)| ...

Filters - Basics

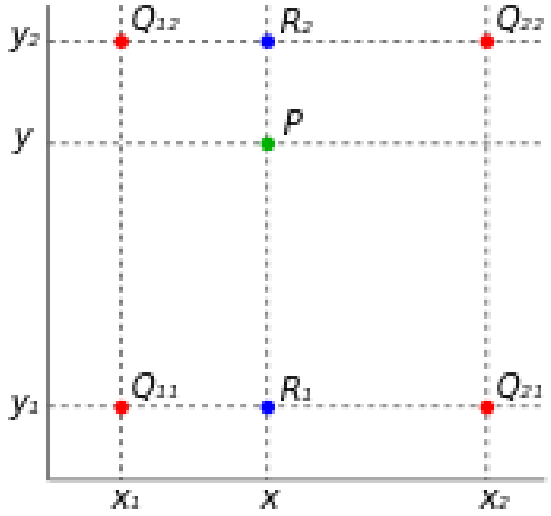
Each pixel of the resulting image is based on one or more pixels of the input image

Remember bilinear filtering for texture lookups

- We looked up the values of 2x2 pixels to get a value for the final pixel

Filter kernel

- Specifies the pixels we need to sample and the weights we sample them with



Box filter

Filter

TECHNISCHE UNIVERSITÄT DARMSTADT

Move a box over the image

- New pixel = Sum of original pixels * weights
- Iterate over the image and calculate new pixels

Minimal Kernel size: 3x3

- Sizes schould be odd numbers (→ central pixel)

In the following slides

- Divide by the sum of the values of the kernel → Normalization
- Alternatively, floating point numbers could be used

How to handle edges?

- Similar to texture lookup
- Extend the image, fill with constant color, …

Image source for next slides: http://tech-algorithm.com/articles/boxfiltering/

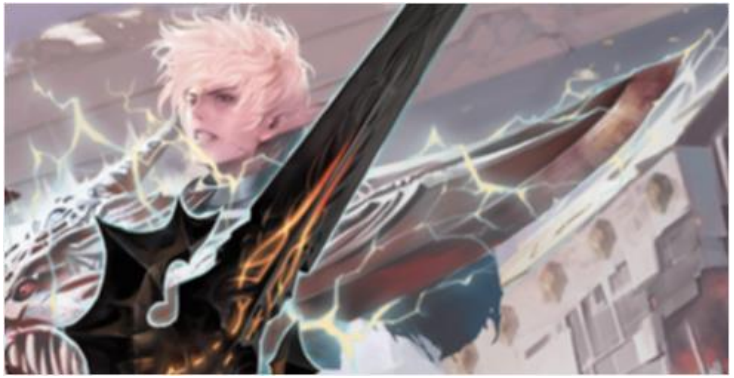
Box filter results

Unfiltered image

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Smoothing

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Box filter results

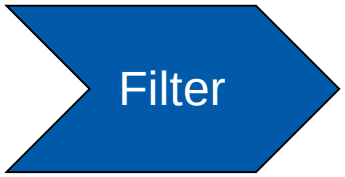
Sharpening

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Raised

$$\begin{bmatrix} 0 & 0 & -2 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Box filter results



TECHNISCHE UNIVERSITÄT DARMSTADT

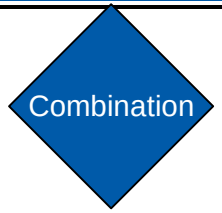
Motion Blur

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Edge Detection

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Combinations



TECHNISCHE UNIVERSITÄT DARMSTADT

Remember the lecture on Alpha Blending → Combination of source and destination pixels

Modes

- Normal blend mode
- Dissolve
- Multiply
- Screen
- Overlay
- Hard Light
- Soft Light
- Dodge and burn
- Divide
- Addition
- Subtract
- Difference
- Darken Only
- Lighten Only

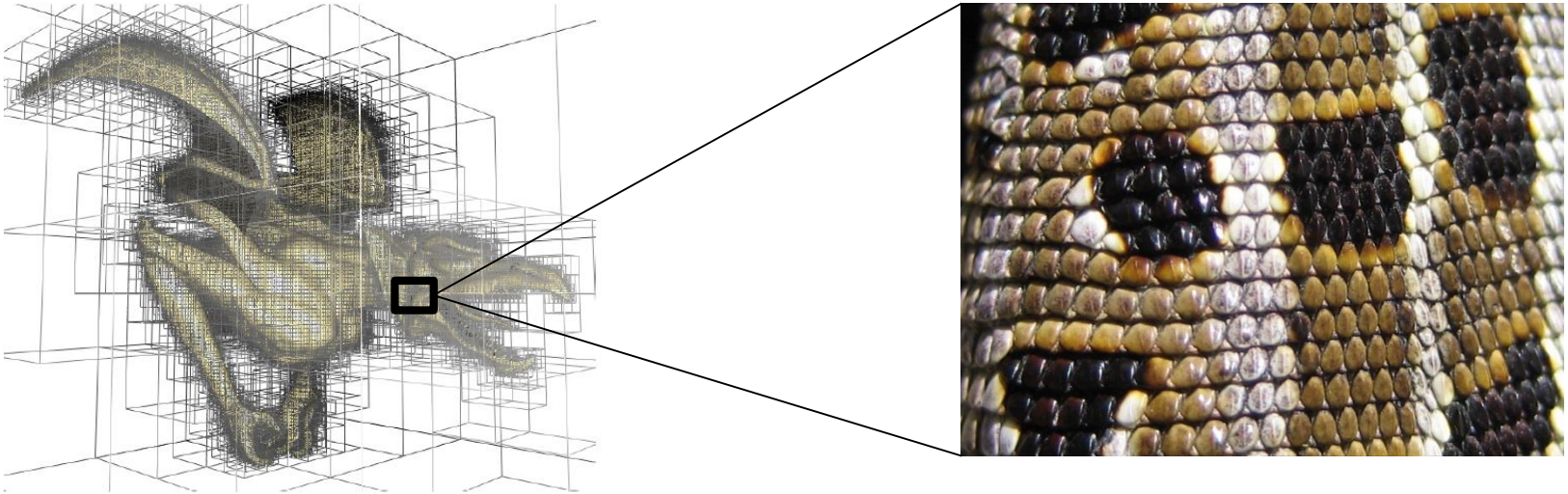
Examples: http://docs.gimp.org/en/gimp-concepts-layer-modes.html

Combine PCG and Streaming

TECHNISCHE UNIVERSITÄT DARMSTADT

As in Minecraft – Generate parts of the world not yet seen

Fill in PCG details after a certain level of detail



Literature

TECHNISCHE UNIVERSITÄT DARMSTADT

Julian Togelius

- IT University of Copenhagen
- http://julian.togelius.com/

Procedural Content Generation in Games - A textbook and an overview of current research

- Available for free at http://pcgbook.com/

PCG Wiki

- http://pcg.wikidot.com

Ebert, Musgrave, Peacheay, Perlin, Worley:

Texturing & Modeling – A procedural approach