Game Technology

Lecture 11 – 16.01.2015

Dipl-Inf. Robert Konrad
Dr.-Ing. Florian Mehm

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

Preliminary timetable

Lecture No.	Date	Topic
1	17.10.2014	Basic Input & Output
2	24.10.2014	Timing & Basic Game Mechanics
3	31.10.2014	Software Rendering 1
4	07.11.2014	Software Rendering 2
5	14.11.2014	Basic Hardware Rendering
6	21.11.2014	Animations
7	28.11.2014	Physically-based Rendering
8	05.12.2014	Physics 1
9	12.12.2014	Physics 2
10	19.12.2014	Procedural Content Generation
11	**16.01.2015**	**Compression & Streaming**
12	23.01.2015	Multiplayer
13	30.01.2015	Audio
14	06.02.2015	Scripting
15	13.02.2015	AI


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

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

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

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

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

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
  - Constant size of a pixel or a pixel block

Possible compression strategies

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

Example 1

TECHNISCHE UNIVERSITÄT DARMSTADT

ETC

- Ericsson Texture Compression

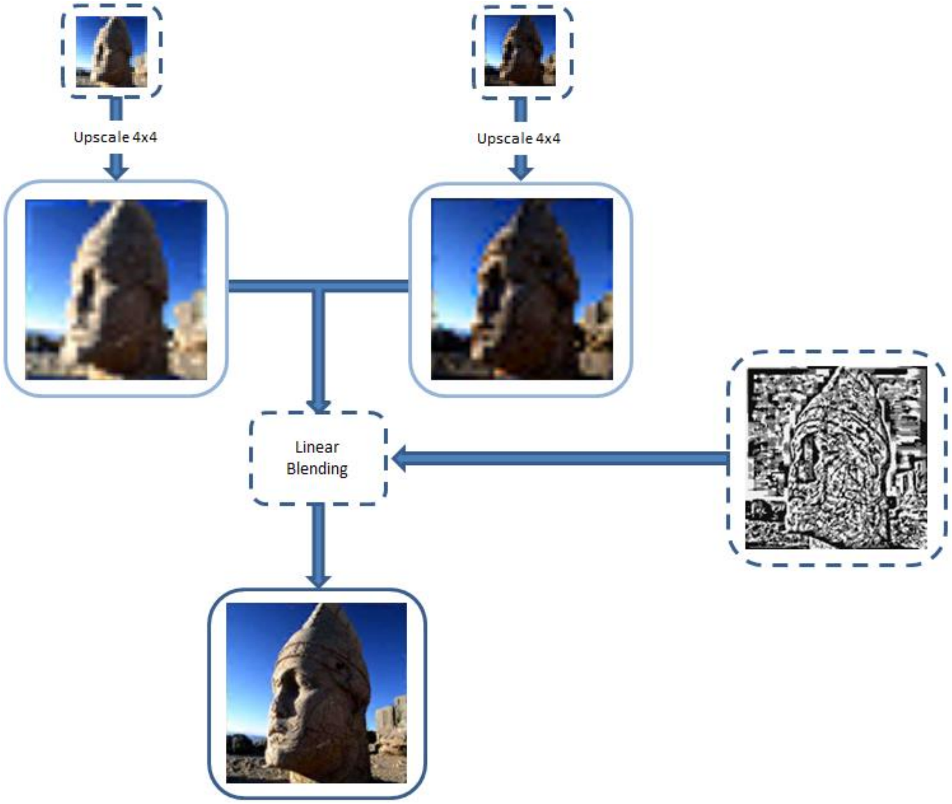
Compresses 4x4 pixel blocks to 64 bits

- Split into two 2x4 groups
- Each gets a 12 bit base color plus 3 bit brightness range selection
- Each pixel gets a 2 bit offset value

Example 2

PVRTC

- PowerVR Texture Compression



Normal Maps,…

Compression for images might not be optimal for other textures

- But it might just work
- Swizzling channels can help

3Dc

- $x^2+y^2+z^2=1$
  - $z^2=1-x^2-y^2$
  - One value can be omitted
- Plus block compression

Manual Compression

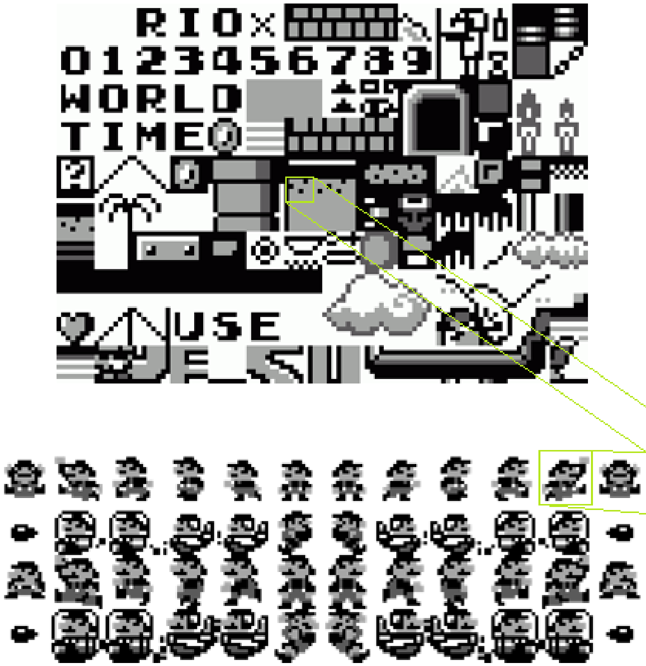
TECHNISCHE UNIVERSITÄT DARMSTADT

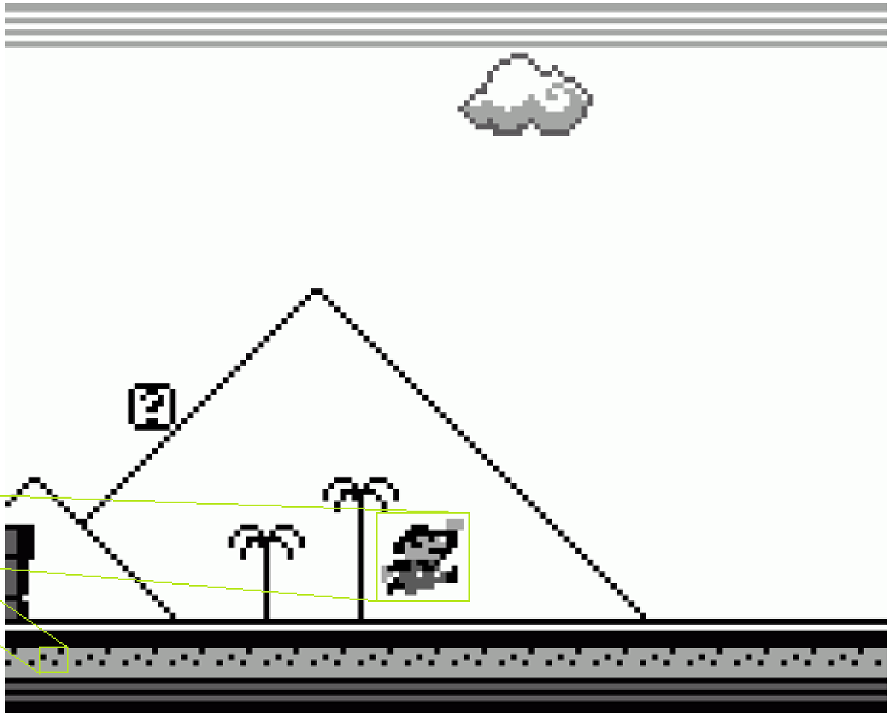
Let the artists do the job

Repeat images over and over

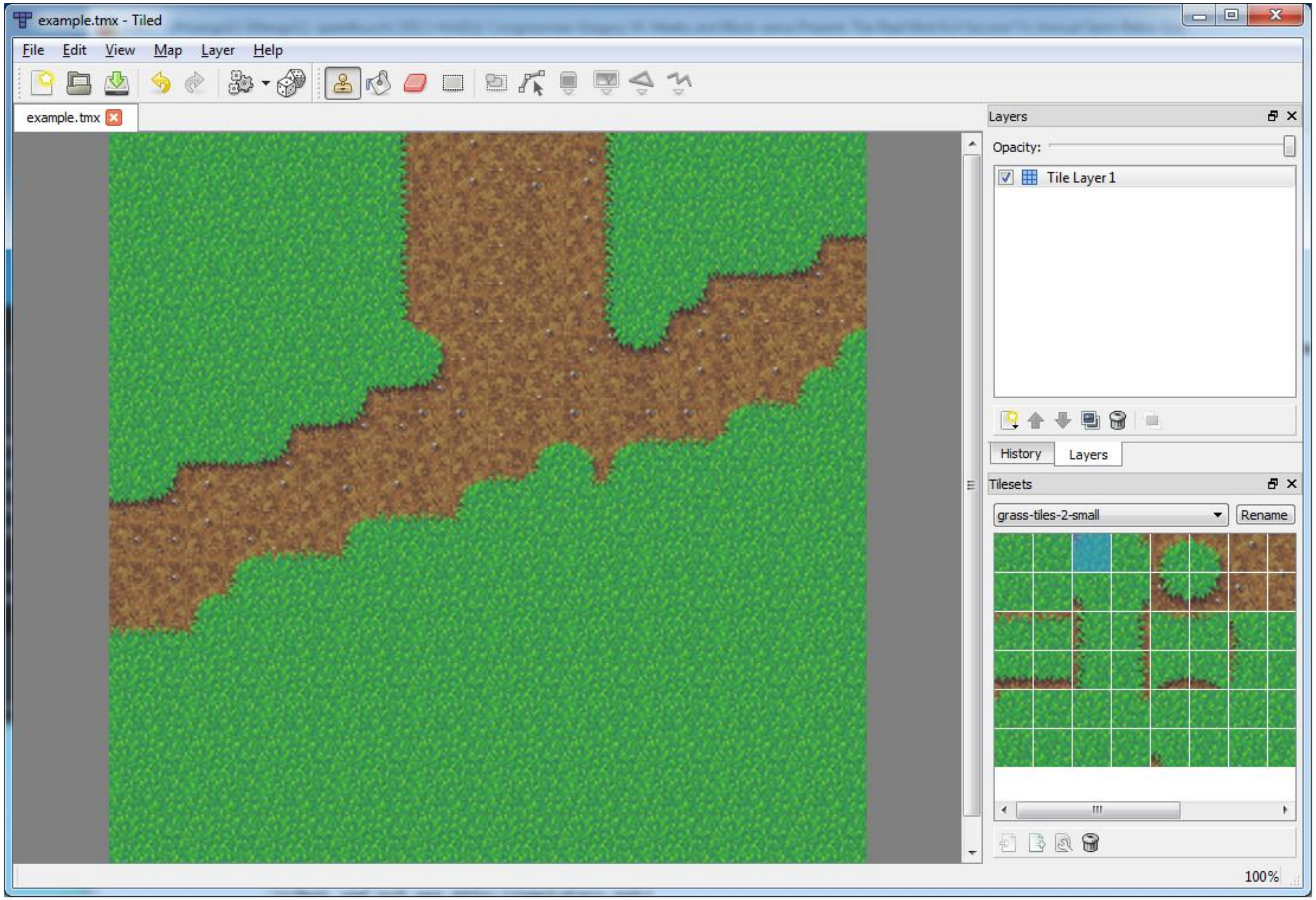
- Nobody might notice it when you do it cleverly

Tilemaps/Tilesets





Tile Editors



Pitfall: The Mayan Adventure

Warcraft 3

Tilemaps in 3D

Bilinear Filtering

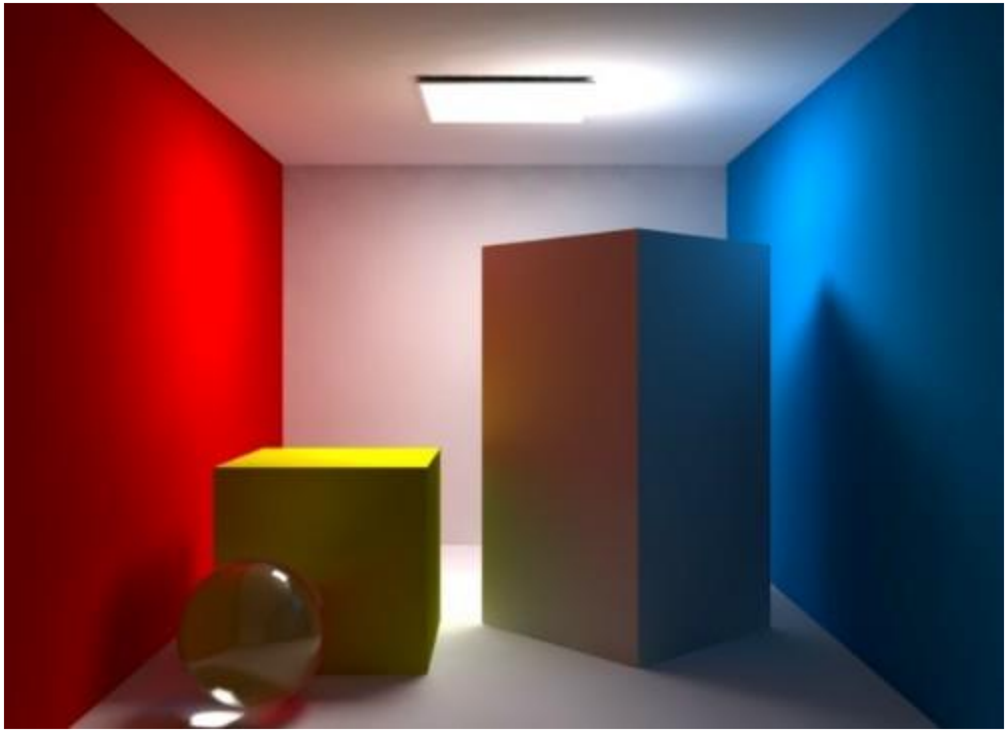
- Would have to use texels from two tiles at tile boundaries
- Complicated
- Expensive
- Rarely used

Multitexturing

Multitexturing

Multitexturing

Good lighting can hide a lack of details



Problems

Performance

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

Needs good tool support

- Scary communication with artists

Streaming

Coarse Streaming

- Load and replace complete assets

Fine Grained Streaming

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

Coarse Streaming

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

SWIV

Problems

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 is problematic

- Driver might decide to convert the texture
- Easier on console
- Probably easier with Direct3D 12

Fine grained texture streaming

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

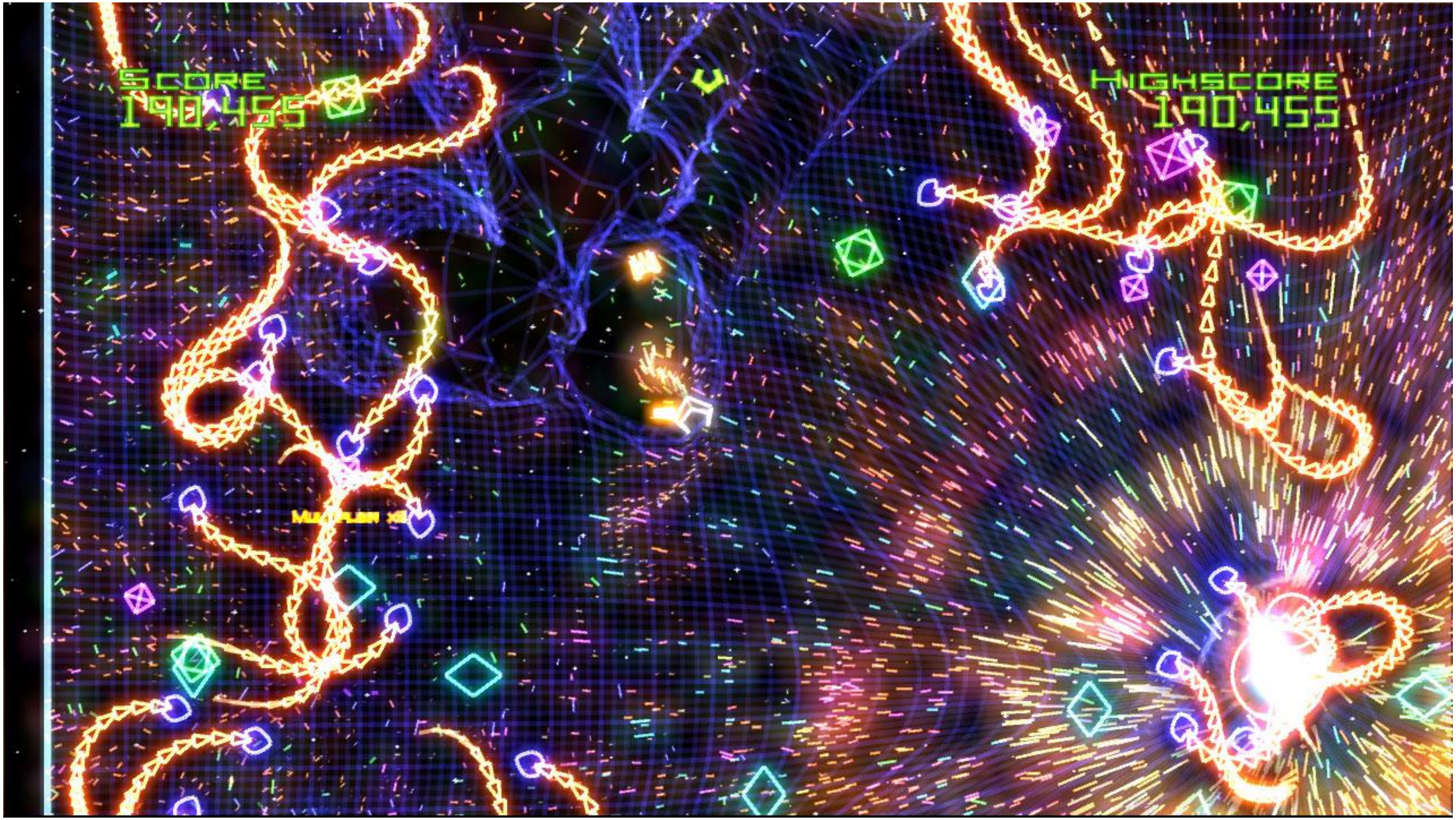
MegaTextures

Geometry is split up in tiles

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

Geometry

TECHNISCHE UNIVERSITÄT DARMSTADT



Geometry Compression

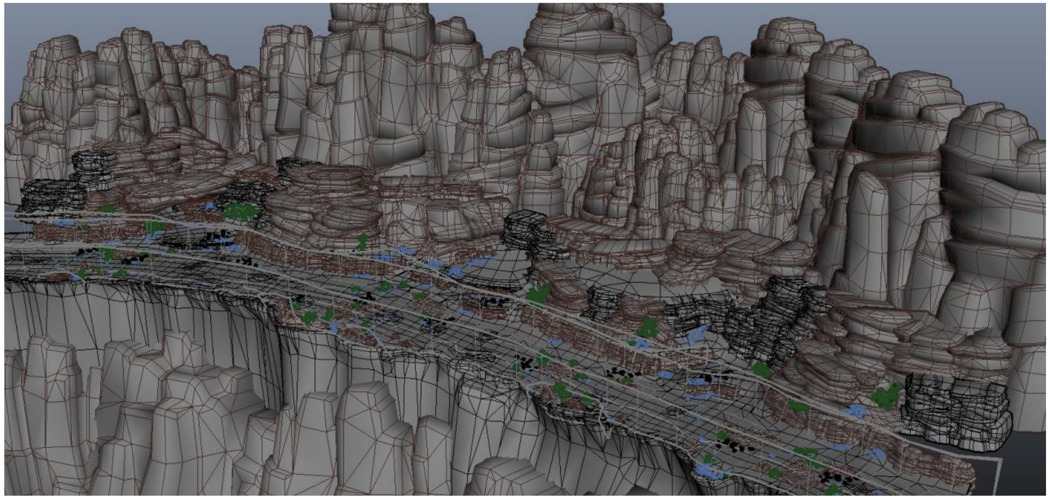
Not widely used

No hardware support

Special strategies for animations

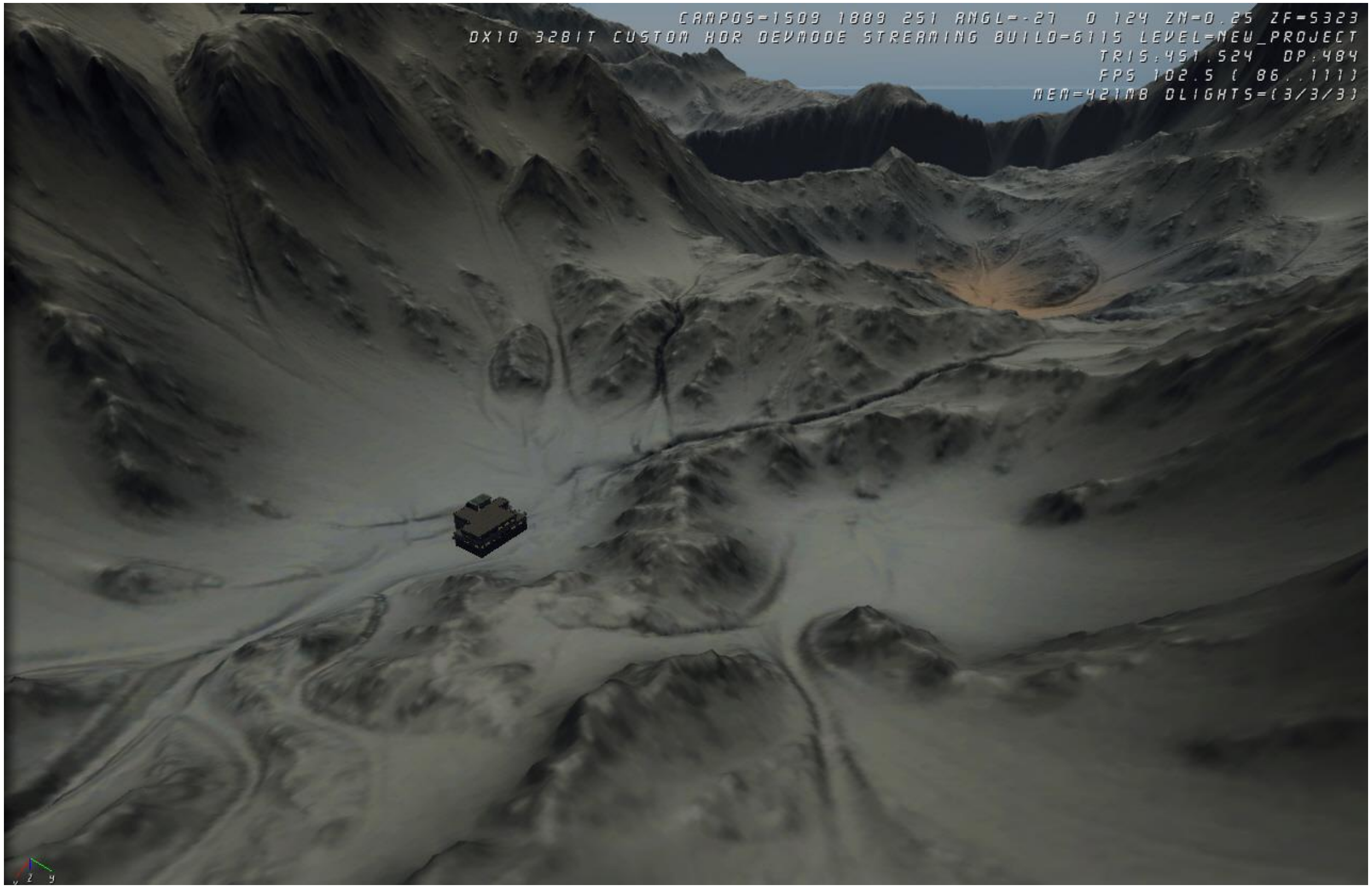
- Like skeletal animations, which are tiny

Manual Compression



Height Maps

Just Y instead of X/Y/Z



Normal Maps

TECHNISCHE UNIVERSITÄT DARMSTADT

Remove super detailed geometry

Replace with normal maps

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

Coarse Geometry Streaming

Same strategies as for textures

- Could be directly plugged into a level of detail system

Fine grained geometry streaming

To be done

Sound

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

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

Profiling

Sampling

- Samples at random intervals
- Does not modify code

Instrumentation

- Adds sampling code to binary

Performance Counters

CPU integrated circuitry that measures certain performance characteristics

- Like number of cache misses,…

Can be read by CPU specific profiling tools