GDC 13 EUROPE

Using Gigapixel Landscape Textures in Dragon Commander: Lessons Learned

Swen Vincke
CEO, Larian Studios

Charles-Frederik Hollemeersch
CTO, Graphine

GAME DEVELOPERS CONFERENCE EUROPE
COLOGNE, GERMANY
AUGUST 19-21, 2013
EXPO DATES: AUGUST 19-20
2013

About Larian Studios

- Divinity II (Dec. 2010)
  - RPG
  - PC & Xbox 360
  - Dragon fights were very popular!
- Dragon Commander (DC) (Aug. 2013)
  - RTS + Action
  - Lots of dragon fights


20
51
+37
30
+10
Bismark

Landscape Texturing in Divinity II

- Looked great on the ground

Landscape Texturing in Divinity II

- Lower quality when seen from the sky

Visible Tiling

Need more dramatic detail

Divinity II Approach

- Masked blending
  - Low resolution masks
  - Tiling detail textures

DC Requirements

- Needed something more visually unique
  - Less tiling when seen from the sky
  - More natural detail: erosion, rivers, cracks, …
- Blending approach not feasible
  - Needs many layers for variety
  - Needs high resolution source art to ensure uniqueness
    - Masks: height, slope, …
    - Textures: diffuse & normal
  - Needs complex blending modes

DC Prototype Solution

- Use uniquely baked texture data on the landscape
  - 64 tiles, 2048x2048 each
  - Equivalent to a 16384 x 16384 texture
  - Textures baked during production
- Statically load all the data
  - 512 MB (DXTn)
- Big, but the artists were happy ☺


Capped: 17 (58 ms) - Real: 18 (55 ms)
The Zealots
The Imperials
35 (3460)
0 / 550
New tutorial available: Movement and Selection
114

Prototype Issues

- Too much runtime memory use
  - 512MB per level landscape
- Too much disc storage
  - 512MB per level landscape

Solution 1: Back to layer blending?

Layer Blending

- Recreate production pipeline in shader:
  - 36 source textures per map
    - 18 diff + 18 norm
    - 1k x 1k: ~ 50 MB total (DXT)
  - 8 mask textures
    - height, erosion(s), normals, roads, walkable
    - 16k x 16k: ~1,3GB total
- Assuming 8k masks: still 391 MB run time memory use
- Complex blending shader accessing many textures

Static Loading

- 2 textures
  - diffuse + normal
  - 512MB run-time meory use (DXT)
- Trivial shader

Solution 2: Texture Streaming?

- A.k.a. don’t load it all at once
- Requirements
  - Low rendering overhead
  - Low memory footprint
  - Low disk storage
  - Future scalability

Solutions overview

Prototype

- Trivial shader accessing 2 textures
- 512 MB memory at run-time
- 512 MB memory on disc

Layer Blending

- Complex blending shader accessing many textures
- 360 MB memory at run-time
- 360 MB memory on disc

Streaming Final

- Relatively simple shader accessing 3 textures
- 60MB ~ 120MB run time memory use (depends on cache)
- ~130MB memory on disc

Texture Streaming

Charles Hollemeersch - Graphine

Background

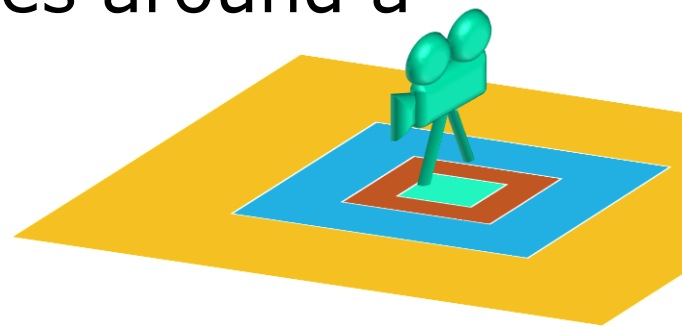
- Mipmap based
- Clipmapping
- Virtual texturing
- Dedicated hardware

Mipmap Based Streaming

- Stream textures one mipmap level at a time
- + Easy to implement
- + No special shader needed
- - Bad granularity
  - Big tiles
    - Seconds, not ms to process for highest mips
    - Memory not efficiently allocated maybe only small area needed
  - Small tiles
    - Batch overhead of splitting geometry

Clip Mapping

- Load fixed resolution rectangles around a certain point
  - + Supports large textures
  - + No need to split geometry
  - - Need roughly planar UVs
  - - Special set-up in shader
  - - Streaming gets more compicated



Clip Mapping in Games

- Difficult to decide a single high resolution area to load

Which area should we load at the highest resolution?

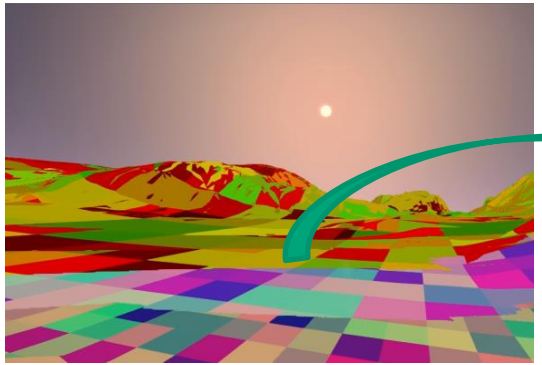
Clip Mapping in Games: Example

Virtual Texturing

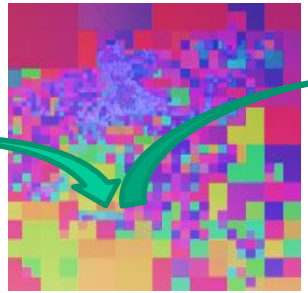
- Stream arbitrary set of tiles based on actual visibility
- + WYSIWYL (a.k.a. what you see is what you load)
- + No special geometry or uv requirements
- + Tiles are small and can be processed fast
- - Most complex to implement
- - More complex shader

Virtual Texturing Overview

Calculated Tile Identifiers and Mip Levels

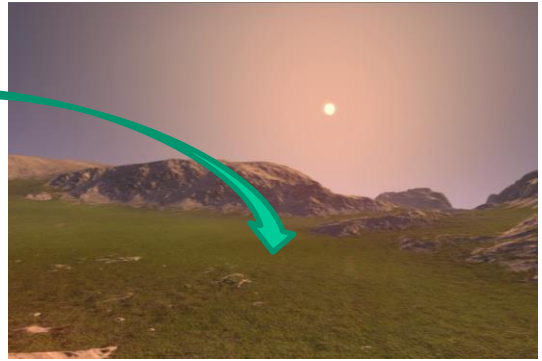


Translation Texture



Cache Texture

Rendered Result



Virtual Texturing in Games

- Has been used in a few high-end titles

Tiled Resources

- Hardware tiled-texture support
- Simlified texture filtering
- Announced at //Build/ 2013
- Confirmed on DirectX 11.2 and Xbox One
- PS4 Partially Resident Textures
- Can be used to implement flexible streaming
  - Clipmapping
  - Virtual Texturing
  - …

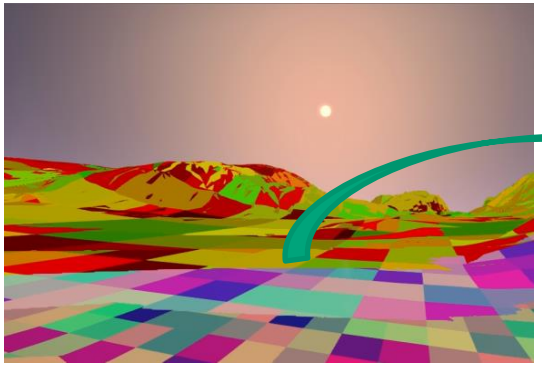
Tiled Resources II

- Still need streaming system to drive hardware
  - Decide what to load
  - Streaming data from disc
  - Compression
- More info in our //Build/ talk
  - http://msdn.microsoft.com/en-us/library/dn312084(v=vs.85).aspx
- Not used in Dragon Commander: DX9

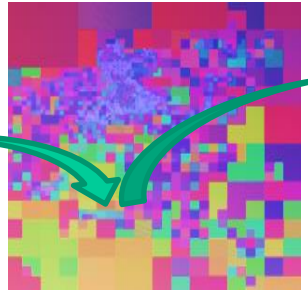
Tiled Resources: Hardware

Indirection step is now hardware accelerated

Calculated Tile Identifiers and Mip Levels

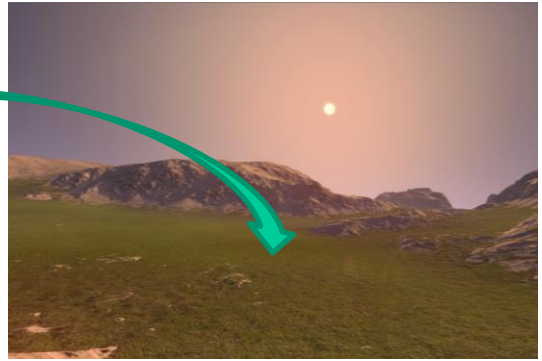


Translation Texture



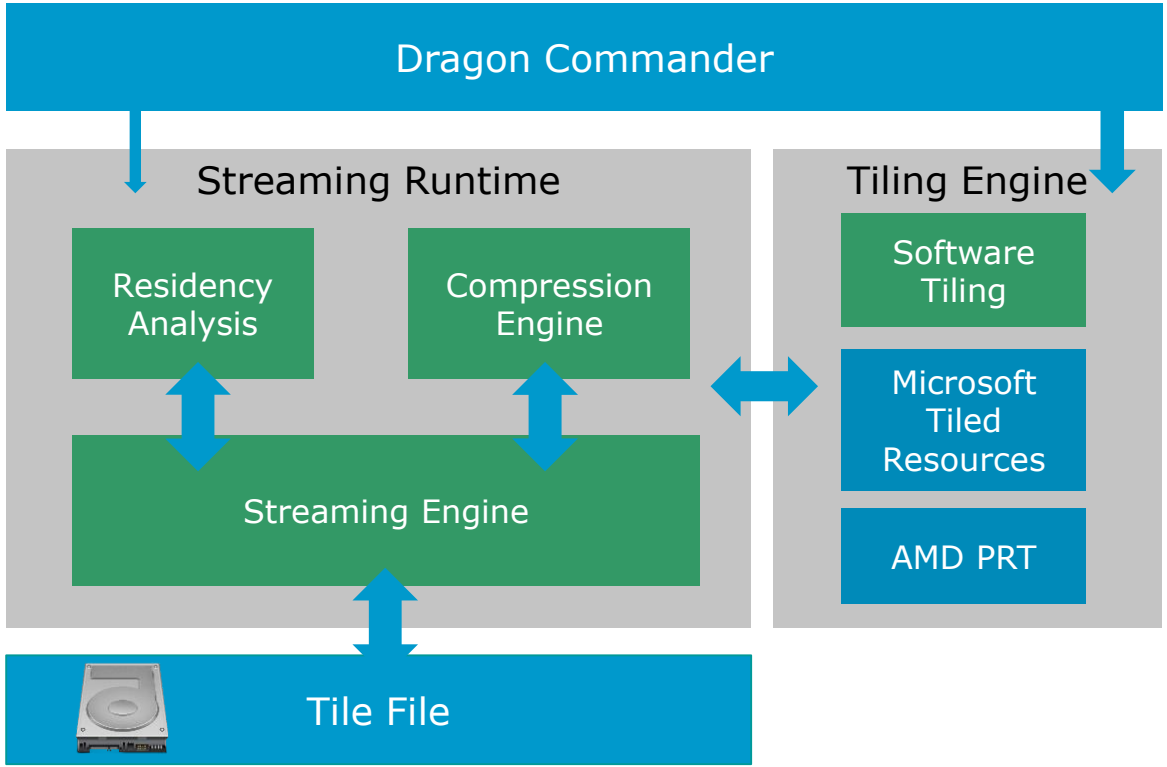
Cache Texture

Rendered Result



Virtual Texturing in Dragon Commander

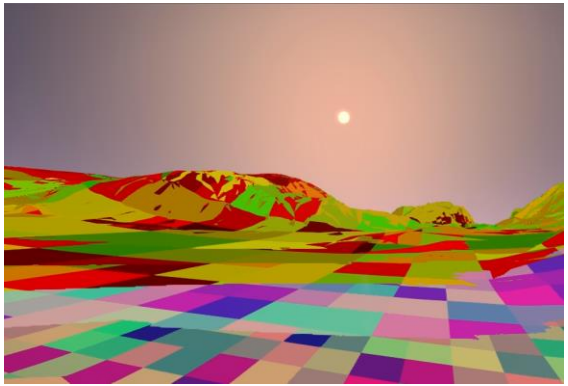
Streaming Runtime Overview



Software Tiling on DX9

- Difficult to get right in all cases
  - Tile border filtering
  - Mipmap filtering
- Been described in detail elsewhere
  - J.M.P. van Waveren - id Tech 5 Challenges
  - Charles Hollemeersch et al – Accelerating Virtual Texturing using CUDA

Residency Analysis in DC



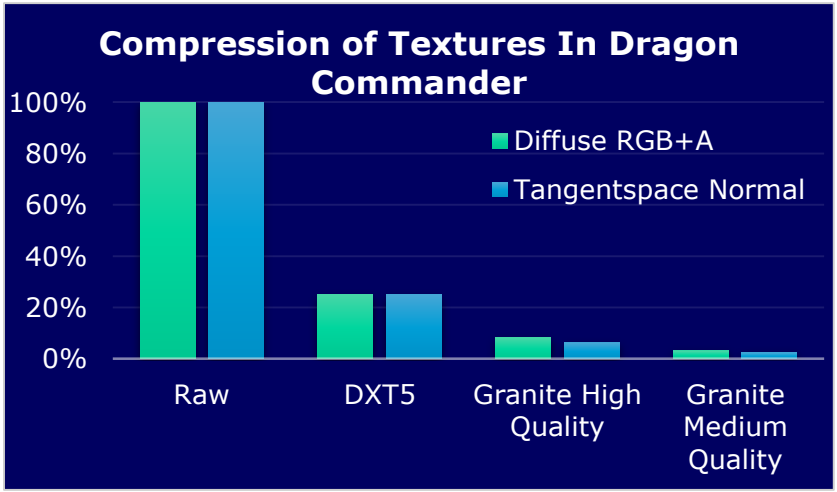
- Decide what tiles to load
  - Try not to load unnecessary tiles
    - Mipmap level never accessed
    - Occluded by other geometry
- Low resolution render of the scene
  - Asynchronous read back to the CPU for analysis
- Only render landscape
  - Main occluder in DC scenes

Need for compression

- DXTn compression: 512 MB/Level
  - DXT5: 16k x 16k Normals + mipmaps
  - DXT1: 16k x 16k Color + mipmaps
- 54 levels in game:
  - 28 Gigabyte
- We needed something better than plain DXTn on disc

Compression

- Graphine compression
  - quantized, block-based
  - predictive
  - adaptive: color + normals
  - support for alpha



- Supports fast transcoding from this format to DXT for optimal storage on GPU

Compression: Example

Compression: Example

fixed 6:1

quantisable ~20:1

Compression in Dragon Commander

- 4.6x compression over DXT (18x vs raw)
- Made the game ~ 1/3rd smaller
  - 6GB level textures
  - Ship on 2 DVDs
  - Download faster

VT Engine Integration

- VT textures are first class citizens
  - Fully integrated in engine resource system and shader node editor, not a special case shader
  - Can use VT on all objects if we want (RAGE style)
  - Currently only used on the landscape
- Integrated in tool chain
  - Compression (16k) adds 45 seconds

Multi-threading

- Asset loading thread
  - Regular assets load through this thread, used at load time
  - Main thread renders loading animations
  - Resource creation invoked on main thread (DX9)
- Texture streaming thread
  - Active at run-time streams compressed pages from disc
  - Mainly sleeping blocked on disc IO
- Texture transcoding thead
  - Convert tiles from compressed format to DXT

Memory use

- 128 MB of system cache memory
- 96 MB of GPU texture memory (-82%)
- Less on low-quality settings

Performance

- Fast even on low-end systems
  - Minimum spec GeForce 8800 (>96% PC Gamers)
- Transcoding of tiles takes up 10-15% of a single core
- Tiles usually available in 2-4 frames
  - Gracefully falls back to lower resolution if data is unavailable
  - No visible popping

Texture Streaming

Swen Vincke - Larian Studios

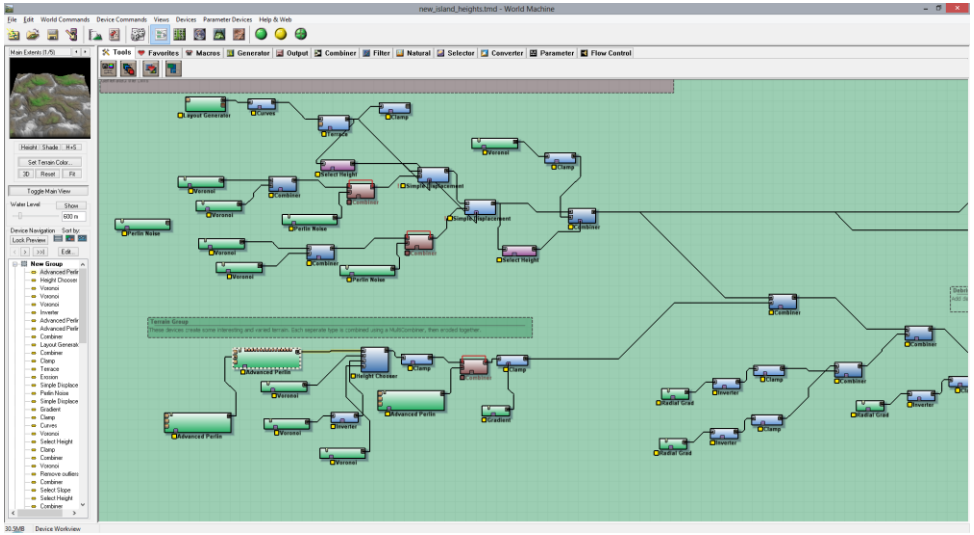
Conclusions on VT

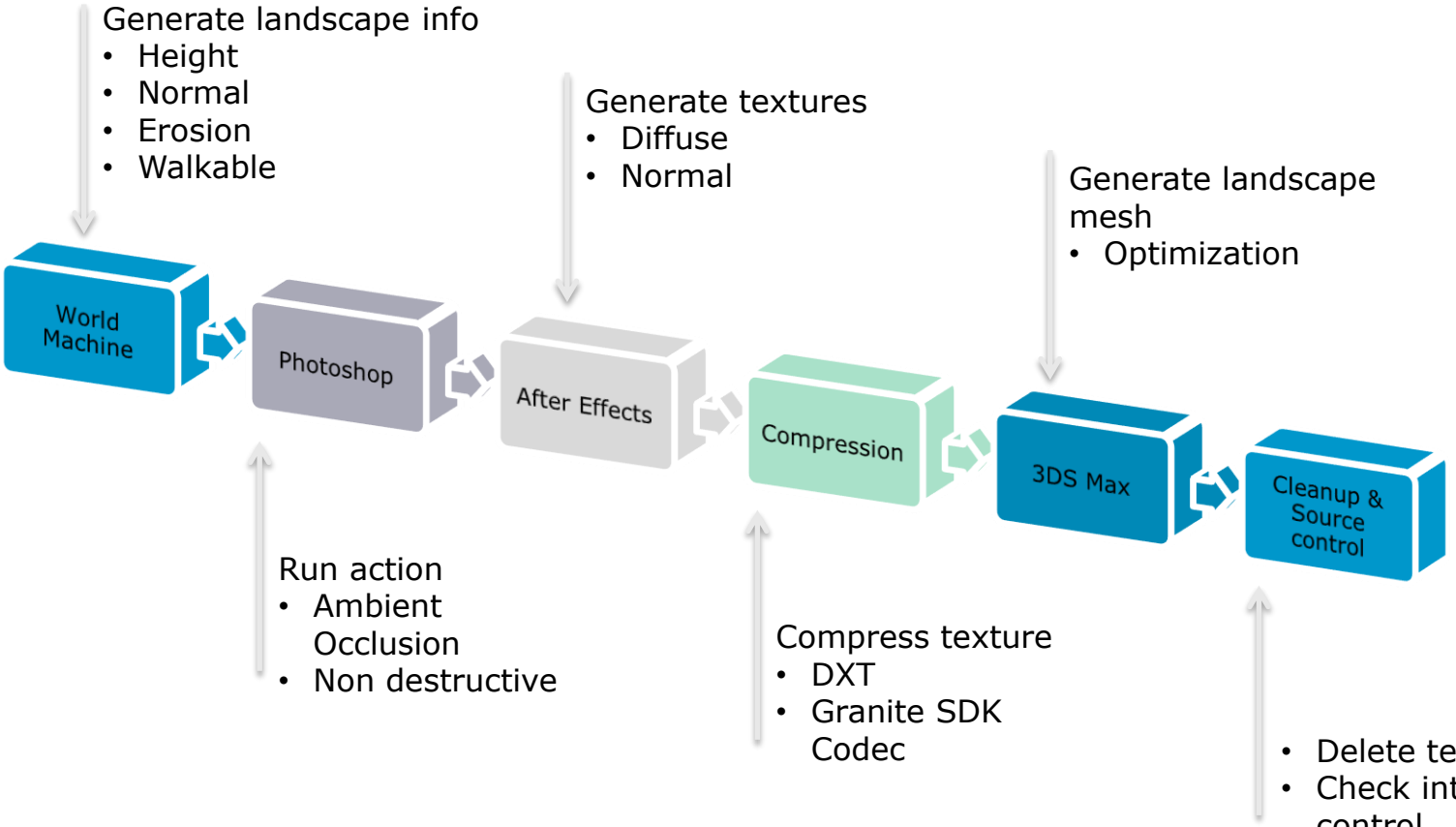
- Removed limit on texture detail
- Reduced storage per map
  - 28GB -> 6GB
  - ~1/3 size reduction overall
- Low runtime memory use
- Easy to integrate in production pipeline

Why we chose to use Granite SDK by Graphine

- Granite SDK: middleware package for advanced streaming and compression
- Easy to integrate
- Years of expertize on streaming and compression
- No codecs to write & optimize
- Future proof: Engine is tiled resources ready

Baking Workflow




Generate landscape info
• Height
• Normal
• Erosion
• Walkable
Generate textures
• Diffuse
• Normal
Generate landscape mesh
• Optimization
World Machine
Photoshop
After Effects
Compression
3DS Max
Cleanup & Source control
Run action
• Ambient Occlusion
• Non destructive
Compress texture
• DXT
• Granite SDK Codec
• Delete temp. data
• Check into source control

Build Server

- Automatically builds landscapes overnight
- Python script controlling all DCC apps
  - Maxscript
  - After effects java script
- Runs through: http://www.cruisecontrolnet.org/

Pipeline problems

- Iteration times to long
  - 2~3 hours for full resolution bakes
- Not enough local control
  - Wanted to add small stamps around bases, forests…
  - Fine control for road layout
- Integrate interactive painting tools in the game editor in the future

Conclusions

Final Conclusions

- Texture streaming allowed us to use less memory at run-time
- On-disc compression allowed us to pack more content in the same download size
- Using middleware gave us high-end technology quickly
- More interactive tools are needed

Questions ?