ADAPTIVE VIRTUAL TEXTURE RENDERING IN FARCRY4

Ka Chen
Technique Architect Ubisoft

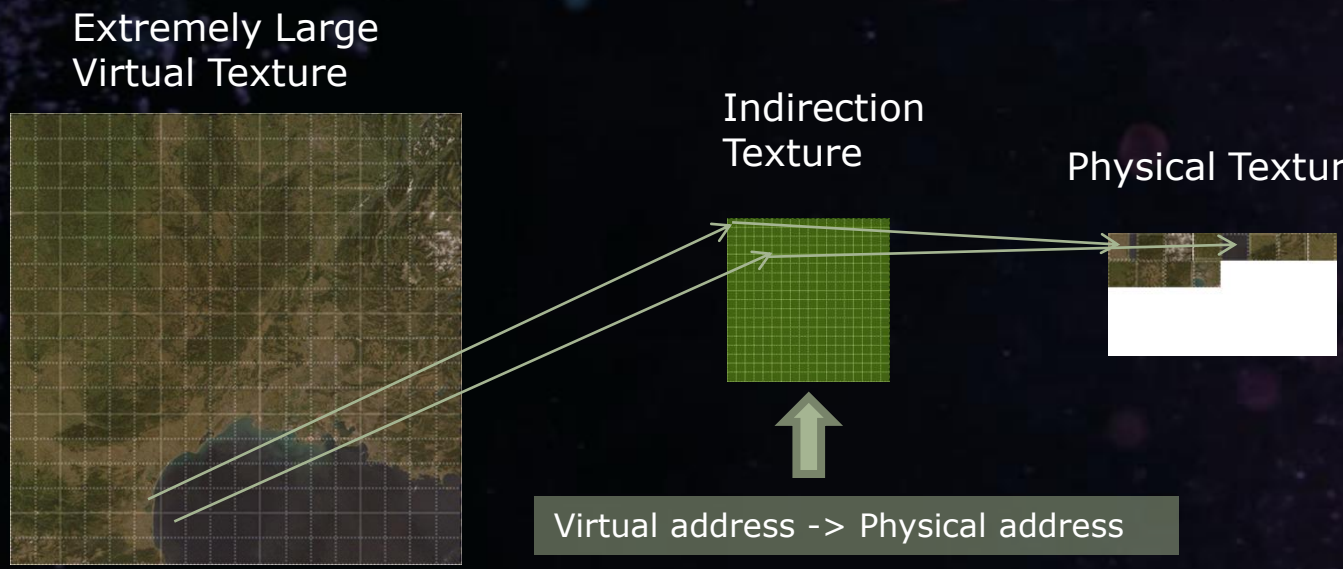
GAME DEVELOPERS CONFERENCE®
MOSCONE CENTER · SAN FRANCISCO, CA
MARCH 2-6, 2015 · EXPO: MARCH 4-6, 2015

Contents

1. Overview of Virtual Texture Techniques
2. Far Cry 4 Terrain
3. Adaptive Virtual Textures - AVT
4. Virtual Texture Rendering Challenges
5. Results, Performance, Summary

1. OVERVIEW OF VIRTUAL TEXTURE TECHNIQUES

Virtual Texturing



- Virtual texturing in games
  - Mega-Textures
  - Procedural Virtual Textures

Mega-Textures

- Developed by id Software for Rage (*Waveren 2013*)
- Texture data is stored on disk and streamed to memory as required
- Runtime determines the required tiles (pages) and requests them from disk
- Tiles are loaded to a tile cache (physical texture cache) and the page table (indirection texture) is updated

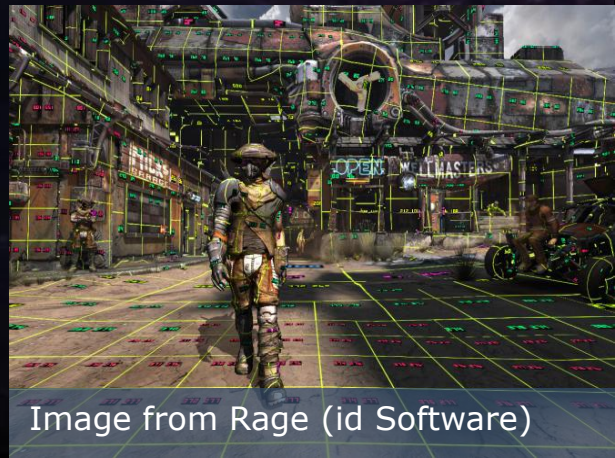


Image from Rage (id Software)

Procedural Virtual Textures

- Used by DICE in Frostbite Engine for Battle Field3 (Widmark 2012)
- Splats terrain rendering into virtual textures at runtime
  - No highly compressed virtual textures from disk
  - Direct render into virtual texture for missing pages
- Leverages frame-to-frame coherency to reduce terrain rendering cost
- Powerful GPU optimization for terrain rendering

2. FAR CRY 4 TERRAIN

Far Cry 4 Terrain

- Cross platform (PS4, Xbox1, PC, PS3, Xbox 360)
- Large world: 10 x 10 KM
- Far terrain(**Vista** terrain > 300 meters away):
  - Offline baked geometry and textures
- Near terrain:
  - Rendered from height-map
  - 4 detail material layers blended with a mask texture
  - Road and decals add unique detail
  - Target resolution: 10 texels / cm
  - Could use virtual textures

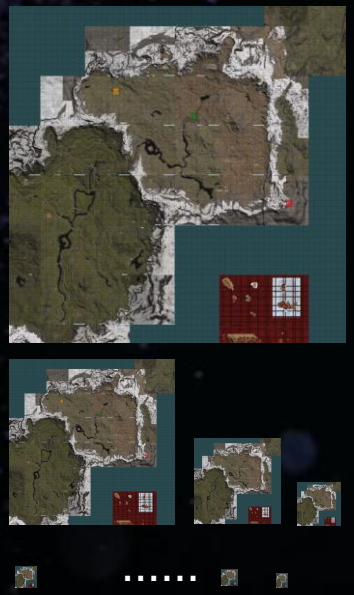
For next gen platforms we want to add a massive number of procedurally placed decals

- Simple deferred decals would be too expensive in this quantity
- So optimize by baking decals into a virtual texture at runtime

Procedural Virtual textures in Far Cry 4

Virtual texture

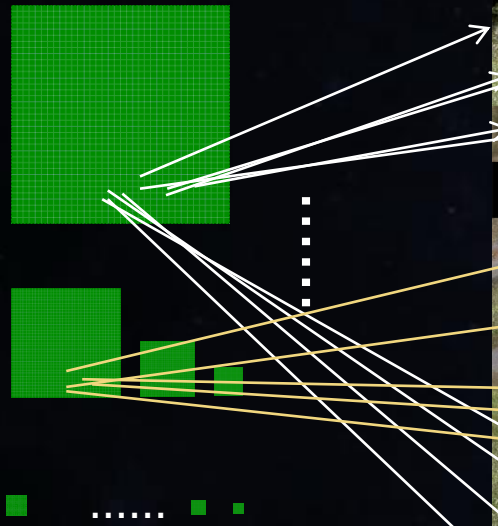
512K x 512K



11 Virtual texture mips

Indirection texture

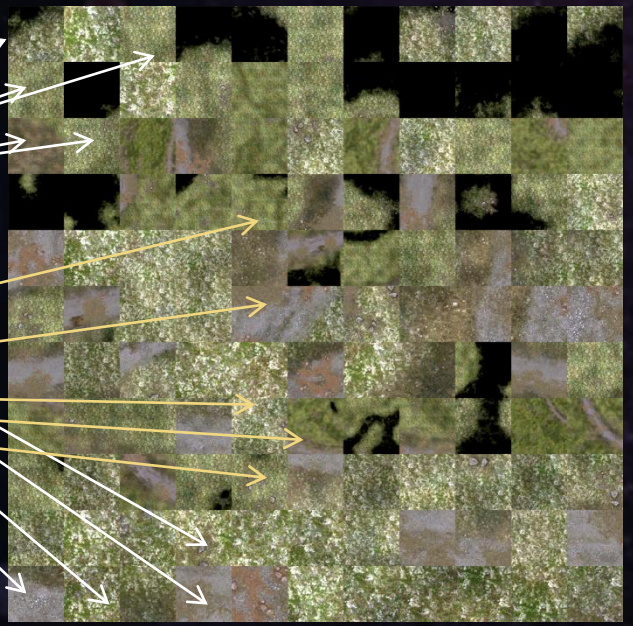
2K x 2K



11 indirection texture mips

Physical texture

9K x 9K



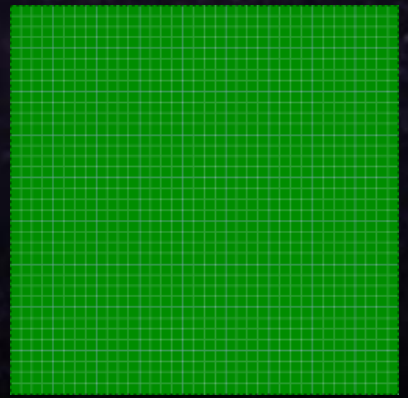
Indirection Texture Format

- Entry coordinate (x, y):
  - Each entry represents one virtual page
  - Entry coordinate = Virtual page coordinate / virtual page size
- Entry content format: 32 bit integer

PageOffsetX : 8	PageOffsetY : 8	Mip : 8	Debug : 8

  - PageOffsetX = Physical page U Coordinate / physical page size
  - PageOffsetY = Physical page V Coordinate / physical page size
  - Mip: Mip-map level of this page
  - Debug: used for debugging only
    - (for example saving a frame counter)

Indirection texture



Motivation

- With conventional virtual texturing
  - 512K x 512K Virtual Texture on 10 x 10 Km world

0.5 texel/cm resolution

10 texel/cm resolution

  - **10 million x 10 million Virtual Texture !**
- Another technique is required.

3. ADAPTIVE VIRTUAL TEXTURES

Adaptive Virtual Textures (AVT)

- Based on procedural virtual textures
- The 10x10KM world is divided into **64x64** meter sectors
- Near terrain sectors:
  - Allocate virtual images in the virtual texture
  - Nearer sectors : larger virtual images
    - 64K x 64K (64K / 64 meter = 10 texels/cm)
  - Farther sectors : smaller virtual images
    - 32K x 32K
    - 16K x 16K
    - …
    - 1K x 1K

Adaptive Virtual Textures (AVT)

- Allocate virtual images inside the virtual texture for all close sectors

Visualize the allocation of virtual images for close sectors inside virtual texture

- Each colored square represents one virtual image for each nearby sector

Camera frustum

2 sectors nearest the camera
64K x 64K virtual images

6 sectors further from camera
32K x 32K virtual images

Multiple sectors slightly further from camera
16K x 16K virtual images

Continue until all nearby sectors are allocated in the virtual texture


0
FarCry4 [Release PC]

Upscale virtual image size when camera moves closer

32K x 32K -> 64K x 64K

Downscale virtual image size when camera moves further

64K x 64K -> 32K x 32K

Upscale a Virtual Image

- We allocate a larger virtual image in the virtual texture and remove the old one
  - In this example, 32K x 32K -> 64K x 64K

Upscale a Virtual Image

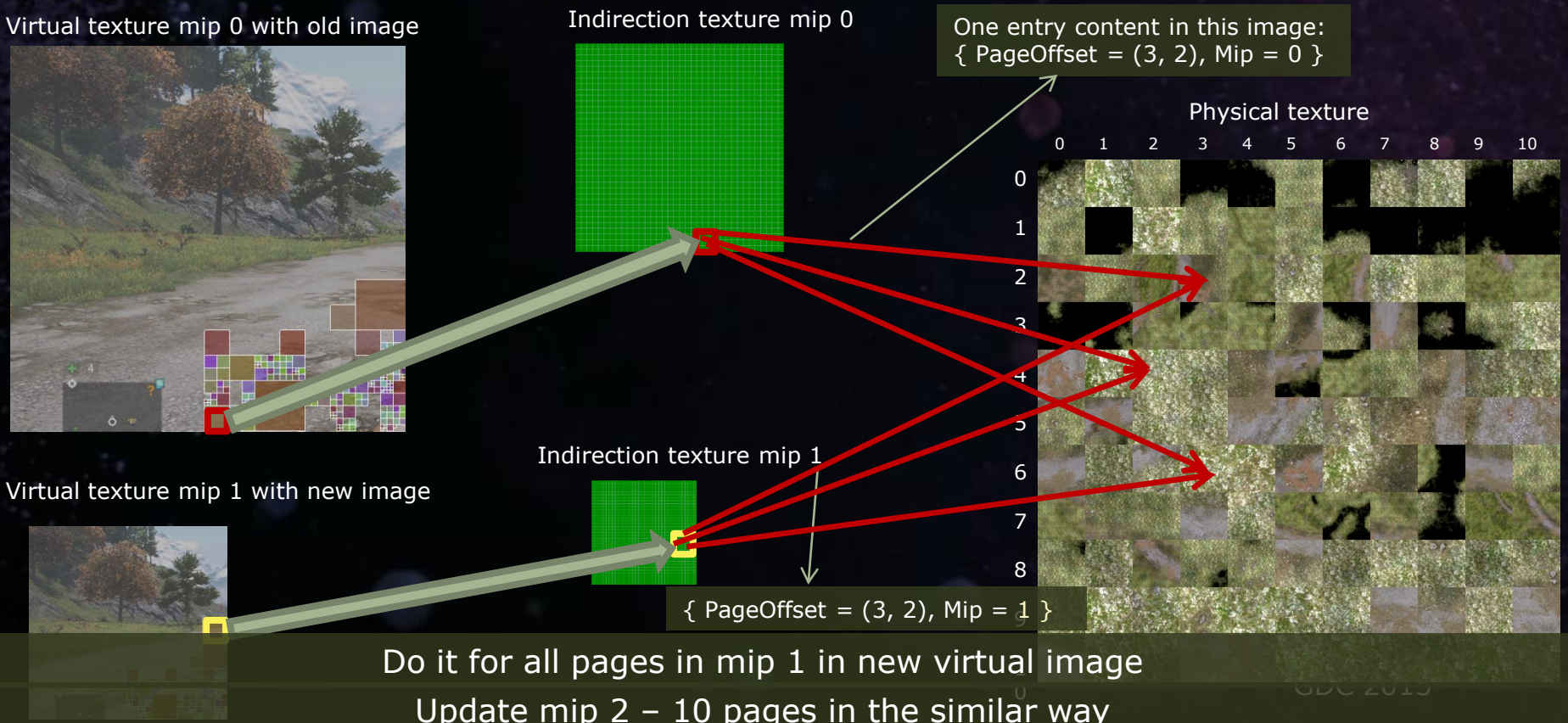
- Terrain material blending with additional decals
  - Already cached in our physical texture cache

Shift and reuse them!

- For all pages that are from mip 1 to mip 10, copy entries of indirection texture from old image to new image while shifting up 1 mip

Upscale virtual image of sector 32K -> 64 K

- Update all mip 1 entries in indirection texture for new virtual image

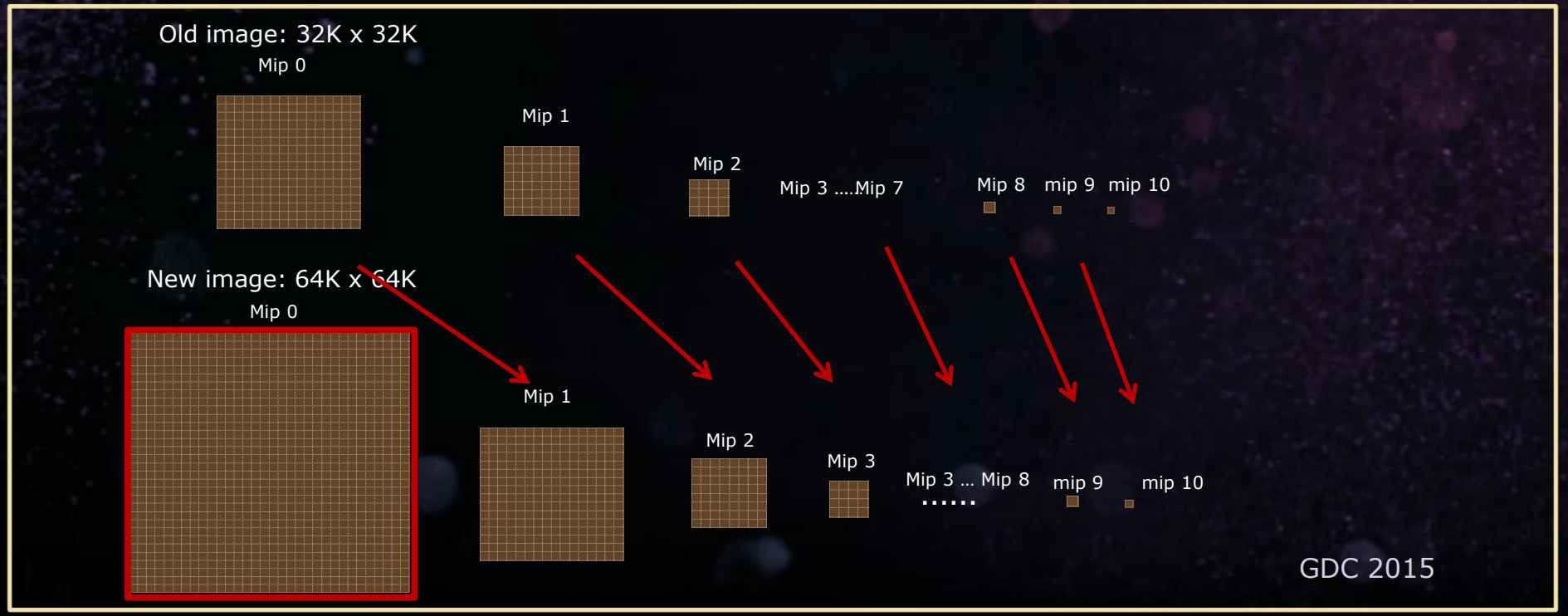


Do it for all pages in mip 1 in new virtual image

Update mip 2 – 10 pages in the similar way

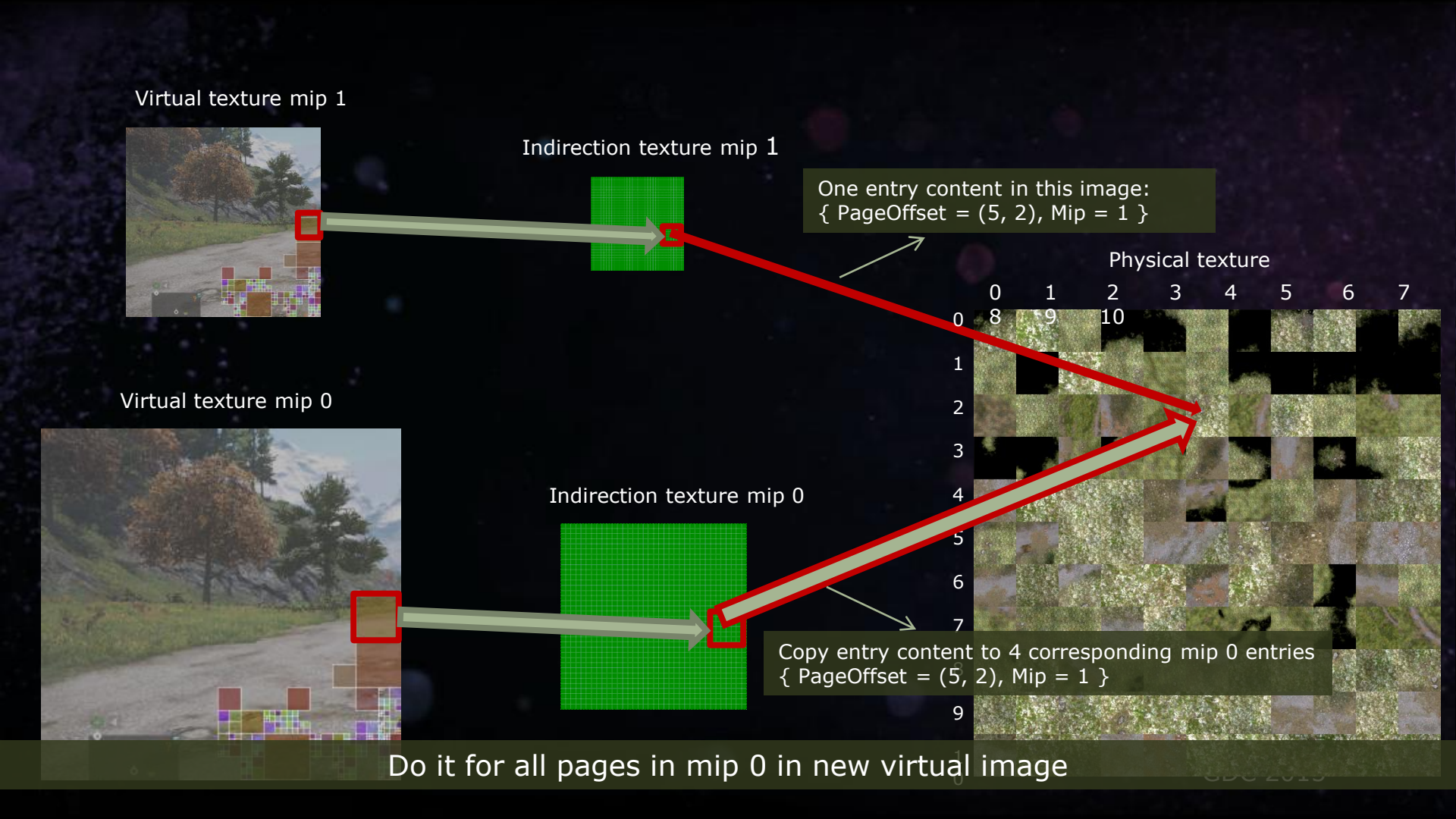
Update mip 0 pages in indirection texture

- Need to handle mip 0 pages
  - They haven't been rendered in the old image



Update mip 0 pages

- 4 mip 0 pages have 1 corresponding mip 1 page
  - Temporarily map to lower mip page
    - Images appear blurred in this frame
    - Will become sharper after correctly updated.


Virtual texture mip 1
Indirection texture mip 1
One entry content in this image:
{ PageOffset = (5, 2), Mip = 1 }
Physical texture
0 1 2 3 4 5 6 7
0 8 9 10
1
2
3
4
5
6
7
9
Virtual texture mip 0
Indirection texture mip 0
Copy entry content to 4 corresponding mip 0 entries
{ PageOffset = (5, 2), Mip = 1 }
Do it for all pages in mip 0 in new virtual image

Update mip 0 pages in indirection texture

- Copy indirection texture entries content

Indirection texture mip 0

- Physical UV is calculated according to the mip in the entry

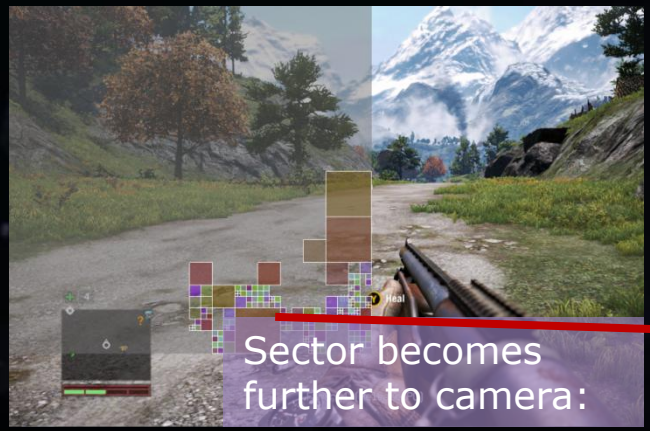
Code snippet in terrain pixel shader

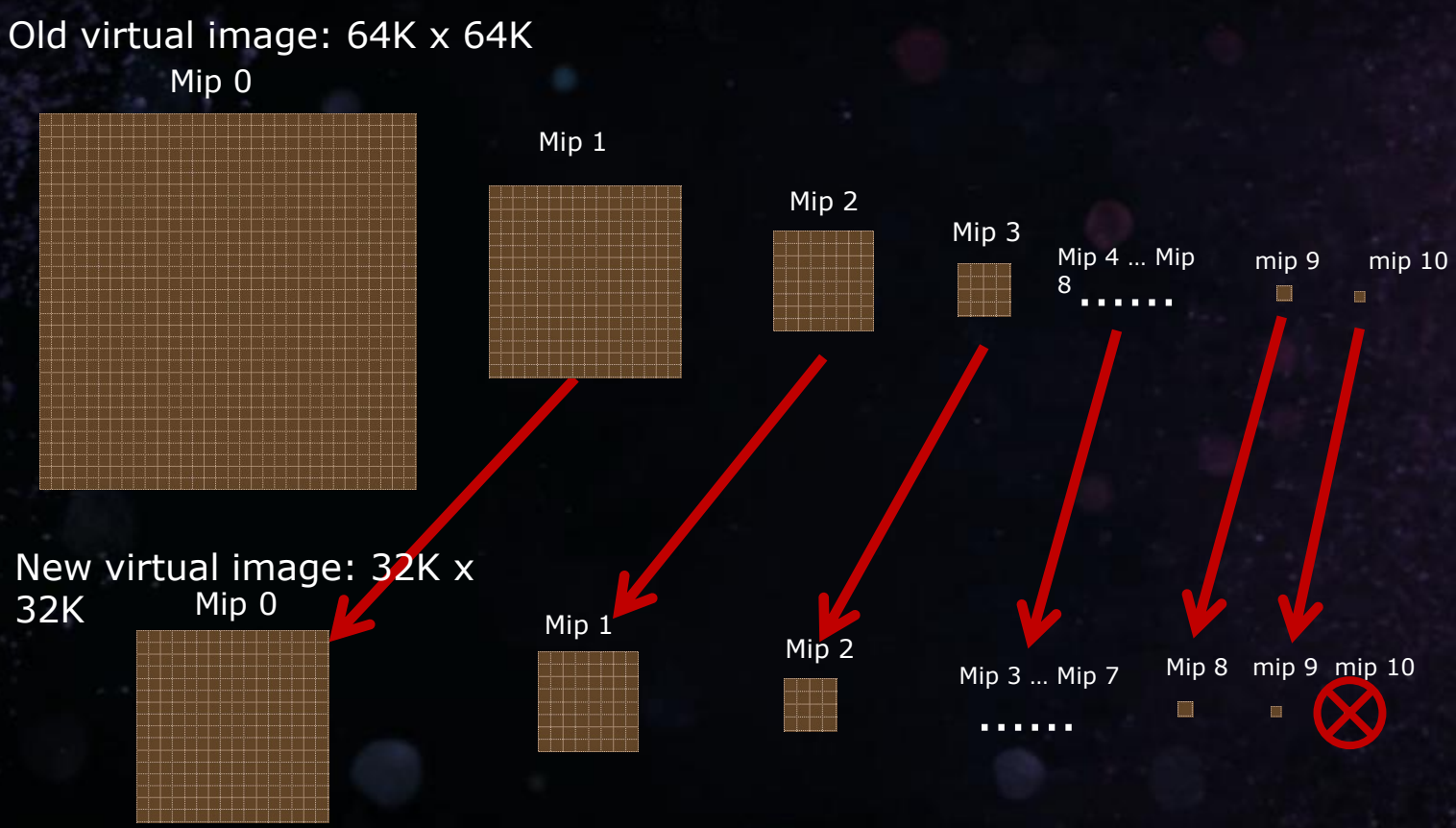
```
scale = (virtual texture size / physical texture size) >> mip
bias = physical page offset – virtual page offset * scale
physical uv = virtual uv * scale + bias
```

- Physical page offset and mip are the entry content

Downscale a virtual image

- We allocate a smaller virtual image in the virtual texture and remove the old one
  - In this example, 64K x 64K -> 32K x 32K
- Reverse the steps of upscaling virtual image




Old virtual image: 64K x 64K
Mip 0
Mip 1
Mip 2
Mip 3
Mip 4 … Mip 8
mip 9
mip 10
New virtual image: 32K x 32K
Mip 0
Mip 1
Mip 2
Mip 3 … Mip 7
Mip 8
mip 9
mip 10

4. Virtual Texture Rendering Challenges

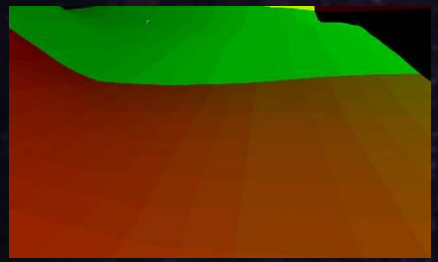
Reduce memory for virtual page id buffer

- Strategy
  - Output page IDs and MIP levels
    - To a Read Write Buffer
    - During G-Buffer pass
- Buffer format (32 bits)

  - PageID XY = Virtual UV / Virtual Page Size
  - Size = log2 (Virtual Image Size)
- Buffer size: 1/8 of resolution of MRTs

Game Scene

1/8 x 1/8 PageID RW Buffer



Limit per frame rendering cost

- Caching virtual textures can be slow
  - Camera moves fast -> need to render a lot of pages
    - When driving a vehicle
    - Flying

Limit per frame rendering cost

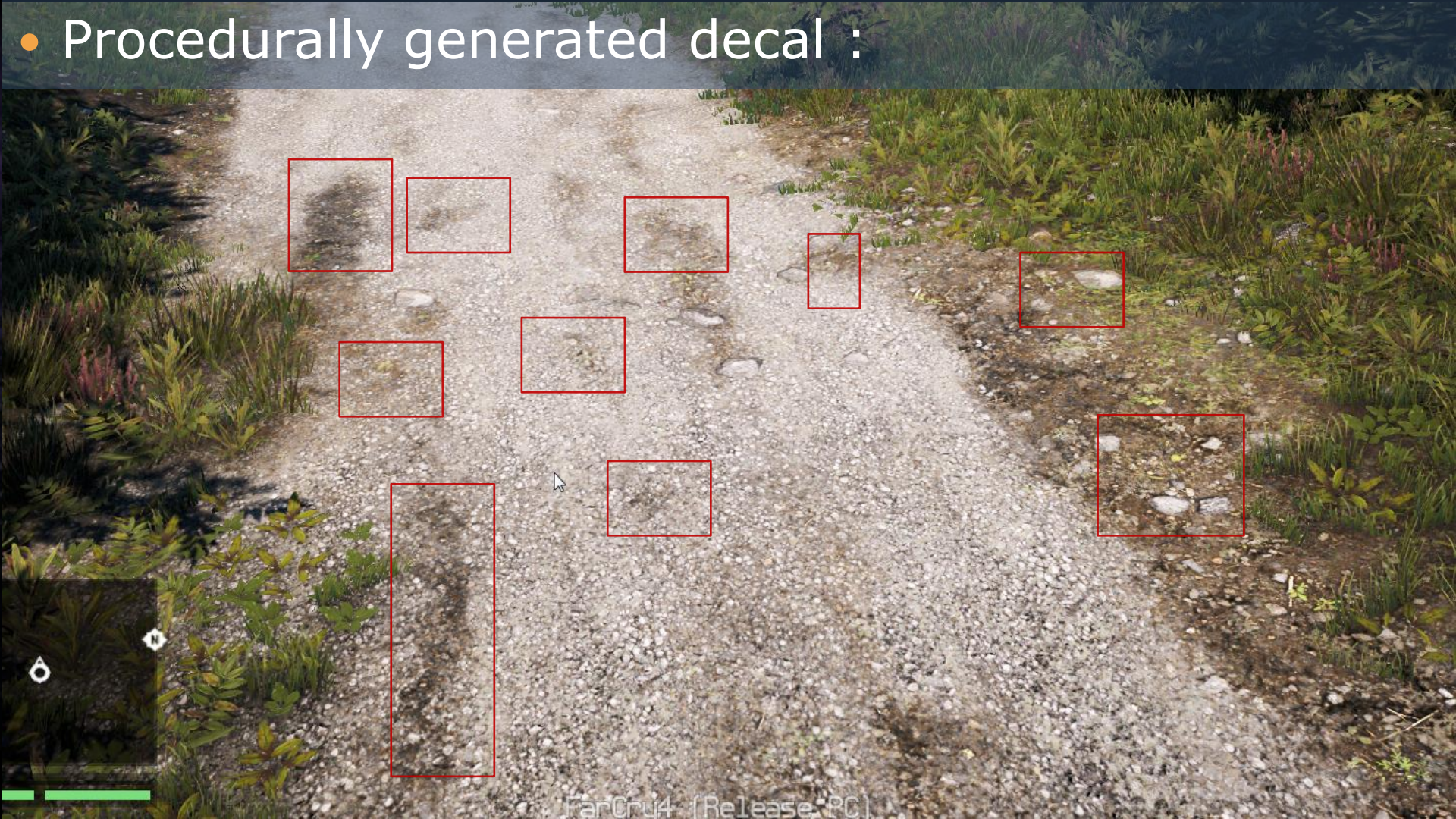
- Solution：Distributed rendering
  - Sort required pages by mip levels
    - low -> high
  - Distribute the rendering of pages into multiple frames


Frame N
Frame N+1
Frame N+2
Frame N+3

Generate massive number of decals

- Artists want to generate many decals efficiently
- Solution: Procedural content generator
  - Automatically attach decals to specified objects
    - Leaves, stones, roots under trees
    - Cracks, dirt on road
    - Much more…

- Procedurally generated decal :

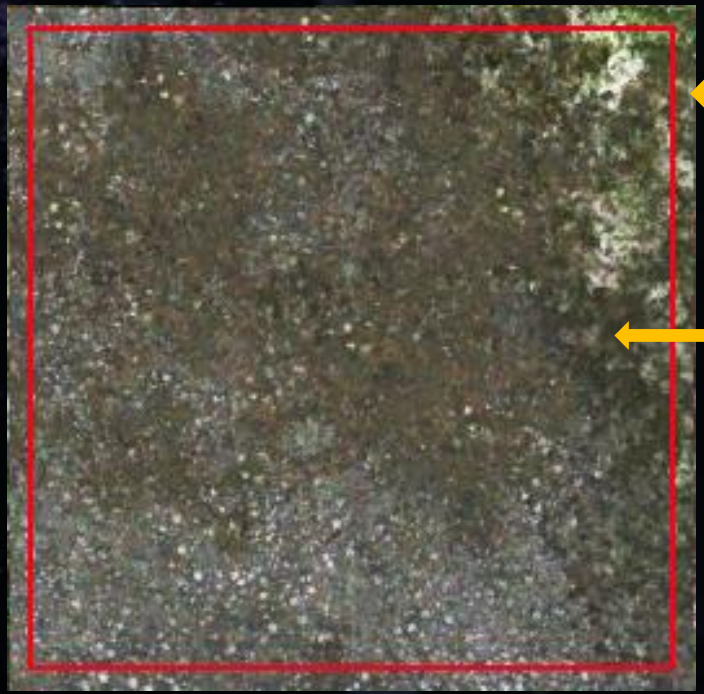


- Procedurally generated decal :

Attach to other procedurally generated content
For example generate fallen leaves under trees

Anisotropic Filtering

- Support 8x anisotropic in Far Cry 4
  - Texels in neighbor pages are not adjacent in world space
    - This could cause color bleeding
- Solution: Add 4 texels border to physical texture pages
  - Physical texture page: 264 x 264
  - Render page with 264 x 264 viewport



4 texels border
Enlarge viewport 264 x 264

256 * 256 original page content

- Physical texture page: 264 * 264
- Can support 8x anisotropic filtering

Support trilinear filtering

- Only bilinear filtering
  - See seams where mip level is changing
- Far Cry 4 Solution: software trilinear filtering
  - Fetch virtual textures twice with mip(x) and mip(x+1)
  - Calculate linear blending of fetched colors in shader
- Other solution: hardware trilinear filtering
  - Create ¼ size mip 1 cache
  - Render into mip 1 cache too
  - Hardware handle the blending between mips
  - 25% more cache memory

Bi-Linear

Anisotropic

Tri-Linear,
Anisotropic

5. RESULTS, PERFORMANCE, SUMMARY


0
FarCry4 (Release PC)
Enabled AVT


0
FarCry4 (Release PC)

Enabled AVT

Rendering Performance

Performance (PS4)	CPU (rendering thread)	GPU
Static scene (Cache primed)	0.2 ms Analyze PageID buffer	0.1 ms Write PageID buffer
Dynamic scene (Caching virtual textures)	0.5 – 1 ms Setup time for render	0.5 – 1 ms Render terrain and decals into virtual textures and compress to BC format textures

Memory Consumption

Memory	PageID Buffer	Indirection Texture	Physical Texture	Total
Memory	0.4 MB	16 MB	202 MB	220 MB

Summary

- Procedural virtual texture is a good fit for terrain rendering
- Using AVT we can increase the resolution of the results
- Great when drawing a massive number of decals on Far Cry 4

Thanks to:

- Far Cry 4 rendering and arts team
- GDC reviewer : Mark Cerny

References

1. *Software Virtual Textures (J.M.P. van Waveren 2013)*
   - http://mrelusive.com/publications/papers/Software-Virtual-Textures.pdf
2. *Advanced Virtual Texture Topics (Martin Mittring)*
   - http://developer.amd.com/wordpress/media/2012/10/S2008_Mittring_AdvVirtualTexTopics.pdf
3. *Terrain in Battlefield 3 (Mattias Widmark 2012)*
   - http://dice.se/wp-content/uploads/gdc12_terrain_in_battlefield3.pdf
4. *Virtual Texturing in Software and Hardware*
   - http://mrelusive.com/publications/presentations/2012_siggraph/Virtual_Texturing_in_Software_and_Hardware_final.pdf
5. *Virtual Texturing*
   - http://aaronm.nuclearglory.com/vt/VirtualTexturing-AC07808876.pdf

Thanks!

References

1. *Software Virtual Textures (J.M.P. van Waveren 2013)*
   - http://mrelusive.com/publications/papers/Software-Virtual-Textures.pdf
2. *Advanced Virtual Texture Topics (Martin Mittring)*
   - http://developer.amd.com/wordpress/media/2012/10/S2008_Mittring_AdvVirtualTexTopics.pdf
3. *Terrain in Battlefield 3 (Mattias Widmark 2012)*
   - http://dice.se/wp-content/uploads/gdc12_terrain_in_battlefield3.pdf
4. *Virtual Texturing in Software and Hardware*
   - http://mrelusive.com/publications/presentations/2012_siggraph/Virtual_Texturing_in_Software_and_Hardware_final.pdf
5. *Virtual Texturing*
   - http://aaronm.nuclearglory.com/vt/VirtualTexturing-AC07808876.pdf