Cloud Terrain System

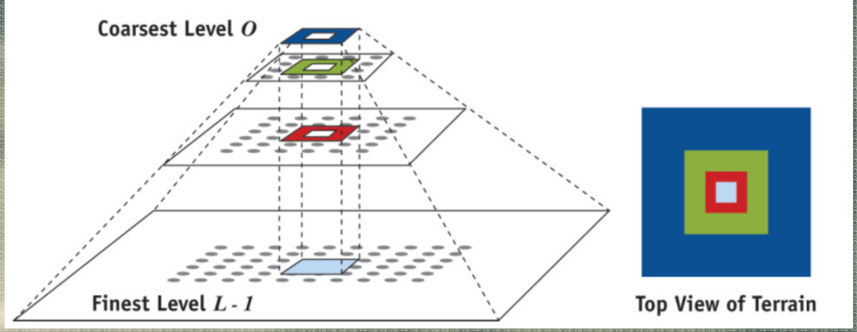
Rodrigo Marques Almeida da Silva

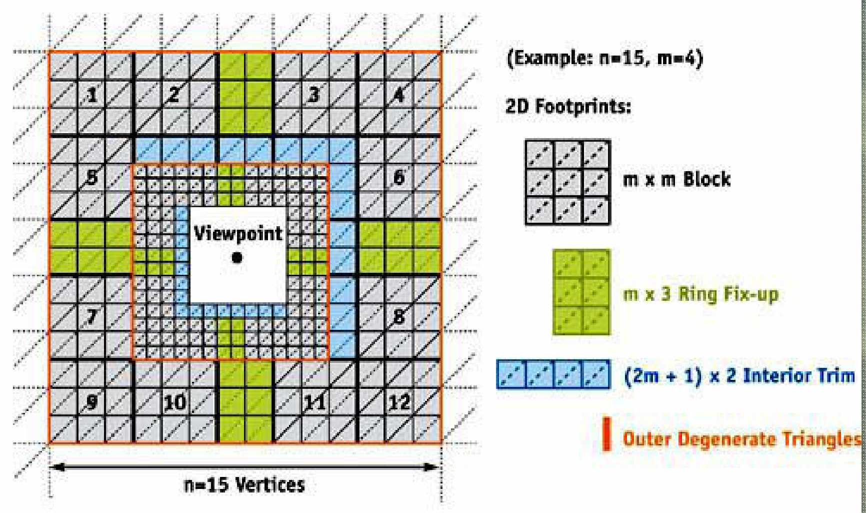
Index

- Terrain Rendering
  - GeoClipmap
  - Tiled GeoClipMap
    - Cracks Resolution
- Load Management
  - Local Load Balance
    - Streaming Core (HD → Memory → GPU Memory)
    - Dynamic Tile Substitution
    - Asyncron File Loading Core
- OGL Render Thread Issues
  - Thread Safe Solution
- Cloud
  - Cloud Tile Streaming
    - Server Load Balancing
    - Caching System
  - Cloud Manager
    - Monitoring
    - Adaptive Server Wakeup
- Test and Results
- Conclusion and Future Works
- Bibliography

Terrain Rendering

- HeightMap Based
- Full GPU Solution
- GeoClipMap
  - Texture Pyramid
  - Only 4 Mesh Patch
    - Low Memory Footprint
    - High Complexity for Rendering
      - A lot context changes
  - Compressed Texture
    - 85% of render time used to decompress texture
  - Pre-Processing Crack Resolution
  - Adaptive LoD
  - Fixed Tile Level of Detail

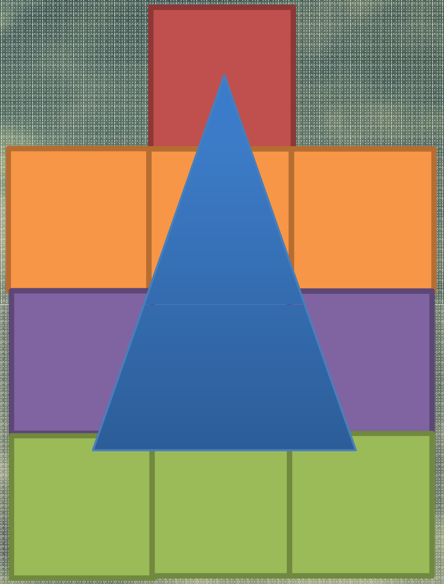




(Asirvatham& Hoppe, 2005)

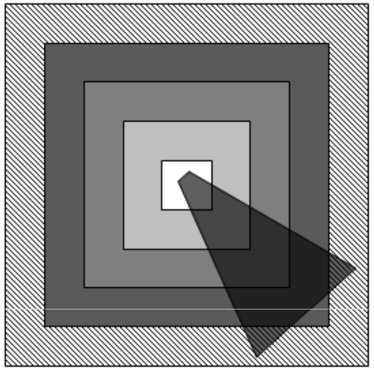
Tiled GeoClipMap

- Hybrid Solution
  - Use CPU for Memory Management and Culling
  - Use GPU for Rendering
    - Mesh, Texture
  - Dynamic Tile Level of Detail
- For each tile in the frustum
  - Select the best resolution based on the distance (the tile variance can be used also)
  - Render the Tile
  - Solve Cracks

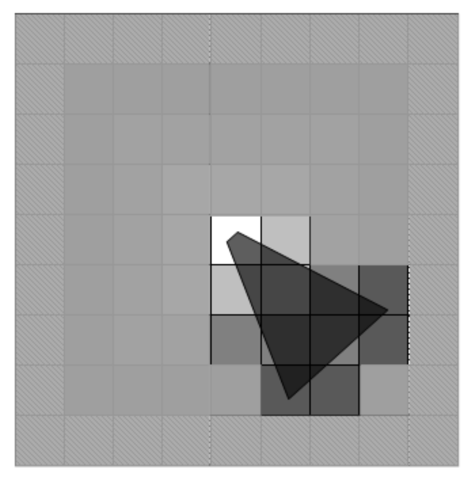


Tiled GeoClipMap

- Pyramid Texture
  - MipMap
- Pyramid Mesh
  - Power of 2 Mesh
    - Static Meshes
- Render Sequence
  - Run Culling
    - Run Frustum Culling (Fast)
  - Run Selector
    - Select the best level for the tile
  - Run Sorting
    - Same Level order by count desc
    - Simple Quicksort
  - Render

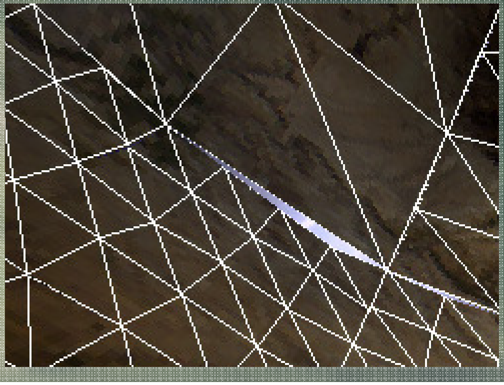


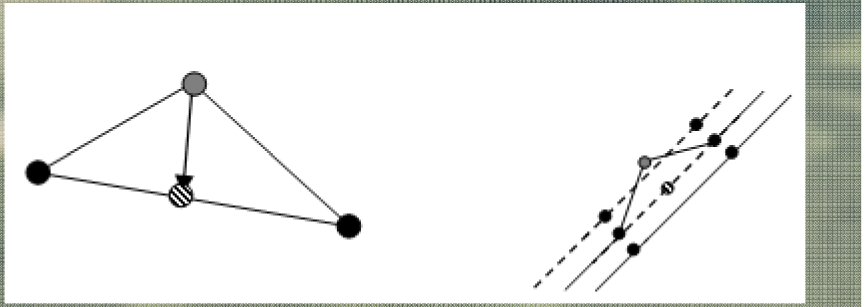
Nível	Dimensão	Cor
0	512 x 512	□
1	256 x 256	░
2	128 x 128	▒
3	64 x 64	▓
4	32 x 32	▨
5	16 x 16	▥
6	8 x 8	▩
7	4 x 4	▤
8	2 x 2	■

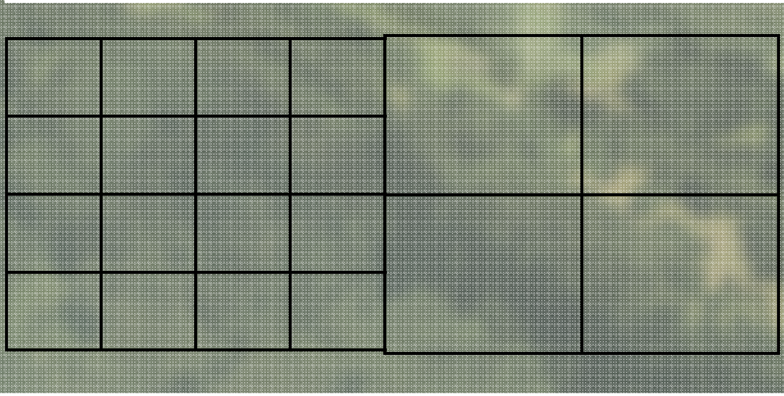


Tiled GeoClipMap

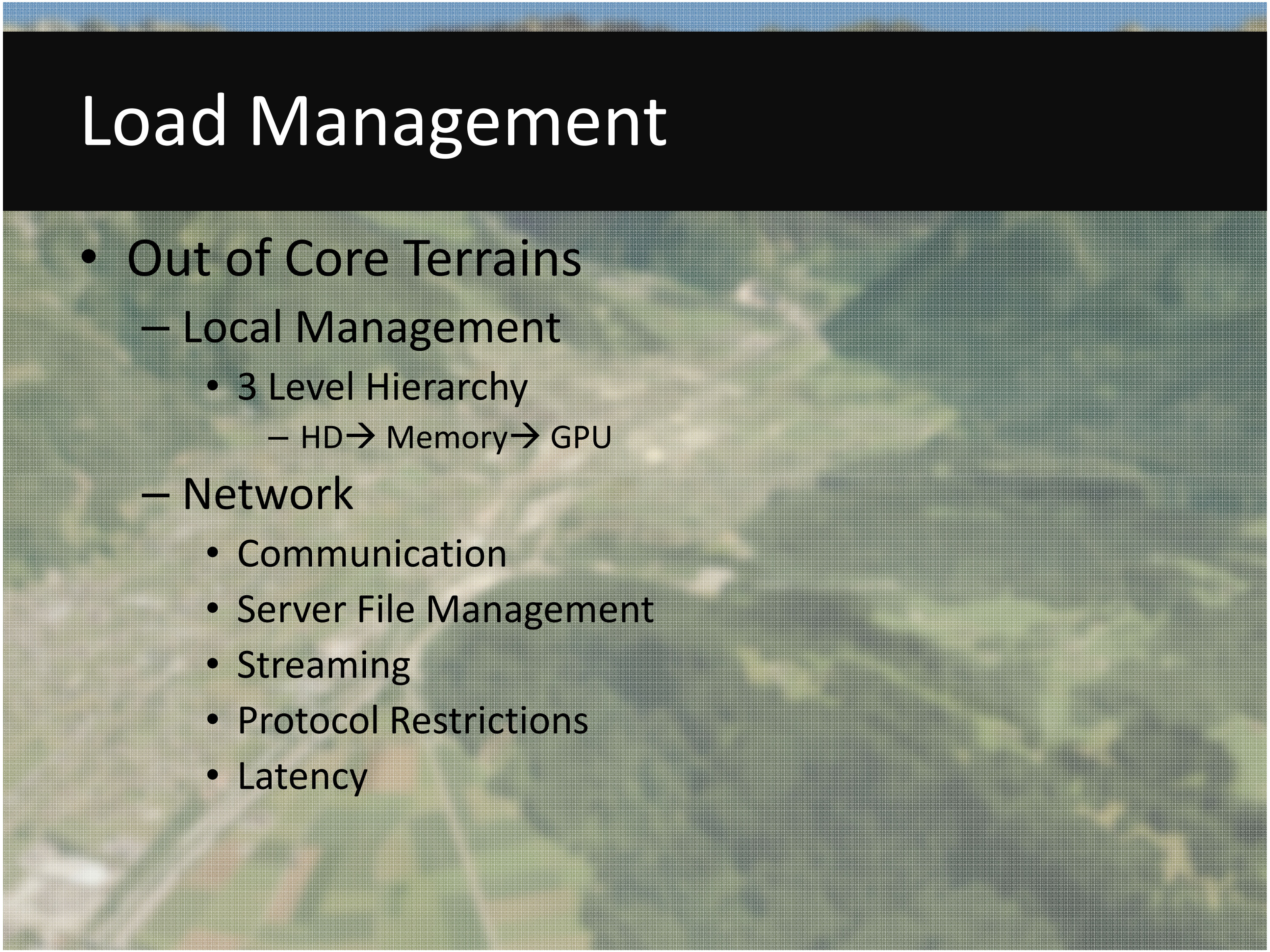
- Cracks
  - At a LoD Gap between 2 tiles
  - The tile with the major LoD must morph its edges to fit the another tile edges.
  - Used as a vertex shader
  - We can use Geometry Shaders to improve the LoD of the minor to the major
  - Work only with 1 level of difference







Load Management



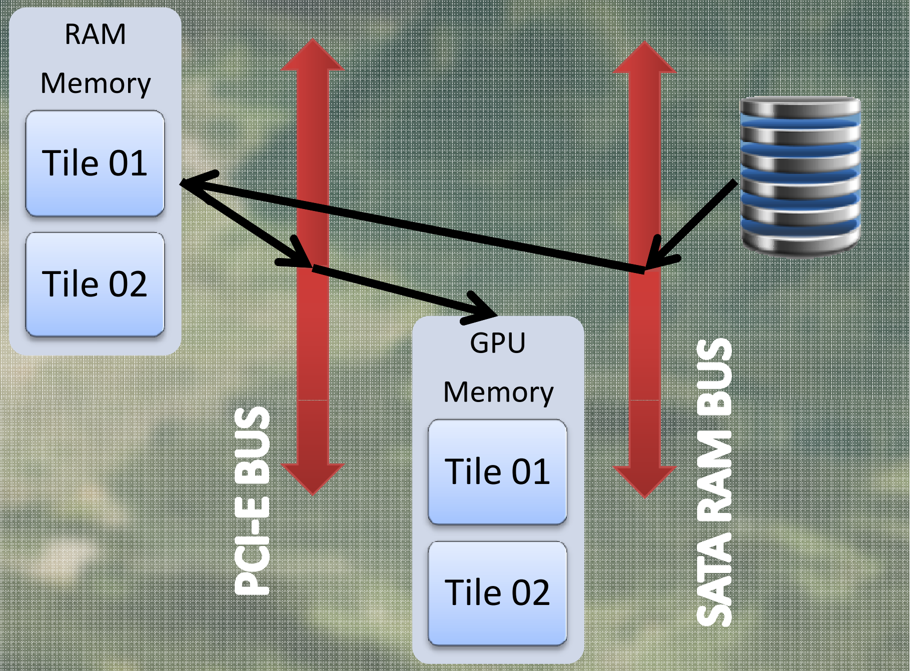
- Out of Core Terrains
  - Local Management
    - 3 Level Hierarchy
      - HD→ Memory→ GPU
  - Network
    - Communication
    - Server File Management
    - Streaming
    - Protocol Restrictions
    - Latency

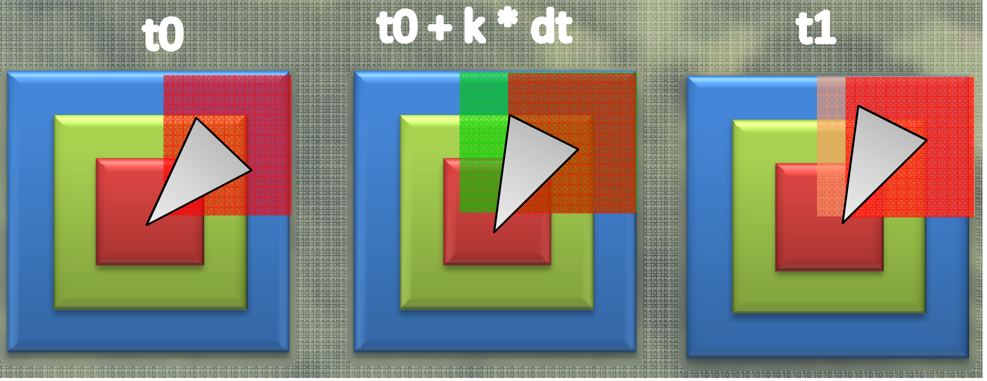
Load Management

- Dynamic Tile Substitution

Processor/% Processor Time/_Total
Memory/
  Available Bytes
  Cache Faults
  Page Faults
LogicalDisk/
  % Disk Read Time
  % Disk Write Time
  % Free Space
  % Idle Time
System
  Process
  Threads

- Asynchronous Load
  - IOCP
    - Creates a Thread To Handle Each Tile Block Load
- Tile Load Prediction
  - Interpolated Camera Move
    - Refresh New Tiles
    - Time Factor





Load Management

- Store File Format
  - Splitted Tiled File (v1)
    - Good
      - Web Services
      - Easy to copy
      - Hard to corrupt the whole terrain
    - Bad
      - Hard to Manipulate
      - Need a lot of File Handles
      - Long time to get the file in the memory
  - Single Huge File (v2)
    - Can use asynchronous callbacks
    - Raw File
      - Multilevel
        - » The Level i has a half tile size of Level i -1
      - Sequential
      - 4K Disk Format Blocks
      - Defragmented Disk

Title	Size	Offset	Value
Identifier	4	0	VTMF
Version	1	4	0x1
# Levels	1	5	6
Width	4	6	(int)
Height	4	10	(int)
TileW	2	14	(short)
TileH	2	16	(short)

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Size = # Comps * TileW * TileH

Offset(i) = i * Size + HeaderSize

i = Row * Width / TileW + Col

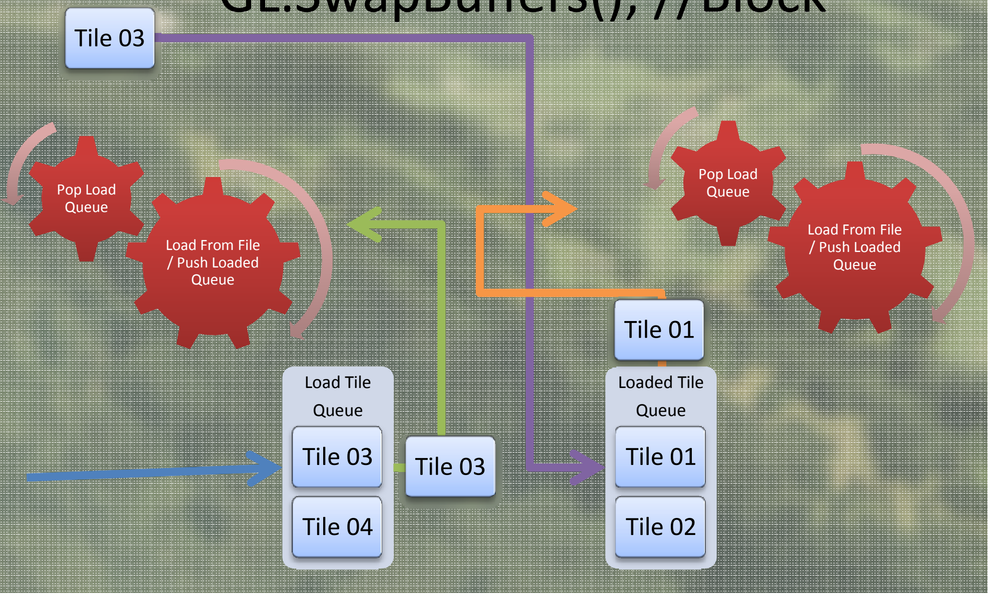
Row(i) = i div (Width / TileW)
Col(i) = i mod (Width / TileW)

OGL Render Thread Issues

- The GPU Loader must be in the Render Thread
  - Load Bottleneck
  - Sync
- Solution
  - Producer / Consumer Problem
    - Producer → File Loader
    - Consumer → Render Thread
  - OGL Render Thread Load After Render the Frame (until swap buffer)

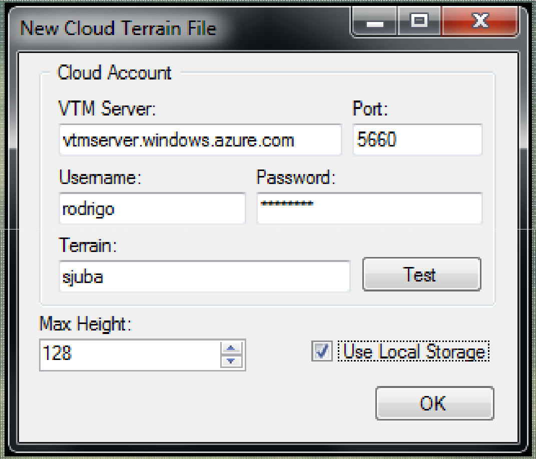
- Ex:
  - GL.MakeCurrent();
  - RenderScene();
  - Update();
  - While(Query(Rendering))
    - Consume();
  - GL.SwapBuffers(); //Block

BeginConditionalRenderNV



Cloud

- A Huge and Dynamic Computer Farm
  - Easy to add a machine
  - Large Storage System
  - Backup
  - No Maintenance
  - Pay as You Go
- Extends a Server Based Tile System
  - Dynamic Load Balance



New Cloud Terrain File

Cloud Account

VTM Server: vtmserver.windows.azure.com

Port: 5660

Username: rodrigo

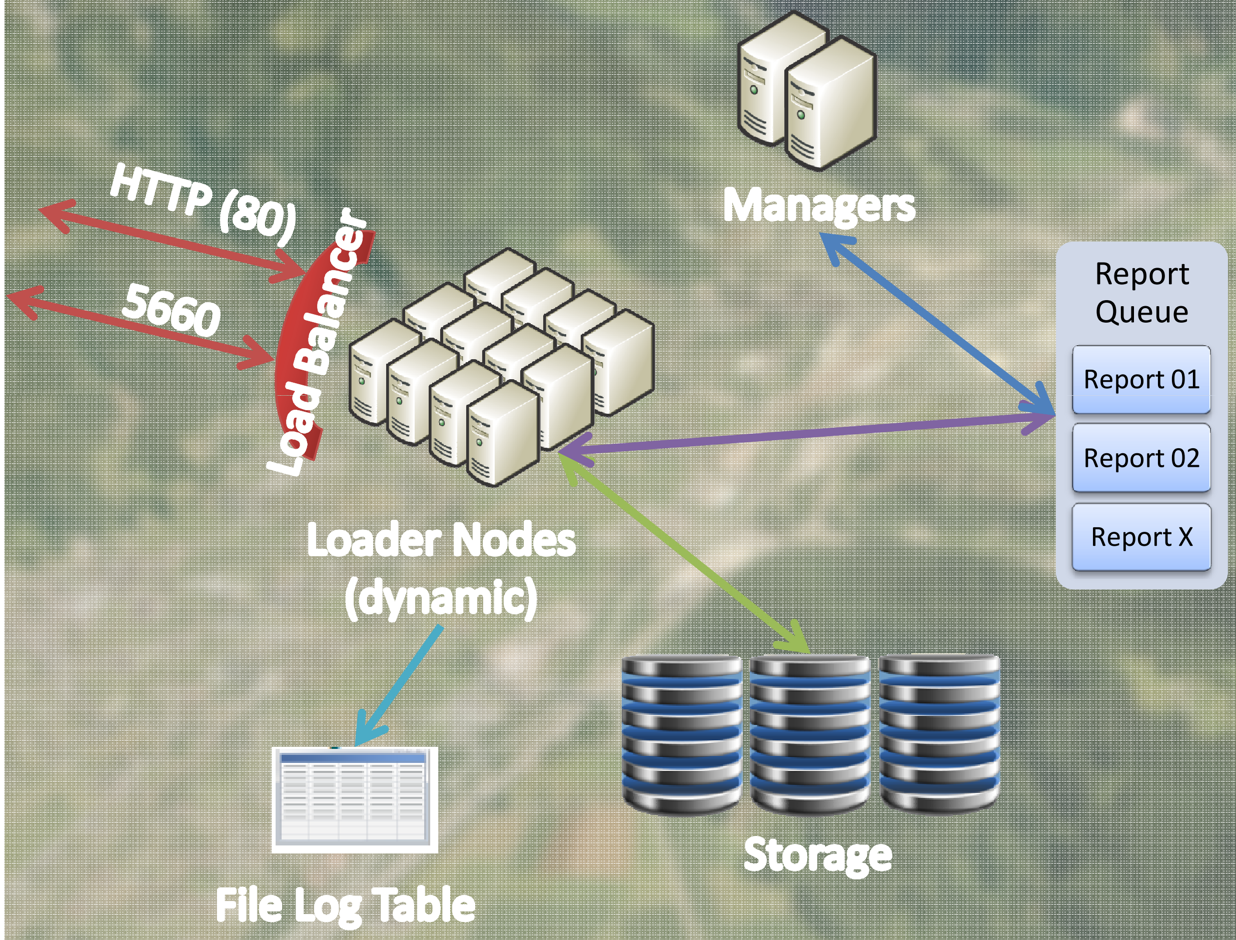
Password: *********

Terrain: sjuba

Max Height: 128

☑ Use Local Storage

Cloud Tile Streaming



- Windows Azure Cloud
  - Performance Analyser
  - Blobs
    - Cloud Drive
  - Tables
  - Queues

Cloud Tile Streaming

- The Loader Role Maps a Blob (Max 1TB) as an NTFS/CIFS Partition Drive
  - The Blob is not local, so, we create a blob cache on the local cloud machine (HD) to improve the performance and the Blob Access.
- When a tile is load, it remains on the machine RAM until:
  - Machine Memory is Low
    - Discard Tiles Based on the Last Access and Aging Data
  - After Loaded Data From HD, the Loader Compress it using GZip Algorithm.
  - Only Compressed Data is in Memory.
    - The server does not need to use the raw data.

Cloud Communication Protocol

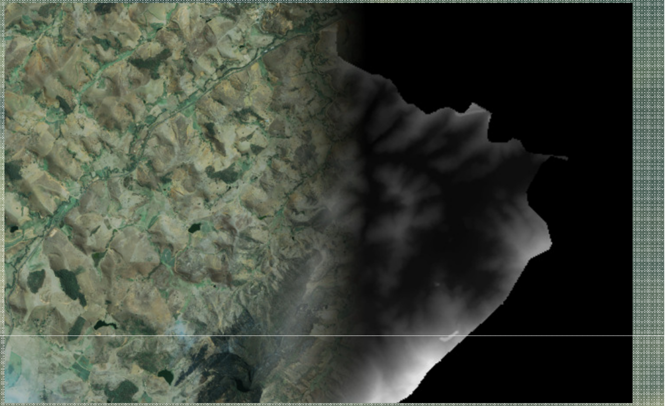
- HTTP interface for administration proposes
  - Not yet ready
- TCP Connection (5660)
  - Based on ISO 8583 Message Format – ASCII Encode
    - CODE+BITMAP+FIELDS
    - 0800/0810 for Connection and Authentication
      - Username, Password (cipher + base64), Max TCP Frame Size
    - 0600/0610 for Terrain Metadata Loading
    - 0200/0210/0202 for Tile Request
    - 0100/0110 for Update Server Information
  - To Send a Tile, we need to take care about TCP Frame Size
    - Split The Tile as a Set of TCP Frames and send it to the client
      - 0210 sends the data
      - 0202 is the client response for a sent data
        - » Remember, the connection can be lost
  - The Data is compressed by GZip
    - The Client need to merge the Frames and decompress it.
- The Load Balancer try to connect the client with its last used machine (to use the cache)

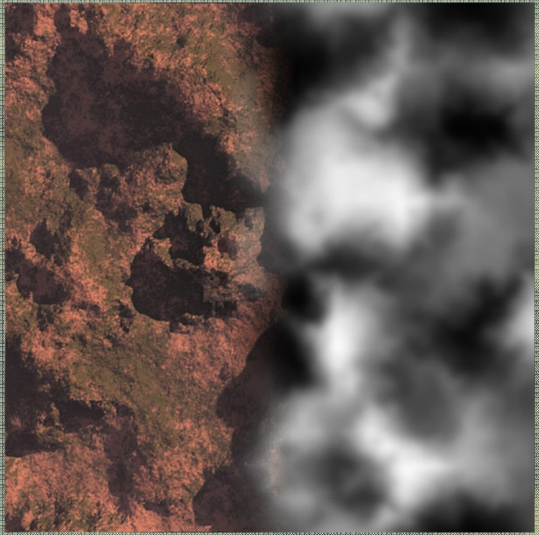
Cloud Manager

- The Manager Read the Reports From the Loaders
  - If all Loaders capacity are over 75% of power, then it raises another loader machine.
  - If all Loaders capacity are below 40% of power, then it shuts down a loader machine
  - After change the cloud capacity, it recalculates the Cloud Capacity and check the conditions again
    - The Cloud must have at least one Loader machine and 1 Manager machine
    - The Cloud Account has a maximum limit. In the test case, we can use 20 processors.

Test and Results

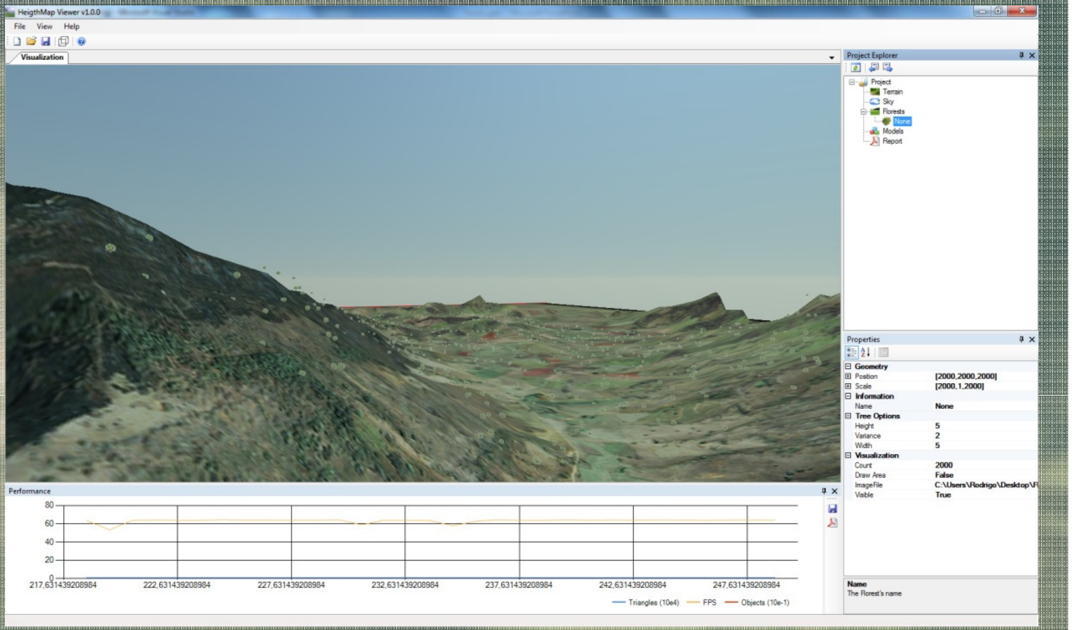
- HD Resolution (1280 x 720) 720p
- Data Sets
  - São José de Ubá Watershed – SET0
    - 16K x 8K File – 1.2 GBytes
      - Color @ 24 Bits (8 bits per channel)
      - Height Map @ 16 Bits
    - Tiled and Raw Version
  - Terragen Height Map – SET1
    - 4096 x 4096 File – 150 MBytes File
      - Color @ 24 Bits(8 bits per channel)
      - Height Map @ 8 bits
    - Raw Version
- Machine
  - Intel Quad Core @ 2.4 GHz
  - 8 Gbytes of RAM
  - NVidia 9800 GT
  - 5.0 Mbps Internet Link
- Cloud
  - Azure
  - 1 Small Instance for Manager
  - 3 Medium Instance for Loaders





Test and Results

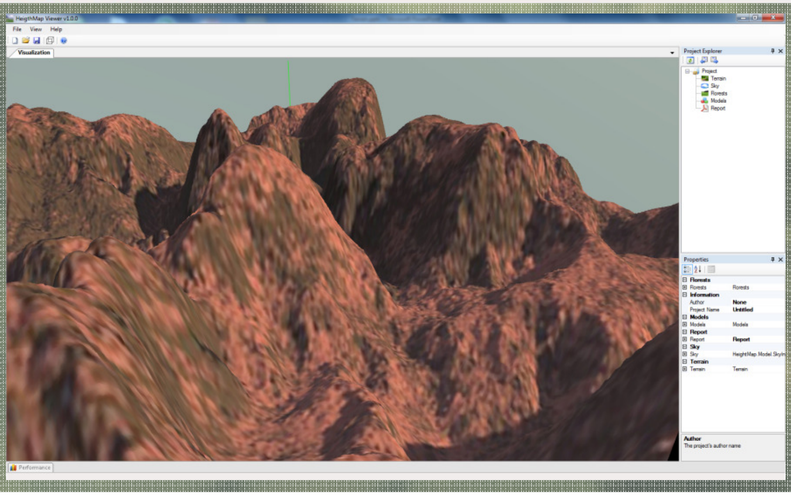
- SET0 – 5 min
  - FPS = (91.2,114.6,163.7)
  - Memory = 0.8 GBytes
  - Tile Lost = 1127



Geometry	
Position	[2000,2000,2000]
Scale	[2000,1,2000]
Information	
Name	None
Tree Options	
Height	5
Variance	2
Width	5
Visualization	
Count	2000
Draw Area	False
ImageFile	C:\Users\Rodrigo\Desktop\F
Visible	True

Name
The Forest's name

- SET1 – 5 min
  - FPS = (92.6,153.2,158.6)
  - Memory = 262 MBytes
  - Tile Lost = 0



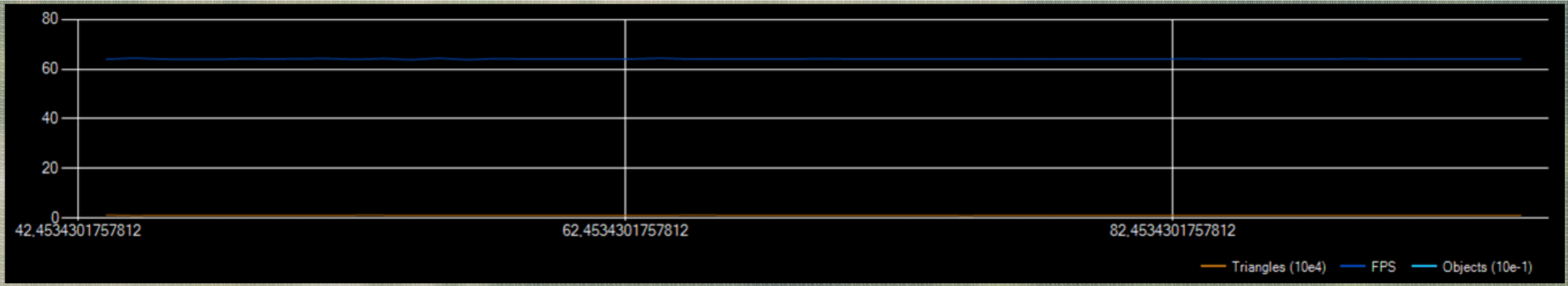
Forests	
Forests	Forests
Information	
Author	None
Project Name	Untitled
Models	
Models	Models
Report	
Report	Report
Sky	
Sky	HeightMap.Model.SkyIn
Terrain	
Terrain	Terrain

Author
The project's author name

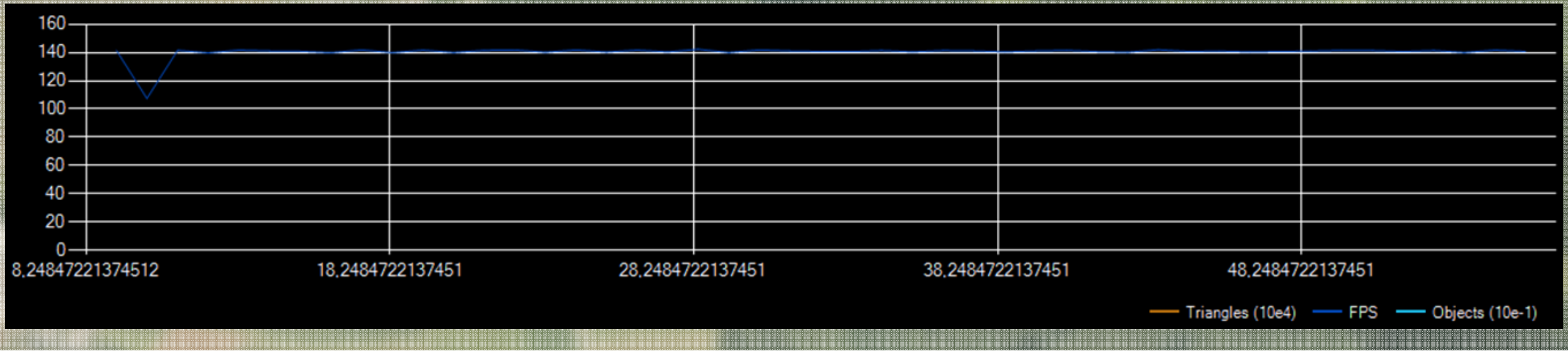
Test and Results

- SET0 – 5 min
  - ROAM
    - FPS = (61.78,64.10,70.48)
  - Tiled GeoClipMap
    - FPS = (102.47,137.32,145.28)

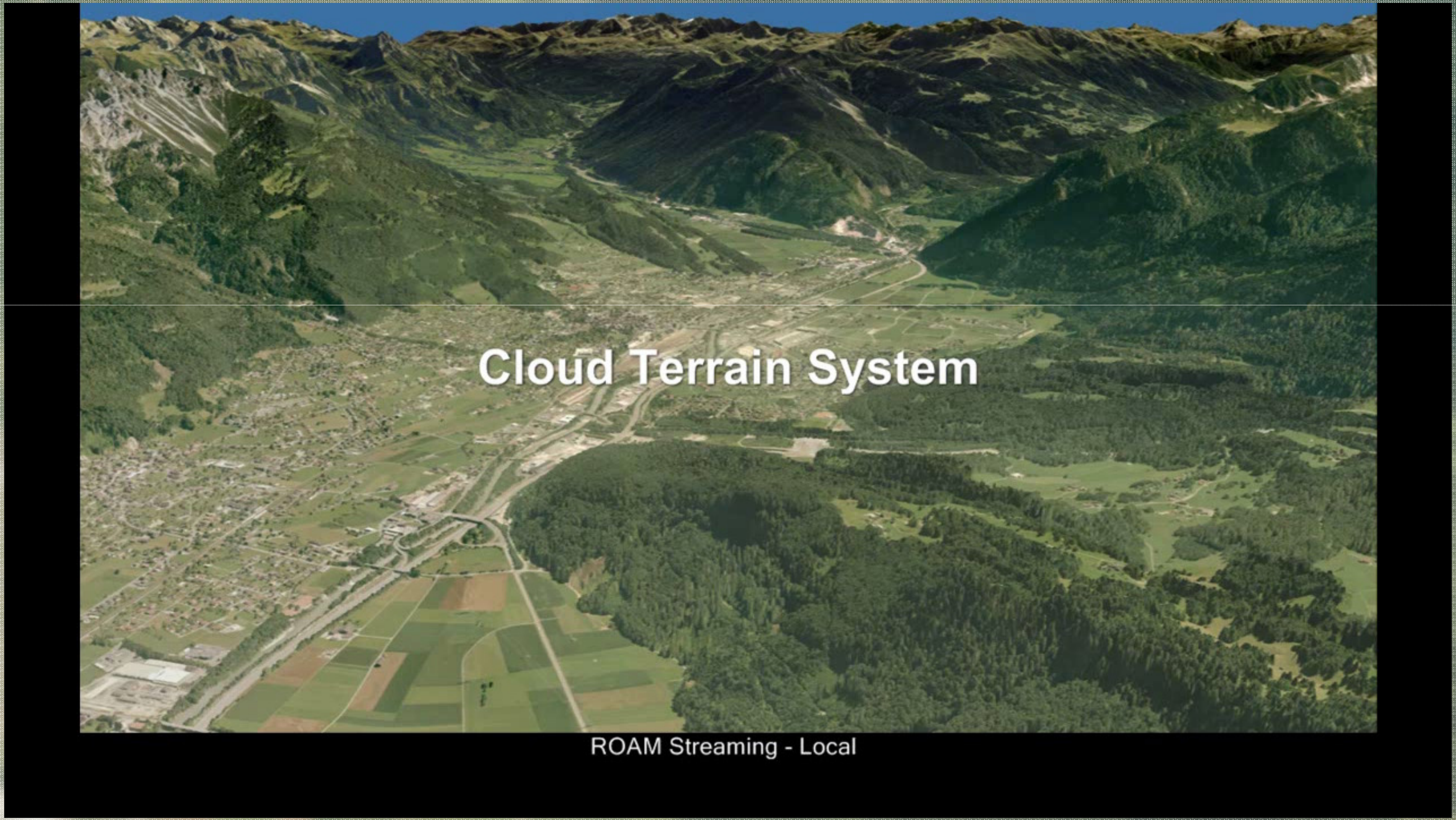
ROAM



TGCM



Test and Results - Videos



Conclusion and Future Works

- **The system is not yet fully implemented.**
- There are a lot o problems with tile loading and memory management
- The Tiled GeoClipMap is a good Algorithm for Out of Core Terrains
- Load Balancing and Management is very difficult.
- The Tile Loss is a very difficult problem to solve.
- Make more validation tests
- Create a Local Network Server
  - It was created but not tested.
- Windows Azure 1.3 is not Compatible with 1.2
  - We need to make changes

Conclusion and Future Works

- Solve The Communications Problems
  - Fix the 0210 package problems
  - Fix the Timer for 0202
  - Use more Internal Endpoints for Inter-Role Cache Share
  - Reduce the Tile Loss
- Tiled GeoClipMap Algorithm
  - Try to use Geometry Shaders to Improve Cracks Resolution
  - Try to use (SM5)
  - Work with compressed data
  - Work with level texture interpolation
- Load Optimizations
  - Batch Loads
  - Improve Prediction Algorithms
  - Finish Administrative Interface
  - Create the 4th Level in Client Machine
    - The Compressed or Not Level
      - HD → Memory Compressed → Memory Raw → GPU
- Compute Cloud Billing

Bibliography

- Real-Time, Continuous LOD Rendering of Height Fields, Lindstrom-Koller, 1996
- Large Scale Terrain Visualization Using The Restricted Quadtree Triangulation, Renato Pajarola
- Willem H. de Boer, GeoMipMap
- Mark Duchaineau et al, ROAM
- Thatcher Ulrich, Adaptive Quadtrees
- Henri Hakl, Diamond Terrain Algorithm
- F. Losasso, H. Hoppe, Geometry clipmaps
- C. Dachsbacher, M. Stamminger, Procedural Terrain
- J. Schneider, R. Westermann, GPU-Friendly High-Quality Terrain Rendering
- Malte Clasen and Hans-Christian Hege (Zuse Institute Berlin), Terrain Rendering using Spherical Clipmaps
- Tatarchuk N.: Dynamic Terrain Rendering on GPU Using Real-Time Tessellation. ShaderX7 (Dec. 2008).
- Tatarinov, A.: Instanced Tessellation in DirectX10. GDC 2008. February 2008.
- Gee, K.: Introduction to the Direct3D 11 Graphics Pipeline. Nvision 2008.
- Castano, I.: Tessellation of Displaced Subdivision Surfaces in DX11. Gamefest 2008
- Windows Azure SDK Documentation
- Amazon Cloud Service
- MapReduce - http://en.wikipedia.org/wiki/MapReduce
- MapReduce: Simplified Data Processing on Large Clusters, Jeffrey Dean &Sanjay Ghemawat, 2004.