An Adaptive Texture Control for a High Fidelity Terrain Visualization

Sang-Hee Kim
Agency for Defense Development, Daejeon, 305-600, Korea
falcon@add.re.kr
Kwangyun Wohn
Korea Advanced Institute of Science and Technology, Dept. of EECS, Daejeon, 305-701, Korea
wohn@vr.kaist.ac.kr

Abstract

For an out-of-core terrain visualization on a resource-limited computing environment, real-time continuous rendering must be crucial to give user more fidelity. Most of the existing techniques often suffer from overloading massive terrain data on-the-fly, especially texture data, by a complex flying-over and by a large view frustum, which results in severe fluctuation in rendering performance.

In this paper, we propose a method, ***multiple VFmaps(ViewFrustum maps)*** which are concentric VF footprints swept on viewpoint, to feasibly work out performance inconsistency by adaptively dealing with large scale of terrain texture. The resolution range and the size of each VFmap are determined by texture mipmap levels contributed to the viewport, viewing parameters, and texture memory capacity. The reason that we have multiple classes of VFmap is to adaptively fit the amount of texture needed to load to the texture memory, because VFmap may cause the excessive data loading in order to cover all area around viewpoint. The position of concentric center of multiple VFmaps plays an important role in controlling the rendered image quality and the performance.

Applying to the terrain-based simulation on PC consolidates the most regulated performance for the abrupt rotation, geo-reference and the large view frustum scale compared to the existing out-of-core terrain processing methods, which contributes to give more visual fidelity.

Key words: Real-time Rendering, Terrain Texture, View Frustum map, Terrain Cell, Adaptive Control

1. Introduction

With the development of satellite technologies, we have to deal with out-of-core terrain data because the high resolution terrain data tends to be growing larger and larger. And also user's need for virtual reality or fidelity in virtual simulation is getting higher and higher. Out-of-core means that all the referenced data during rendering is too large to fit in main memory or texture memory, but exist in secondary disk or data server[Davis 98].

When we perform photogrammetric modeling on the stereo terrain images from satellite, we always get terrain texture higher than DEM (Digital Elevation Model) in resolution. For terrain visualization to deal with out-of-core terrain data on-the-fly on resource-limited PC under the low-to-high altitude flight-through, real-time continuous rendering with promising image quality must be crucial to give user more fidelity[Rabinovich 97].

The whole gamut of terrain-based virtual flight is from simple level flight to very complex maneuvers such as abrupt rotation and geo-reference to the arbitrary position around a viewpoint. Level flight is a normal translation which causes rarely problematic rendering. However, there exists an noticeable discrepancy in frame rates when we perform the abrupt rotation which means drastic attitude change in roll and yaw, and the geo-reference which observes objects at an arbitrary direction out of the viewpoint. The abrupt rotation often needs to load considerably much data entered into current VF compared to the previous VF. For the geo-reference, the situation tends to be catastrophic because almost all data should be newly loaded. Therefore, we have to work out efficiently to these kinds of abrupt movements to keep orientation-invariant continuous rendering.

IMAGINEVirtualGIS[ERDAS99]suggested reasonable viewing range to be [0.5 to 1.5]$\times$[image scale]$\times$[(1 for meters) or (3.281 for feet)]. According to this equation, for example, most popular map of 1:50,000 scale image (roughly equivalent to 5~10 meter resolution image) can have 25km to 75km visibility. For the large view frustum, all the terrain data, especially texture can't be loaded at the very time, so we have to efficiently work out about view frustum with this kind of visibility range.

Most of the existing techniques for orientation-invariant, scale-invariant huge terrain visualization often suffer from overloading massive terrain data, especially terrain texture due to sudden viewpoint change, which results in severe fluctuation in rendering performance. Therefore, efficient texture management is crucial to alleviate excessive load into the graphic processor and present the proper data at the right time.

In this paper, we propose a method, multiple VFmaps (ViewFrustum maps), to feasibly work out performance inconsistency by adaptively dealing with large scale terrain texture in orientation-invariant and view frustum scale-invariant fashion. The resolution range and the size of each VFmap are determined by texture mipmap levels contributed to the viewport, viewing parameters, and texture memory capacity.

We have applied our technique to visualize large scale terrain(for instance, larger than 1000km×1000km) that are composed of 20GB-texture and 5GB-DEM, with view frustum of 90°H×73.7°V×75km and it shows the most regulated performance for the abrupt rotation, geo-reference and view frustum scale compared to the existing out-of-core terrain processing methods, which contributes to give more visual fidelity. Among polygon-based approaches, we think the proposed multiple VFmaps must be the first software-wise approach to adaptively deal with the complex flying-over out-of-core terrain data on resource-limited PC environment.

Fig.1 shows the proposed rendering pipeline schema. We mainly focus on the adaptive per-cell processing considering system resources, application requirements, data scale, and so on. Adaptive texture control uses multiple VFmaps and controls them by terrain cell indexing and prediction-based prefetching. Fig.2 presents the overall data structure which the quadtree of terrain cells makes up the whole terrain and each terrain cell is formed in pyramid (Multi-Resolution in a Single File, MRSF).

We define some terminologies presented in this paper.

(1) Scene map : 2D matrix of maximum terrain cells made by projection of view frustum on the terrain surface

(2) VFmap : bounding square of scene map swept on projected viewpoint

(3) multiple VFmaps : concentric VFmaps which are classified by mipmap levels and cell array determined by factors on texture memory capacity, application, viewing parameters, cell size, etc.

(4) Terrain cell : smallest unit of terrain data handled in per-cell processing

(5) Terrain pool : pointer array corresponding to each terrain cell in whole VFmap

(6) Old visible list : list of cells rendered in previous frame among cells in pool

(7) New visible list : list of cells to be rendered in current frame

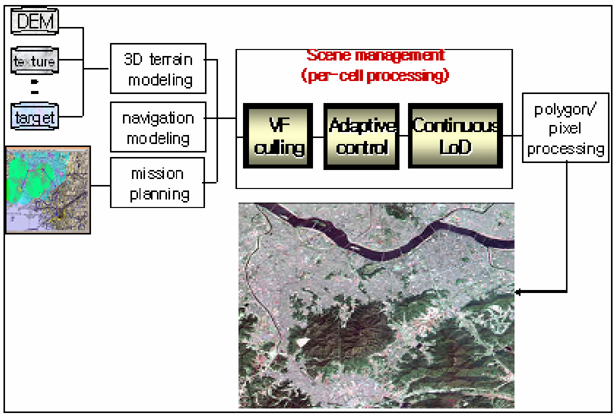


Fig. 1 Real-time rendering pipeline

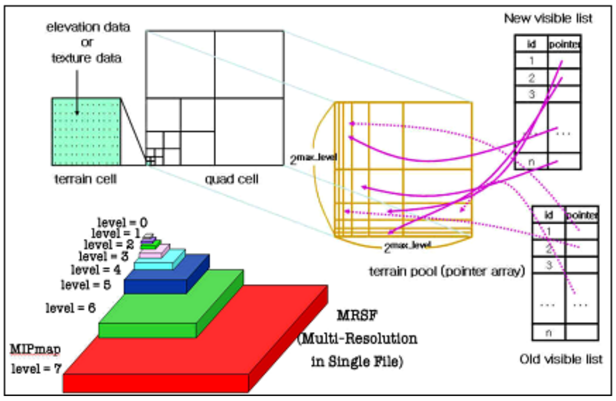


Fig. 2 Terrain cell processing structure

2. Related Works

As an efficient texture management to adjust to the system performance, mipmap texturing[Williams 83] has been popularly used. Tanner proposed the clipmap method to work out a texture-based large terrain visualization by removing unnecessary detail in excess of the system resolution from mipmap. It also manages dynamic loading of subdivided tiles by toroidal wrapping[Tanner 98]. However clipmap method depends on the specific hardware support, and can't keep continuous real-time rendering during geo-referencing.

Blow gives a fundamental texture caching system on low-end PC[Blow 98]. It determines the mipmap level on the basis of the nearest vertex to perform texture caching efficiently and considers larger level first to handle many texture demands. But it still suffers from overloading by sudden orientation change, and sometimes degenerates the rendered image quality by using default stand-in textures.

Cline's approach is to construct large texture as mipmap pyramid grid(MP-grid), make progressive data transfer of cells by priority. But as it keeps the full mipmap textures of terrain cells necessary for rendering, it has some limits to adaptive texture management, which results in an irregulated rendering performance to substantial number of cells newly incoming to view frustum.

A prioritized prefetching technique concerning visibility and LoD switch has been tried[Varadhan 02]. Terrain cells included in prefetching volume larger than the view frustum can be loaded by angular priority. It seems to work well about abrupt-rotation, but turns out severe degeneration to geo-reference.

A large terrain can be treated in the view-dependent fashion[Hoppe 98], but it needs much memory overhead and 2 passes to simplify intra-cell and to stitch inter-cells. Even though it employes the output-sensitive data structure, it can't endure the incoming data due to sudden viewpoint change.

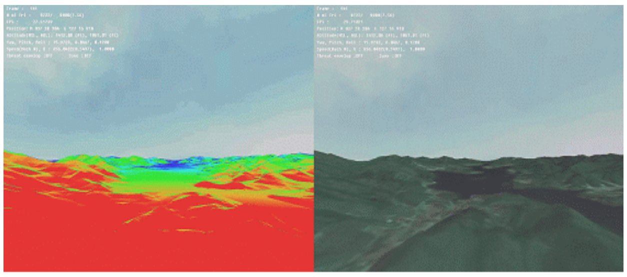
Rather than subdivision scheme, Lindstrom handled out-of-core data by memory mapping using clustered data layout according to data access order, but it can be applied to only DEM which can be clustered individually by access order because texture data is usually managed in arbitrary data block, there doesn't seem to get merit by applying it to the texture [Lindstrom 02]. And also it needs quite a lot preprocessing time and can't work well with the complex flying-over.

3. Adaptive Texture Control by Multiple VFmaps

We propose this method to make an orientation-invariant, VF scale-invariant smooth rendering feasible for out-of-core terrain on resource-limited computing platform. The core idea is to adaptively control data flow from HDD(Hard Disk) to MM(Main Memory) and from MM to TM(Texture Memory) based on preloading by prediction and to deal with the total data volume to meet orientation-invariant and VF scale-invariant performance with negligible loss of image quality.

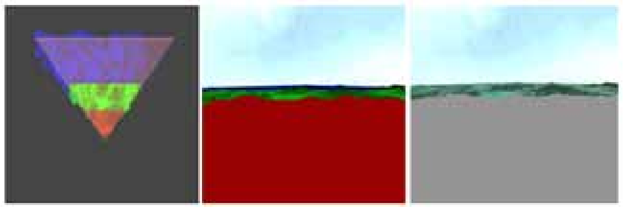
3.1 Memory Configuration

When we investigated the contribution of texture mipmap levels to the rendered scene, we could find out that only some portions of higher levels was delivering most of the visual cue to users, no matter how big the view frustum was. Fig.3 describes the distribution of mipmap level in rendered scene. The meaning follows rainbow color legend, that is, red color means the highest mipmap level, orange color does next lower level, and so forth. Fig.4 shows level coverages in view frustum and in viewport with the same scene as in Fig.3. Both figures explain most of the part in the scene are covered with the highest two levels, even though they just occupy less than a quarter of the view frustum.



(a) mipmap levels (b) rendered image

Fig. 3 Snapshot of mipmap level distribution on viewport



(a) view frustum (b) levels $\geq$ 6 (c) levels $\leq$ 5

Fig. 4 Mipmap level coverage

Making use of these facts, we suggest the memory configuration capable of minimizing data flow between HDD or network server and MM, or between MM and TM(Fig.5). It constructs areas near the view point to be texture-mapped by full range of mipmap levels and far away areas by restricted lower mipmap levels, which results in adaptive texture memory control with promising image quality.

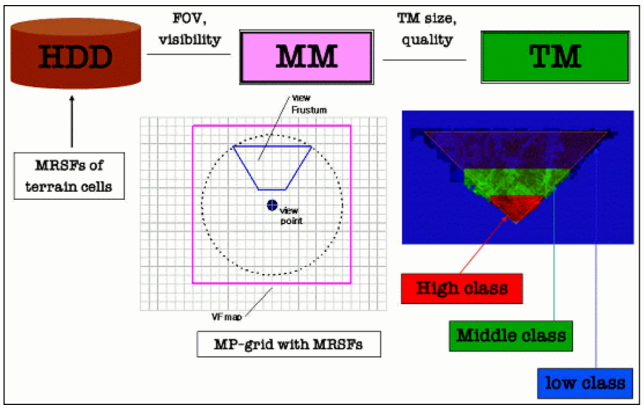


Fig.5 Memory configuration (hard disk, main memory, texture memory)

3.2 Multiple View Frustum maps(VFmaps)

We define multiple VFmaps to be concentric VF footprints swept on viewpoint, to feasibly work out performance inconsistency by adaptively dealing with large scale of terrain texture. The resolution range and the size of each VFmap are determined by texture mipmap levels contributed to the viewport, and viewing parameters like FOV, viewing range, attitude, viewport size, and error metric for simplification, data structure like terrain cell size, cell indexing, and texture memory capacity. The reason that we have multiple classes of VFmap is to adaptively fit the amount of texture needed to load to the texture memory to content with complex flights, because VFmap may cause the excessive data loading in order to cover all area around viewpoint.

Multiple VFmaps form concentric squares, since they are supposed to be texture, considering textures mapped by means of viewing direction and impact position on terrain surface (Fig.6), and Fig.7 presents the corresponding coverage of each VFmap class in view port.

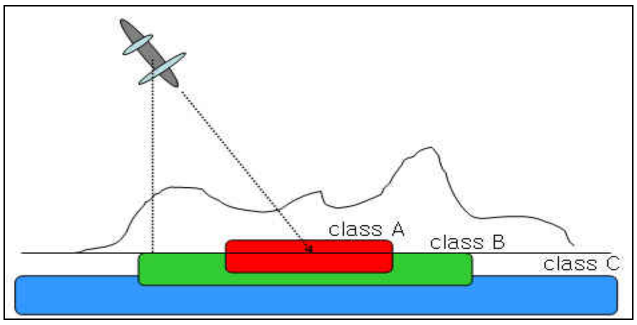


Fig.6 VFmap and viewing direction

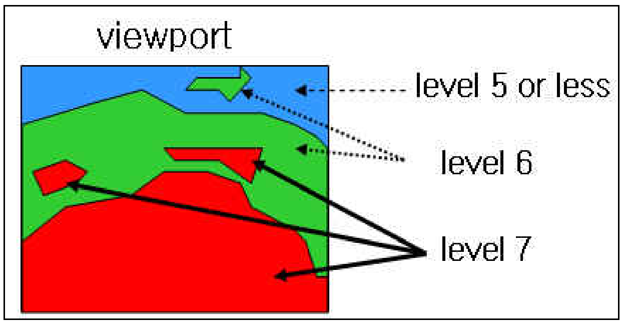


Fig.7 The coverage of VFmap classes

One instance of the resolution range, the number of cells, and the total size that each VFmap class holds shows in Table 1. Those values can be determined dynamically by concerning system performance or by 0/1 knapsack solution. More important thing we must catch is the size. For example, if a view frustum contains 100 cells, it needs about 100 MB of texture memory per every frame, but our VFmap classes never exceed 30 MB.

Table 1. Multiple VFmap classes

	class A (high)	class B (middle)	class C (lower)
resolution	0～7	0～5	0～3
# of cells	4×4	8×8	≤ 32×32
total size	16 MB	4 MB	≤ 4 MB

3.3 Finding the Proper Center of VFmaps

For the sake of generating high quality image, we have only to assign higher VFmap classes to as much area as possible. It can be done by finding an optimal center position of concentric VFmaps.

$$C_{map} = P_{xy}(v) + \lambda \bullet S_c \bullet \frac{P_{xy}(u_v)}{\left\| P_{xy}(u_v) \right\|}$$

…… (Eq. 1)

where C_{map} = Center of View Frustum map classes

P_{xy} = projection of the 3D point on xy plane

v = current view point in (x,y,z)

λ = offset control parameter (non-negative integer)

u_v = unit vector of viewing direction

S_c = size of one side of terrain cell (ex> 5120)

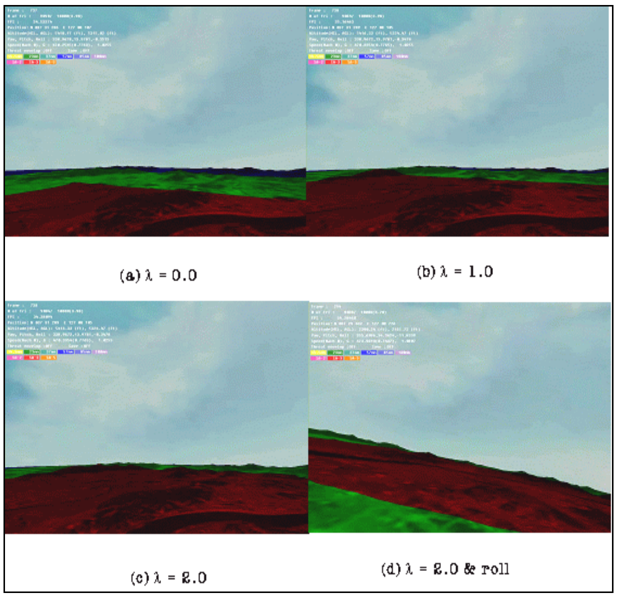


Fig.8 VFmap centers for various λ s

We propose the (Eq. 1) that makes the center position to adjust by way of normal vector of viewing direction. Normal viewing vector length is coincident with one terrain cell size and helps to measure the moving distance of the VFmap center to the viewing direction by the amount of an offset control parameter (λ).

As λ is chosen properly, rendered image in viewport would be better, but if it increases to some degree rendering system might suffer from indexing and maintaining too many cells for that VFmap class, which degenerates rendering performance. Therefore, a meditation between performance and image quality would be needed.

Fig.8 shows the viewport states of changed VFmap center when λ equals to 0.0, 1.0, 2.0. Larger value helps to increase image quality, but shows poor performance to complex flights. As shown in Fig.8-(d), it would happen that temporary degeneration to map the lower class to the area near the viewpoint.

3.4 Terrain Cell Indexing

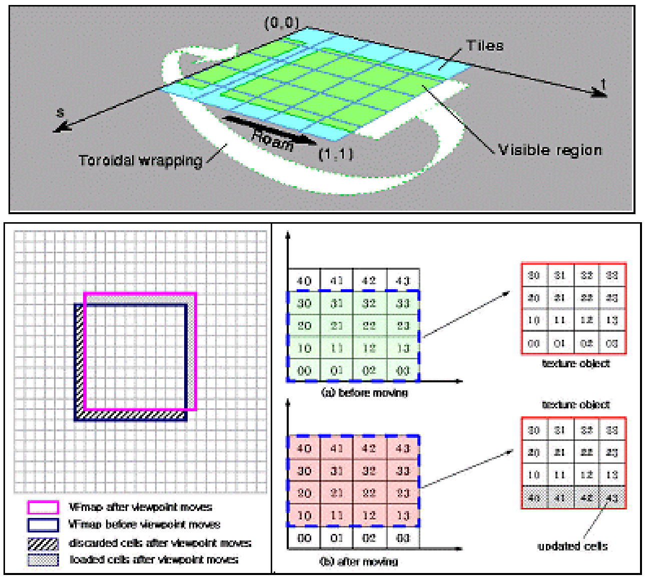

(0,0)
Tiles
t
s
Roam
(1,1)
Toroidal wrapping
Visible region
VFmap after viewpoint moves
VFmap before viewpoint moves
discarded cells after viewpoint moves
loaded cells after viewpoint moves
(a) before moving
texture object
texture object
updated cells
(b) after moving

Fig.9 Dynamic update by toroidal addressing

Each VFmap class has its own cells dynamically loaded and also is used as texture template. Therefore, we have as many textures as the number of VFmap classes. All the texture for terrain cells would be loaded to the corresponding VFmap class by texture subloading. Just like the terrain pool, each VFmap texture object has only to utilize toroidal addressing scheme. Texture template looks like symmetry around view point so that we may use unoccupied areas.

4. Experimental Results

4.1 System and Data Configuration

4.1.1 Computing System

We have used two kinds of PCs, one for portable PC and the other for low-end PC.

(1) Portable PC

- Dual CPUs of 2.0GHz with 1GB RAM

- GeForce FX Go4600 with 64MB texture memory

(2) Low-end PC

- Dual CPUs of 2.8GHz with 2GB RAM

- Quadro FX 2000 with 128MB texture memory

4.1.2 Terrain Data

- Datum : WGS 84 (cf, height : mean sea level)

- Map projection : UTM (zone 52)

- DEM : 20m resolution

- Texture : 10m resolution pan-sharpened color image

- Terrain cell size : 5.12km×5.12km (256×256 for a DEM cell and 512×512 for a texture cell)

- Region of interest : 65,536 cells (256×256)

4.1.3 Viewing Environment

- viewport : 1024×768

- Field of view : 90°H × 73.7°V

- visibility range : 25km ～ 75km (14NM ～ 42NM)

4.2 Results

4.2.1 VF Scale-invariant (Mipmap vs. Vfmap)

We'd like to show that VFmap method operates superior to Mipmap method for various VF scale. Contemporary flight simulation using 10m resolution requires up to 75km with frame rates of more than 20 fps. However mipmap approach can't afford visibility range farther than 60km on PC platform, but VFmap has got no problem even with 75km visibility range.

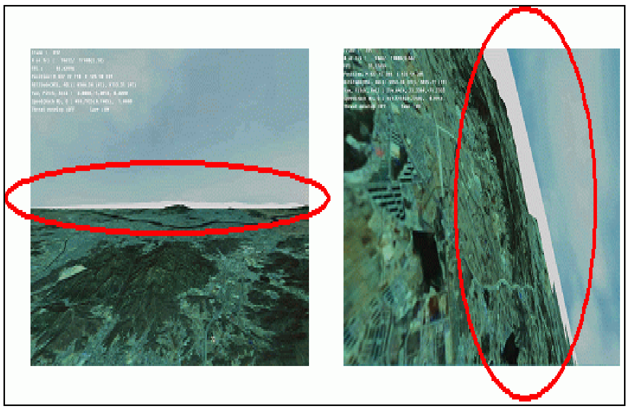


Fig.10 Lack of texture in mipmap with 63km visibility

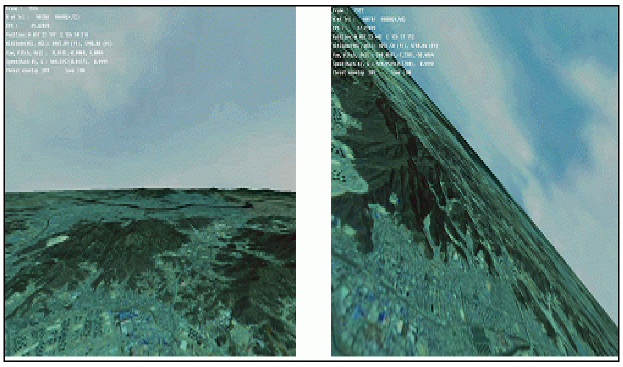


Fig.11 VF scale-invariant in VFmap with 75km visibility

4.2.2 Orientation-invariant (Mipmap vs. VFmap)

(1) path : 4th lag in Fig.12 for a short flight and from 4th lag through 5th lag for a long flight

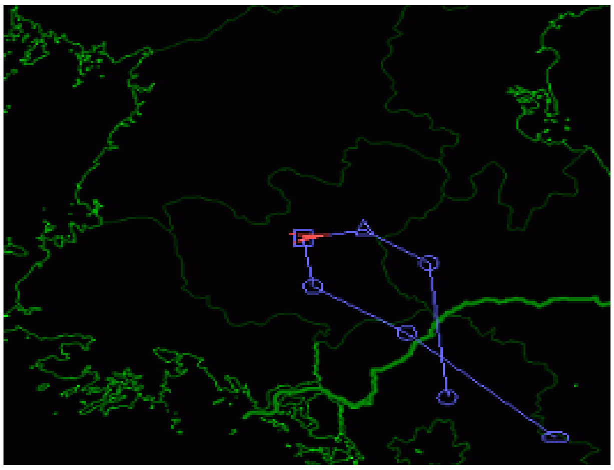


Fig.12 Flight path for orientation-invariant

(2) High Flight Maneuver

- system : Low-end PC

- path type : from level to abrupt rotation (4300 frames)

- visibility range : 48km

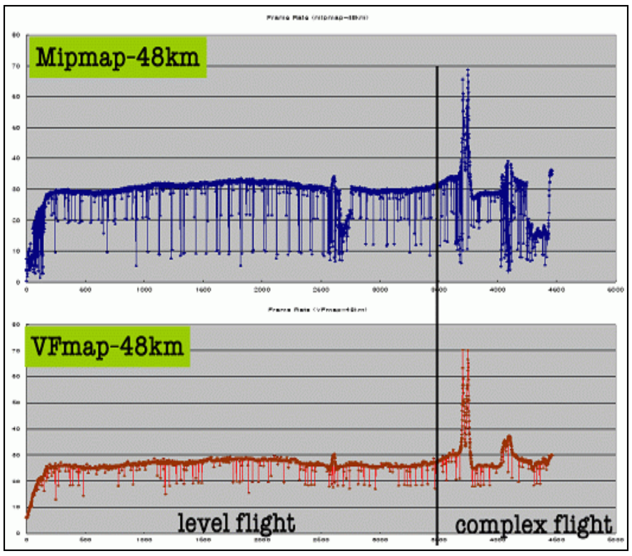


Fig.13 Orientation-invariant for complex flight

4.2.3 MapViewofFile vs. VFmap (on portable PC)

(1) path 1: small area & abrupt maneuver (3,500 frames)

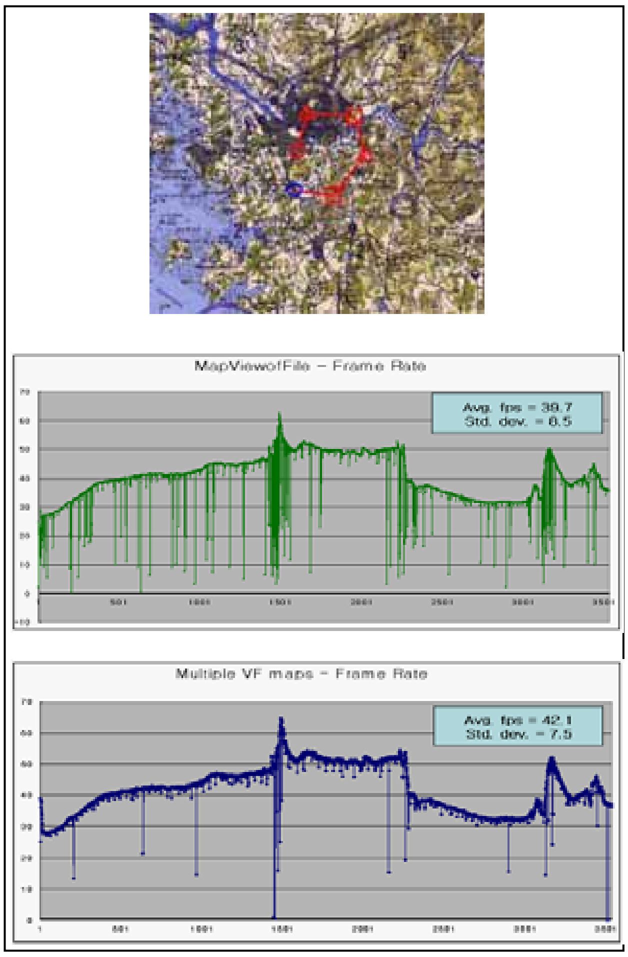


Fig.14 Frame rates of MapViewofFile and VFmap for an abrupt rotation over a small area

(2) path 2 : large area & normal flight (23,000 frames)

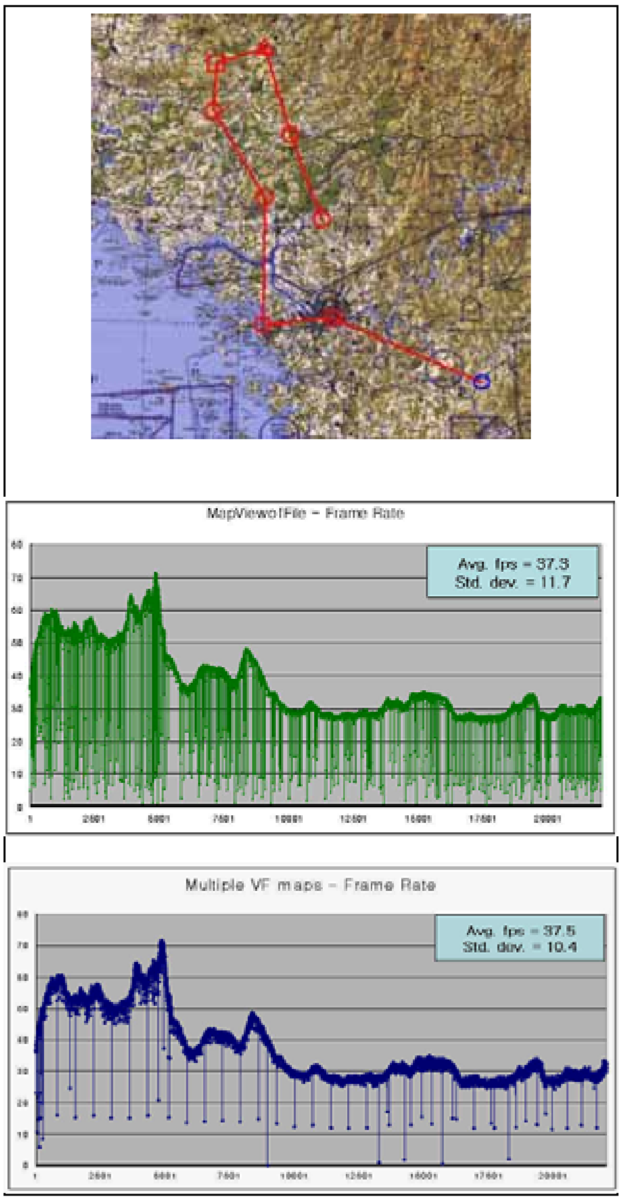


Fig.15 Frame rates of MapViewofFile and VFmap for a normal flight over a large area

Let's summary all the experiments by means of rendering performance.

- Level flight for small area

  : VFmap ≅ MapViewofFile ≅ Mipmap

- Level flight for large area

  : VFmap ≅ MapViewofFile > Mipmap

- Abrupt rotation for small area (64MB TM)

  : VFmap ≅ MapViewofFile

- Abrupt rotation for large area (64MB TM)

  : VFmap >> MapViewofFile

- Abrupt rotation : VFmap > Mipmap

- Geo-reference : VFmap > MapViewofFile >> Mipmap

- VF scale (big) : VFmap ≅ MapViewofFile >> Mipmap

- VF scale (small): VFmap ≅ MapViewofFile ≅ Mipmap

5. Conclusion

We described the real-time scene control to efficiently deal with the large scale of terrain data, which makes orientation-invariant, view frustum scale-invariant smooth rendering possible while maintaining image quality.

Multiple VFmap classes we presented is an efficient per-cell processing to adaptively manage bulky terrain data, and subsequent per-polygon/per-pixel processing given out-of-core terrain data (especially huge terrain texture), various kinds of flight patterns, and resource-limited computing environment.

This method has feasibly worked out rendering performance inconsistency and proved the efficiency by comparing to mipmap-based process and memory mapping process (MapViewofFile).

Since higher resolution terrain data is getting huge and user's region of interest is getting vast and our approach is totally software-wise, we believe the proposed approach may be utilized in any kind of real-time terrain rendering needed to deal with out-of-core data.

6. Acknowledgment

This work was performed under auspices of Agency for Defense Development (Geo-Image Acquisition and Processing System project). It was also supported in part by Virtual Reality Research Center of Korea Advanced Institute of Science and technology.

References

1. J.Blow, "Implementing a Texture Caching System", Game Developer Conference, 1998.

2. D. Cline and P.K. Egbert, "Interactive Display of

Very Large Textures", Proc. of IEEE Visualization, pp.343-350, 549, 1998.

3. D.Davis, T.Y.Jiang, W.Ribarsky, and N.Faust, "Intent, Perception, and Out-of-Core Visualization Applied to Terrain", Proc. of IEEE Visualization, pp.455-458, 566, 1998.

4. "Optimizing IMAGINE VirtualGIS Performance", ERDAS, Nov. 1999.

5. H. Hoppe, "Smooth View-dependent Level-of-Detail Control and its Application to Terrain Rendering", Proc. of IEEE Visualization, pp.35-42, 1998.

6. S.H. Kim, "Efficient Real-time Terrain Rendering System for Mission Flight Simulation", TM-2001-14 , VR lab of Dept. CS of KAIST, 2001.

7. P. Lindstrom and V. Pascucci, "Terrain Simplification Simplified : A General Framework for View-dependent Out-of-core Visualization", IEEE Trans. on Visualization and Computer Graphics, vol. 8, no. 3, pp.239-254, 2002.

8. B. Rabinovich and C. Gotsman, "Visualization of Large Terrains in Resource-Limited Computing Environments", Proc. of IEEE Visualization, pp.95-102, 1997.

9. S. Rottger, W. Heidrich, P. Slasallek, and H.P. Seidel, "Real-time generation of Continuous Levels of Detail for Height Fields", 6th International Conf. in Central Europe on Computer Graphics and Visualization, 1998.

10. C.C. Tanner, C.J. Migdal, and M. T. Jones, “ The Clipmap: A Virtual Mipmap” , Proc. of SIGGRAPH, pp.151-158, 1998.

11. G. Varadhan and D. Manocha, "Out-of-Core Rendering of Massive Geometric Environments", Proc. of IEEE Visualization, pp.69-76, 2002.

12. L. Williams, "Pyramidal Parameters", Computer Graphics, vol. 17, no. 3, pp.1-11, 1983.