ENABLING VIRTUAL-GLOBE BROWSING ON MEMORY-CONSTRAINED PLATFORMS

Cyril MINOUX [a,]

[a] Département Environnement Géophysique – Centre d'Expertise Parisien
16bis avenue Prieur de la Côte d'Or – 94114 Arcueil Cedex – France

Commission IV, WG IV/4

KEY WORDS: Virtual globe, vector draping, terrain rendering, memory constraints, seamless interaction and visualization

ABSTRACT:

In this paper we investigate techniques for virtual-globe visualization of 2D geospatial data draped on terrain while limiting as much as possible the amount of data rendered and loaded in random access memory, in order to enable seamless interaction with heterogeneous geospatial data on memory-constrained platforms. Using our two-steps rendering technique, which uses a texture-based approach for rendering vector data and a multiple level of details quad-tree data structure for rendering raster data, we were able to measure that such virtual-globe interactions can be considered for browsing huge heterogeneous datasets, even with limited available RAM resources.

1. INTRODUCTION

1.1Motivation

Geospatial information targets portable devices such as handhelds on the low-end side, and service oriented architectures on the high-end side. Wether for minimizing the hardware requirements or maximizing the scalability of an architecture, there is a drastic need for geospatial information dissemination software to minimize the resources consumption.

Virtual-globe ergonomics is becoming a standard as it allows geospatial data to be presented as close as possible to the way it appears in the real-world, and interaction to take place in a user-centered schema : the user does not move along a given CRS axis, but moves along horizontal and vertical directions respective to “the screen”, i.e. its own local 3D cartesian reference system, and 3D navigation has been enriched in order to benefit from aircrafts' motion capabilities - "pitching", "rolling", "yawing" as defined in (NasaGRC).

For the purpose of this paper, the challenge consecutive to these capabilities is to take into account that the user can "tilt" the camera in order to have a perspective view of the landscape. Tilting the camera, the user sollicitates a much more significant part of the underlying datasets than in a vertical view. And as geospatial datasets are usually huge, they usually only fit in mass-storage devices. Despite increasing developments in flash-ram-based solid-state-drives, which appear as a promising alternative to conventional hard-disks by providing numerous significant advantages such as lower latency, these emerging technologies do not improve at this time the rate for sequential I/O access (SSD). As a consequence, data-subsets required to be rendered on-the-fly still need to be streamed from persistent storage and cached in volatile random access memory. As for today, PDAs typically come with 64 to 256 MBytes RAM, while servers need to share their few GBytes with tens or hundreds of users. However, given the current area of interest of a user, there is basically no need for loading more information in RAM than the exact information that is going to fit into the screen at the current pixel resolution.

1.2 Related work

Rendering virtual landscapes requires methods for rendering textured-terrain for dealing with raster data, and methods for rendering vector on that terrain.

Huge efforts have been committed to address as fast and as accurate as possible landscape terrain rendering. (Hoppe, 1998) demonstrates the use of multiple-level-of-details terrain and on-the-fly approximation of triangle meshes for efficient terrain rendering maximizing the accuracy of the landscape as it appears on the screen. (Pouderoux, Marvie, 2005) focus on adapting the quality of the terrain approximation to match a target number of frames per second, selecting the appropriate triangles with respect to a metric in order to adapt to the devices' graphic processing capabilities.

These work however focus on terrain data with precomputed textures. If we want to enable seamless interaction with heterogeneous data, the order, visibility or opacity of the various layers can be changed at any moment by the user. As a consequence, this working context requires on-the-fly computation of terrain's textures.

For raster data, this essentially means loading the appropriate datatiles with the relevant level of detail. Handling vector data rendering on 3D landscape is a little bit more tricky. At this time, two types of method coexist : texture-based approaches and geometry-based approaches.
In (Kersting, Döllner, 2002), vector data is rasterized on-the-fly to create a quadtree containing multi-resolution textures, which are then used at the appropriate level of detail depending on the resolution requirements of the terrain triangles.
In (Schneider, Guthe, Klein, 2005), the technique is refined in order to improve the quality of the rendering and minimize aliasing, by applying a perspective transform to the generated-

textures' coordinates. Those techniques for rendering vector data on terrain are known as "texture-based" approaches.
(Schneider, Guthe, Klein, 2005) proposes an alternative method, in which the 2D vector data is preprocessed in order to compute each node's height and in order to associate multi-resolution geometries to each feature. This last method has the drawback of tightly coupling the terrain and vector data : updating the terrain data will require to reprocess every vector layer on the area of interest.
Recently, a more sophisticated geometry-based approach has been proposed by (Schneider, Klein, 2007). It is based on polyhedral extrusion, which enables to precisely control the area which is drawn into when rendering a given feature. This method advertises high-quality rendering, and does not suffer the above-mentioned drawback of previous geometry-based approaches which tight vector and terrain data together in a preprocessing step. However, while it seems efficient for rendering area and line features with simple styles, it is not that clear that it is an efficient method when using line styles with repetitive patterns such as railways, which are likely to increase the complexity of the extrusion polyhedra.

2. TWO-STEPS TEXTURE-BASED RENDERING USING EARTH'S SURFACE APPROXIMATIONS

Our rendering method addresses issues that can be encountered on vector and raster data respectively.

2.1Vector data : need for a continuous texture space

In (Kersting, Döllner, 2002) the visible space is divided into tiles of relevant level of detail which are then used for texturing the terrain. From our experience, dividing the texture space into several tiles leads to visual discontinuities on linear vector features at the edges and at the corners of the tiles as soon as they are portrayed with a thickness greater than 1.

Two types of discontinuities are encountered :
1/ features that go through a tile's vertex will be considered to be out of the area of two of the four adjacent tiles, which geometrically is right, but dismiss part of the symbology

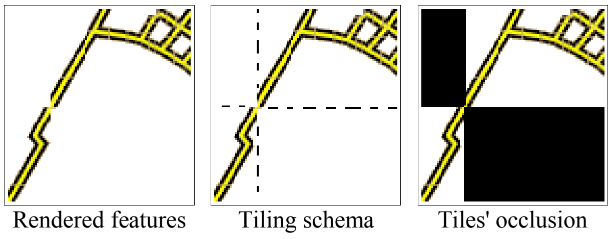

Figure 1. discontinuity type 1 when using tiled texture buffer

2/ features that go across tiles of different level of detail will be affected by the gap of resolution existing between the two tiles

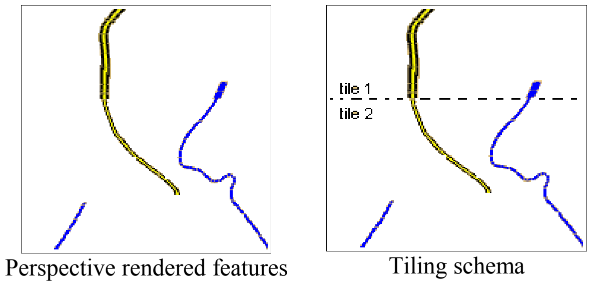

Figure 2. discontinuity type 2 when using tiled texture buffer

On top of these vector portrayal issues, dividing the texture space into tiles will put some constraint on the terrain data sampling schema : allowing terrain triangles to spread over several texture tiles would complexify the terrain rendering step and slow down the whole process. To avoid this, terrain's samples would need to match textures' tiling schema.

Given these three issues we propose texture-based approaches should use a continuous texture space for the whole scene.

2.2 Raster data : need for real-world perspective correction

The best way of having a continuous texture space while limiting the growth of the texture buffer is to have a perspective corrected texture.
(Schneider, Guthe, Klein, 2005)'s perspective correction is used to minimize aliasing that would occur when the frame buffer undersamples the texture buffer's primitives. Although not clearly stated, it is likely that this method could be used to overcome the second type of discontinuity depicted in figure 2. However the two other issues remain to be addressed. Their parametrization is homothetic. Applying this homothetic transformation in the context of a single squared texture buffer is equivalent to considering that the texture space's shape is trapezoidal.

Let's remind that our concern is to avoid loading and rendering data that lies outside of the area pointed at by the viewing frustum. In general intersection of a pyramidal frustum with an ellipsoid does lead to more complex areas than trapezoids.
The following figures show the WGS84 ellipsoid on the left and the shape of the intersection in a plate-carree projection on the right. Whenfar away from the earth, the complexity of modelling this intersection is obvious (figures 3-4).

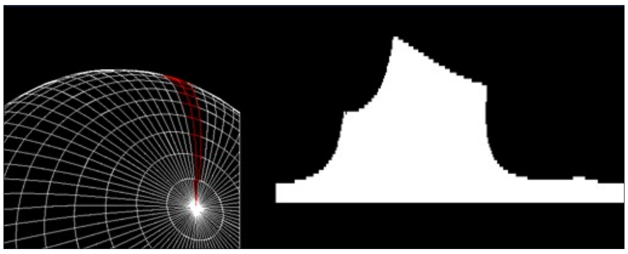

Figure 3. Intersection of a frustum and a WGS84 ellipsoid

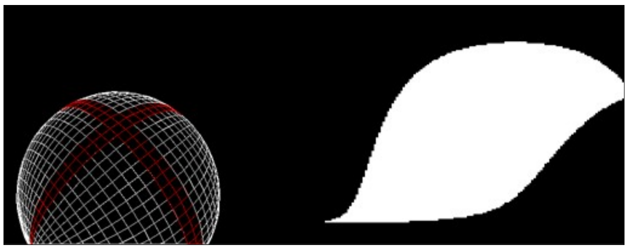

Figure 4. Intersection of a frustum and a WGS84 ellipsoid

As we get closer to the earth's surface - and this is what we're basically interested in - the area gets closer to a trapezoidal area. However, as shown in figure 5, using an enclosing trapezoidal area will lead to select a larger area than the actually visible one. In the end, this will result in requesting more datatiles than necessary (figure 6).

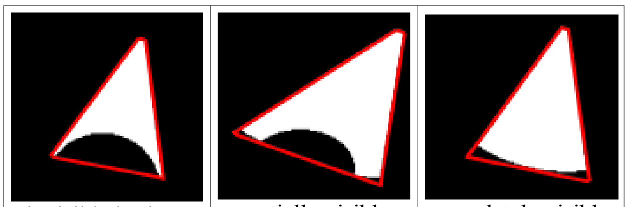

Figure 5 : three possible shapes of the intersection of a frustum and an ellipsoid at a close distance

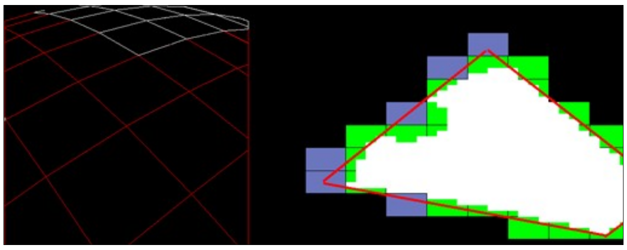
Figure 6. In this example, a trapezoidal approximation of the intersection will require to load 6 additional data tiles (in blue) than when using the actual intersection shape (38 green tiles), which results here in a loss of 6/38 = 15% of cache memory.

The most straightforward way of letting a rectangular buffer
1/ model a shape as close as possible to the actual intersection of a frustum and the earth's surface
2/ while providing a finer resolution in the foreground and a coarser resolution in the background like in perspective correction
3/ and providing the best possible match (without knowledge about terrain) between the resolution dedicated to each tile's information and the resolution required by the terrain's texture
is to apply the viewpoint's current projection matrix to the texture buffer.

2.3 The two step rendering process

In order to address the three issues mentionned in paragraph 2.1 and to minimize the amount of data required as explained in paragraph 2.2, we propose the following two-step rendering process.

In the first step 2D data is rendered in a 3D space without taking terrain data into account. Vector features are simply transformed from longitude-latitude 2D geographic coordinates to 3D using the surface's approximation model. This is done on-the-fly. One can use ellipsoid equations as described in (Bouteloup, 2003) or any other approximation of the earth's surface, including geoids. The choice of the earth's approximation is essentially a trade-off between accuracy and efficiency. As it is simply about selecting the relevant data tiles, the granularity should not require a much more precise approximation than the ellipsoid.

Imagery data is structured as a quadtree, with constant pixel size and increasing geographic resolution. High resolution / small extent tiles are used in the foreground while low resolution / wide extent tiles are used in the background. The four vertices of the tile are processed like vector vertices. This adaptative method allows to adjust at the same time both the resolution of the data and the accuracy of the earth's surface approximation.

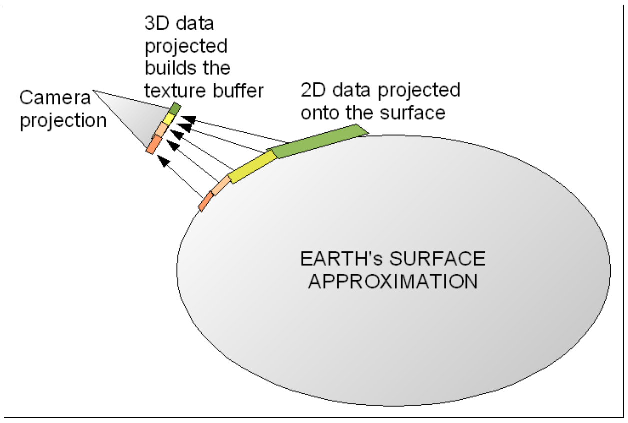
3D data projected builds the texture buffer
Camera projection
2D data projected onto the surface
EARTH's SURFACE APPROXIMATION

Figure 7 : Rendering step 1 : building the texture buffer

As depicted in figure 7, this first rendering step builds the texture buffer's content.

The second step consists in rendering terrain using this texture buffer. The only prerequisite on terrain data structure is that it is available as a triangle mesh.
As described in figure 8, for each vertices of each triangle, the projection of the vertex on the earth surface's approximation is computed and projected into the texture buffer by applying the projection transformation of the current point of view. This gives pixel coordinates corresponding to the texture's coordinates to be used for this vertex when rendering the terrain triangle on the screen.

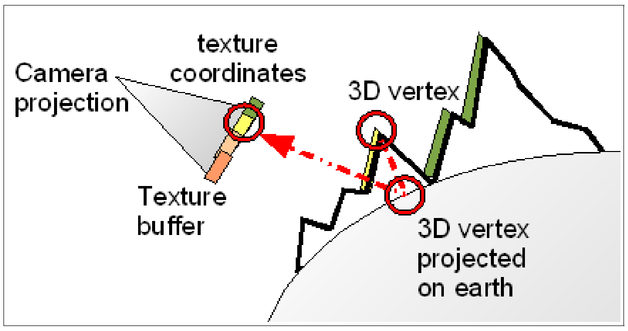

Figure 8 : Rendering step 2 : computing texture coordinates

Figures 9 and 10 illustrate the process on a real-world example.

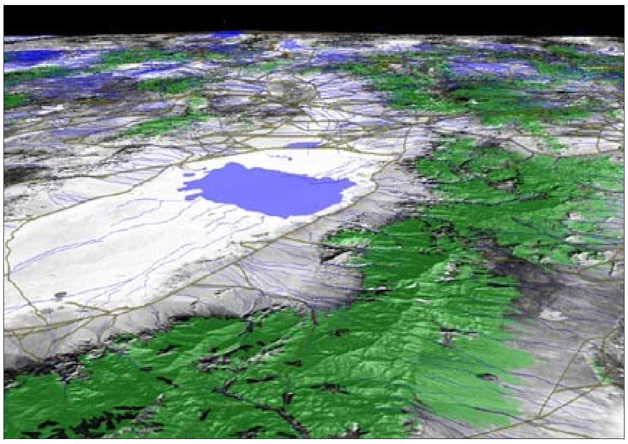
Figure 9 : Example of texture buffer generated in step 1

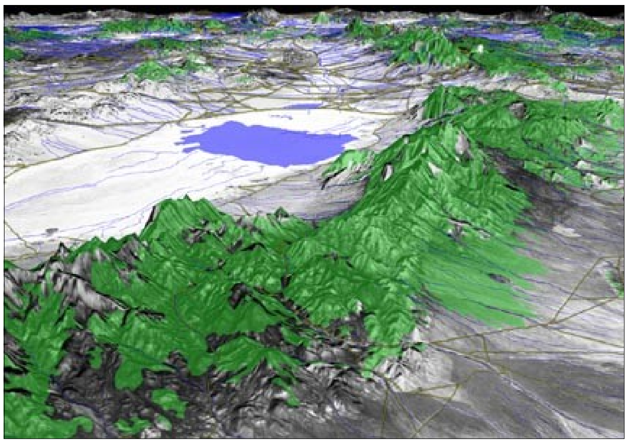
Figure 10 : Terrain generated with the above texture buffer

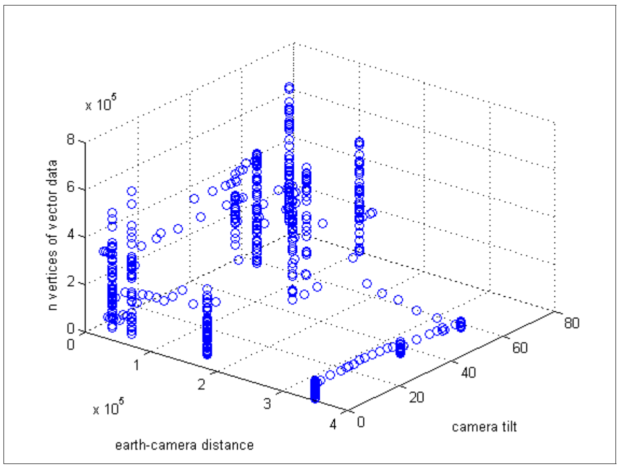

Figure 12 : number of vector data vertices required for building the texture given the viewpoint's tilt and distance to earth

3. IMPLEMENTATION AND TESTS

In our implementation, the WGS84 ellipsoid was chosen as the approximation of the earth's surface. The client window is sized 1024x1024 pixels. The size used for the imagery data quadtree tiles is 512x512. A basic cache mechanism was implemented in order to avoid streaming again data that is already in memory from one frame to another, but was configured so that it can just fit the maximum number of tiles required experienced.

The following datasets were used for these tests, covering an area of 10°x4° (W120-110 longitude, N36-40 latitude) collected from (USGS) and (MapAbility)

- a 28800x72000 pixels matrix of Landsat imagery (1.44 GBytes in GeoTIFF format), turned to grayscale to improve vector features readability ;
- a 7200x12000 pixels matrix of SRTM terrain (213 MBytes in DTED format) ;
- five layers from a 223 MBytes VMap Level 1 dataset chosen for their representativity : tree areas from the vegetation layer ; lake areas and watercourses from the hydrographic layer ; roads from the transportation layer ; builtup areas from the population layer

3.1Memory requirements measures

A monitoring process enabled to record various parameters of interest and produced the following observations.

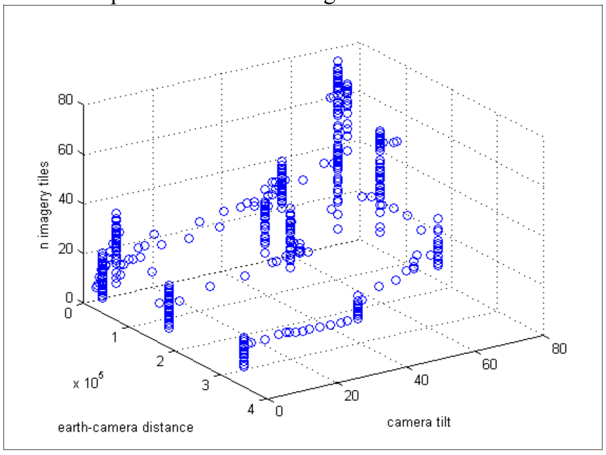

Figure 11 : number of data tiles required for building the texture given the viewpoint's tilt and distance to earth

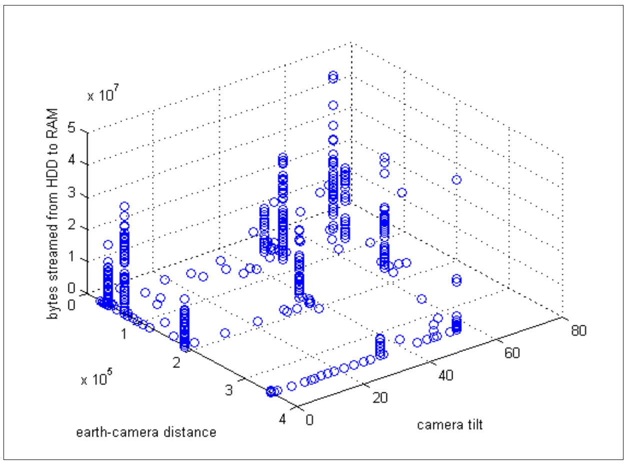

Figure 13 : number of bytes streamed from HDD to RAM when moving, given the viewpoint's tilt and distance to earth

camera's tilt	**camera - earth distance**	**number of visible imagery tiles**	**number of visible vector vertices**	**MBytes of data streamed (imagery + vector)**
	350 km		9 000 to 90 000	0 to 1 MB
vertical (0°)	186 km	6 to 24	60 000 - 350 000	0 to 11MB
	40 km		50 000 - 550 000	0 to 20MB
tilted, no	350 km		20 000 - 90 000	0 to 5 MB
horizon visibility	186 km	9 to 50	90 000 - 550 000	0 to 20MB
(40-50°)	40km		100 000 – 580 000	0 to 32MB
tilted,	350 km		10 000 - 100 000	0 to 5 MB
horizon visible	186 km	11 - 80	140000 - 600000	0 to 20MB
(65-70°)	40km		100 000 - 700 000	0 to 50MB

Table 1 : influence of camera's tilt and distance to earth on number of visible tiles, vector vertices and data streams.

The number of 512x512 tiles necessary for rendering a 1024x1024 screen is a little bit astonishing at first but can be explained. A coarser level of detail for a given tile is selected only if the data resolution is more than twice the screen pixel resolution. As a consequence a 257x257 screen area will still require a 512x512 data tile. As a consequence, up to 4x4=16 512x512 data tiles may be required for filling the screen, provided that the tiles and the screen are perfectly aligned and the viewing frustum is vertical. Should the viewing frustum be misaligned with the tiling schema of the data, this can increase to 25 (figure 14). Also, depending on how data was produced, and specifically if the angular pixel resolution is the same in both north/south and east/west directions, 512x512 datatiles will be rectangular once projected on earth. As a consequence, at 38° northern latitude ceil(5 / cos(38°)) = 7 datatiles will be selected instead of 5 in the longitudinal direction, leading to selecting 5x7 = 35 datatiles.

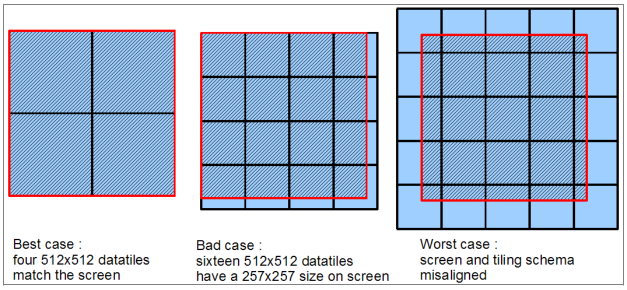

Figure 14 : number of datatiles required for building texture

As expected, camera's tilt is a key factor on the number of data tiles required for building the texture. As shown in figure 15, some small portions of the texture buffer may require a full tile.

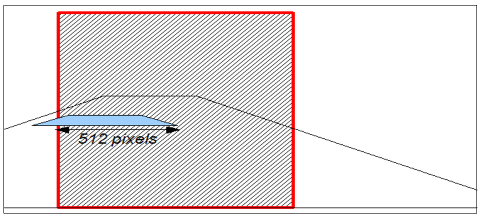

Figure 15 : example of a small area requiring a full datatile

Fortunately the multiple level of details data structure of imagery and the distance dependant vector data portrayal rules are able to maintain the number of tiles streamed to the texture buffer below 80 tiles and the amount of vector vertices below 700 000 : this represents 1% of original imagery dataset (8037 tiles) and 14.2% of the original vector dataset (4 942 539 vertices).

As each tile is stored in memory as a RGBA uncompressed texture, it represents about 1MBytes (1 048 576 exactly). This means that a 84 MBytes cache is necessary for imagery data. Each vertex is stored as a double-precision floating point triplet (X,Y,Z) which means that a 3x8x700000 = 16.8 MBytes cache is necessary for vector data. This amounts to a total of 101 MB.

Our measures demonstrate that using this technique, devices with 128 MBytes should be able to browse on a 1024x1024 screen and with virtual globes ergonomics, heterogeneous data like the huge imagery and vector set we tested. For portable devices with 6x smaller screens (320x480) memory requirements can be extrapolated to about 17 MBytes.

3.2Rendering quality considerations

One specificity of the proposed two-step rendering technique is that when in perspective view, the more distant areas are provided a lot less texture-buffer surface than the closer areas, and since line feature rendering occurs with constant pixel width on the texture buffer, the aliasing of line feature rendering can strongly affect readability.

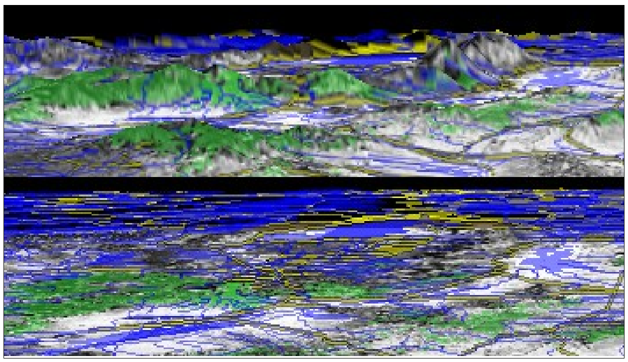

Figure 16 : zoom on the background of a landscape with terrain (top) / without terrain (bottom)

Readability of vector data in contexts that would clutter the features is however an issue in itself that does not affect our approach specifically : as shown in figure 16, far-away vector information is not much more informative without than with terrain. Distance-dependent portrayal rules for vector features are necessary not only for performance considerations to keep the amount of data reasonable when tilting the camera, but also for maintaining some readability. We use this mechanism in the step 1 of our rendering algorithm, which provides more pleasant results in the background while being more selective on the relevant information presented to the user.

Still, aliasing will affect line features rendering, even at the middle of the screen, whenever there are steep slopes (figure 17). This problem however has been acknowledged by the cited papers addressing vector data draping on terrain, so it is not specific to our method.

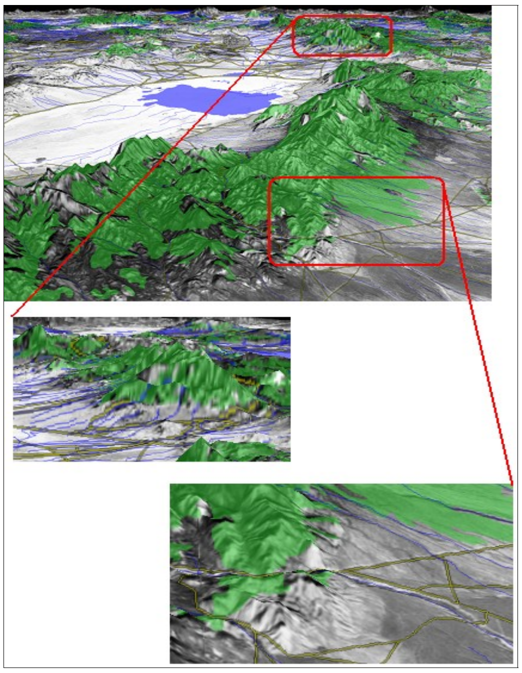

Figure 17 : foreground / background quality and steep slopes

A specificity of our implementation was to directly use the framebuffer for step 1 (building the texture). No framebuffer color clearing is done between the two steps, thus there is already a default view of the landscape when step 2 is triggered. This way we don't have to render a default WGS84 terrain where no terrain data is available. This has been thought as a basis for best-effort continuum between areas where there is terrain data and areas where there is none. However this suffers from the caveat that terrain data is usually visualized with accentuation, and this accentuation also affects the mean height at the current viewpoint, in which case unpleasant – but meaningful – gaps will appear (figure 18).

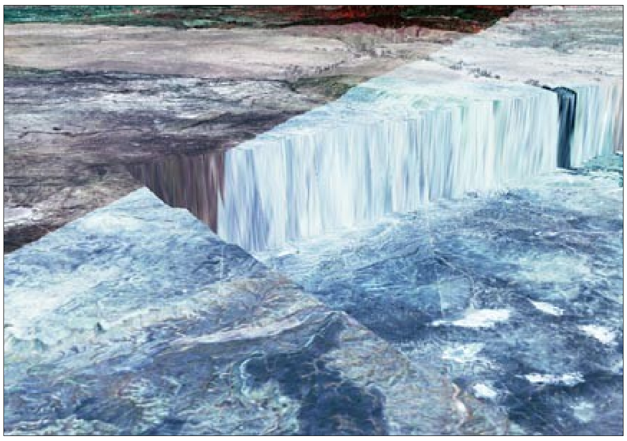
Figure 18 : a step between no-data terrain areas and terrain areas, with 120x terrain accentuation

4. CONCLUSIONS AND WAY FORWARD

This paper proposes a refinement of texture-based vector draping approaches

- which minimizes the amount of data needed to be loaded into memory ;
- which provides on-the-fly computation of the terrain texture, which enables the user to manipulate layers' order, visibility or opacity at any time ;
- which keeps independence between the terrain and the imagery and vector layers, which means that each of this source can be streamed and updated independently ;
- which uses a continuous texture-space, addressing tricky issues on vector data portrayal ;
- which is open to arbitrary complex portrayal rules for vector data ;
- which is open to any efficient algorithm for terrain rendering, enabling to benefit from latest developments in that respect ;

Our memory-benchmarks demonstrated that using such an approach, virtual globe browsing can be considered on memory-constrained platforms.

There are still further developments to undertake to address some rendering quality issues. For example, only part of the texture buffer pixels are used whenever the skyline is visible within the viewport when the camera is strongly tilted. Finding a way to overcome this limitation could help limiting aliasing of line features in steep slopes.

5. REFERENCES

References from Journals :

(Hoppe, 1998) H. Hoppe - “Smooth View-Dependent Level-of-Detail Control and Its Application to Terrain Rendering” In *IEEE Visualization'98*, 35-42, 1998

(Kersting, Döllner, 2002) O. Kersting, J. Döllner - « Interactive 3D Visualization of Vector Data in GIS ». In *Proceedings of the 10th ACM International Symposium on Advances in GIS*, 2002

(Pouderoux, Marvie, 2005) J. Pouderoux, J.-E. Marvie - "Adaptive Streaming and Rendering of Large Terrains using Strip Masks". In *Proceedings of ACM GRAPHITE 2005* pp. 299-306, 2005

(Schneider, Guthe, Klein, 2005) M. Schneider, M.Guthe, R.Klein « Real-time Rendering of Complex Vector Data on 3D Terrain Models ». In *Proceedings of the 11th International Conference on Virtual Systems and Multimedia*, 2005

(Schneider, Klein, 2007) M. Schneider, R. Klein "Efficient and Accurate Rendering of Vector Data on Virtual Landscapes", in *15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision,* 2007

References from Books :

(Bouteloup, 2003) Didier Bouteloup – "Cours de géodésie", Chapter 2 "Géométrie de l'ellipsoïde" – *Ecole Nationale des Sciences Géographiques, Institut Géographique National, France* – March 1st 2003

References from websites :

(MapAbility) http://mapability.com/info/vmap1_index.html

(NasaGRC)http://www.grc.nasa.gov/WWW/K-12/airplane/pitch.html
http://www.grc.nasa.gov/WWW/K-12/airplane/roll.html
http://www.grc.nasa.gov/WWW/K-12/airplane/yaw.html

(SSD) http://en.wikipedia.org/wiki/Solid_state_drive
http://en.wikipedia.org/wiki/Access_time

(USGS) Data available from U.S. Geological Survey, EROS Data Center, Sioux Falls, SD – http://seamless.usgs.gov