J. Alonso, R. Joan-Arinyo. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY : 1– March 2018

Back-to-Front Ordering of Triangles in DTMs Over Regular Grids

J. Alonso[1] and R. Joan-Arinyo[1,2], *Corresponding author*

[1] *Informàtica a l'Enginyeria Group, Universitat Politècnica de Catalunya, Barcelona 08028, Catalonia.*
[1,2] *Group d'Informàtica a l'Enginyeria, & Visualization, Interacction and Virtual Reality Group, Universitat Politècnica de Catalunya, Barcelona 08028, Catalonia*

E-mail: jalonso@cs.upc.edu, robert@cs.upc.edu

Revised March 23, 2018

Abstract Visiting triangles that conform a digital terrain model is a core operation in a number of fields like animation and video games or generating profiles, cross-sections and contours in civil engineering. Performing the visit in an efficient manner is an issue specially when the output of the traversal depends in some way on additional parameters or informations that change over time, for example a moving point of view.

In this work we report a set of rules such that given a digital terrain model defined over a regular grid and an arbitrary point outside the terrain, for example a point of view, defines a total back-to-front order in the set of digital terrain model triangles with respect to the point. The set of rules is minimal, complete and correct.

To assess how the rules perform, we have implemented a CPU based algorithm for realistically rendering height fields defined over regular grids. The algorithm does not make use of the z-buffer or shaders featured by our graphics card. We show how our algorithm is implemented and we show visual results obtained from synthetic and real data. We further discuss the algorithm performance with respect to two algorithms: a naive algorithm that visits triangles according to grid indices which does not solve the hidden line problem, and the z-buffer provided by the graphics card featured by our computer. Our algorithm allows real time interaction when the point of view arbitrarily moves in 3D space and we show that its performance is as good as that of the z-buffer graphics card.

Keywords back-to-front ordering, digital terrain models, elevation terrain models, triangle strips, visibility.

1 Introduction

Digital terrain models (DTM) provide a topographic model of the earth surface including just terrain features and are widely applied in many fields such as computer graphics, resource management, earth and environmental sciences, civil and military engineering, surveying and photogrammetry, and interactive 3D game programming.

Among the requirements that impose strict efficiency constraints on the algorithms that traverse DTMs we can find: the need of encoding large area terrain maps, the ever increasing precision that applications demand, and the steadily increasing number of DTM applications that require real-time response. These requirements clearly make brute force traversal unpractical.

In this work we report a minimal, complete and correct set of visiting rules which define a back-to-front ordering of triangles in a coherently triangulated DTM with respect to an arbitrary point placed outside the terrain, in what follows called point of view. With the point of view, we associate an axis-aligned local reference framework. Projections on the XY plane of the local axis and the bisector of the first and third quadrants define six sectors. Specific visiting rules for collections of triangles that project on each sector are then defined. The set of rules includes rules for uniformly triangulated DTMs and additional rules for fans of triangles that seamless stitch triangles belonging to different levels of detail in level of detail-based renderings.

We have implemented an algorithm based on the set of visiting rules defined. The algorithm is simple, does not exploit edge graphics cards rendering technology, say z-buffer or shaders, and allows for real time interaction when the viewing position freely moves in 3D space. The results of our experiments show that the algorithm performance is as good as that of the z-buffer of our computer's video card.

The manuscript is organized as follows. Section 2 is devoted to reviewing previous related work. In Section 3 we describe the topology of the triangulation we will use and briefly justify how the 3D problem boils down to a 2D problem. The set of rules for back-to-front visiting triangles are defined in Section 4. In Section 5 we develop a case study where our rules are applied to realistically rendering three different DTMs. Here we also detail and discuss the results of the study. In Section 6 the basic set of visiting rules is generalized to deal with multi-resolution representations of terrains. We close with a summary in Section 7.

2 Previous work

A number of techniques have been proposed that deal with simplified representations of terrains [1]. Here we are interested in those algorithms developed to take advantage of the data structures underlying grid surfaces. Specifically, we focus on surfaces defined as bivariate functions computed on a set of regular grid points and back-to-front visibility ordering.

The most widely used approach relies on the multi-resolution concept. The approach reduces the number of triangles or polygons to be visited according to an adaptive level-of-detail control that adjusts the terrain tessellation as a function of the view parameters [2]. However, the level-of-detail introduces some new problems that violate the terrain coherence in both space and time. Spatial coherence is threatened because of cracks generated along an edge between two adjacent regions of different levels of detail if the region of less details cannot represent the height at a point where the region of higher detail can do it. Concerning temporal coherence, difficulties arise because a temporally coherent terrain does not rapidly change over time resulting in the so-called popping effect. Fixing cracks requires an accurate strategy to define transition regions to guarantee geometric continuity [3]. Suppressing popping requires specific techniques like geomorphing [4].

Back-to-front ordering was first introduced in [5]. The ordering is based on the simple observation that, given a volume grid and a view plane, for each grid axis there is a traversal direction that visits geometric elements defined over the grid in order to decrease distance to the viewing point. Authors of [5] only reported results for implementations considering orthographic projections.

The floating-point perimeter algorithm reported in [6] considers nine different regions defined in the viewing plane by four lines, $x = Xmin$, $x = Xmax$, $y = Ymin$, and $y = Ymax$. With each region we associate an enumeration that defines a previously computed sequence to process faces and edges with respect to an active perimeter. The algorithm needs to process sets of edges individually to figure out crossing points with respect to the floating perimeter. The paper does not discuss whether the method supports geometric elements straddling over regions. The work in [7] gives a formal theoretical basis for the algorithm in [6].

Ordering algorithms for rectilinear grids have been thoroughly studied in [8]. After showing that the basic back-to-front approach as well as a number of variations fails for perspective projections, a correct perspective back-to-front ordering is introduced. In [9] authors described an algorithm based on the implicit back-to-front ordering defined by nodes in an quadtree.

A technique based on predefined configurations to visit DTM triangles is reported in [10]. Configurations define a back-to-front ordering of quad cells. The set of predefined configurations suffers from some drawbacks. For example, configurations for some quadrants are redundant and no specific configurations are given to render quads overlapping more than one sector. In these conditions, the approach can lead to quads which are wrongly sorted thus turning the approach useless.

Concerning rendering of 3D geometric models, recent work reported in [11] reports on an approach to traverse static 3D models in a front-to-back or a back-to-front order with respect to a set of predefined points placed outside the model bounding volume. The 3D space is partitioned into several subspaces according to a set of planes. Then triangles within each subspace are depth-sorted and stored in a graph. Finally triangles are visited after identifying the region where the predefined point under consideration is placed. The approach shows a state-of-the-art run-time performance but at a high cost in the preprocessing step.

3 Terrain Representation

First we fix the topology of the DTM triangulation we shall use. Then we show how ordering along a straight line 3D triangles on a DTM can be solved as a 2D problem.

3.1 Triangulation Topology

We consider terrains represented as DTMs defined over regular grids along both the X and Y axis. Every pair of neighbor heights in a DTM along a sampling axis defines a DTM edge. A loop of four edges defines a DTM cell. Each DTM cell is subdivided into two surface triangles. There are two possible different ways of subdividing DTM cells as shown in Fig. 1 where triangle vertices are labeled with grid coordinates. In the sequel, we consider the cells subdivided into triangles as shown in Fig. 1a. There is nothing essential in the choice but it has an effect on the resulting set of cell configurations needed to properly define a sequence to traverse the triangles.

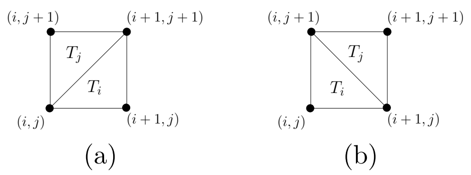

Fig. 1. Projections on the XY plane of two different possible triangulations associated with a DTM cell.

To allow fast processing, compact representations and easy traversing [4, 12], we organize the terrain into squared blocks each of which consists of a number of tiles. In the current implementation, blocks are of size 512×512 and tiles of size 64×64. A block is the basic unit our algorithm considers. The block under consideration is updated as the point of view changes. Then tiles in the block are visited.

3.2 Ordering Triangles Along a Line

Ordering triangles in a DTM over a regular grid hit by a given straight line takes advantage of the fact that height fields do not allow terrain overhangs and that a triangle is a convex shape. In these conditions, it is easy to see that sorting 3D triangles in the DTM can be solved by projecting the DTM, the point of view, and the straight line on the XY plane and considering the 2D induced problem. Fig. 2 shows a DTM with a camera placed on it and the projection on the XY plane of the triangulation.

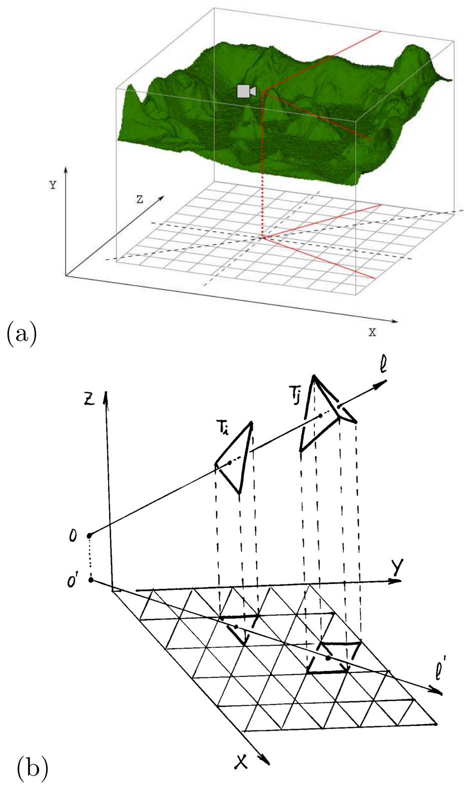

Fig. 2. (a) DTM, the singular point pictured as a camera and the projection plane. (b) Projections on the XY plane of: the DTM triangulation, the point of view, the line of sight and the 3D X and Y axis.

Let T_i and T_j be two different triangles in the DTM surface triangulation. Clearly, triangles T_i and T_j share at most one common edge.

Let O be the point of view and l the line of sight as illustrated in Fig. 2. Let p_i and p_j be the points where the line of sight l intersects triangles T_i and T_j, respectively.

Let T'_i, T'_j, O', l', p'_i and p'_j denote the parallel projections onto the XY plane of the corresponding geometric elements in the 3D space. Clearly T'_i and T'_j are convex. Since projections preserve incidence, p'_i is on l' and belongs to T'_i and p'_j is on l' and belongs to T'_j.

Assume that p_i is closer to O than p_j and that the line of sight l through p_i and p_j is not parallel to the Z axis. Then clearly, the distance from p'_i to O' is samller than the distance from p'_j to O'. As considered, DTM triangulations do not allow terrain overhangs. Taking into account that DTM projected triangles are convex and do not overlap, triangles in a DTM triangulation over a regular grid can be sorted according to distances to the viewing point just by considering the projection of the 3D geometry onto the XY plane.

4 3D Back-to-Front Ordering

A back-to-front ordering of the triangles in the DTM defines a closer-than relationship between triangles with respect to the point of view. This ordering is at the heart of the algorithms that explore triangulations over regular grids for fast processing to solve a number of problems, for example the hidden line problem for a fixed point of view.

To define a complete and correct set of orderings, we split the projection of the DTM triangulation onto the XY plane as follows. Let $O = (x, y)$ denote the projection of the point of view, which is not necessarily a grid point. We define a set of local orthogonal axis, X and Y, with origin at the projected point of view, O, and the axis aligned with the terrain sampling directions. Now let B be the bisector of the first and third quadrants defined by axis X and Y. The triple $\{X, Y, B\}$ partitions the terrain into different regions that we call sectors. When the point of view projects within the triangulation projection, there are six sectors that we label as NE1, NE2, NW, SW1, SW2 and SE as shown in Fig. 3(a). When the projection of the point of view falls outside the triangulation projection, the number of sectors is at most three as depicted in Fig. 3(b).

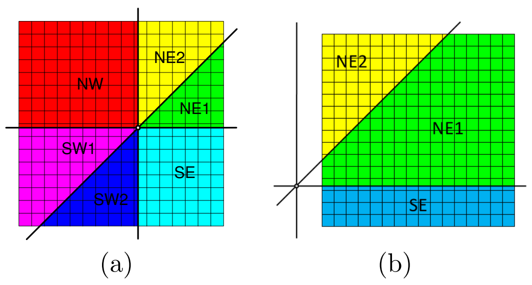

Fig. 3. Sectors defined by a viewing position in a DTM. (a) Point of view projects within the DTM projection. (b) Point of view projects outside the DTM projection.

To describe the set of rules to visit triangles in a tile to guarantee back-to-front ordering, we first consider the case where tiles belong to one sector and then we consider those tiles which straddle over different sectors.

4.1 Tiles Within One Sector

We associate each sector with a unique and particular rule that defines the path in which DTM cells must be visited to guarantee a back-to-front ordering of triangles. First, let us consider a set of tiles placed within the NW sector with respect to the projection of the point of view and the projection of the viewing frustum, as depicted in Fig. 4(a).

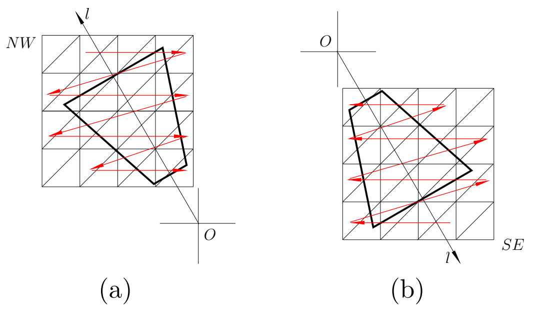

Fig. 4. Back-to-front orderings for terrain tiles within sectors. (a) NW sector. (b) SE sector.

Visiting the DTM cells following the red arrows from top to bottom and from left to right guarantees a back-to-front ordering for any line of sight l that starts at the point of view and runs through the frustum. When the set of tiles to be visited is within the SE sector all what we need to do is to follow the path defined for the NW sector but in a bottom-up and right-left order as shown in Fig. 4(b).

Now consider terrain tiles within the NE quadrant. For the DTM cell triangulation we have chosen, (see Fig. 1(a)), the relationship "closer than" applied to DTM triangles within a DTM cell, as discussed in Subsection 3.2, depends on the slope of the line of sight. Assume that the slope of the line of sight l through a DTM cell is smaller than 45° as shown in Fig. 5(a). Clearly triangle T' is closer to the point of view than T. However, if the line of sight is l' with a slope larger that 45°, triangle T is closer to the point of view than triangle T'.

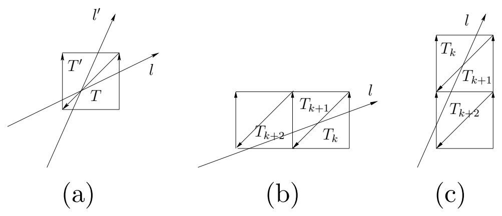

Fig. 5. "Closer than" relation among triangles in DTM cells within the NE quadrant.

A similar situation arises when considering triangles in rows or columns of a triangulated DTM. If the line of sight slope is smaller than 45° the correct back-to-front ordering of triangles is given by visiting the first rows, Fig. 5(b). When the line of sight slope is larger that 45°, triangles are properly scanned visiting cells by columns, Fig. 5(c). The same rationale applies to DTM tiles within the SW quadrant. Visiting rules for sectors SW1 and SW2 are symmetric with respect to the point of view of rules for sectors NE1 and NE2. The set of rules to be applied to tiles in sectors within either NE or SW quadrants is illustrated in Fig. 6.

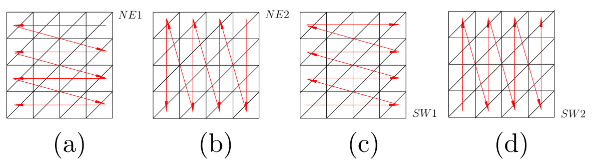

Fig. 6. Sorting triangles within the NE and SW quadrants. (a) NE1 sector. (b) NE2 sector. (c) SW1 sector. d) SW2 sector.

When the projection of the point of view falls outside the triangulation projection, some of the sectors discussed above do not appear on the projection plane (see Fig. 3(b) for example). However, the rules for visiting triangles in tiles within sectors described above still apply.

4.2 Tiles Straddling Over Sectors

In general, frustum angles are smaller than 90°. Thus the projection of the frustum onto the XY plane straddles at most over three different sectors. For a field of view on the X and Y axis of 60°, Fig. 7 shows the projected frustum as a triangle in dashed lines when the viewing position projection falls within the terrain projection.

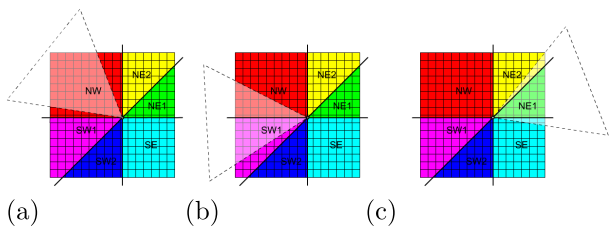

Fig. 7. Number of sectors overlapped by the field of view. (a) One sector. (b) Two sectors. (c) Three sectors.

When triangles in a DTM tile straddle over two or more sectors, an approach to solving the triangles ordering would consist in two steps. First, one could compute the set of triangles in the tile within each terrain sector overlapping the field of vision. Then we could apply the corresponding visiting rule defined in Subsection 4.1 to each set of triangles within a sector. However considering the terrain tile as the unit to be visited leads to a simpler approach.

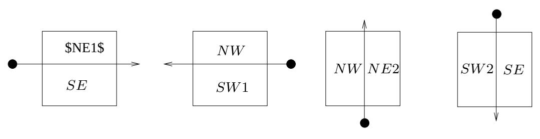

Fig. 8. Sectors involved in the ordering when tiles straddle over X and Y axis. The viewing point projection is outside the tile.

In what follows we shall call configuration the set of regions induced on a tile by the viewing position O, the local X and Y axis and the bisector B as depicted in Fig. 8 and Fig. 9.

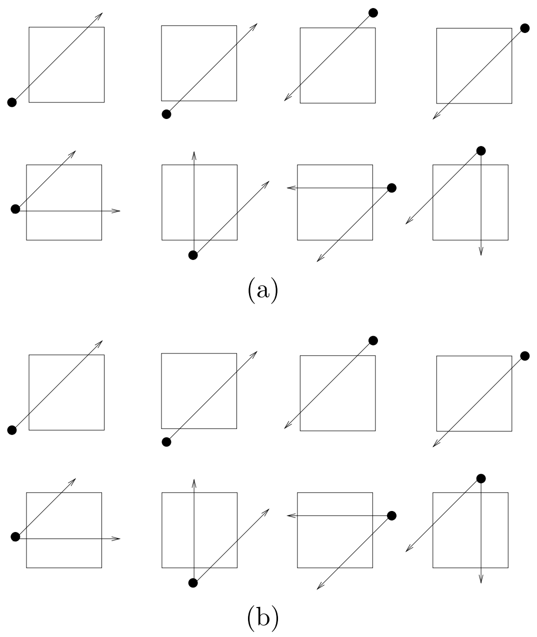

Fig. 9. Sectors involved in the ordering when tiles straddle over bisector B. (a) The point of view is projected outside the tile. (b) The point of view is projected within the tile.

In our approach, when triangles in a DTM tile straddle over two or more sectors, we first classify tiles according to the tile configuration. Then we define specific rules to visit triangles within each region in the configuration. We distinguish two situations depending on whether the projection of the point of view is outside or inside the tile. Then, within each family we consider different configurations depending on the geometry of the regions.

Terrain tiles intersected by just the local X or Y axis result in four possible types of regions shown in Fig. 8. Triangles within these tiles have in general vertices that belong to different sectors. Taking into account that triangles in these cells are trivially ordered and that the precision of the DTM representation is given by the triangle size, we coherently assign them to one of the two neighboring sectors and triangles are sorted according to the configuration specific for each sector.

When the bisector B intersects the terrain tile there are 16 different possible sector configurations shown in Fig. 9. The two rows at the top correspond to configurations where the point of view is projected outside the tile. The two rows at the bottom include configurations where the point of view is projected within the tile. Now sorting triangles in each sector is a little bit more complex.

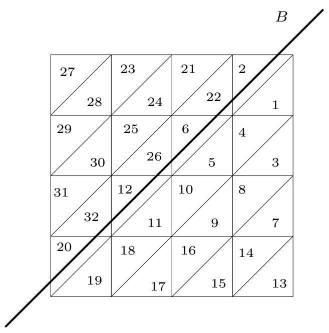

Fig. 10. Tile intersected by bisector B in the first quadrant. Pairs of triangles (22, 6), (26, 12), and (32, 20) in cells intersected by the bisector are incorrectly sorted.

To illustrate the situation, consider a tile in the NE quadrant crossed by the bisector B as depicted in Fig. 10 and assume that triangles are labeled according to the sequence generated by the back-to-front configuration associated with the NE1 sector. Assume that triangles labeled 2, 6, 12 and 20 are assigned to the NE1 sector while triangles labeled 22, 26 and 32 are assigned to the NE2 sector. Then according to what has been said in Subsection 4.1 and illustrated in Fig. 5, pairs of triangles (22, 6), (26, 12) and, (32, 20) are incorrectly sorted. Notice that labeling triangles according to the back-to-front ordering associated with the NE2 sector just would yield the symmetric incorrect result.

A way to solve the problem would be to exclude from the ordering those triangles in the cells intersected by the bisector and sort them on their own. But this would preclude from defining convenient data structures such as triangle strips. Thus, we look for a different approach by considering rectangular subregions defined by the X and Y axis and the bisector B in the terrain tile.

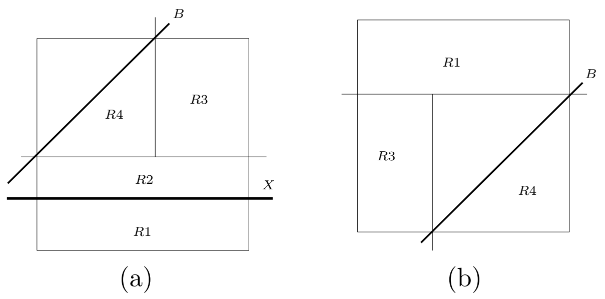

Fig. 11. Tiles intersected by bisector B in the first quadrant. a) Bisector intersects tile boundary at left and top edges. b) Bisector intersects tile boundary at bottom and right edges.

First we consider the case where the bisector B intersects the terrain tile boundary at the left and top edges as depicted in Fig. 11(a) for the NE quadrant. Here we group DTM cells into subregions labeled R1, R2, R3 and R4 respectivelly. Each of the subregions R1, R2 and R3 is fully within a different sector. Therefore triangles in each of them are sorted applying the specific already defined back-to-front ordering, that is, the ordering associated with the SE sector for subregion R1 and the ordering associated with sector NE1 for subregions R2 and R3.

Subregion R4 is always squared and we define a specific back-to-front visiting rule for the triangles in it, say the NE41 rule. The visit starts in triangles in the column of cells with the largest column grid index within the lower side of R4 with respect to the bisector. Then triangles in the terrain row with the largest row grid index within the upper side of R4 and not yet visited are considered. Successive strip triangles are sorted by alternatively visiting columns and rows of terrain cells. Fig. 12 shows the NE41 visiting rule and the correct back-to-front ordering of triangles resulted from applying the rule to the set of triangles in Fig. 10.

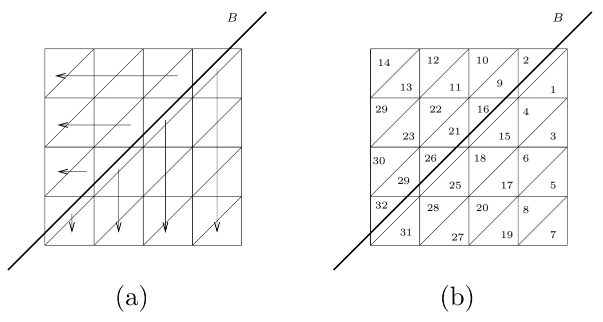

Fig. 12. Tiles intersected by bisector B in the first quadrant. (a) NE41 ordering rule. (b) Resulting correct back-to-front ordering of triangles.

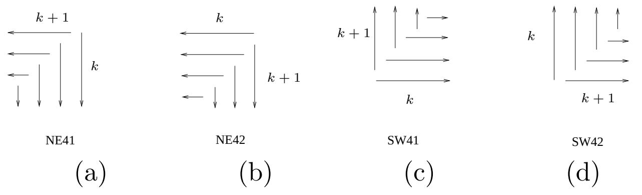

Fig. 13. Sorting rules for subsectors intersected by the bisector B. (a), (b) Sequences for alternatively traversing row triangle strips. (c), (d) Sequences for alternatively traversing column triangle strips.

When the bisector B intersects the tile boundary at the bottom edge and at the edge on the right side, the terrain tile is subdivided into regions R1, R2 and R4 (see Fig. 10(b)). Now columns and rows interchange their roles with respect to those played in the ordering for the NE41 subsector. The ordering for the NE42 subsector is shown in Fig. 13(b).

A similar rationale allows to define rules for subsectors SW41 and SW42 originated in the quadrant SW by the bisector B, illustrated in Fig. 13(b) and Fig. 13(c). Notice that these rules are symmetric with respect to the point of view of those given for NE41 and NE42 subsectors. Back-to-front visiting rules for triangles in tiles intersected by the bisector are illustrated in Fig. 13 where triangles in the strip labeled $k+1$ are closer to the point of view than those in the strip labeled k.

In conclusion, our approach includes a total of 10 different back-to-front rules to visit triangles in a triangulated DTM. Six rules correspond to tiles that are projected within a single terrain sector. Four rules are associated with tiles whose projections straddle over more than one terrain sector. On the one hand, we have considered all the possible tile-terrain sector combinations and thus the set of visiting rules is complete. On the other hand, no rule in the set can be reduced to a combination of other orderings in this set; therefore the set is minimal. Since the set of rules always sort the triangles in the terrain correctly, the resulting set of rules is correct.

5 Case Study

Based on the back-to-front ordering strategy using the rules described in Section 4 we have implemented an algorithm to realistically render triangulated DTMs. Fig. 14 illustrates conceptually how our algorithm works. Let us consider a terrain with orthographic projection onto the XY plane as shown in Fig. 14(a) and assume that the point of view is such that tiles depicted in grey are the projections of the X and Y local axis and bisector B.

To render triangles in a tile, for example the tile selected in Fig. 14(b), the algorithm identifies the sectors defined on the tile and the set of triangles in each sector as shown in Fig. 14(c). Then triangles within each sector are rendered by visiting them according to the specific rule. The result is shown in Fig. 14(d).

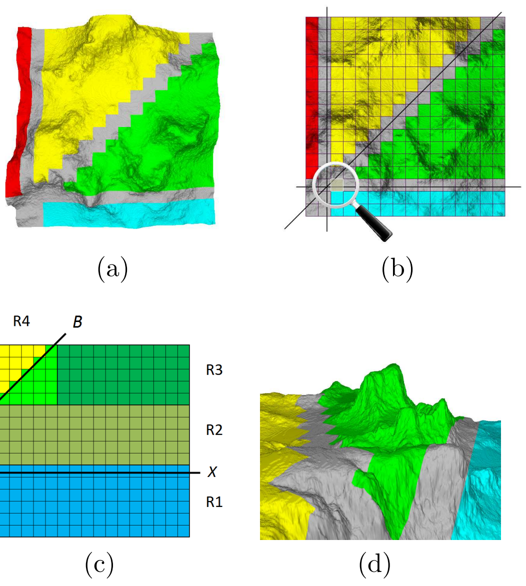

Fig. 14. Axis and bisector project on tiles in grey color. (a) Synthetic terrain viewed from the top. (b) Selecting a tile to be rendered. (c) Regions defined on the selected tile. (d) Rendered terrain.

Pseudo-code for the algorithm is listed in Algorithm 1. We assume that the algorithm is fed with the set of terrain tiles to be rendered which have been properly selected in the DTM model, the point of view O, the line of sight L and the culling quadtree depth D. The output is the rendered terrain.

The algorithm has two main parts: data initialization and rendering. In the data initialization part, the algorithm first organizes the projection onto the set of terrain tiles on the XY plane to be rendered in a uniform depth quadtree. Terrain tiles included in each quadtree node are recursively visited following a standard back-to-front quadtree ordering [13]. Tiles placed outside the viewing frustum are removed using the quadtree to speed up the culling process.

Then for each remaining tile the specific ordering rule is identified according to the terrain sector it belongs to. If the tile is included within a unique terrain sector, triangles in it are visited and rendered. When the tile straddles over more than one terrain sector, the tile is split according to the specific configuration determined by the axis and bisector on the tile boundary as shown in Fig. 8. The splitting algorithm associates with each tile fragment a visiting rule. Then triangles in each tile fragment are visited and rendered.

5.1 Results and Discussion

To compare and assess the performance of the algorithm described, we have implemented two extra algorithms. One is a DTM rendering algorithm using the standard z-buffer provided by the graphics card at hand. The other just renders triangles in tiles always using the NE rule. Clearly, this algorithm does not solve the hidden-surface problem but, as the experiments show, it yields the highest rendering frame rate among the tested algorithms. It is used as a reference and we shall refer to it as the *naive* algorithm.

Algorithm 1: Displaying a DTM by Visiting Triangles in a Back-to-Front Order

```
input  : $DTM$: a block of terrain tiles
         $O$: viewing point
         $L$: line of sight
         $D$: quadtree depth
output: A render of the $DTM$

$Q = quadtree(DTM, D)$
$CT = tilesCulling(Q, O, L)$
for each tile $T$ in $CT$ do
    $R = identifyVisitingRule(T, O, L)$
    if $R \in \{NE1, NE2, NW, SW1, SW2, SE\}$ then
        $renderTile(T, R)$
    else
        $S = splitTile(T, O, L)$
        for each fragment $S.F$ in $S$ do
            $renderFragment(S.F, S.R)$
        end
    end
end
```

The experiments have been conducted on a laptop Pentium Intel Core i7 at 2.20 GHz, with 8GB RAM, featuring an AMD Radeon HD6750M graphics card with 1GB running Visual Studio 2010 under Windows 7. The graphics API used is OpenGL and the GLUT library is used for events and window management.

The benchmark consists of in three different terrains shown in Fig. 15 represented as digital elevation models of height fields sampled on a regular grid aligned with the X and Y terrain axis. The terrain in Fig. 15(a) is a synthetic terrain. Fig. 15(b) is a section of the Grand Canyon carved by the Colorado River in Arizona (USA).[1] Fig. 15(c) shows Mount Ruapehu and Mount Ngauruhoe in New Zealand.[2]

For each terrain in the benchmark, we consider two different series of experiments. In one series, the point of view is static and in the other series the point of view moves along an arbitrary 3D path. For each case, we test three different terrain resolutions with 512×512, 1024×1024 and 2048×2048 uniformly distributed grid points respectively. For the lowest and middle resolutions, eight different quadtree subdivision depths are considered. Due to the limited available storage space, the maximum quadtree depth tested for the highest resolution case is 7.

Figs. 16 and 17 plot the number of frames per second rendered for each grid resolution and quadtree depth by the naive algorithm, the graphics card z-buffer algorithm, and our ap-

[1] US Geological Survey. Grand Canyon, USA. http://www.usgs.gov.
[2] Koordinates. Mount Ruapehu and Mount Ngauruhoe, New Zealand. https://koordinates.com.

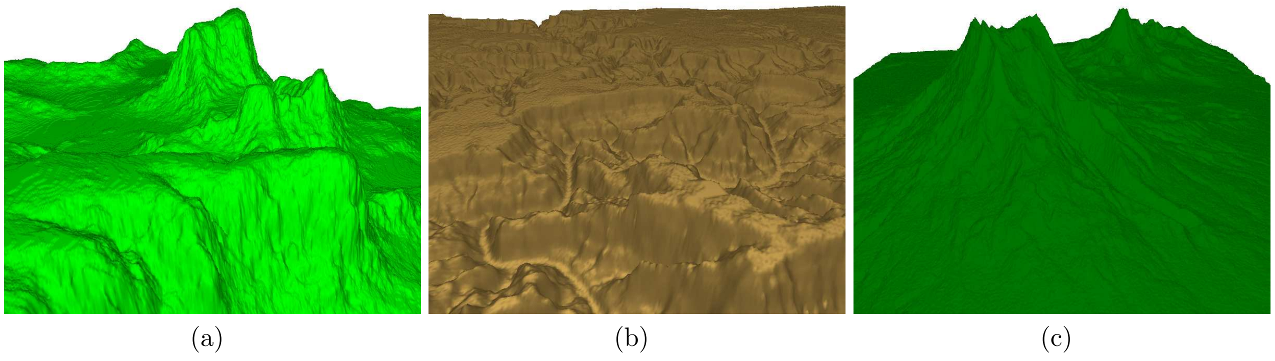

Fig. 15. Benchmark terrain models. (a) Synthetic landscape. (b) Grand Canyon, Colorado River (USA). (c) Mount Ruapehu and Mount Ngauruhoe (New Zealand).

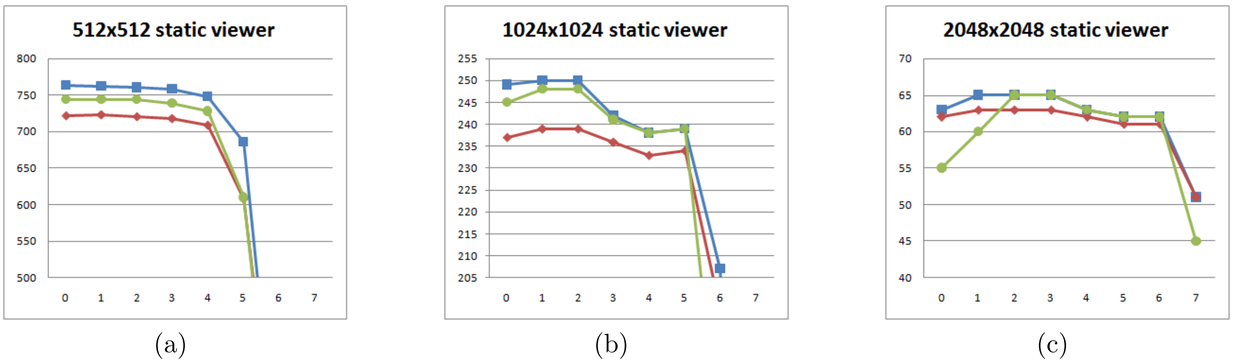

Fig. 16. Frame rate for static point of view versus quadtree depth. Height points grid of (a) 512 × 512. (b) 1024 × 1024. (c) 2048 × 2048.

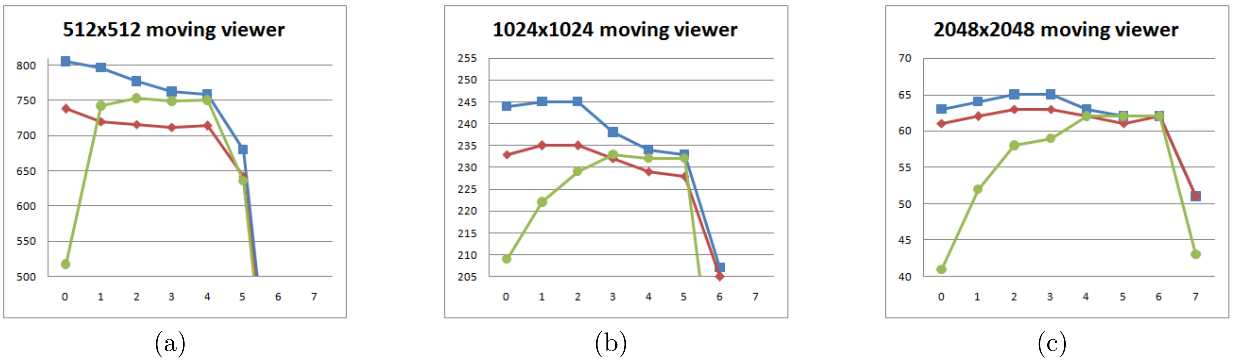

Fig. 17. Moving point of view. Frame rate versus quadtree depth. Height points grid of (a) 512 × 512. (b) 1024 × 1024. (c) 2048 × 2048.

proach respectively.

For the static point of view and 512 × 512 and 1024 × 1024 terrain precisions, plots of frame rates follow the same pattern. For small quadtree depths, curves show a plateau where the number of frames per second rendered is almost constant. Then the frame rate drops off sharply. As expected, the naive algorithm always performs better than both the graphics card z-buffer and our algorithm. In general, our algorithm performs as well as the graphics card z-buffer.

When the point of view arbitrarily moves in 3D space, the number of frames per second rendered for the terrain precisions considered shows patterns consistent with those yielded for the static point of view. as illustrated in Fig. 17. The drop off also appears for quadtree depths of about 6 and the rationale given for the static point of view also applies. For small quadtree depths, our approach always performs worse than the graphics card z-buffer. However, the performance of our approach steadily improves with the increase of quadtree depth. When the quadtree depth reaches a value of 4, the frame rate reaches the plateau where the number of frames per second rendered by our algorithm is equal to that yielded by the graphics card z-buffer.

6 DTMs with Level of Detail

Visiting triangles of large DTMs with high resolution is a challenging problem. Consequently, a number of algorithms have been developed that just visit simplified representations of terrains [1].

Multi-resolution terrain models provide efficient mechanisms to represent and manipulate DTM by optimizing the tradeoff between complexity and accuracy of representation. In level of detail multi-resolution DTM simplification schemes, terrain regions close to the point of view are approximated more accurately than regions that are far away. The reduction of the number of triangles to be visited is in general a significant part of the triangles in the DTM.

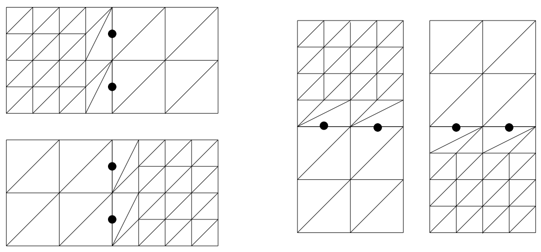
Fig. 18. Neighboring tiles with different levels of detail and stitching triangle fans.

We adopt the multi-resolution level of detail approach reported in [14] applied, for example, in [15]. In this work the level of detail of two neighboring terrain tiles can differ at most in one level. Geometry gaps at common edges of tiles with different levels of detail are avoided by changing the connectivity of height points in the higher detail tile and building a stitching fan of triangles.

Fig. 18 shows the four possible cases that arise in neighboring tiles of 5 × 5 height points. Note that stitching fans have been defined to agree with the topology of the underlying triangulation. Height points marked with a small filled circle in Fig. 18 are still part of the triangulation but they do not define any triangle to be visited.

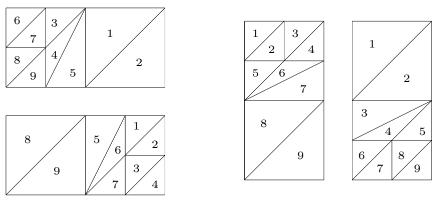
Fig. 19. Visiting sequence for triangles in a stitching fan located within viewing sector NE2.

To deal with different levels of detail, we need to define the ordering to visit the triangles in each stitching fan. Since there are four possible tile neighborhoods and each of them can be found within each terrain sector, we have to consider 24 cases. Fig. 19 shows the rules to visit triangles in the stitching fan according to decreasing distances to the point of view along the line of sight when the fan is located within viewing sector NE2. Labels in triangles define the visiting sequence. Similar rules have been defined for stitching fans located in NW, SW1 and SW2 terrain sectors.

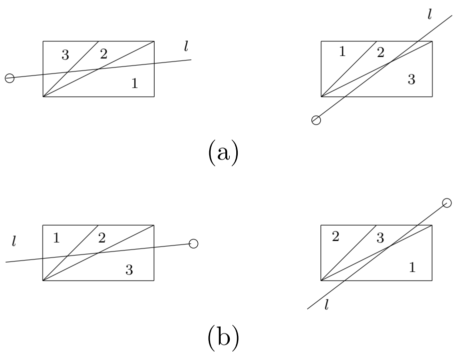
Fig. 20. Visiting sequences depend on how the line of sight intersect the stitching fan. (a) NE1 terrain sector. (b) SW1 terrain sector.

When stitching fans are located in sectors NE1 and SW1, there are two neighborhoods that require two different orderings depending on how the line of sight intersects the stitching fan. Fig. 20 depicts the orderings for these cases. The slope of the line of sight l for the NE1 sector is in the range $[0, 45°]$ while for the SE1 sector the slope is in $[180°, 225°]$. The row at the top includes the orderings for the NE1 terrain sector while the bottom row shows the orderings for the SW1 terrain sector.

7 Conclusions

Fast traversing of triangles in a DTM over a regular grid is a challenging problem in a number of fields like animation, video games and civil engineering applications.

In this paper we provided a complete and correct set of rules that define a back-to-front order according to distances to a point of view. The set includes six rules for terrain tiles the projection of which falls within one of the six sectors defined on the projection plane by the point of view and the local set of axis plus four rules to visit triangles in tiles straddling over different terrain sectors. In addition, we provide rules to visit fans of triangles that seamlessly stitch different levels of triangulations when a level of detail approach is applied to visit triangles in the DTM.

As a proof of concept, we have implemented an algorithm to realistically render DTMs defined over regular grids based on the described set of rules. To assess and compare the performance of our approach, we implemented two additional algorithms: a naive algorithm to render DTM models, and the other one using the native z-buffer offered by the available graphics card. Experimental results showed that in all cases our approach performed as well as the naive algorithm which does not solve the hidden line problem. For a static point of view, our approach performs as well as the native z-buffer algorithm. When the point of view moved along an arbitrary 3D path the performance of our approach depends on the quadtree depth used to organize the set of tiles to be rendered. Frame rate plots show a plateau for quadtree depths ranging from 3 to 6. Frame rates yielded by our implementation compare to those yielded by the z-buffer. Currently we are trying to find a rationale to explain why the algorithm performance plateau shows just for these quadtree depths.

The new algorithm described is easy to implement and experimental results show that it is robust and supports real time interaction without exploiting cutting edge graphics cards technology. The approach does not suffer from popping or cracking effects.

The visiting rules described were defined using the specific DTM triangulation chosen in Section 3 and illustrated in Fig. 1(a). If the triangulation of interest is the one shown in Fig. 1(b), the path to visit triangles that project on tile rows or columns in each rule should be reversed.

The approach described can be applied to visit triangles in a front-to-back order. All what is needed is to reverse the whole path described by each rule.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper. The material in this paper is based upon work supported by the Spanish Ministerio de Economia y Competitividad and by FEDER Euroepan Union funds under Grant No. TIN2017-88515-C2-1-R.

References

[1] Fan M, Tang M, Dong J. A review of real-time terrain rendering techniques. In *Proc. the 8th International Conference on Computer Supported Cooperative Work*, May 2004, pp.685–691.

[2] Pajarola R, Gobetti E. Survey of semi-regular multiresolution models for interactive terrain rendering. *International Journal of Computer Graphics*, 23(8):583–605, 2007.

[3] Losasso F, Hoppe H. Geometry clipmaps: Terrain rendering using nested regular grids. *ACM Transactions on Graphics*, 23(3):779–776, 2004.

[4] Wagner D. *ShaderX2: Shader Programming Tips & Tricks with DirectX 9*, chapter Terrain Geomorphing in the Vertex Shader. Wordware Publishing Inc., 2004.

[5] Frieder G, Gordon D, Reynolds R A. Back-to-front display of voxel-based objects. *IEEE Computer Graphics and Applications*, 5(1):52–60, January 1985.

[6] Wang S L C. Visibility determination on projected grid surfaces. *IEEE Computer Graphics & Applications*, 10(4):36–43, July 1990.

[7] Anderson D P. Hidden line elimination in projected grid surfaces. *ACM Transactions on Graphics*, 1(4):274–288, October 1982.

[8] Swan J E. *Object-ordered rendering of discrete objects* [PhD thesis]. Department of Computers and Information Science, The Ohio State University, 1997.

[9] Yoon H S, Jung M J, Han J H. Alternation of level-of-detail construction and occlusion culling for terrain rendering. In *CIS 2004*, LNCS 3314, pages 1161–1167. Springer-Verlag, 2004.

[10] Bhattacharjee S, Narayanan P J. Real-time painterly rendering of terrains. In *Sixth Indian Conference on Computer Vision, Graphics and Image Processing*, pages 568–575, Bhubaneswar, India, December 16-19 2008. IEEE Computer Society Press.

[11] Chen G, Sander P V, Nehab D, Yang L, and Hu L. Depth-presorted triangle lists. *ACM Transactions on Graphics*, 31, November 2012. DOI 10.1145/2366145.2366179.

[12] Deb S, Bhattacharjee S, Patidar S, Narayanan P J. Real-time streaming and rendering terrains. In *ICVGIP'06*, LNCS 4338, pages 276–288. Springer-Verlag, 2006.

[13] Samet H. *The Design and Analysis of Spatial Data Structures*. Addison Wesley Publ., Reading, MA, 1990.

[14] De Boer W. Fast terrain rendering using geometrical mipmapping. E-mersion Project.

[15] Agrawal A, Radhakrishna M, Joshi R C. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models. In *WSCG 2006*, Plzen, Czech Republic, January 30-February 3 2006. UNION Agency, Science Press.

Jesús Alonso Alonso is an Assistant Professor at the Department of Computer Science of the Universitat Politècnica de Catalunya (UPC). He obtained a Computer Science Engineering degree from the UPC and he is a researcher at the Grup d'Informàtica a l'Enginyeria (GIE) of the UPC where his research is mainly devoted to video games. He has been the Director of the UPC Multimedia and Image Center as well as the manager of a number of teaching programas including Video Games Development, Game Design, Digital Art, Animation and Mobile Apps.

Robert Joan-Arinyo is a Professor at the Computer Science Department of the Universitat Politècnica de Catalunya. He received a diploma in Chemical Engineering in 1971 and a MSc in Nuclear Power Plants in 1975 both from the Universitat Politècnica de Catalunya. After expending some years working in private companies, he received a PhD in Computer Science in 1988. His research interests include computer graphics, high-level representations for geometric design and analysis, solid modelling, geometric constraint solving, dynamic geometry and computerized medical applications. His research has been funded by NATO, Catalan Government, Spanish Ministry of Education and by a number of private companies.