Journal of Mobile Multimedia, Vol. 9, No. 1&2 (2013) 083–0100
© Rinton Press

INTERACTIVE MESH DEFORMATION IN MULTIRESOLUTION THROUGH AUGMENTED REALITY

RENAN A. DEMBOGURSKI RODRIGO LUIS DE SOUZA DA SILVA[a]
Department of Computer Science, Universidade Federal de Juiz de Fora (UFJF)
DCC/ICE, R. Loureno Kelmer, 36036-330, Juiz de Fora, MG, Brazil
ad.renan@gmail.com

BRUNO J. DEMBOGURSKI
Computer Institute - IC, Universidade Federal Fluminense (UFF) - IC/UFF
Rua Passo da Ptria 156 - Bloco E, 24210-240, Niteri, RJ, Brazil
bdembogurski@ic.uff.br

This work presents a method that allows the deformation of a terrain by modifying its heightmap in an augmented reality environment. The hierarchical structure of A4-8 meshes was used to represent terrains. This structure defines a parameter space to calculate the coordinates of a field in the $\mathbb{R}^3$ Euclidean space. In particular, this paper deals with the problem of modeling spherical terrains. An error metric dependent on the observer and the geometry of the land used for its observation and modeling. The results demonstrate that the use of A4-8 mesh combined with the tangible augmented reality system is flexible to shape spherical terrains and can be easily modified to deal with other topologies, such as the torus and the cylinder. The development of an efficient and intuitive to use method for mesh generation, based on augmented reality markers, is the main contribution of this work.

Keywords: Augmented Reality, Mesh Deformation, Terrain Generation, Parametrization

1 Introduction

Mesh deformation is an important resource for the object modeling area, allowing the modification of a surface to suit a particular purpose. Due to its importance, several deformation techniques have been developed, as seen in [1], [2] and [3], where some of these take into account the decomposition of highly detailed surfaces in hierarchical levels [4], [5]. Representing a surface by multiple levels of detail allows changes to be made at any of these levels, resulting in a fine control of the mesh.

A hierarchical structure that has favorable properties for representation and modeling of a terrain at various levels of detail is a semi-regular 4-8 adaptive mesh (A4-8 mesh [6]). Through this structure, it is possible to represent terrains with different resolutions, allowing punctual control over the generated mesh. The A4-8 can also define a parametric space to calculate coordinates on a $\mathbb{R}^3$ Euclidean space.

One of the possible ways to interact with 3D models is through fiducial markers in an augmented reality environment. In this scenario, real and virtual elements are combined to create the impression that they coexist in the same scene. To interact with virtual models an interface called Tangible Augmented Reality (TAR) [7] can be used, where each virtual

[a]Author address: Av. Rio Branco n 2403 apto 01 - Juiz de Fora. MG - Brazil

object is associated with a physical object and the user interacts with the virtual objects by manipulating tangible objects.

This work proposes a method to deform spheric terrains through the combination of an A4-8 structure and tangible markers. Local deformations are applied to a fraction of the overall terrain, resulting in an application that allows both visualization and modification of a terrain in real time. The A4-8 hierarchical structure provides the method with a precise control over the surface to be modified.

2 Related Work

There are not many works that deal with the problem of deforming meshes through augmented reality. Also, the focus of most studies involving object modeling and augmented reality is in the deformation models in general, or the development of applications for remote areas such as surgical operations simulation [8], clothes modeling [9] and accidents modeling [10].

A work that allows interactive modification of multiresolution polygon meshes through a 3D interface can be found in [11]. The authors propose an interface called *inTouch*, which contains a projected 3D scene where the model is modified and a 2D menu for operations. The interface has a multiresolution meshes editing subsystem, which takes as input the position of the haptic device probe projected onto the scene and the direction of the applied force.

Fig. 1. 3D model created with *inTouch* [11].

When the mesh is modified at a particular resolution, a set of vertices is moved. The change is then propagated to the higher detailed levels through mesh subdivision and to the less detailed levels through smoothing. The mesh editing subsystem receives a triangle, a contact point and a motion vector as input. In our method, the control point is a vertex and the deformation is propagated only to the resolution levels which are higher than that point.

An example of work that uses augmented reality and a tangible user interface for modeling 3D objects, can be found in [12], where a system called *3DARModeler* is presented. With this system it is possible to create a 3D model through one or several primitive geometries, apply textures, add animations, estimate real light sources and cast shadows on it. The difference between this system and the one developed in this work, with respect to modeling, is that the 3D models used by the *3DARModeler* are static. In this paper, the proposal is to dynamically change the geometry of a surface at multiple levels of detail, providing greater user interaction with the model and, also, overall control over the mesh.

The model visualization through the *inTouch* interface always shows the most refined level of the mesh. This can lead to problems when dealing with a large number of triangles in real time. An important factor to be mentioned is that, in areas far from the observer, a high level of resolution is not necessary to visualize an object. In this work, the level of detail depends not only on the object's geometry, but also the distance from the observer.

Another study that follows the same idea of the *inTouch* interface can be seen in [13]. The authors present a system for modeling 3D objects, called *ArtNova*, using haptic interface. Unlike the aforementioned interface, this system has several dynamic visualization techniques, such as vision-based navigation and automatic repositioning.

Also, in all mentioned works, the object to be deformed does not consist of thousands of triangles as the meshes required to adapt an actual terrain representation. No work in the literature, dealing directly with the interaction between user and 3D models composed of thousands of triangles in multiresolution through tangible augmented reality has been found.

3 A4-8 Meshes

Among the various categories of regular meshes, it is necessary to choose one that not only represent a terrain, but also can support simplification and refinement methods. The semi-regular meshes of type 4-8 (vertices with valence equal to 4 or 8, except at the edges) fit the desired profile to solve the problem addressed in this work. The semi-regular 4-8 mesh is a hierarchical structure for subdivision surfaces, or a complex cellular homeomorphic to a $[4, 8^2]$ Laves tiling [6].

The semi-regular 4-8 meshes can be divided into adaptive and non-adaptive. In the case of non-adaptive, there is only one hierarchy in the mesh, meaning that the modifications made in a region of the mesh affect the resolution of the entire grid. In the case of adaptive ones, there is a family of hierarchies, so that there is no interference between local modifications at each level.

This work uses adaptive meshes in order to deform a terrain at different levels of detail, independent of each other, and view it in different resolutions through simplifications or refinements. An example of this structure can be seen in Figure 2, where it is shown a basic block of the structure and two steps of consecutive refinement over it.

Consider a mesh $M = (\mathbf{V}, \mathbf{A}, \mathbf{P})$, where $\mathbf{V}, \mathbf{A}$ and $\mathbf{P}$ is the set of vertices, edges and faces (polygons) that belong to this mesh. Consider also that the set of vertices with valence equal to 4 is $\mathbf{V_4}$ and equal to 8 is $\mathbf{V_8}$. The *quaternary subdivision* refinement procedure of a mesh M can be seen in Figure 3a and is described by the following steps:

1. Split all edges $a \in \mathbf{A}$ in their average points m;

2. Subdivide all faces $f \in \mathbf{P}$ into four new faces, connecting the vertex of valence $4, v \in \mathbf{V_4}$ to the average point m of the opposite edge and connecting m to the average points of the other edges.

The *binary interleaved subdivision* procedure is illustrated by Figure 3b and is achieved through the following steps:

1. Split all edges $a = (v_i, v_j) \in \mathbf{A}$ that are made using two vertices of valence $8, v_i, v_j \in \mathbf{V_8}$;

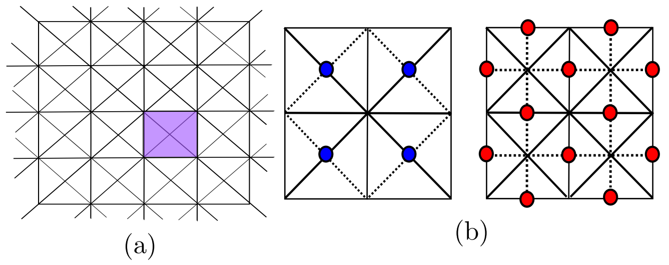


Fig. 2. Structure of a 4-8 mesh, (a) shows the basic construction block in purple and (b) shows two steps of refinement, the first generates the vertices in blue and the second, the vertices in red

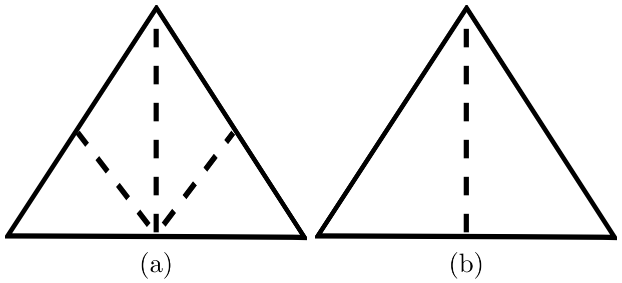


Fig. 3. (a) Quaternary subdivision prototype and (b) binary subdivision prototype.

2. Subdivide all faces $f \in \mathbf{P}$ into two sub-faces, connecting the vertex of valence $4, v \in \mathbf{V_4}$, to the average point m of the opposite edge.

An A4-8 mesh can be constructed using two approaches: top-down and bottom-up. The top-down approach, also known as refinement method, characterizes itself for creating a mesh in the lowest possible resolution and, progressively, increasing the resolution through the creation and addition of new triangles to the mesh. The bottom-up approach, or simplification method, creates a mesh in the highest possible resolution and, in a similar way, reduces the resolution by uniting the mesh triangles.

In this work, the mesh is constructed using a strictly top-down approach in the tessellation process. This way, it is possible for the final application to maintain its efficiency even with modifications being done to the mesh in real time.

3.1 ***Triangle*** *Bintree*

This work uses the binary triangle tree structure, or triangle *bintree* [14], to represent a terrain. Consider an A4-8 mesh M with a set of vertices $\mathbf{V} \subset \mathbb{R}^2$ given by $\mathbf{V} := \{\vec{v_i} := (u_i, v_i) \in \mathbb{R}^2, i = 1, ..., n\}$, where n is the number of vertices. A triangle bintree is a binary tree where, geometrically, every node of the tree represents a triangle.

The root triangle, $T = (\vec{v_a}, \vec{v_0}, \vec{v_1})$ where $\vec{v_a}, \vec{v_0}, \vec{v_1} \in \mathbf{V}$, is define as a right isosceles triangle in the coarser level, $(l = 0)$. In the next level, $l = 1$, the children of the root are defined through the insertion of an edge defined between the apex vertex $\vec{v_a}$ and the middle point $\vec{v_c} \in \mathbf{V}$ of the base edge $(\vec{v_0}, \vec{v_1})$. The left children is $T_0 = (\vec{v_c}, \vec{v_a}, \vec{v_0})$ and the right children is $T_1 = (\vec{v_c}, \vec{v_1}, \vec{v_a})$. Repeating this process recursively, the rest of the tree is obtained. Figure 4 shows a few levels of a triangle bintree.

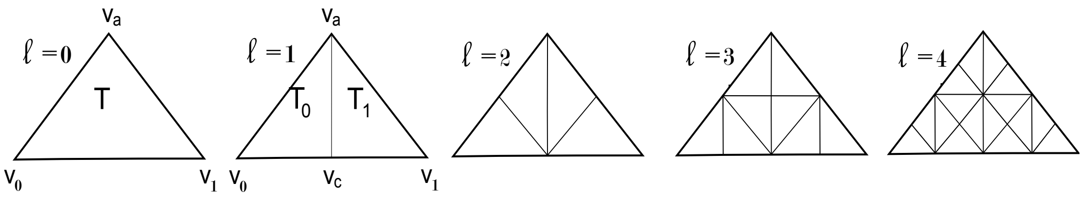


Fig. 4. Levels 0−4 of a triangle bintree.

It is important to note that the described structure needs to be adapted in a way that the A4-8 mesh can represent a spherical terrain. When subdividing a triangle at the first or last meridian, a middle point vertex must be created for both meridians. These vertices are "connected", meaning that any modification done to one of them must be applied to the other in order to prevent cracks on the mesh. This procedure ensures that the plane is conformed according to the cylinder topology.

Another important factor to be mentioned is that, for each vertex $\vec{v} \in \mathbf{V}$, two values must be stored, w e ε. The first value, w, represents the height of a vertex defined as $w = g(f(u,v)) = g(x,y,z)$, where f is a mapping function $f : \mathbb{R}^2 \to \mathbb{R}^3$ and g is a function that uses the mapped vertex coordinates, x, y and z, to calculate a height value. The resulting scalar is stored in a scalar matrix (heightmap M_a) at the vertex $\vec{v}$ coordinates u and v.

The second value, ε, represents the error value of each vertex $\vec{v} \in \mathbf{V}$. This value will be detailed in Section 5.1. In a resumed way, the vertex error is defined as $\varepsilon = \varepsilon_T$, where ε_T is the triangle error that has $\vec{v}$ as apex vertex. In practice, a triangle error value ε_T, for $T = (\vec{v_a}, \vec{v_0}, \vec{v_1})$, is stored in a scalar matrix (error map M_e) at the triangle T apex vertex $\vec{v_a}$ coordinates, u and v.

4 Parametrization

A parametric surface in the Euclidean space $\mathbb{R}^3$ is defined by a parametric equation with two parameters, named in this work as u and v. In particular, we want a mapping $f : \Omega \to \mathbf{S}$, where f is a mapping function, Ω is an open subset $\Omega \subset \mathbb{R}^2$ and $\mathbf{S}$ is a surface $\mathbf{S} \in \mathbb{R}^3$.

Orthographic projection was used as a function f to map Ω in S. More specifically, a plane, represented by A4-8 mesh, in a sphere. Consider, the vertex set $\mathbf{V}$ of a A4-8 mesh is Ω and all vertices $\vec{v} \in \mathbf{V}$ are associated with coordinates u and v in the plane. Consider also, that $0 \le u \le m_u$ and $0 \le v \le m_v$.

To define a function f in terms of latitude and longitude of a vertex $\vec{v} \in \mathbf{V}$, it is necessary to calculate the latitude of $\vec{v}$ using Equation 1 and its longitude using Equation 2.

$$latitude = \frac{(v * \pi)}{m_v} - \frac{\pi}{2}. \tag{1}$$

$$longitude = \frac{(2u * \pi)}{m_u} - \pi. \tag{2}$$

Now consider that the radius of the sphere defined by $\mathbf{S}$ is r. The domain surface V and S can be written using Equations 3 and 4, respectively.

$$\mathbf{V} = \{(u,v) \in \mathbb{R}^2 : 0 \le u \le m_u, 0 \le v \le m_v\}. \tag{3}$$

$$\mathbf{S} = \{(x, y, z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 = r^2\}. \tag{4}$$

A plane to sphere mapping function is defined in Equation 5.

$$f(u, v) = \begin{pmatrix} r * \cos(longitude) * \cos(latitude), \\ r * \sin(longitude) * \cos(latitude), \\ r * \sin(latitude) \end{pmatrix}. \tag{5}$$

From this point on in this work, $\vec{v}$ will be used as a notation for describing a vertex that belongs to $\mathbf{V}$, which is a vertex of the mesh in discrete domain. To differentiate, $\vec{\mathbf{v}}$ will be used when the vertex is in continuous domain, in other words, the vertex mapped by $f(u, v)$ which has coordinates x, y and z.

When the set of vertices to be deformed is defined, the values of height are calculated in the continuous domain for each one of these vertices. The calculation is done through a function g and the values obtained are stored in the heightmap. This way, the terrain deformation is the modification made to the values of height w of the vertices inside a deformation area, in the discrete domain.

4.1 Parameter Space Partitioning

To obtain a partition $\mathbf{N} \subset \mathbf{V}$ endowed with all vertices $\vec{v}$ that when mapped to $\mathbb{R}^3$ are within a certain distance of a control vertex $\vec{\mathbf{v_c}} \in \mathbb{R}^3$, it is necessary to discretize the continuous object that represents the area of deformation in $\mathbb{R}^3$. More specifically, it is desired to obtain a partition of $\mathbf{V}$ having all the vertices $\vec{v}$ that, when mapped by $f(u, v)$, reside within a certain distance from a control vertex $\vec{\mathbf{v_c}} \in \mathbb{R}^3$. In other words, a partition that has all vertices $\vec{v} \in \mathbf{V}$ that reside in a deformation area defined in Euclidean $\mathbb{R}^3$ space.

To obtain $\mathbf{N}$, the Euclidean distance d in $\mathbb{R}^3$ between the vertices $\vec{\mathbf{v}}^T = [x, y, z]^T$ and the control vertex $\vec{\mathbf{v_c}}^T = [x_c, y_c, z_c]^T$ is initially calculated for all vertices $\vec{v} \in \mathbf{V}$. Then, a function is defined to classify the vertices $\vec{v_i}$ as

$$I(u, v) = \begin{cases} 1, & \text{if } d \leq r_d \\ 0, & \text{otherwise} \end{cases}, \tag{6}$$

where $I(u, v)$ is a function that uses the coordinates u and v of a vertex $\vec{v}$ to indicate if it is inside or outside of a deformation area. The value r_d is defined dynamically by the user.

This classification of the vertex according to the I function will be called property $I(u, v)$. If the function returns one, the vertex has the property, otherwise, it has not. If the vertex has the property $I(u, v)$, it is placed in the set of vertices that are inside the deformation area. This set of vertices will be called $\mathbf{V}_d$.

It's Important to note that the set $\mathbf{V}_d$ is defined in the discrete domain, but the distance is calculated in the continuous domain. This happens because, if the distance was calculated in the discrete domain, the resulting deformation area would be distorted by the mapping according to its location on the sphere.

5 Error Metric

This work uses the error metric presented by [14] in their algorithm named Real-time Optimally Adapting Meshes (ROAM). To decide if a triangle must be refined or simplified, the ROAM algorithm uses the concept of nested spaces. Being $\mathbf{C}$ the descendants of a vertex $\vec{v_i} \in \mathbf{V}$, the error $\varepsilon_{\vec{v_i}} \geq e_{\vec{v_j}}$, $\forall j \in \mathbf{C}$. This means that the error metric guarantees the monotonicity of the error values, in other words, the "parents" in a coarser level always have a higher error than their children, that reside in finer levels of the terrain. To use this error metric, the geometric error and the view-dependent error must be calculated for each triangle of the mesh.

5.1 Geometric Error

The ROAM algorithm uses the concept of a *wedgie* to define the geometric error. Consider that T is a triangle of the triangle bintree and the same notation of neighborhood from Section 3.1 will be used. A *wedgie* is defined as the world volume that has the vertices (u, v, w), in a way that $(u, v) \in T$ and $|w - w_T(\vec{v})| \leq e_T$, for a certain *wedgie* thickness $\varepsilon_T \geq 0$. The line segment from $(u, v, w - e_T)$ to $(u, v, w + e_T)$ is called thickness of a segment for a vertex $\vec{v}$.

The nested errors of the *wedgies* are calculated in a bottom-up manner. It is assumed that $\varepsilon_T = 0$ for all triangles in the finest possible level. The *wedgie* thickness of a triangle T is calculated based on the *wedgie* thickness of its children, ε_{T_0} and ε_{T_1}. Consider a triangle T with a left neighbor T_0 and right neighbor T_1. The error ε_T can be calculated as

$$\varepsilon_T = max(\varepsilon_{T_0}, \varepsilon_{T_1}) + |w(\vec{v_c}) - w_T(\vec{v_c})|,$$

where $\vec{v_c}$ is the vertex obtained from the bisection of the edge opposite to the apex vertex in the triangle T and $w_T(\vec{v_c})$ is given by

$$w_T(\vec{v_c}) = \frac{(w(\vec{v_0}) + w(\vec{v_1}))}{2}.$$

5.2 View-dependent Error

The view-dependent error, called geometric screen distortion, is simply the calculation of the geometric distortion. This calculation represents the distance between the position where each vertex of the surface $\mathbf{S}$ should be in the screen space and where the triangulation touches the screen.

Be $s(\vec{\mathbf{v}})$ the correct position in screen space of a vertex $\vec{\mathbf{v}}$ given by $f(u_{\vec{v}}, v_{\vec{v}})$ and $s_T(\vec{\mathbf{v}})$ its approximation by the triangulation T. The error in this point is defined as:

$$dist(\vec{\mathbf{v}}) = \|s(\vec{\mathbf{v}}) - s_T(\vec{\mathbf{v}})\|$$

In the entire image, the maximum error is given by $dist_{max} = max_{\vec{v} \in \mathbf{V}_f}(dist(\vec{\mathbf{v}}))$, where $\mathbf{V}_f$ is the set of vertices $\vec{v}$ of the mesh that, when mapped to the sphere, reside in the view frustum (region in space that appear on the screen).

In practice, a superior limit is calculated for the maximum distortion. For each triangle T in the triangulation, a local limit is calculated, projecting the *wedgie* in the screen space. This limit is defined as the higher thickness size ε_T of all vertices $\vec{v} \in T$ projected to the screen.

6 Proposed Method

In order to deform a terrain through the modification of its height map, a method is proposed with the following steps: Definition of the control point, definition of a deformation area and propagation of heights and errors. The first stage defines where on the terrain the deformation is made. The second stage uses the control point to define a dynamic area of deformation. The last step is the most important, where error and height values are propagated to the vertices of the terrain contained within the area of deformation. Figure 5 shows the steps of the proposed method and their relation with the overall system. The system parts that are independent from the method are shown in blue, the method steps are depicted in red and the inputs controlled dynamically by the user are shown in green.

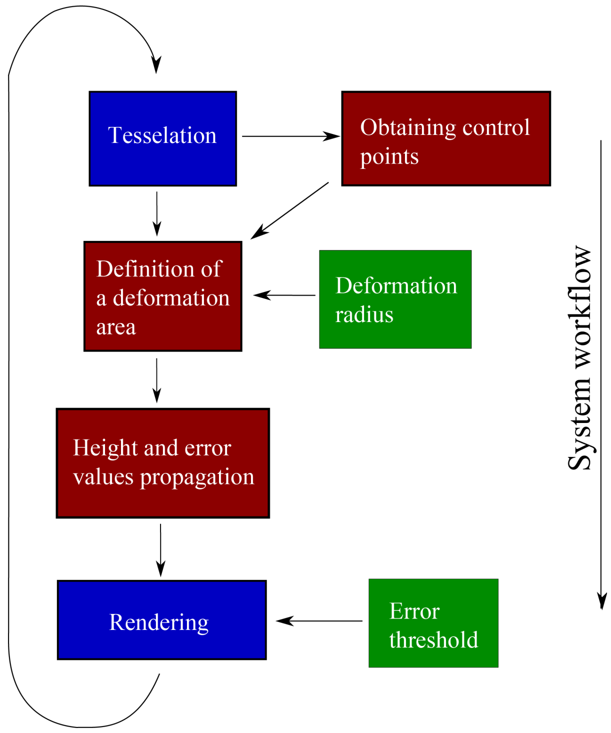


Fig. 5. Overview of the proposed method.

6.1 Control Point Definition

This work uses one control vertex $\vec{v_c}$, which defines the center of the propagation area of height and error values, where the proposed method starts. It is obtained during the mesh tessellation, representing the vertex of the mesh $\vec{v} \in \mathbf{V}$ that, when mapped to the sphere, is the closest one to the deformation object (pre-defined fiducial marker).

If the camera position approaches the area pointed by the user, it increases the model scale and the mesh goes through a refinement process that can modify the control vertex $\vec{\mathbf{v_c}}$. Likewise, if the user moves the deformation object, $\vec{\mathbf{v_c}}$ changes dynamically.

6.2 Defining the Deformation Area

The area of deformation is a set of vertices $\mathbf{V}_d \subset V$ that, when mapped to the sphere through $f(u,v)$, reside at a distance less than or equal to a radius in relation to a control point $\vec{\mathbf{v_c}} \in \mathbb{R}^3$ (Section 4.1). This value is dynamically defined by the user through the user interface and will be called, henceforth, *radius of deformation*, or r_d. The area of deformation is constructed according to a set of constraints described below.

Let $\mathbf{V}$ be the set of vertices of the A4-8 mesh. Subregions $\mathbf{V}_i \in \mathbf{V}$, representing partitions of $\mathbf{V}$ endowed with some property $I(\mathbf{V}_i)$, can be constructed by the following restrictions:

1. $\bigcup_{i=1}^{n} \mathbf{V}_i = \mathbf{V}$;
2. $\mathbf{V}_i$ is a connected region, $i = 1, 2, ..., n$;
3. $\mathbf{V}_i \bigcap \mathbf{V}_j = \emptyset \mid \forall\ i,\ j$ where $i \neq j$;
4. $I(\mathbf{V}_i) = 1$ for $i = 1, 2, ..., n$;
5. $I(\mathbf{V}_i \bigcup \mathbf{V}_j) = 0$ for $i \neq j$, where $\mathbf{V}_i$ and $\mathbf{V}_j$ are adjacent.

The first restriction means that the partition should be complete, in other words, all vertices of $\mathbf{V}_i$ must be contained in $\mathbf{V}$. The second requires that all vertices of the sub-region are "connected". The third constraint indicates that different subregions should be disconnected. The fourth requires that all vertices $\vec{v_i} \in \mathbf{V}_i$ must satisfy the properties $I(\mathbf{V}_i)$. The last restriction means that distinct sub-regions have different properties.

The restrictions shown above can be reinterpreted according to the problem addressed here. A property called $I(u,v)$ is defined for each vertex of the mesh according to Equation 6. This way, it is desired to partition the terrain into two disconnected sets, a set $\mathbf{V}_d$ that has vertices with the property $I(u,v)$ and another complementary to it $\bar{\mathbf{V}}_d$. Henceforth in this paper, the vertices of $\mathbf{V}_d$ will be called internal and vertices of $\bar{\mathbf{V}}_d$ will be called external.

The formulation proposed above follows the same principle of the image segmentation procedures, where usually one wants to separate the background and a region of interest. This choice is due to the fact that the almost completely regular structure of the semi-regular meshes allows the use of traditional algorithms of image processing.

6.3 Region Growing

The region growing technique was chosen as the implementation of the expansion function to define the set $\mathbf{V}_d$ ([15]). The starting point is the control vertex $\vec{v_c}$ and the expansion occurs in all of its neighbors directions while there are vertices with the property $I(u,v)$. Henceforth, the set of neighbors of $\vec{v_c}$ that have the property $I(u,v)$ will be called $\mathbf{V}_{d_c}$.

Initially, the algorithm tests the neighbors of $\vec{v_c}$ and the vertices with the property $I(u,v)$ are put in a set of vertices called $\mathbf{V}_{d_c}$. This set is passed to a recursive expansion function so the set $\mathbf{V}_d$ can be obtained. This function expands $\mathbf{V}_{d_c}$ from $\vec{v_c}$ toward the direction of each neighboring vertices $\vec{v_i} \in \mathbf{V}_{d_c}$. The neighborhood is covered in the A4-8 mesh, but the distance test is done in Euclidean space $\mathbb{R}^3$. An example of the test of a vertex $\vec{v_i}$ neighbor of $\vec{v_c}$, through the distance $\vec{\mathbf{d_{ic}}}$ between them, can be seen in Figure 6(a). The final set of vertices $\mathbf{V}_{d_c}$ can be seen in Figure 6(b).

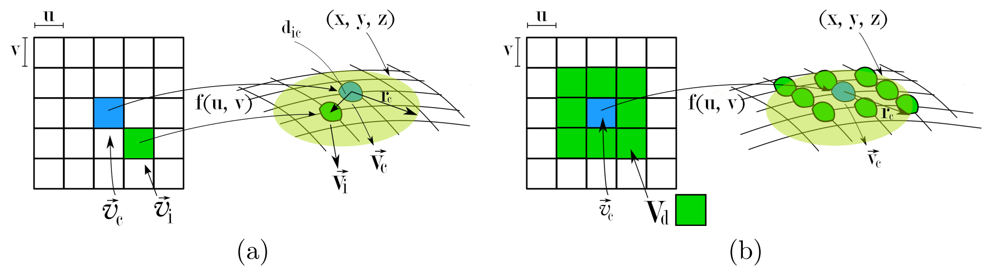


Fig. 6. (a) Beginning of the expansion process. The control vertex is $\vec{v_c}$ in the discrete domain and $\vec{\mathbf{v_c}}$ in continuous domain. The radius that defines the area of deformation is r_d and $\vec{\mathbf{d_{ic}}}$ is the distance in $\mathbb{R}^3$ between the control vertex $\vec{v_c}$ and of its neighbors $\vec{v_i}$. (b) The set $\mathbf{V}_{d_c}$, in green, represents the vertices that have the property $I(u, v)$.

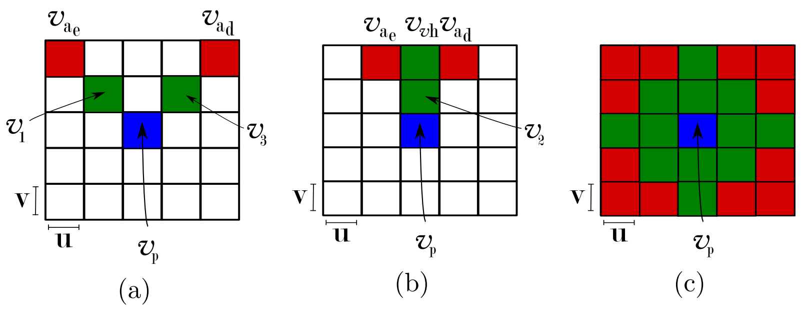


Fig. 7. Diagonal expansion example (a) and vertical/horizontal example (b). A single algorithm pass, after all neighbors of $\vec{v_c}$ are calculated, is presented in (c).

The recursive function is responsible for "walking" across the mesh in the direction of each of the neighbors of $\vec{v_c}$. It is possible to separate the eight neighborhoods of a vertex in two cases, diagonal direction and vertical/horizontal direction. A neighbor is in a diagonal direction if both coordinates of the original vertex in the parameter space, u and v, change in one step of the algorithm in that direction. For the vertical/horizontal direction, one step of the algorithm modifies only one coordinate, u or v, of the original vertex.

The combination of these two cases mentioned above allows the algorithm to go through the entire deformation area over the mesh. If the direction is diagonal, the recursion just repeats the direction of expansion of the previous step (Figure 7a). In the case of the vertical/horizontal direction, the recursion is called for the expansion direction of the previous step and its two adjacent diagonal directions (Figure 7b). The situation after one step of the algorithm for all internal vertices neighboring $\vec{v_c}$ is shown in Figure 7c.

6.3.1 Border Treatment.

When a vertex $\vec{v_i} \in \mathbf{V}_{d_c}$ is at the border of the mesh, meaning it has coordinates $u = \{0 \text{ or } m_u\}$ and/or $v = \{0 \text{ or } m_v\}$, the method needs to modify its routine to promote expansion. If the vertex coordinate reaches $u = 0$ the next step of the algorithm will map this coordinate to $u = m_u - 1$ and, similarly, if the coordinate is $u = m_u$ the next step of the algorithm will map this coordinate to $u = 1$. In the case of the v coordinate (latitude), if a vertex reaches a border coordinate, the algorithm simply stops propagating.

The leap of one coordinate for the longitude case is deliberate, reflecting that for coordi-

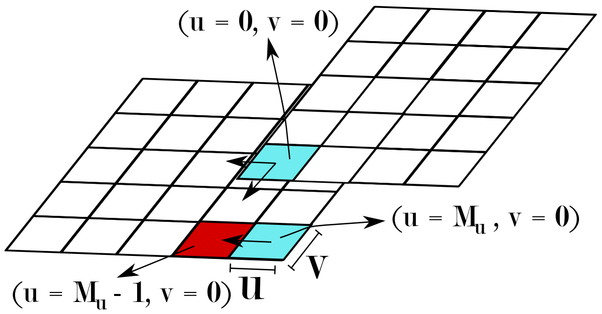


Fig. 8. Border treatment in the case of vertices trying to expand beyond the dimensions of the terrain. For the latitude, if a vertex $\vec{v}$ goes beyond the dimensions of the terrain, the expansion is interrupted in that direction. In the case of longitude, the coordinate of the vertex u is modified so that it can continue expanding.

nates $u = 0$ and $u = m_u$ there is an overlap. If a vertex has a border coordinate u on the terrain, any change made in the heightmap and error maps for this vertex must be repeated at the other end of the heightmap to prevent cracks on the terrain. An example of border treatment can be seen in Figure 8.

6.4 Error and Height Propagation

Determined the area of deformation, or set $\mathbf{V}_d \in \mathbf{V}$, it is necessary to propagate error and height values for all vertices in the set $\mathbf{V}_d$. The propagation algorithm can be defined as an extension of the area determination algorithm, since both run through all vertices that have the property $I(u, v)$.

6.4.1 Height Definition and Propagation.

The function used in this paper to calculate the height values is the Gaussian distribution function $g(x)$ described by Equation 7. The function has a maximum at the control vertex and a minimum at the border of the area of deformation. In this function, the standard deviation is σ and the average is μ.

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{7}$$

The process involves a mapping from the discrete to the continuous domain and vice versa. A vertex $\vec{v} \in \mathbf{V}$ is mapped on the sphere and tested to see whether it is or not within the area of deformation. If it is an internal vertex, the Euclidean distance in $\mathbb{R}^3$ of the vertex $\vec{\mathbf{v}}$, mapped by f, to the control vertex $\vec{\mathbf{v_c}}$, is passed to the Gaussian function defined by Equation 7 and a height value is obtained. This value is then associated with a scalar w and stored in the heightmap at its coordinates (u, v). The new value of w is accessed on the next tessellation step, where the mapping function will transmit the modification made on the heightmap to the surface of the terrain.

6.4.2 Metric and Error Propagation

In the same way as the propagation of height values, it is desired to cover all vertices $\vec{v} \in \mathbf{V}_d$ for the error propagation. One restriction to be recalled is that the error metric must preserve the monotonicity, meaning that the parents have higher error value than their children.

To propagate error in the correct way, it is necessary to use an expansion function that traverse the mesh in a hierarchical manner. This is possible using the concept of alternate step. Consider a number of iterations $i = 0, 1, 2, ..., n$ and a step size p_a. Knowing that every leaf node in the triangle bintree has error $\varepsilon_T = 0$ and that the error grows as we reduce the resolution level until the root node is reached, it is possible to notice in the tree that the position of the nodes is spaced in a homogeneous way for similar error values.

Consider, for instance, Figure 9a that shows the displacement of the vertices that have error $\varepsilon_T = 0$ in a triangle in the highest level of resolution. It is possible to notice that one can walk on the structure covering all vertices in this resolution, using steps p_l of odd steps in the vertical direction, followed by "walks" in both diagonal directions of the vertex reached by an initial step.

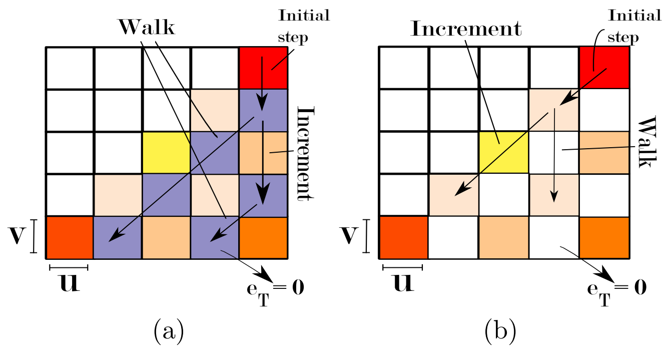


Fig. 9. Hierarchical expansion example. In (a) the first case occurs, vertical step and diagonal walk. In (b) the second case happens, diagonal step and vertical walk, both on the zero value iteration.

In a more specific way it is possible to state in this case, that the initial step of each iteration i is given by Equation 8 and the step increment i_c is given by Equation 9. In this case where the resolution is maximum, the iteration is $i = 0$, the initial step $p_l = 1$, the step increment $i_c = 2$ in the vertical direction and the step for the diagonal walk $p_d = p_l$.

$$p_l = 2^i. \tag{8}$$

$$i_c = 2 * p_l. \tag{9}$$

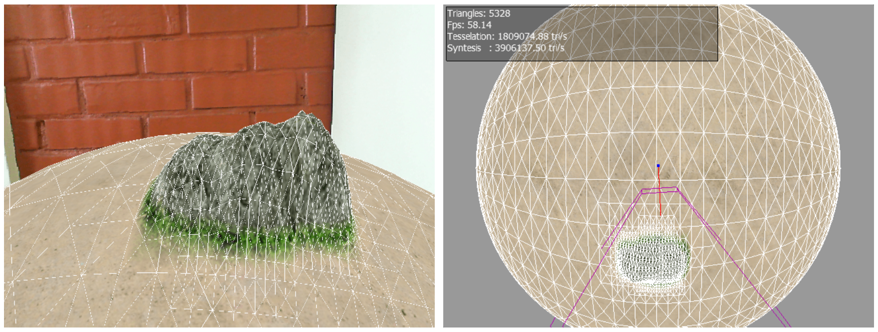
To the level immediately below the maximum, the model modifies itself (Fig. 9b). The initial step and the increment are obtained in the same way as in the previous case, following Equations 8 and 9. The iteration is again $i = 0$, $p_l = 1$ and $i_c = 2$. In this case, although, the initial step is done in the diagonal direction and the walk through the structure is done in the vertical direction (the inverse of the previous case). The recursion walks vertically with a step $p_a = i_c$, being $p_a = 2$ for this resolution.

The combination of these cases allows that all vertices are traversed. An algorithm can be easily extracted from these two cases, representing a hierarchical manner to cover the A4-8 mesh. The monotonicity is guaranteed this way, because the error can still be calculated with a bottom-up approach.

7 Results

All experiments were done on a computer with an AMD FX(tm)-6100 six cores processor, 8GB RAM and GeForce GTX 550 Ti with 2GB of RAM. The Microsoft LifeCam camera was used as video capture device. A resolution of 640×480 *pixels* was defined for all visualization windows. In order to validate the method, meshes with maximum resolution of 2049×2049 and 8193×8193 vertices were chosen. The maximum number of triangles generated by the tesselation process was defined as 70000.

Thinking of user interactivity with the spherical terrain, two viewing windows were defined. Thus, the user can see the results of tesselation on a global and local scale. In the local view window, the mesh is immersed in an augmented reality environment. In the overview window, the same mesh is viewed in a purely virtual environment so that the user can visualize the local *frustum* and the planet tesselation, enabling greater control over the modeling processes as a whole.



```
Triangles: 5328
Fps: 58.14
Tesselation: 1809074.88 tri/s
Syntesis   : 3906137.50 tri/s
```

(a) (b)

Fig. 10. Spherical terrain view. Local view window(a) and global (b).

The user is able to make basic operations such as translation, rotation and scale to better visualize the terrain. To translate it, the user moves the fiducial marker using his hands. The rotation can be done through keyboard shortcuts or hand rotation movements holding the marker. To change the terrain scale, the user must use the mouse to change the marker distance from the camera.

As the user modifies the distance from the terrain to the camera, the view-dependent error changes, and the mesh adapts itself through refinement or simplification. Figure 11 shows an example of mesh change due to terrain scale modification. The initial position of the terrain can be seen in Figure 11b, mesh modifications when the camera approaches the terrain is seen in Figure 11c and when it moves away in Figure 11a.

The deformation of a spherical terrain with the proposed method can be seen in Figure 19. The image shows a sequence of frames, from left to right, where the user performs a movement with the marker through the terrain surface causing deformation.

Here, the maximum image error $dist_{max}$ tested varied from 0.1 to 6.0. Experiments shown that the ideal value, that promote balance between visualization quality and number of triangles, is $dist_{max} = 4.0$.

The radius of deformation r_d used during testing ranged from 150km to 600km, being the radius of the spherical terrain equal to 6372.79km. In all experiments, the area of deformation

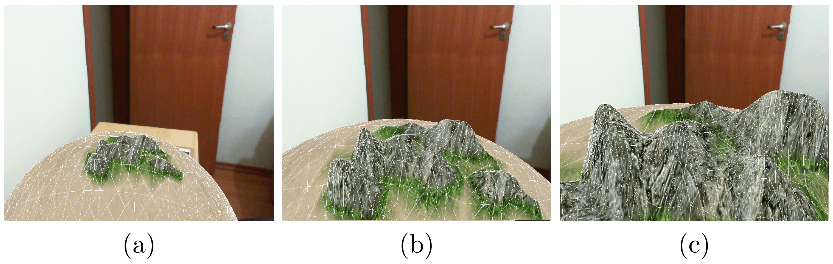


Fig. 11. Mesh modifications related to the distance between the camera and the terrain. Terrain initial position (b), camera close to the terrain (c) and moved away from it (a).

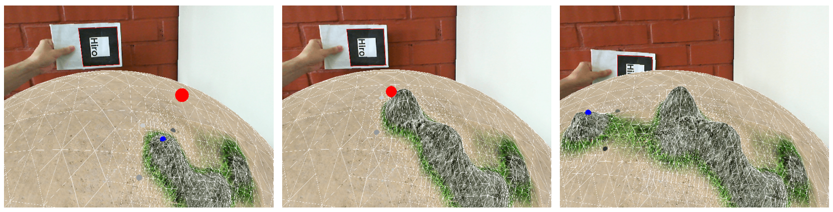
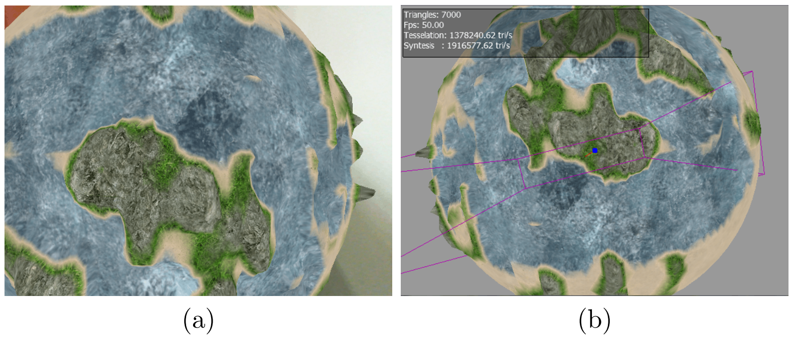


Fig. 12. A sequence of frames spaced in time showing the mesh being deformed using the proposed method.



```
Triangles: 7000
Fps: 50.00
Tesselation: 1378240.62 tri/s
Syntesis   : 1916577.62 tri/s
```

Fig. 13. Final terrain visualization generated with the proposed method. Global scale (a) and local scale (b).

change in the $\mathbb{R}^3$ space for three different radius of deformation defined as: 150km, 300km and 600km

The average time that the application takes to render one frame with the terrain being deformed was calculated for different resolutions. The camera was fixed in a single position for all values contained in Tables 15 and 17. The chosen position was kept at a distance of three times the radius of the sphere to its center, being approximately $19,000$km.

In the case of deformation, the worst case scenario was chosen to evaluate the method. In other words, the longitude was set at 180°, where the vertices have coordinates $u = 0$ and $u = m_u$ for variables values of latitude. Latitude values begin at 0° and increase 30° until they reach 180°, in other words, from one pole to the other. The average time to render a frame without deformation being applied to the terrain was 0.033 seconds.

The values in Table 15 represent the average time to render a frame with the terrain being deformed for a maximum resolution of 2049×2049 vertices. It is possible to affirm that, for a radius of deformation $r_d = 150$km, any region of the planet may be deformed without the user noticing drops in the number of frames per second. For a radius of $r_d = 600$km, the application takes three times longer to render a frame with deformation occurring at the pole than at the equator, but the number of vertices deformed at the pole is about 90 times greater when compared to the equator region.

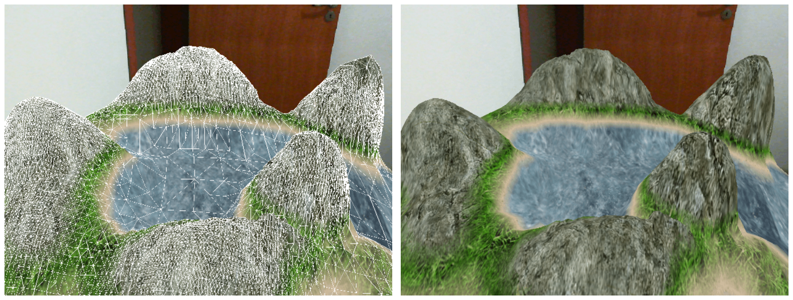

(a) (b)

Fig. 14. A complete deformation scenario. Terrain with edges drawn (a) and without (b).

Latitude	**Average time(s)**			**Latitude**	**Modified Vertices**		
0	0.064	0.096	0.117	0	32764	63471	126700
30	0.041	0.047	0.051	30	698	2789	11467
60	0.041	0.047	0.044	60	397	1593	6650
90	0.041	0.046	0.043	90	360	1431	5737
120	0.038	0.044	0.050	120	405	1629	6560
150	0.045	0.047	0.050	150	704	2916	11543
180	0.063	0.086	0.116	180	32759	63463	126888
$r_d(km)$	150km	300km	600km				

Table 1. Average rendering time of the terrain in seconds when a number of vertices are modified at a certain latitude. Maximum resolution of 2049×2049 vertices.

Table 16 shows the same values now for a resolution of 4097×4097 vertices. It is possible to notice that the same situation of the lower resolution repeats itself. The number of modified vertices increases much faster than the average time of rendering. This means that the method is efficient to deal with the number of vertices as the resolution increases. Their amount is enough to make the average rendering time increase significantly in the poles.

Latitude	**Average time(s)**			**Latitude**	**Modified Vertices**		
0	0.117	0.211	0.368	0	126955	253855	503383
30	0.043	0.050	0.067	30	2808	11356	45087
60	0.043	0.046	0.048	60	1672	6540	25920
90	0.041	0.043	0.048	90	1403	5660	22484
120	0.041	0.043	0.049	120	1627	6518	26389
150	0.044	0.047	0.065	150	2756	11267	45102
180	0.116	0.203	0.369	180	126959	253755	503400
$r_d(km)$	150km	300km	600km				

Table 2. Average rendering time of the terrain in seconds when a number of vertices are modified at a certain latitude. Maximum resolution of 4097 × 4097 vertices.

Table 17 lists values for the higher terrain resolution tested, 8193 × 8193 vertices. For a deformation radius of r_d = 600km, one realizes that the average time start to rise for all tested latitudes. This means that, as the resolution increases, it is necessary to reduce the deformation radius in order to exchange application performance and display quality.

Latitude	**Average time(s)**			**Latitude**	**Modified vertices**		
0	0.356	0.711	1.328	0	507744	1006861	2013967
30	0.052	0.053	0.114	30	11302	45437	180019
60	0.045	0.052	0.098	60	6532	25752	106635
90	0.035	0.047	0.082	90	5765	23056	89330
120	0.046	0.051	0.083	120	6511	26024	101521
150	0.045	0.059	0.107	150	11329	45625	179929
180	0.350	0.720	1.333	180	507805	1007216	2013721
$r_d(km)$	150km	300km	600km				

Table 3. Average rendering time of the terrain in seconds when a number of vertices are modified at a certain latitude. Maximum Resolution 8193 × 8193 vertices.

A complete deformation scenario can be seen in Figure 14. A planet generated with the proposed method is shown in Figure 13. The implementation of the proposed method can cope with deformations in real time even if the speed of rotation of the planet is set to a high value. It is important to note that adaptive meshes are usually used only for visualization, here the final application can both visualize and modify a terrain in real time.

7.1 Particularities

One the the relevant factors to deform a terrain is the calibration of the standard deviation in the deformation function $g(x)$ calculated by Equation 7. Like mentioned in Section 6.4.1, standard deviation values that are too small or big can lead into undesired aspects of the terrain, as it can be seen in Figure 18. Sample resulting deformations can be seen in Figures 18a and 18b, using standard deviation values of $\sigma = 0.17$ and $\sigma = 0.75$, respectively. The average used in both cases was $\mu = 1$.

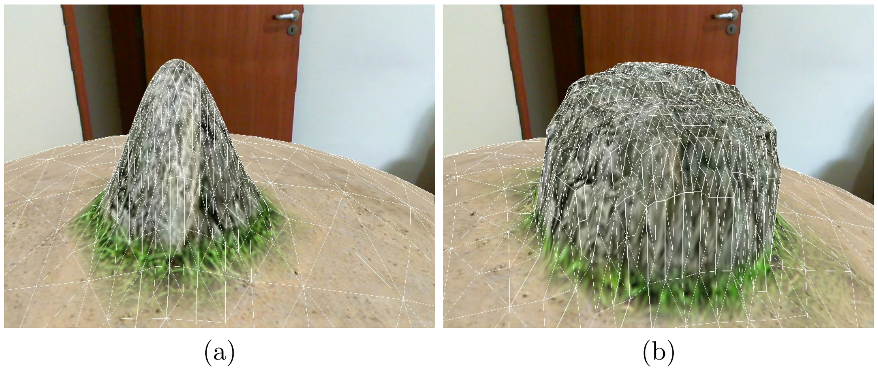


Fig. 15. Terrain deformation using the Gaussian distribution function. The deformation achieved with a value of $\sigma = 0.17$ and average $\mu = 1$ can be seen in (a). In (b), $\sigma = 0.75$ and $\mu = 1$.

Another important factor mentioned in Section 6.3.1 is the border treatment. To demonstrate how the proposed method deals with vertices that lie in the meridian, consider Figure 19. Figures 19a and 19b show the deformation result before and after the border treatment has been applied, respectively.

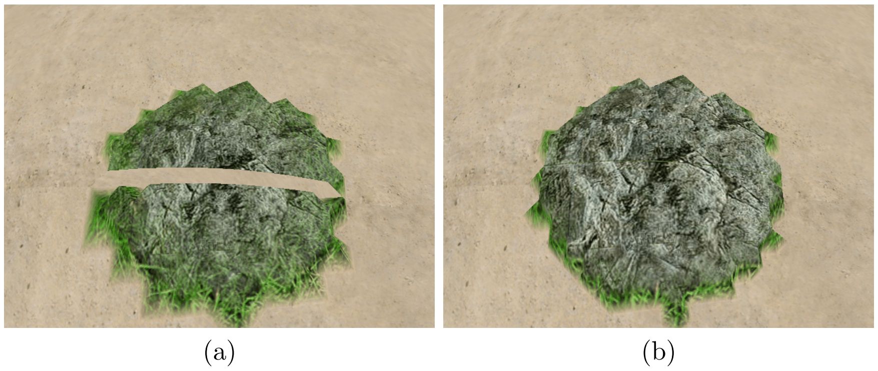


Fig. 16. Border treatment. A terrain deformed in the border region without treatment can be seen in (a). In (b) the same region, now with border treatment applied to it.

A planet generated with the proposed method is shown in Figure 13. This terrain was modeled in less than three minutes. The implementation of the proposed method can deal with deformations in real time even if the rotation speed is defined with a high value. It is possible to state that the final interface is easy to be manipulated and the deformation process is intuitive to the user that sculpts the terrain in real time.

8 Conclusion

Interactive systems providing real 3D modeling freedom are rare. In this work, we propose an interactive method for 3D modeling of spherical terrains in real time through augmented reality. The entire process is easy to be controlled by the user. Nevertheless, the generated terrain can be quite complex, natural-looking, even if defined by a small number of user actions.

The proposed method proved to be efficient in dealing with different resolutions. The increase of the average rendering time is lower than the increase in the number of modified vertices for all tests. The hierarchical propagation allows that all vertices in an area of deformation are visited, complying with the monotonicity constraint of the error metric.

The use of a physical marker to deform a mesh, provides an intuitive interface to create shapes and patterns across the terrain. This interface allows easy surface modeling and manipulation, without requiring prior knowledge in 3D modeling softwares such as *Maia* and *3D Studio*.

The combination of known techniques with the A4-8 mesh structure produced the desired results, it is possible to not only generate a terrain with a few movements, but also add detail at any resolution. Although the visualization and modeling steps are done sequentially, the application runs in real time for various resolutions. This means that a large but highly detailed terrain can be generated with the proposed method.

9 References

1. Coquillart, S.: Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In: Proceedings of the 17th annual conference on Computer graphics and interactive techniques. SIGGRAPH '90, New York, NY, USA, ACM (1990) 187–196
2. Singh, K., Fiume, E.: Wires: A geometric deformation technique (1998)
3. Sumner, R.W., Schmid, J., Pauly, M.: Embedded deformation for shape manipulation. ACM Trans. Graph **26** (2007) 80
4. Kobbelt, L.P., Bareuther, T., peter Seidel, H.: Multiresolution shape deformations for meshes with dynamic vertex connectivity (2000)
5. Sha, C., Liu, B., Ma, Z., Zhang, H.: Multi-resolution meshes deformation based on pyramid coordinates. In: CGIV, IEEE Computer Society (2007) 200–204
6. Velho, L., Gomes, J.: Variable resolution 4-k meshes: Concepts and applications. Comput. Graph. Forum **19**(4) (2000) 195–212
7. Kato, H., Billinghurst, M.: Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In: Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on. (1999) 85 –94
8. Nealen, A., Mueller, M., Keiser, R., Boxerman, E., Carlson, M.: Physically Based Deformable Models in Computer Graphics. Computer Graphics Forum **25**(4) (December 2006) 809–836
9. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques. SIGGRAPH '98, New York, NY, USA, ACM (1998) 43–54
10. O'Brien, J.F., Hodgins, J.K.: Graphical modeling and animation of brittle fracture. In: Proceedings of the 26th annual conference on Computer graphics and interactive techniques. SIGGRAPH '99, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (1999) 137–146
11. Gregory, A.D., Ehmann, S.A., Lin, M.C.: intouch: Interactive multiresolution modeling and 3d painting with a haptic interface. In: Proc. of IEEE VR Conference. (1999) 45–52
12. Do, T.V., Lee, J.w.: 3darmodeler : a 3d modeling system in augmented reality environment. Systems Engineering **4** (2010) 2
13. Foskey, M., Otaduy, M.A., Lin, M.C.: Artnova: touch-enabled 3d model design. In: ACM SIGGRAPH 2005 Courses. SIGGRAPH '05, New York, NY, USA, ACM (2005)
14. Duchaineau, M.A., Wolinsky, M., Sigeti, D.E., Miller, M.C., Aldrich, C., Mineev-Weinstein, M.B.: Roaming terrain: real-time optimally adapting meshes. In: IEEE Visualization. (1997) 81–88
15. Stockman, G., Shapiro, L.G.: Computer Vision. 1st edn. Prentice Hall PTR, Upper Saddle River, NJ, USA (2001)