Interactive Virtual Terrain Generation using Augmented Reality Markers

Renan Dembogurski, Dhiego O. Sad,
José Luiz de Souza Filho, Rodrigo Silva,
Marcelo B. Vieira
Department of Computer Science
Universidade Federal de Juiz de Fora (UFJF)
DCC/ICE, R. Loureno Kelmer, 36036-330,
Juiz de Fora, MG, Brazil
Email: ad.renan@ice.ufjf.br

Bruno Dembogurski
Instituto de Computação
Universidade Federal Fluminense (UFF) - IC/UFF
Rua Passo da Ptria 156 - Bloco E, 24210-240,
Niterói, RJ, Brazil
Email: bdembogurski@ic.uff.br

***Abstract*—This paper presents an application that allows the generation of virtual terrains interactively, using augmented reality markers. This application also allows the user to navigate in the generated virtual environment. To demonstrate how the process is done, a terrain generation scenario was chosen. Virtual objects were augmented using markers and the detection is done through the ARToolKit framework. A particle system was used to simulate deformation to better incorporate the needs of terrain generation. The deformation itself follows an interparticle force between the particles attached to a movable physical marker and the particles attached to a fixed multi-marker representing the mesh. A viscous force is also used to generate a plastic material effect ensuring permanent deformation. The resulting application although conceptually simple and easy to use, can produce an immersive output environment that the user can freely navigate.**

***Index Terms*—Navigation Environment, Augmented Reality, Terrain Visualization, Particle System.**

I. Introduction

Recently, augmented reality has been the focus of several studies. In the general context of this field, it has been used to provide a better understanding of our surroundings adding some sort of virtual information and/or content to real scenarios. In fact, it is possible to develop complete applications with a very limited amount of resources, including applications that involve computational geometry.

In this context, 3D User Interfaces present themselves as a natural way of using the inherently spatial nature of the augmented reality environment. Among the categories of the 3D UIs research field one to be noted is spatial arrangement, which studies, by definition, the properties of an array of things that have space between them. This scenario can serve as a powerful tool for developers, for instance, when generating terrains, or deforming meshes in an augmented reality environment. In fact, when associating a mesh to a physical marker it is not only possible to visualize in 3D, but the user can also rotate and translate freely using its hands, providing a higher interaction level between user and model. The final terrain can also be used as base for a virtual navigation environment.

Considering this, one of the main contributions of this work is to present an intuitive interface for modeling terrains in augmented reality and generating a virtual navigation environment based on it. We use a previously defined virtual object to deform a mesh by direct interaction, calculating collision and potential forces between them. The user interacts with the system by manually moving the marker related to the deformation object along the area of the mesh he wants to deform.

A. Related work

In the research field of 3D UIs a good survey on the categories and goals of this area can be found in [1], [2]. More relevant works closer to what is proposed in this paper can be found in [3], where the user makes freeform sketches in a 2D sketching interface so the system can generate a plausible 3D polygon mesh, and in [4] where the authors present an interactive system for generating photorealistic, textured, piecewise-planar 3D models of architectural structures and urban scenes from unordered sets of photographs. The essence of these works is followed here, the user can freely deform a 3D mesh using physical markers and then the system generates a virtual environment based on it.

There are not many works which deal with the problematic of physically based terrain deformations and the use of augmented reality. In many studies, the focus is laid on deformable models in general or applied to other areas such as surgical operation simulation [5], clothes modeling [5], [6] and crash modeling [7]. In all these cases the model which is deformed in time does not correspond to large terrains, which consists of thousands of triangles and is described by its surface only.

Regarding deformation and Augmented Reality, a related work can be found in [8], where the authors present a 3D modeling system in an Augmented Reality environment called *3DARModeler*. With this system it is possible to create a 3D model, apply textures, add animations, estimate real light sources and cast shadows. The contributions of this work related to modeling in an Augmented Reality environment include the possibility of customizing objects and the capacity of creating complex models by combining primitive geometries.

A more specific paper regarding terrain deformation in real-time is presented in [9]. This paper enumerates the methods that can be used to generate plastic deformation in a physical body represented by a mesh, and why a dynamical system (such as the one used in the presented paper) was chosen to simulate the deformations occurred.

Another work to be mentioned can be found in [10]. The authors present a deformable material to free modeling in Spatial Augmented Reality. This material can be sculpted with bare hands and, by having invisible markers immersed in a layer of white silicone, can still count with the capacity of having images projected and mapped over its surface. In the end the material represents a low fidelity sketch of a final product but already with details on it, speeding up the production process.

II. Augmented Reality

An Augmented Reality (AR) system supplements the real world with virtual (computer-generated) objects that appear to coexist in the same space as the real world. While many researchers broaden the definition of AR beyond this vision, [11] says an AR system is capable of:

- combine real and virtual objects in a real environment;
- run interactively, and in real time;
- register, or align, real and virtual objects with each other.

The ambitious goal of AR is to create the sensation that virtual objects are present in the real world. To achieve this effect, software combines virtual reality (VR) elements with the real world. Obviously, AR is most effective when virtual elements are added in real time. Because of this, AR commonly involves augmenting 2D or 3D objects to a real-time digital video image. The simplest example of visual AR is overlaying a 2D image on digital video. However, it is also possible to add 3D objects - they can be rendered so that they appear to belong to a scene containing real 3D objects. Generally speaking, adding a 3D object to real-time video improves a more impressive demonstration of AR technology.

With that in mind, Augmented Reality is a natural way to explore 3D objects and data, as it brings virtual objects into the real world where we live, rather than forcing us to learn how to navigate inside the computer. With video-see-through technology, AR handheld devices such as tablet PC™s, PDA™s, or camera cell phones (or in many cases even just a webcam and our standard computer monitor), one can hold the device up and see through the display to view both the real world and the superimposed virtual object. It is also possible to move around the display and see virtual objects, models, animations, or even a game from different views as the AR system performs alignment of the real and virtual cameras automatically.

A. ARToolKit

ARToolKit is a software library for building Augmented Reality (AR) applications. The basic idea of this framework is to add virtual objects into the real world [12].

ARToolKit uses computer vision algorithms to detect markers in an image captured by a camera. From the optical tracking of the marker, one can make the adjustment of position and orientation allowing the rendering of the virtual object, so that seems to be linked to the marker. The user can then manipulate the virtual object using a real object.

In this work two markers were used: a fixed one with multi pattern and a movable one with the 'Hiro' pattern to be detected by the ARToolKit framework. The markers are shown in Figure 1.

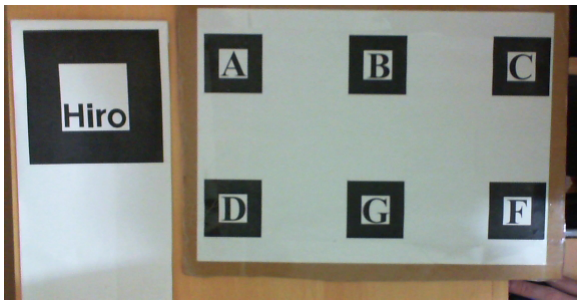

Fig. 1. Physical markers for camera position and orientation calculation.

III. Deformation Models

Deformation is a process of changing, interactively, the surface of a model in response to a control mechanism [13]. In this work, the main goal is to generate a virtual environment based on a permanently deformed 3D object represented by a mesh in an Augmented Reality environment. This mesh can be defined as a collection of vertexes, edges and faces that represents the shape of a polyhedral object.

There are several ways of implementing a system in order to generate a plastic deformed mesh. The first naive approach is to simply offset the mesh points according to some pre-defined conditions (like canonical form functions). Another way is to use physical based systems, like the spring-mass, with a viscous force in its formulation. One can also use a particle system with the same viscous force combined with an inter particles' force based on the Lennard-Jones potential. The rest of this section presents a naive method to permanent mesh deformation, a brief review of the geometry based models and a more in-depth description of the physical based models that were used in this work.

A. Naive Method

In the augmented reality environment, a simple but effective scheme can be formulated using geometric functions in their canonical forms. A fixed physical marker can serve as the coordinate system origin and a canonical volume function can be set in another movable marker to test collisions and offset mesh points. Utilizing the ARToolKit framework the distance between two markers can be obtained through the position and orientation of both markers as follows:

- For a given 3×4 matrix M of an initialized object, the first three columns contain the orientation of the marker

and the fourth column has the relative translation between the marker and the camera.
- To obtain the relative translation between two markers is necessary only to multiply the inverse matrix of the first marker by the second marker matrix. So, given that the camera is the same for both markers, the fourth column of the resulting matrix will have the translation values.
- $M_t = M_1^{-1} * M_2$.

Considering the first marker as the origin of the coordinate system, it is possible to set a mesh (physical body as a virtual object) related to this marker with a simple translation. With this relation it is possible to obtain the relative distance between the mesh and the second marker. Including the volume canonical function as another virtual object, now related to the second marker, it is possible to calculate the distance between the mesh and this volume.

To serve as an example, consider a sphere S with radius r and center $c = (x_0, y_0, z_0)$. The distance d between a mesh point $p = (x_p, y_p, z_p)$ and the center of the sphere is given as

$$d = \sqrt{(x_p - x_0)^2 + (y_p - y_0)^2 + (z_p - z_0)^2}.$$

The canonical volume form of the sphere can be expressed by:

$$r^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2.$$

In the deformation scheme, one can conclude that if a point p has its distance d to the center of the sphere inferior to the radius of the same sphere, this point resides inside it. If this happens it means that this point collided with the sphere at some point and it is necessary to offset it back to the sphere surface. This can be achieved by making a vector $\vec{v} \in \mathbb{R}^3$ to represent the direction that the point p must move in order to reach the sphere surface. This vector can be represented by:

$$\vec{v} = \frac{(p - c)}{||(p - c)||}. \quad (1)$$

The final part is to move p along the normalized vector $\vec{v}$ until the sphere volume canonical function reaches a value equals to the radius r. The overall process can be observed in 2.

B. Geometry based models

One of the most relevant approaches for deformation of 3D objects was proposed in [14], where rules are presented for global deformations (which modify the global space coordinates of a point) and local deformations (which modify the tangent space of a model). Another widely cited technique in the literature [15], [16], [17], [18], [19] is called Free-Form Deformation *(FFD)* and was presented in [20]. The main idea is to deform the space in which an object is contained (defined by a grid on it) and the object itself, through the interpolation of Bernstein polynomials.

The *FFD* was extended by S. Coquillart [21], which made possible the addition of arbitrary projections on the surface

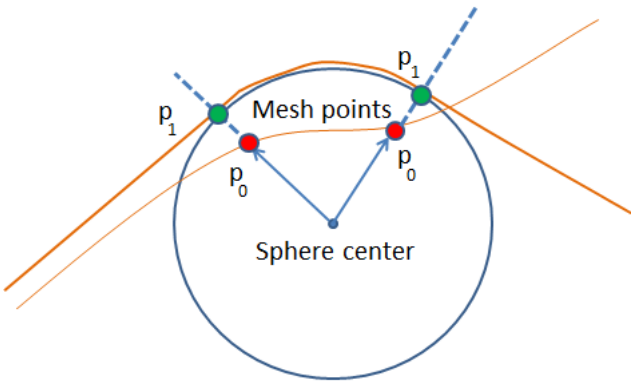

Fig. 2. Naive method, example of mesh particles offset towards the surface - from p_0 to p_1.

and fold it along a curve of arbitrary shape, i.e., the user can set their own bars on the object. The extension of Coquillart originated the name Extended Free Form Deformation of *(EFFD)*.

Other authors such as Yu-Kuang Chang and Alyn P. Rockwood [22] used a different approach to *FFD* where the distortion of the object along the Bezier curve is made by Casteljau algorithm [23].

C. Mass-Spring System

In the classical mass-spring system a physical body can be represented by a mesh with n nodes. Each node i of this mesh is then associated with a particle of mass m_i and each edge has an elastic constant K_{ij}, denoting the connection between nodes i and j. It is then calculated the force Q applied to the particle i by the spring $i \rightarrow j$ as

$$Q_i^j = K_{ij} q_j^i,$$

being the set of forces acting on a node i calculated by

$$Q_i = \sum_{j=1}^{k} K_{ij} q_j^i, i = 1, 2, ..., n.$$

A system dynamic movement equation is finally determined. This equation can be shown in a simplified form as

$$m_i \ddot{r}_i - \sum_{j=1}^{k} K_{ij} q_j^i + \sum_{j=1}^{k} \beta_{ij} \dot{q}_i^j - F_i^e = 0, \ (i = 1, 2, ..., n) \quad (2)$$

where the first term represents the mass of node i multiplied by its acceleration value, the second term relates to the elastic force, the third to a damping factor and the last to the external forces acting on the system. Being a physical based system, this equation is necessary by Newtons' second law. In other words, to have an equilibrium of the system, the sum of the forces acting on it must be equal to zero. It is often necessary to make a permanent deformation to a mesh so the system equilibrium is not reached at the original position of

the nodes. This plastic deformation effect can be achieved by incorporating a viscous force to the deformation model [24]

$$f_{\vec{v}_i} = W_d(\vec{v}_j - \vec{v}_i), \quad (3)$$

where $f_{\vec{v}_i}$ is the viscous damping force acting on the mass i, W_d is the viscous damping coefficient, $\vec{v}_j$ and $\vec{v}_i$ are the velocities of the masses i and j. Although possible, calibrating a mass-spring system for any significant model can represent a challenge to the developer [25] due to many free parameters. A possible solution that was used in this work is to develop a loosely coupled particle system.

D. Loosely Coupled Particle System

Following the same idea of physically based systems using Newtonian physics, one can replace the fixed springs between nodes (particles) present in the system by a potential field representing the interparticle relation. Associated to this field there is a potential force which must be repulsive when the distance between two particles is smaller than the rest distance, and attractive when its bigger. One possible way to define this force is to model it according to the molecular level, being an example of approximation the Lennard-Jones potential defined as

$$f_{ij} = \left(\frac{48\epsilon}{\sigma^2}\right)\left[\left(\frac{\sigma}{r_{ij}}\right)^{14} - \frac{1}{2}\left(\frac{\sigma}{rij}\right)^{8}\right] r_{ij}.$$

The f_{ij} term represents the potential magnitude, ϵ and σ are specific parameters varying with the desired material effect to be achieved, r_{ij} is the distance between two particles.

IV. Proposed System

To achieve the desired plastic material deformation effect as the terrain gets modified, the system dynamic movement equation was defined as

$$m_i\ddot{r}_i - \sum_{j=1}^{k} W_d(\vec{v}_j - \vec{v}_i) - \sum_{j=1}^{k} f_{ij} - F_i^e = 0, \quad (i = 1, 2, ..., n)$$

where the first term is the same as in Equation 2, the second term is from Equation 2, the third represents the interaction potential force and the last term relates to the external forces.

To incorporate this interaction potential, the Lennard-Jones potential, and the viscous force to the system it is necessary to know and update the position, velocity and acceleration of the system particles. This leads to the necessity of solving second order differential equations, which can be achieved using Verlet Integration. Since the velocity is implicit in the original Verlet algorithm, the Velocity Verlet algorithm [26] was used and it can be summarized as follows:

After calculating the values for time $t+\Delta t$ the acceleration can be modified by the viscous force. In this work the potential interaction force and the viscous force are calculated between a group of particles on a movable marker and the particles of the mesh. When the movable marker gets closer to the mesh a repulsion force makes the mesh particles move, gaining velocity and acceleration. As the mesh particles velocity increase, the viscous force decelerates them until they stop (see Equation 3), creating the plastic deformation effect.

Define a small Δt;

Velocity Verlet();

$x(t+\Delta t) = x(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2$;

$a(t+\Delta t) = -(\frac{1}{m})\nabla V(x(t+\Delta t))$;

$v(t+\Delta t) = v(t) + \frac{1}{2}(a(t) + a(t+\Delta t))\Delta t$;

End Velocity Verlet();

Algorithm 1: Verlet integration algorithm.

Analyzing the deformation models and the objective to be achieved, the proposed work chose to use the potential force between particles to model deformation due to its local character, meaning that deformations realized in a specific region will not affect or change the rest of the mesh. The real-time constraint also restrict the choices for this model, since updating a large number of particles has a high computational cost.

A. Post Processing

The last stage of generating a terrain is the post processing, used by the user to make fine adjustments to the mesh. In the proposed application it is possible to erode the terrain using a thermal erosion scheme and also save and export the results to different formats, which can be used by different applications. In order to smooth eventual errors introduce by the user an average neighborhood position scheme can be used.

Regarding erosion, the presented work follows the modified thermal erosion proposed in [27]. Using a Von Neumann neighborhood, the difference in height between a particle and its neighbors is checked. If it is greater than a talus angle, the mass is distributed from the central particle to the neighbors that satisfy the condition. Otherwise, the height stays the same. An example of the erosion can be seen in Figure 3, after 5 iterations.

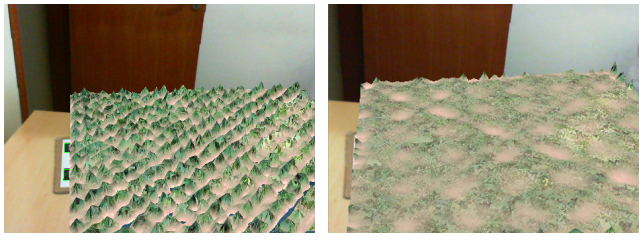

Fig. 3. The erosion post processing, on the left a noisy terrain generated and to the right the eroded terrain.

Another useful resource which helps in the creation of realistic landscapes are procedural functions such as the Perlin noise, white noise and pink noise. Those are available to the

user as a way of quickly adding detail to a mesh in any particular moment, not just a processed one. An example of a initial mesh with noise applied over it is presented in 6.

The last post processing resource in the application is the average position of neighbors. This deal with abrupt movements made by the used and the radial aspect of the potential force when deforming the mesh. The same Von Neumann neighborhood was used and, in this case, the horizontal axis positions x and z of the central particle neighbors is obtained and a simple average is applied to this central particle. This softens high frequencies that might appear throughout the mesh. The results with 10 iterations of this technique are demonstrated in Figure 4.

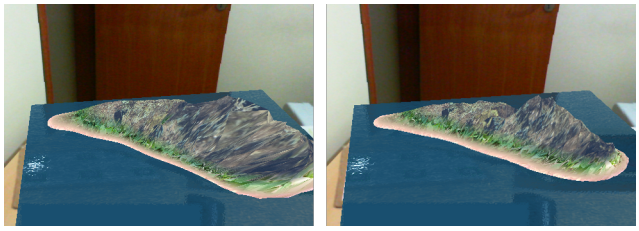

Fig. 4. Example of an abrupt terrain generated to the left, and the fixed terrain to the right.

V. System Usage

This section presents the system usage. The input methods, user interaction, post processing and connection with other applications will be described next. The whole process can be seen in Figure 5.

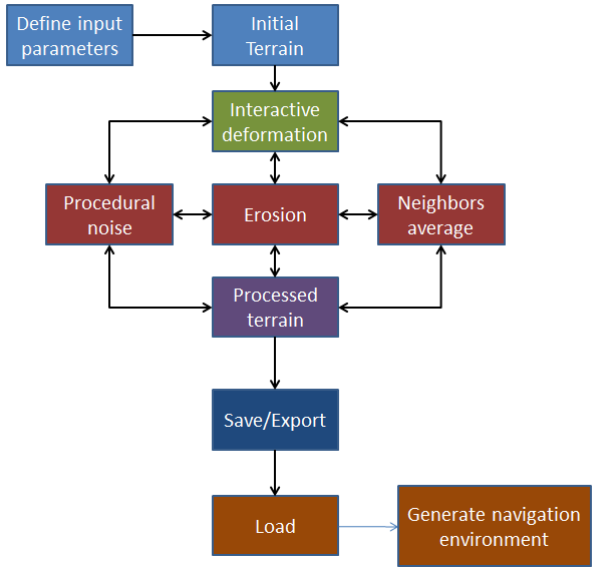

Fig. 5. Pipeline of the proposed system.

By default the system loads a planar surface as the initial mesh and the user choses the desired number of particles that will be evenly distributed over the mesh. The option of loading a precomputed heightmap as the initial mesh is also available, as it might be interesting for the user a higher level of initial detail. Procedural generation methods are also presented as another option for the initial terrain, where the user can adjust parameters such as octaves, frequency and amplitude.

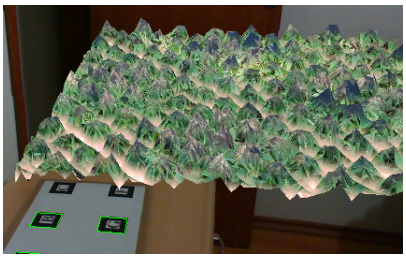

Fig. 6. A possible initial terrain generated with Perlin noise.

Following the initial configuration, the user now proceeds to deform the mesh interactively. The desired deformations are created using a movable marker, which has a specific pattern recognized by the system as the deformation tool. The user can also define the number of particles attatched to this marker, as they will be used to collide against the mesh particles. The more particles on the marker, wider is the area of contact between particles, and less accurate are the modifications on the mesh.

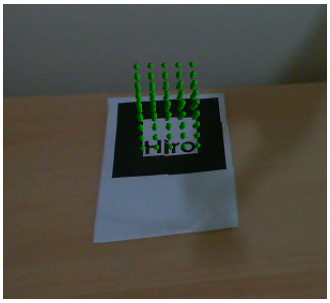

Fig. 7. Particles attached to the movable marker.

The next feature available is the post processing already mentioned in the previous section. The user can erode the terrain, generate noise over the mesh and correct abrupt movements. It is important to emphasize that both deformation and post processing can be done interchangeably. When all modifications made to the terrain are finalized, the user can proceed to generate the virtual navigation environment.

The system has a save command so the user can start to carry his augmented reality terrain into the virtual environment. The particles representing the vertices of the mesh are distributed in a matrix structure, so the system can easily read and write them as a raw format file, or as a grayscale image representing the heightfield, or even as a point cloud scalar field. With the saved terrain coordinates file in hands, the user can load this file with another system command and start to navigate through it. The final result can be seen in Figure 8.

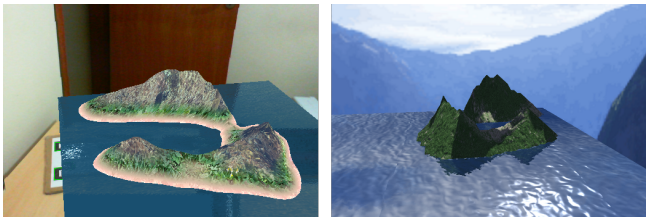

Fig. 8. Example of generated terrain to the left and to the right the visualization of the same terrain in another application.

VI. Results

The visual effects were obtained using the graphical library OpenGL and its high-level shading language GLSL. The generated images contain a multitexture support and a moving water texture with refraction and reflection. The results obtained with both the naive method and the loosely coupled particle system are shown as follows:

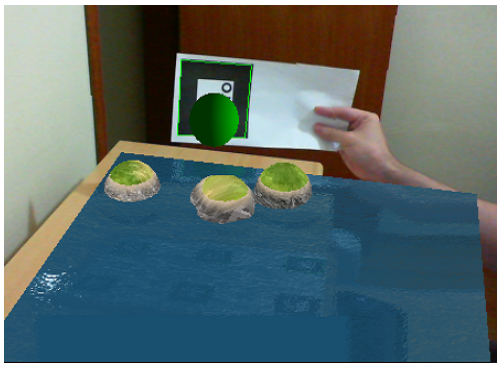

Fig. 9. Deformation using the sphere volume function.

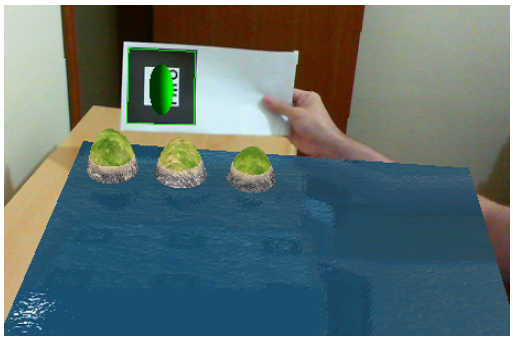

Fig. 10. Deformation using the ellipsoid volume function.

Figures 9, 10, 11 and 12 show the results using the naive method for a sphere, ellipsoid and hyperboloid canonical volume functions respectively. These examples were made utilizing 10000 particles distributed along a 400×400 grid, and although very simple and with a low computational cost the figures show that is possible, in our application case, to generate complex shape terrain.

Figures 13, 14, 15 and 16 show the results for the loosely coupled particle system used in this work. These examples

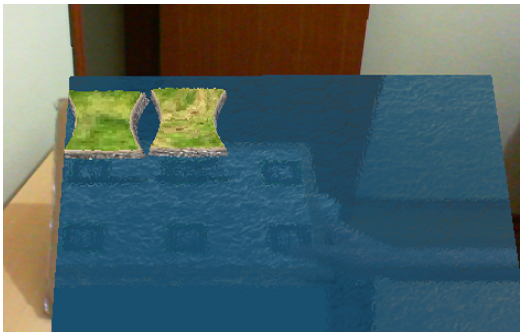

Fig. 11. Deformation using the hyperboloid volume function.

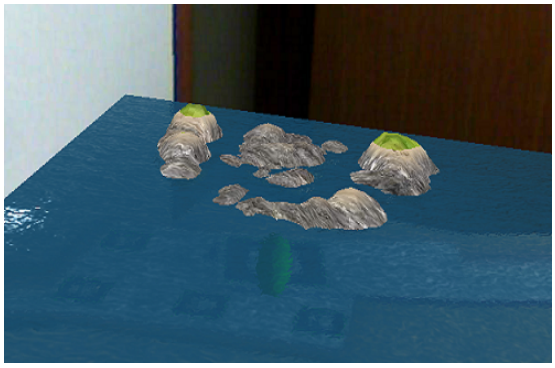

Fig. 12. Complex terrain generated using multiple volume functions.

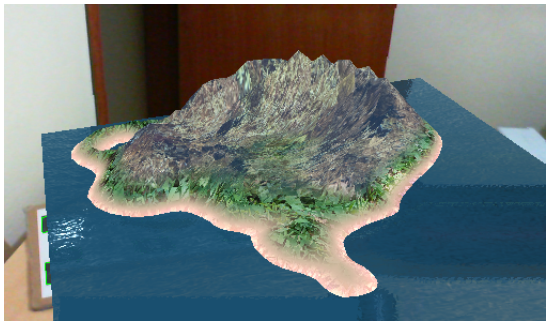

Fig. 13. Simple island created using the proposed system.

Fig. 14. Another example terrain generated with the system.

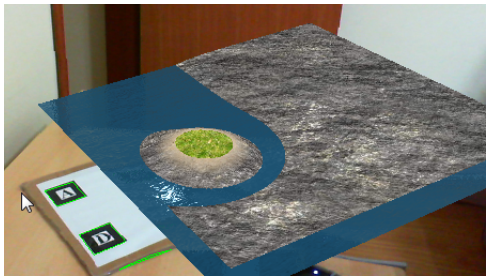

Fig. 15. Attraction force example.

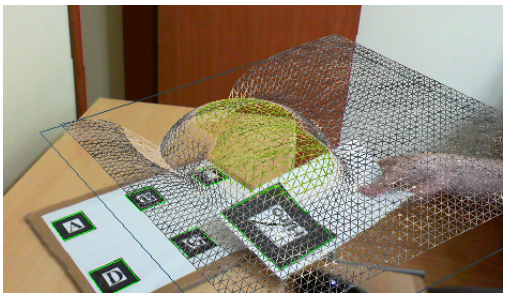

Fig. 16. Wireframe view of the surface.

Fig. 17. Final terrain to the left and the virtual environment generated based on it to the right.

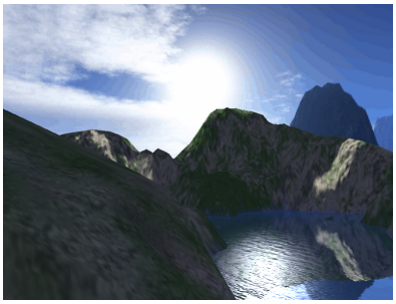

Fig. 18. A navigation moment example.

were made using 2500 particles distributed in a 400×400 grid. Since this is a more expensive method less particles were used. It is also important to note that a larger grid can be used, but for practical reasons these dimensions were chosen. The Lennard-Jones potential was calibrated with a $\sigma = 4$, $\epsilon = 20$, and the viscous damping coefficient was set to 1. In our experiments these values can be modified to increase or decrease the deformation effects.

As seen above, a simple set of parameters were used to obtain different kinds of terrains with various levels of complexity. Regardless of its simplicity, the naive method proved itself able to create terrain deformations through simple point displacement. Although the same complex terrain can be generated with the naive approach, the proposed system using a potential force provides additional advantages such as, undoing a deformation or dynamically increasing/decreasing the deformation obtained both through the interaction potential formulation. It is worth mention that all results were obtained using the same camera resolution, 640×480 pixels.

After the terrain is generated and modified by the user, the final virtual environment can be seen as in 17. It is possible to navigate through this virtual environment, moving and rotating in any direction as seen in 18. There is also collision detection implemented, so the camera will not be able to go through the mesh when forced towards it.

VII. Conclusion

Interactive systems providing real 3D freedom for modeling are rare. In this work, we propose a virtual environment generator based on an augmented reality terrain which provides an interactive 3D modeling framework. The whole process is simple to setup and easy to be controlled by the user. Despite this, the terrains generated in real time can be quite complex with natural look even with a small number of user actions.

The marker utilization, in order to deform the mesh, provides an intuitive method to create forms and patterns across the terrain. This allows an easy surface modeling without any skills on computer programming or rendering knowledge. The post processing techniques complement the initial terrain, making it simple to generate a terrain with details to some extent, and to use it in another application.

As mentioned before, it is not very common to have real time terrain deformation in an augmented reality environment. The combination of these well-known techniques produced the expected results, being possible not only to generate a terrain in a matter of seconds with few intuitive movements, but also to add details if desired.

A limitation of this work is the resolution of the equipment used. Depending on the camera one may have a small visualization volume available to create the mesh and manipulate it. Other limitation is the problem of keeping a high and stable frame rate when increasing the number of particles present on the system, after a determined limit, in order to keep processing in real time. Our work also needs a better interface to change basic desired properties such as the shape of the virtual objects dynamically or inserting procedural noise.

References

[1] D. A. Bowman and B. Frohlich, "New directions in 3d user interfaces," in *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, VR '05, (Washington, DC, USA), pp. 312–, IEEE Computer Society, 2005.

[2] D. Bowman, S. Coquillart, B. Froehlich, M. Hirose, Y. Kitamura, K. Kiyokawa, and W. Stuerzlinger, "3d user interfaces: New directions and perspectives," *Computer Graphics and Applications, IEEE*, vol. 28, pp. 20 –36, nov.-dec. 2008.

[3] T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: a sketching interface for 3d freeform design," in *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, (New York, NY, USA), ACM, 2007.

[4] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," in *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, (New York, NY, USA), pp. 159:1–159:10, ACM, 2008.

[5] A. Nealen, M. Mueller, R. Keiser, E. Boxerman, and M. Carlson, "Physically Based Deformable Models in Computer Graphics," *Computer Graphics Forum*, vol. 25, pp. 809–836, Dec. 2006.

[6] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, (New York, NY, USA), pp. 43–54, ACM, 1998.

[7] J. F. O'Brien and J. K. Hodgins, "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), pp. 137–146, ACM Press/Addison-Wesley Publishing Co., 1999.

[8] T. V. Do and J.-w. Lee, "3darmodeler : a 3d modeling system in augmented reality environment," *Systems Engineering*, vol. 4, p. 2, 2010.

[9] R. Dvorak and M. Drahansky, "Physically based real-time terrain deformations: Extensions," in *Proceedings of 18th International Conference on Computer Graphics and Vision*, pp. 74–78, Lomonosov Moscow State University, 2008.

[10] E. T. A. Maas, M. R. Marner, R. T. Smith, and B. H. Thomas, "Quimo: A deformable material to support freeform modeling in spatial augmented reality environments.," in *3DUI'11*, pp. 111–112, 2011.

[11] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Comput. Graph. Appl.*, vol. 21, pp. 34–47, Nov. 2001.

[12] H. Kato and M. Billinghurst, "Marker tracking and hmd calibration for a video-based augmented reality conferencing system," in *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, (San Francisco, USA), Oct. 1999.

[13] A. Sheffer and V. Krayevoy, "Shape preserving mesh deformation," in *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, (New York, NY, USA), pp. 39–, ACM, 2004.

[14] A. H. Barr, "Global and local deformations of solid primitives," in *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, (New York, NY, USA), pp. 21–30, ACM, 1984.

[15] D. Peled, P. Pelliccione, and P. Spoletini, *Wiley Encyclopedia of Computer Science and Engineering 6th Edition*, vol. 3, ch. Model Checking, pp. 1904–1920. New York, NY, USA: Benjamin W. Wah, 2009.

[16] W. Zhang and M. C. Leu, "A spatial warping method for freeform modeling based on a level-set method," *Comput. Aided Des.*, vol. 41, pp. 765–771, Nov. 2009.

[17] Y. Jung, H. Graf, J. Behr, and A. Kuijper, "Mesh deformations in x3d via cuda with freeform deformation lattices," in *Proceedings of the 2011 international conference on Virtual and mixed reality: systems and applications - Volume Part II*, (Berlin, Heidelberg), pp. 343–351, Springer-Verlag, 2011.

[18] S.-H. Kim, K.-H. Shin, and W. Chung, "A method for modifying a surface model with non-uniformly scattered displacement constraints for shoe sole design," *Adv. Eng. Softw.*, vol. 39, pp. 713–724, Sept. 2008.

[19] M. Pithioux, O. López, U. Meier, C. Monserrat, M. C. Juan, and M. Alcañiz, "Technical section: Parsys: a new particle system for the introduction of on-line physical behaviour to three-dimensional synthetic objects," *Comput. Graph.*, vol. 29, pp. 135–144, Feb. 2005.

[20] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, (New York, NY, USA), pp. 151–160, ACM, 1986.

[21] S. Coquillart, "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), pp. 187–196, ACM, 1990.

[22] Y.-K. Chang and A. P. Rockwood, "A generalized de casteljau approach to 3d free-form deformation," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, (New York, NY, USA), pp. 257–260, ACM, 1994.

[23] P. de Casteljau, "Outillages méthodes calcul," tech. rep., A. Citroen, Paris, 1959.

[24] D. L. Tonnesen, *Dynamically coupled particle systems for geometric modeling, reconstruction, and animation*. PhD thesis, University of Toronto, Toronto, Ont., Canada, Canada, 1998. AAINQ41520.

[25] D. Morris and K. Salisbury, "Automatic preparation, calibration, and simulation of deformable objects," *Comput Methods Biomech Biomed Engin*, vol. 11, no. 3, pp. 263–79, 2008.

[26] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson, "A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters," *The Journal of Chemical Physics*, vol. 76, no. 1, pp. 637–649, 1982.

[27] J. Olsen, "Realtime procedural terrain generation," tech. rep., University of Southern Denmark, 2004.