Real-time Multiresolution Rendering for Dynamic Terrain

Dong Wang
Chongqing Key Laboratory of Software Theory & Technology, Chongqing University, Chongqing China
Wuhan Ordnance Noncommissioned Officers Academy, Wuhan, China
Email: wangdong@cqu.edu.cn

Qing-sheng Zhu
Chongqing Key Laboratory of Software Theory & Technology, Chongqing University, Chongqing China
Email: qszhu@cqu.edu.cn

Yi Xia
Chongqing Aerospace Polytechnic College, Chongqing, China
Email: xiayi_840128@126.com

***Abstract*—This paper presents a novel dynamic terrain multiresolution rendering method by utilizing the capabilities of current generation GPUs. Firstly, the terrain depth offset map texture that represents the appropriate offset values is generated through rendering to texture, which is used to deform terrain in vertex shader. Then in order to accurately represent the fine terrain detail created by deformation, an adaptive geometry tessellation technique is implemented in geometry shader. Moreover, to update deformation area texture, we apply procedural texturing based on constraint conditions in fragment shader. In the end, the experiments prove that our method is feasible and efficient.**

***Index Terms*—dynamic terrain, geometry tessellation, multiresolution, GPU**

I. INTRODUCTION

The real-time visualization of the terrain plays an important role in computer graphics, three-dimensional geographic information systems, virtual reality and 3D games. Many excellent algorithms are proposed to realize the large scale terrain rendering. Along with the terrain visualization technology progressing, high quality and reality deformable terrain systems is more desired than before. Deformable terrain or dynamic terrain has become an increasingly important requirement for ground-based simulation systems. When it comes to virtual battlefield, the dynamic terrain techniques are essential to the visualization of crater resulting from explosion.

In this paper, we present a real-time GPU-based multi-resolution dynamic terrain visualization method to simulate crater in virtual battlefield. we develop a novel terrain deformation algorithm based on the programmable Graphical Processing Unit (GPU). The core of this algorithm is using frame buffer object(FBO) render to texture functionality to store terrain deformation process. To ensure efficient rendering of deformed terrain mesh, we introduce a new adaptive tessellation scheme for dynamic extension of resolution in deformation area that works completely on the GPU. To synthesize crater texture, the procedural texturing method based on constraint conditions that totally implemented on the GPU is proposed. The tests prove that our method is feasible and high performance. Our method can be used in dynamic terrain systems, such as war games with bomb explosions, animation with terrain deformation, etc.

The rest of the paper is structured as follows. Section 2 reviews some related work by previous researchers. Section 3 presents a terrain deformation algorithm. Section 4 describes dynamic extension of resolution technique. The texturing of the deformed area is explained in section 5. In section 6, we implement our method and show the result. Section 7 presents some conclusions on the techniques developed and outlines future work.

II. RELATED WORK

Although many existing terrain visualization algorithm focus on static terrain rending, there are still a few methods used for dynamic terrain.

Sumner et al. [1] proposed an appearance-based solution for the display of dynamic terrains. They used a four step execution cycle to create a visually-convincing depiction of terrain surface interactivity. Their method needed to manually adjust rendering parameters to produce a visually-convincing image. The need for manual adjustments suggests that this technique may not be suited for an interactive system.

He et al. [2] extended the ROAM (Real-Time Optimally Adapting Mesh) algorithm to render dynamic terrain mesh. Their method known as Dynamic Extension of Resolution(DEXTER), dynamically extended the geometry hierarchy only where necessary. This method

doi:10.4304/jsw.9.4.889-894

was a milestone in the dynamic terrain visualization. Wang et al. [3] proofed the maximum extension of transition region based on DEXTER. The ROAM algorithm was also extended to offer preservation of vertex properties and relationships with the use of a Direct Acyclic Graph (DAG).

Cai et al [4]. provided a multi-samples texture synthesis method for dynamic terrain texturing. However the main shortcoming was that they did not make use of GPU.

With the development of the graphic hardware, in order to make use of the feature of the latest graphic process unit, in 2006 Anthony et al. [5] presented a new GPU-based terrain deformation algorithm for dynamic terrain simulation. However, their approach did not dynamically extend the resolution in deformed area; moreover, the algorithm was relatively complicated that made it suboptimal.

In 2011, Wang et al. [6] presented a real-time physics-based method to simulate crater in virtual battlefield. Their crater model took account of the crater direction, and they used dynamically-displaced height map(DDHM) and crater offset map to simulate the crater deformation. However, the crater offset map was generated offline at the initialization of the whole system, so their method could not generate arbitrary position and shaped crater on the fly. Moreover they used the procedural texture technique to generate crater texture on the CPU, which was deprecated, meanwhile, the dynamic extension of resolution in crater was also absent.

In 2011, Justin Crause et al. [7] presented a new terrain deformation framework which was able to produce persistent, real-time deformation by utilizing the capabilities of current generation GPUs. Their method utilized texture storage, a terrain level-of-detail scheme and a tile-based terrain representation to achieve high frame rates. To accommodate a range of hardware, they developed two deformation schemes: one based on fragment shader, and another based on geometry shader tessellation.

III. GPU-BASED TERRAIN DEFORMATION ALGORITHM

In virtual battlefield, explosions of ammunition on soft terrain would change the topology of terrain surface and then create craters. To simulate the crater, the deformation algorithm must be presented. The core of the algorithm is using 3 framebuffer objects to generate initial terrain depth texture, crater depth texture, and depth offset map individually.

First of all, translate the height-map texture that every pixel representing the original terrain height to the initial terrain depth texture using FBO render to color texture method(Fig. 1 (a)).

Secondly, to generate crater depth texture, we need to draw a solid surface to represent crater shape, such as sphere for simplicity. Then the crater depth texture is generated through a special modelview and projection transformation using FBO render to depth texture(Fig. 1 (b)).

Upon completion of the upper two render steps, we subtract crater depth texture from initial terrain depth texture in fragment shader, and the result is rendered to depth offset map texture using FBO render to color texture method(Fig.1(c)). The depth offset map represents the vertical elevation offsets for vertices in the terrain depth texture that are impacted due to external force.

Upon completion of upper three steps, in vertex shader program we sample the initial terrain depth texture and then subtract depth offset map texture sample values to obtain the final vertices height value before further subdivision described in next section(Fig. 1 (d))

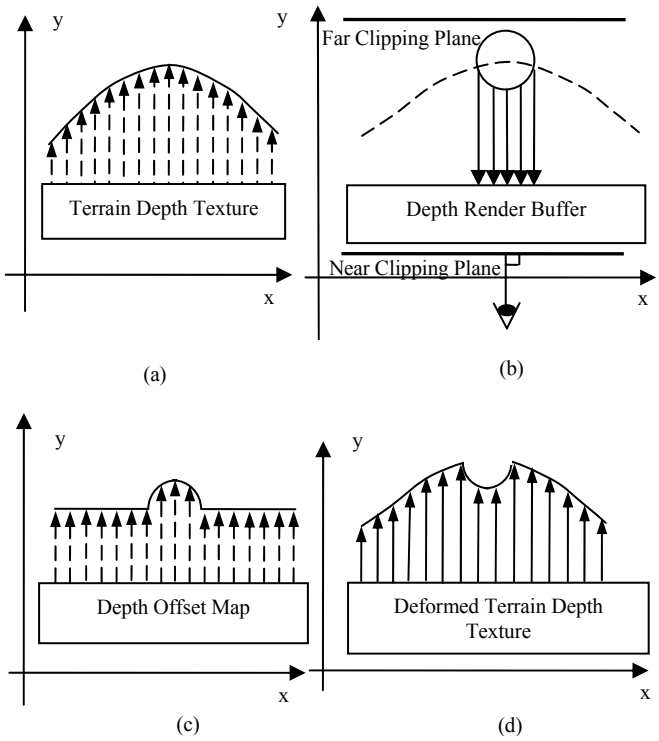

Figure 1. The dynamic terrain deforming algorithm

The pseudo-code of our GPU-based dynamic terrain visualization algorithm is as follows:

```
begin
initialize terrain depth texture
while no exit signal do
begin
   generate crater depth texture
   generate terrain depth offset map
   generate deformed terrain
end
end
```

IV DYNAMIC EXTENSION OF RESOLUTION

In static terrain rendering, the highest detail available to approximate any part of the surface is pre-determined. However in dynamic terrain applications, greater interest may be put on the deformed regions, requiring higher resolution there than on untouched regions. The dynamic extension of resolution provides additional levels of detail at the modified regions without wasting memory space

representing untouched terrain at unnecessarily high resolution. Through dynamic extension of resolution we can integrate a high detail crater into a coarse terrain mesh.

In order to realize dynamic extension of resolution, a geometry tessellation technique is required after terrain deformation implemented in vertex shader. There are two tessellation technique, one is basic tessellation using the geometry shader [8], the other is hardware tessellation using the tessellation shader [9]. Since the terrain deformation area is local to the whole terrain, the geometry shader is adequate to small-scale amplification of vertex data [10].

Deformed terrain mesh vertices generated in the vertex shader program are then assembled into triangles in graphic rendering pipeline, which go to the next shader stage–geometry shader. If the triangle located in deformed area it need to be tessellated according to some refinement pattern. Refinement patterns define how a triangle will be tessellated into sub-triangles. Each pattern can be defined as a set of barycentric coordinates. The difference between these patterns is based on the tessellation states of three vertices that make up a triangle. Each vertex can be tessellated(represented as 1) or not(represented as 0), therefore there are 8 different patterns. The pattern index ρ can be calculated from the tessellation states t_i of the three vertices v_0, v_1 and v_2 in the following formula.

$$\rho = t_0 + t_1 * 2 + t_2 * 4 \quad (1)$$

If t_0=1,t_1=1,t_2 =0,then ρ=3, means the edge formed of v_0 and v_1 need to be subdivided.

If t_0=1,t_1=0,t_2 =1,then ρ=5, means the edge formed of v_0 and v_2 need to be subdivided.

If t_0=0,t_1=1,t_2 =1,then ρ=6, means the edge formed of v_1 and v_2 need to be subdivided.

If t_0=1,t_1=1,t_2 =1,then ρ=7, means all edges need to be subdivided.

In other cases, ρ=0,1,2,4, means there are no two vertices that need to be subdivided simultaneously, so the triangle remains unchangeable.

Fig. 2 presents, on the left side, an initial rectangular triangle whose vertices are labeled as t_0, t_1, and t_2 respectively. Next, the 8 tessellation patterns are presented, with the edges of the original triangle that need refinement depicted in red.

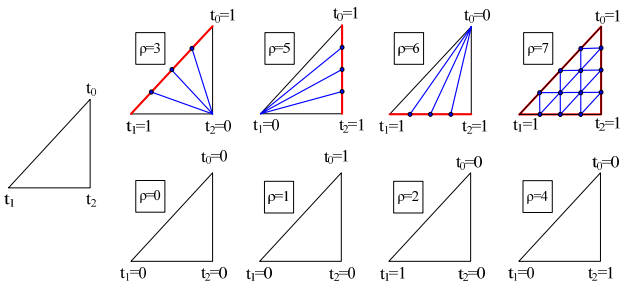

Figure 2. Tessellation pattern. The red color indicates the edges that need refinement.

Having different patterns is essential to combine tessellated and non-tessellated triangles, removing the occurrence of cracks and holes between triangles in different resolution.

Suppose we have an original triangle mesh that is composed of 3 triangle strips, and each triangle strip has 6 triangles. Fig. 3 presents one adaptive tessellation result to the original mesh using upper defined tessellation patterns. In fig. 3 the vertex, whose tessellation state t_i is equal to 1, is described as red character T, otherwise the vertex is described as blue character F, meanwhile the edge that need to be subdivided is drawn in red.

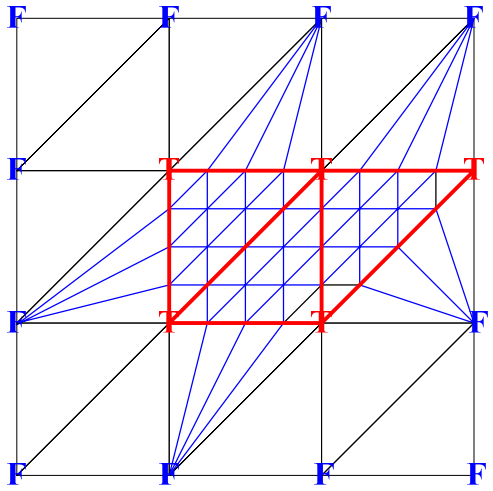

Figure 3. Example of adaptive tessellation. If vertex tessellation state=1, then represent as T. Otherwise represent as F.

Furthermore, to draw a plane terrain mesh, which has a local deformed area in it, the vertex height value can be used to calculate it's tessellation state. If the height value of a vertex is larger or less than the original terrain height value, then its tessellation state t_i will be set to 1, otherwise will be set to 0. In addition, the subdivision level that represent how many segment an a edge was divided into can be set to other value according to system requirement. We just use 4 for illustrative purposes.

V Dynamic Terrain Texturing

After subdivision, it is time to texture the deformed area. Aimed at restrict of the traditional dynamic terrain procedural texture generation [4, 6], a new dynamic terrain procedural texturing method based on GPU is proposed. The algorithm sampled multiple sample textures in the fragment shader, then use functions as constraint condition and terrain depth offset map as alpha map to synthesize crater texture [11].

We observe and study the craters on the grass. Considering a simple cases, the crater texture can be defined as 3 rings, the center of rings is the bomb point [12]. Each ring includes a smooth blending with two sample textures. The innermost ring is generated by blending charring texture with adustion texture, the middle ring is a blend of adustion texture and soil texture, the outermost ring is a blend of soil texture and grass texture, and the rest area is grass texture, just as fig. 4.

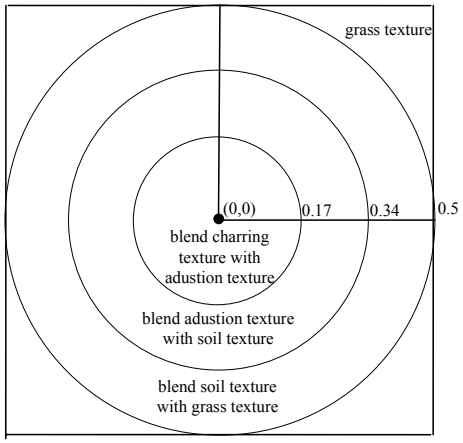

Figure 4. The constitution of crater texture

To generate the crater texture, the 4 sample textures are blended together in the fragment shader according to the distance between texture coordinate and the center of rings. So each crater texel is the result of the mix of two of them according to the following scheme(fig. 5).

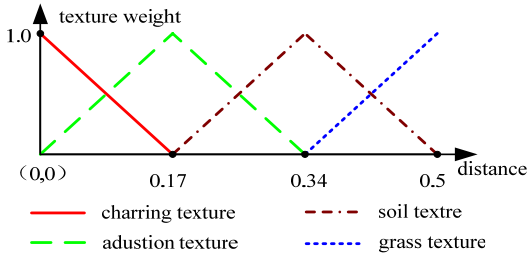

Figure 5. The constraint condition of texture blending

If the distance value is in the range [0.0, 0.17], the mix is done between charring texture and adustion texture. The red solid line represents that the charring texture weight value is 1 in the center of rings, and then decline following the increase of distance from the center. While in the same distance range, the green dot line explains that the adustion texture weight increases from 0 to 1.

Similarly, if the distance value is in the range [0.17, 0.34], the mix is done using adustion and soil textures. If the distance value is in the range [0.34, 0.5], the mix is done between soil and grass textures.

The formula used to compute the weight of a texture in the texel is the following:

$$weight=(dist - mindist) / (maxdist - mindist) \quad (2)$$

where *dist* is the distance from texture coordinate to the center of rings, *mindist* is the minimum distance related to the first texture and *maxdist* is the maximum distance related to the second texture in the blending range.

For example, if the distance value is below the limit value 0.17, the weight of the adustion texture is computed use following formula:

$$weight = (dist - 0.0) / (0.17- 0.0) \quad (3)$$

The weight of the charring texture will be *(1.0 - weight)*. In fragment shader program it's computed by the mix function in the following line of code that returns the final color of the crater texel:

crater_color = mix(charring_texel, adustion_texel, weight);

When texturing a crater in large grass terrain, the depth offset map calculated in section III can be used as the alpha map to texture the final deformed terrain.

VI IMPLEMENTATION

We have implemented our method in a simulation of virtual battlefield. Our implementations are running on a Intel Core i3 2.9GHz computer with 2GB RAM, and NVIDIA GT430 graphics card with 1G RAM, under Windows 7 system, Visual studio 2010 and OpenGL 4.3 environment. Our implementation uses GLSL for shader programming.

The rendering system has a 1024×768 size view port. The size of the initial terrain height map and the 3 render targets are all 256×256. The algorithm is implemented as a research prototype with no code tuning or low-level code optimization, and view-frustum culling is also absent. The frame rate of our system is over 150 fps. Here is a series of screen shots of our implementation.

In the implementation of the rendering system, we can change the viewpoint position easily through keyboard and mouse. The system includes two different rendering mode, fill or wire frame mode. Moreover, we can change the subdivision level through keyboard as needed at run time.

Fig. 6 shows that after generate the vertex of deformed terrain in vertex shader , 4 different subdivision levels were used in geometry shader to generate final deformed terrain mesh. The choice of subdivision level offers the opportunity to alter the granularity of the deformation, which can be used to throttle the simulation and visuals as deemed necessary by the application.

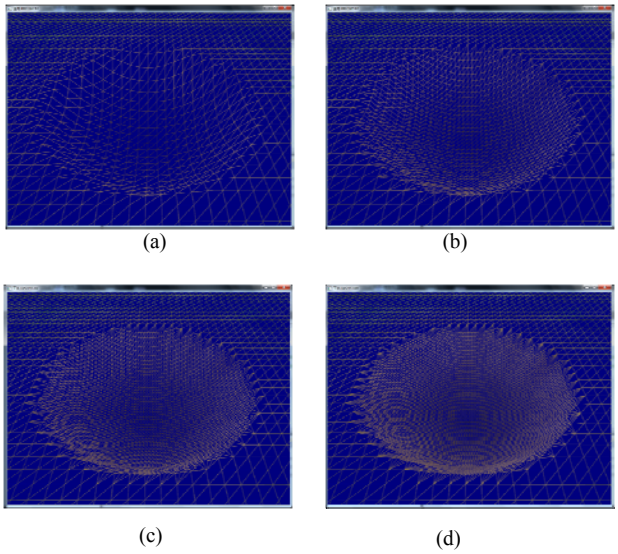

Figure 6. Geometry tessellation of deformed terrain mesh. From left to right and top to bottom, 4 different subdivision level is 1,2,3 and 4 respectively.

Fig. 7 shows the sphere that is used to generate the crater depth texture in the scene. From the figure we can observe that the crater surface envelopes the sphere.

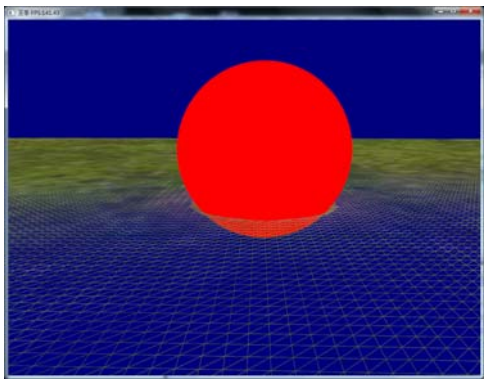

Figure 7. The sphere that partially sink into terrain is used to generate the crater.

Fig. 8 shows the craters on the grass terrain. The crater texture getting from our method enhances the sense of reality.

Figure 8. Crater on soft grass terrain

VII Conclusion and Future Work

In this paper we present a multiresolution rendering method of crater deformation on soft terrain. Firstly, we present a novel crater deformation algorithm. Our method uses 3 framebuffer objects to generate initial terrain depth texture and depth offset texture, then calculate the deformed vertex height value in the vertex shader program. Then, the deformed triangles are subdivided into different patterns in geometry shader according to the vertex tessellation state, which avoid T- junctions. Finally, a method of procedural texture based on constraint conditions is implemented in fragment shader to generate crater texture. Unlike previous work, the terrain deformation, multi-resolution rendering and texturing are all manipulated on the GPU. The experiments show that our method is feasible and high performance.

As a future possibility, we are working on using NURBS surface to generate more realistic crater shape , using view dependent method to calculate triangle subdivision level, and extending our method to realize large-scale dynamic terrain visualization.

Acknowledgment

This research is supported by the Chongqing Postdoctoral Science Foundation(No. xm201325) and the National Natural Science Foundation of China(No. 61073058 and No. 61272194).

References

[1] R. W. Sumner, J. F. O'Brien, J. K. Hodgins, “Animating sand, mud, and snow,” *Computer Graphics Forum*. vol.18(1), pp. 3-15, 1999

[2] Y. He, J.Cremer, Y. Papelis, “Real-time extendible-resolution display of on-line dynamic terrain,” *Proceedings of the 2002 Conference on Graphics Interface*, Calgary, Alberta, Canada, 2002, pp. 151-160

[3] W. Linxu, L. Sikun, P. Xiaohui, “Real Time Visualization of Dynamic Terrain,” *Chinese Journal of Computers*, vol.26(11) , pp. 1524-1531, 2003

[4] X. Cai, J. Li, H. Sun, J. Li, “Multi-samples texture synthesis for dynamic terrain based on constraint conditions,” in *Transactions on Edutainment VI,* Z. Pan, 2012, pp. 188-196

[5] A. S. Aquilio, J. C. Brooks, Y. Zhu, a. G. S. Owen, “Real-Time GPU-Based Simulation of Dynamic Terrain,” *ISVC,* Springer Berlin/ Heidelberg, 2006, pp. 891-900

[6] D. WANG, C. WANG, “Real-time GPU-based Simulation of Dynamic Terrain in Virtual Battlefield,” *Journal of Computational Information Systems*, vol. 7, pp. 1924-1933, 2011

[7] C. Justin, F. Andrew, M. Patrick, “A system for real-time deformable terrain,” *Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment*, Cape Town, South Africa, 2011, pp. 77-86

[8] O. Ripolles, F. Ramos, AnnaPuig-Centelles, MiguelChover, “Real-time tessellation of terrain on graphics hardware,” *Computers &Geosciences*, vol. 41, pp. 147-155,2012

[9] E. Yusov, M. Shevtsov, “High-Performance Terrain Rendering Using Hardware Tessellation,” *Journal of WSCG*, vol. 19, pp. 85-92, 2011

[10] X. Lai, Z. Han, "High Fidelity DEM Generation Based on LiDAR Data," *Journal of Computers*, vol. 7, pp. 2071-2077,2012

[11] Z. Chen, J. Ji, R. Li, "Asynchronous Parallel Computing Model of Global Motion Estimation with CUDA", *Journal of Computers*, vol. 7, pp. 341-348, 2012

[12] D. Su, et al, "Design and Implementation of Drainage Network Extraction Algorithm Based on Binary Linear Regression", *Journal of Computers*, vol. 8, pp. 2277-2283, 2013

Dong Wang Born in 1981. Received his M.S. degree and Ph D. degree from Academy of Armored Force Engineering, Beijing, China in 2010. Post-doctoral fellow in computing science from Chongqing University, China. His main research interests include computer graphics, virtual reality, and computer vision.

Qingsheng Zhu Born in 1956. He is a Ph D. Professor, Ph D. supervisor, College of Computer Science, Chongqing University, Chongqing, China. He was a Visiting Scholar (1993-1994) at Dept. of Computer, Birkbeck College, University of London, UK. He was a Visiting Professor (2001-2002) at Dept. of Computer Science, University of Illinois at Chicago, USA. His main research interests include business intelligence and image processing.

Yi Xia Born in 1984. Received her M.S. degree from Chongqing University, China in 2012. Her main research interests include computer graphics and computer vision.