Computers & Graphics 38 (2014) 140–149

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on CAD/Graphics 2013

Effective traversal algorithms and hardware architecture for pyramidal inverse displacement mapping

Hyuck-Joo Kwon [a], Jae-Ho Nah [b], Dinesh Manocha [b], Woo-Chan Park [a,*]

[a] Department of Computer Engineering, Sejong University, 98 Gunja-Dong, Gwangjin-Gu, Seoul 143-747, Republic of Korea
[b] University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

ARTICLE INFO

Article history:
Received 5 August 2013
Received in revised form
12 October 2013
Accepted 19 October 2013
Available online 6 November 2013

Keywords:
Graphics processors
Hardware architecture
Displacement mapping
Image pyramid

ABSTRACT

We present an effective traversal algorithm and a hardware architecture to accelerate inverse displacement mapping. This includes a set of techniques that are used to reduce the number of iterative steps that are performed during inverse displacement mapping. For this purpose, we present two algorithms to reduce the number of descending steps and two algorithms to improve the ascending process. All these techniques are combined; we observe up to 66% reduction in the number of iterative steps as compared to other pyramidal displacement-mapping algorithms. We also propose a novel displacement-mapping hardware architecture based on the novel techniques. The experimental results obtained from the FPGA and ASIC evaluation demonstrate that our novel architecture offers many benefits in terms of chip area, power consumption, and off-chip memory accesses for mobile GPUs.

1. Introduction

Displacement mapping is a widely used computer graphics technique that shows the effect of the actual movement of geometric points according to a given height field. Modern GPUs offer hardware support for displacement mapping [1–3], and it is widely used to convey depth and details. However, displacement mapping is regarded as more expensive than other mapping techniques, as it involves dealing with a lot of additional geometry and therefore has a high computational load.

Inverse displacement-mapping [4] algorithms, such as parallax mapping [5], parallax occlusion mapping [6,7], relief mapping [8,9], and pyramidal displacement mapping [10–12], have been proposed to improve the performance of displacement mapping. These approaches can determine the intersection point between a ray and a height field by projecting the ray on the height field without changing the geometry. Moreover, inverse displacement mapping has been widely used in CPU and GPU implementations [13].

State-of-the-art methods for inverse displacement mapping can be categorized into two classes [12,13]: approximation (unsafe) and accurate (safe) algorithms. The approximation algorithms, such as parallax occlusion mapping and relief mapping, are fast, but are relatively low-accuracy: there is no guarantee that they will find the correct intersection. A more accurate algorithm that has been proposed in [10–12,14,15] is the per-pixel ray-tracing technique with an image pyramid. The pyramidal displacement map is an image pyramid of the mipmap consisting of many levels of sub-images. By gradually descending from the root level to the leaf level of this map, the accurate intersection point between a ray and a height field can be computed. In [10], visiting a node one level down or visiting a neighbor node using node crossing occurs during the traversal of an image pyramid. Meanwhile, Tevs et al. [12] and Drobot [15] proposed ascending techniques for effective empty-space skipping, which reduced the number of iteration steps for traversal at grazing angles compared with [10]. Dick et al. [16] applied a pyramidal mipmap to GPU ray casting for terrain rendering.

According to [10,12], pyramidal displacement mapping provides several advantages over other accurate algorithms, such as relaxed cone stepping [17] and safety zone techniques [18–20]. First, pyramidal displacement mapping has lower memory requirements and needs only a simple mipmap construction; the other methods require a long off-line preprocessing time and have large memory requirements. Thus, pyramidal displacement mapping is more suitable for large-scale or dynamic height fields. Second, pyramidal displacement mapping provides better image quality than relaxed cone stepping because relaxed cone stepping can miss thin geometry [12].

Main results: In this paper, we present a set of improvements to maximize the traversal performance of pyramidal displacement mapping. Our approach consists of four sub-algorithms: start-level decision, multi-level down, selective level-up, and coherent level-up. The first and second sub-algorithms effectively reduce the required number of iteration steps at front angles. The start-level decision

* Corresponding author. Tel.: +82 2 3408 3752; fax: +82 2 3409 3755.
E-mail addresses: pwchan@sejong.ac.kr, pwchan@sejong.edu (W.-C. Park).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.024

algorithm directly calculates the start traversal level of an image pyramid using only the ray information, and the multi-level down algorithm descends multiple levels at one time before the first node crossing. Both algorithms can be used simultaneously. The third and fourth sub-algorithms improve the ascending process of [12,15] by removing unnecessary level switching. The selective level-up algorithm ascends a level only if the predicted intersection point is not located in the neighbor node in the node crossing. The coherent level-up algorithm ascends a level when the consecutive node crossing at the same level occurs. The former is suitable for dedicated hardware architecture due to low iteration steps and the latter is suitable for current programmable GPUs due to its simplicity. According to our experimental results on a NVIDIA GTX460 GPU, the combination of the presented sub-algorithms increases frame rates by up to 85% and 56% compared to [10] and [12], respectively. More importantly, our approach is more robust for both front and grazing angles than previous algorithms [10,12]. This feature helps to maintain frame rates in interactive graphics applications (e.g., games).

We also present a hardware architecture consisting of a height-map traverse pipeline and a texture-mapping unit. The traverse pipeline was specially designed to accelerate the presented traversal algorithms. To evaluate the feasibility of our architecture, we integrated the presented hardware unit into an existing ray-tracing hardware architecture [21,22]. This approach can also be combined with current programmable GPUs, which are primarily designed for rasterization. According to our ASIC evaluation, our hardware architecture can achieve real-time performance with fewer hardware resources, memory accesses, and lower power consumption. Thus, our proposed hardware unit has high potential utility in desktop/mobile GPUs.

The rest of the paper is organized in the following manner. We give an overview of pyramidal displacement mapping in Section 2. In Section 3, we present a set of improved traversal algorithms. In Section 4, we present the hardware architecture and its implementation. In Section 5, we provide the experimental results.

2. Pyramidal displacement mapping

A pyramidal displacement map is a quad-tree image pyramid created through a pre-computing process. An image pyramid is a hierarchical collection of sub-level images from $2^0 \times 2^0$ to $2^n \times 2^n$, where n is the maximum level. The original displacement map data might be specified as mipmap level 0, so that each leaf texel (or node) indicates a displacement value for the actual surface. In the upper-level image, each inner texel (i, j) is obtained by storing the maximum value among the four texels $(2i, 2j)$, $(2i+1, 2j)$, $(2i, 2j+1)$, and $(2i+1, 2j+1)$ in the lower-level image. Thus, the root texel denotes the globally maximum height value, whereas an inner texel indicates the locally maximum height value [10,13].

To find the accurate intersection point between a ray and a height field, the image pyramid is traversed from root level n and leaf level 0. An example of this process is shown in Fig. 1. First, at the texture coordinate P of the ray, the height value d_1 of the image pyramid's root level is read. If the current position P of the ray is advanced to P_1 where the ray and d_1 meet, the image pyramid is descended by one level. Then, the height value d_2 of the current mipmap level is read at P_1. If d_2 is greater than d_1, P_1 is advanced to P_2 where the ray and d_2 meet, and the image pyramid is descended by one level; otherwise, the position P_1 does not move, but the image pyramid is descended by one level. This process is repeated until it reaches the leaf level.

While the image pyramid is searched, the ray cannot be advanced over the boundary of the current node, because we have no information out of the current node. To address this problem, we must check whether the advanced position of the ray lies inside the boundary of the current node. If it crosses, the ray is moved to the boundary of the crossed node. This process is called "node crossing" and the neighbor node is visited as a result.

In [12,15], traversal algorithms were proposed to reduce the number of node crossings in [10] through ascending the mipmap level. Tevs et al.'s method [12] ascends the mipmap level by one in cases where the ray resides at a node boundary divisible by two. This reduces an unnecessary level ascension when traversing to sibling nodes. In contrast, Drobot's method [15] simply performs the mipmap level ascension if node crossing occurs. This algorithm costs very little to implement, but it may cause an unnecessary level ascension. According to our experiments, [12] is faster than [15] in general cases. Therefore, we will only consider [12] when we discuss level-up algorithms.

3. Proposed traversal algorithm

Fig. 2 shows the processing flow of the proposed traversal algorithm, which consists of three main steps. In the first step, the algorithm proposed in Section 3.1 determines the start level of the traversal for the generated ray. This process mainly consists of end-position calculations and start-level decisions. In the second step, the traversal to find the intersection point between the ray and the height field is accomplished from the start level. The process flow varies by the occurrence of node crossings. If node crossing occurs, the level-up method proposed in Section 3.3 is applied; otherwise, the level-down method in Section 3.2 is applied. Lastly, when the traversal finishes, the target texture coordinate is calculated using the intersection point between the view vector and the height map.

3.1. Start-level calculation

The previous traversal algorithms for image pyramids such as [10,12] initiate the traversal from the root level. The proposed

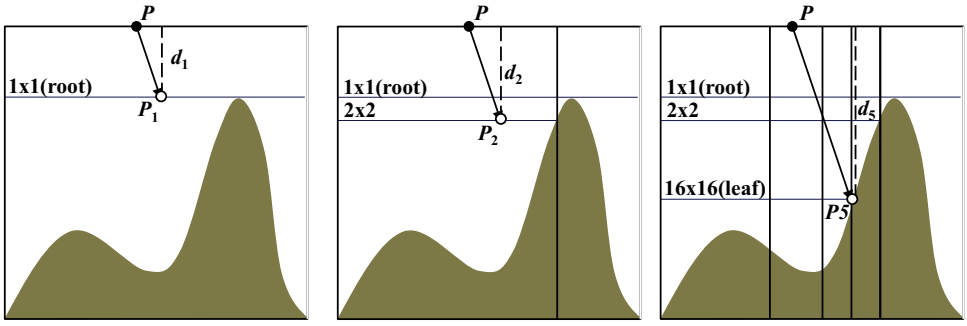

Fig. 1. An example of the traversal process of the pyramidal displacement map.