2012 IEEE International Conference on Information Science and Technology
Wuhan, Hubei, China; March 23-25, 2012

Dipmap : A New Texture Filtering Method for Texture Mapping

Houqing Du and Yi Wan

Abstract— **High image quality and efficiency are usually conflicting goals in texture mapping, especially for real-time rendering. Mipmap is a typical texture filtering method for texture mapping. Due to simplicity and efficiency, it has been popularly used in current graphics hardware. However, its implementation efficiency is at the cost of image rendering quality. In this paper, we first present an improved sampling scheme using the same Mipmap pyramid. In this scheme, for each screen pixel in the scanline, instead of just using the texture scaling ratio in a single direction to determine the Mipmap pyramid level for sampling, we take a row sampling and a column sampling respectively in different Mipmap pyramid levels to generate its value. These two pyramid levels are determined independently by the texture scaling ratios in the row and column directions. The double sampling scheme can overcome the aliasing and overblurring problems caused by just sampling using the texture scaling ratio in a single direction. Motivated by this, we propose a new texture filtering method called Dipmap (Double Mipmap) to further improve image rendering quality. In Dipmap, we establish a row pyramid for row sampling and a column pyramid for column sampling to make these samplings completely independent of each other. Experiments show that Dipmap can deliver much improved image rendering quality than the state-of-the-art texture filtering methods like Mipmap, Ripmap and FAST, while being nearly as efficient as Mipmap.**

I. Introduction

TEXTURE mapping [1] is an important technique in computer graphics whose purpose is to increase the realism of scene rendering. Texture source can be images, functions and some other data. In this paper, we take the two-dimensional image as the texture source.

The pixel in texture image is usually called texel. The key purpose of texture filtering is for each screen pixel to determine the contributing texels in texture image and to convolute the texels with a proper filter to obtain the final pixel color. Usually, a screen pixel may be projected to many texels. Oppositely, there may be multiple screen pixels are projected to one texel. These situations are uniformly called texture scaling, whose degree is measured by texture scaling ratio, expressed in $texels\ per\ pixel$. If scaling ratio is greater than one texel/pixel, we call it texture minification; otherwise, it is termed texture magnification. For pixels in different areas of one image, their scaling ratios are usually different. Even for one pixel, scaling ratios in different directions may be not the same. Since the texture image is two-dimensional, scaling ratios mainly include two : row scaling ratio in row direction and column scaling ratio in column direction. We refer to the direction with the bigger texture scaling ratio as the major direction and the other as the minor direction.

Many texture filtering methods have been proposed for texture mapping. The fastest texture filtering method is point sampling [4] which works relatively well on unscaled images, however the projection of a screen image pixel is usually an area and space variant. Point sampling method generally generates image with aliasing. Direct convolution algorithms [1], [12], [13] directly compute a weighted average of texture samples in screen pixel projection. They can deliver high image quality with very low filtering efficiency. In order to improve filtering efficiency, prefiltering techniques [3], [4], [6] are proposed. Instead of directly sampling in it, they prefilter the original texture image to simplify the calculation of texture filtering. The high efficiency is achieved at the cost of reducing image quality. Anisotropic filtering methods [7], [9] which utilize prefiltering and multisampling technique successfully gain a balance between image quality and filtering efficiency. However, they are more difficult on hardware implementations.

Mipmap [3] is a classic prefiltering technique. Due to simplicity and efficiency, it has become the most popular texture filtering method and been supported by most current graphics hardware [5]. Its basic idea is to previously establish an image pyramid by 2×2 downsampling the original texture image, and then generate screen pixel value by sampling in the appropriate pyramid level determined by the texture scaling ratios. The shortcoming of Mipmap is aliasing or overblurring. Ripmap [4] is an improved algorithm of Mipmap, it extends Mipmap pyramid to include the rectangular area of the texture image. Ripmap can obtain better image quality than Mipmap and has been realized in Hewlett-Packard high-end graphics accelerator in the early 1990s. Its main drawback is that the extending Mipmap pyramid requires four times as much memory as the base Mipmap pyramid.

Texram [7] and Feline [9] belong to anisotropic filtering algorithm which have been realized in special graphics hardware. They use the minimum of the scaling ratios to choose pyramid level for sampling and carry out multisampling along the anisotropic axis determined by the maximum of the scaling ratios. FAST [10] is also an anisotropic filtering method. The difference is that FAST carries out multisampling in screen pixel square rather than along the anisotropic axis. Its basic idea is to adapt sampling rate in each level separately. By utilizing coherence, prefiltering, and area sampling scheme, FAST can achieve higher image quality than Feline in the case of having almost the same efficiency.

This research project is supported by the Fundamental Research Funds for the Central Universities, and is also supported by Program for New Century Excellent Talents in University (Project code: NECT-08-0258).

Houqing Du and Yi Wan are with the Institute of signals and information processing, Lanzhou university, Lanzhou, 730000, China (e-mail: duhq05@gmail.com, wanyi@lzu.edu.cn).

In this paper, we propose a simple and fast texture filtering method called Dipmap. First we propose an improved sampling scheme with Mipmap pyramid. To obtain screen pixel value, instead of sampling in a singel level determined by the minimum or maxmum of texture scaling ratios, we use the row scaling ratio to choose the pyramid level for row sampling, and the similar operation is applied to column sampling, then the screen pixel value is generated by averaging row sample and column sample. The double sampling scheme with Mipmap pyramid can obviously balance the aliasing in minor direction and the overblurring in major direction. Then in order to further improve the image quality, we establish a row pyramid for row sampling and a column pyramid for column sampling, which can make these samplings completely independent to each other. Compared with the existing texture filtering methods such as Mipmap [3], Ripmap [4] and FAST [10], Dipmap not only delivers higher image quality, but also is nearly as efficient as Mipmap.

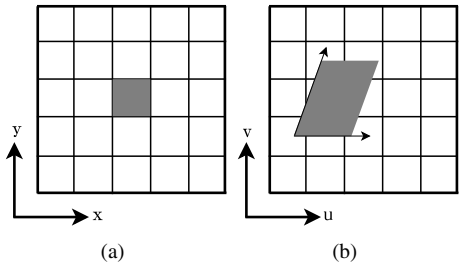


Fig. 1. Screen pixel and its projection in texture image. (a) Screen image, (b) Texture image.

II. DIPMIP

A. Double sampling scheme

Like Mipmap, we consider each screen pixel as a top-right square and its projection in texture image is approximated by a parallelogram described by axis vector $\mathbf{r}_x$ and $\mathbf{r}_y$, as shown in Figure 1. The vector $\mathbf{r}_x = (\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x})$ approximates the scaling ratio in column direction, while $\mathbf{r}_y = (\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y})$ approximates the scaling ratio in row direction, details in [8]. We set that for a two-dimensional image coordinate, column coordinate is in front. Let F be the corresponding relationship between screen image coordinates (x, y) and texture image coordinates (u, v), then

$$u = F_u(x, y), \quad v = F_v(x, y) \tag{1}$$

The partial derivatives can be represented as

$$\begin{aligned} \frac{\partial u}{\partial x} &\approx F_u(x+1, y) - F_u(x, y) \\ \frac{\partial v}{\partial x} &\approx F_v(x+1, y) - F_v(x, y) \\ \frac{\partial u}{\partial y} &\approx F_u(x, y+1) - F_u(x, y) \\ \frac{\partial v}{\partial y} &\approx F_v(x, y+1) - F_v(x, y) \end{aligned} \tag{2}$$

The magnitudes of $\mathbf{r}_x$ and $\mathbf{r}_y$ are approximated with

$$\begin{aligned} \|\mathbf{r}_x\| &= \max(|\frac{\partial u}{\partial x}|, |\frac{\partial v}{\partial x}|) \\ \|\mathbf{r}_y\| &= \max(|\frac{\partial u}{\partial y}|, |\frac{\partial v}{\partial y}|) \end{aligned} \tag{3}$$

Let S denote the scaling ratio, so pyramid level $L = \log_2(S)$. Texture scaling ratio in row direction is $S_y = \|\mathbf{r}_y\|$ while $S_x = \|\mathbf{r}_x\|$ is the texture scaling ratio in column direction.

In Mipmap, if choose $S = \max(S_x, S_y)$, screen image will be blurry, as shown in Figure 2(a), while $S = \min(S_x, S_y)$ screen image will be awry, as shown in Figure 2(b). Traditional Mipmap supported by OpenGL [11] choose $S = \max(S_x, S_y)$. Anisotropic filtering algorithms choose $S = \min(S_x, S_y)$ and take multisampling along the direction determined by $S = \max(S_x, S_y)$. In order to improve the aliasing and overblurring caused by just using a single texture scaling ratio to determine the Mipmap pyramid level for sampling, we propose an improved sampling scheme called double sampling scheme. The double sampling scheme first takes a row sampling in row pyramid level determined by row scaling ratio $L_y = \log_2(S_y)$ and a column sampling in column pyramid level determined by column scaling ratio $L_x = \log_2(S_x)$ separately, then averages row sample and column sample to generate the final screen pixel value. This double sampling scheme is able to balance the aliasing and overblurring because it can avoid oversampling in minor direction and undersampling in major direction, as shown in Figure 2(c).

B. Dipmap pyramid

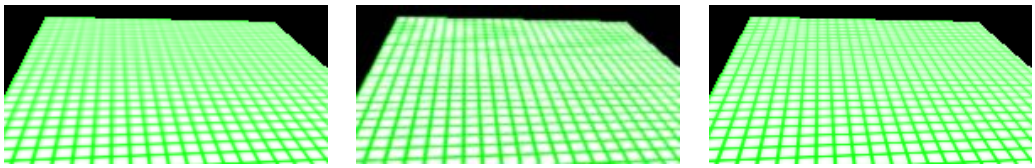
The double sampling scheme introduced in section II-A still uses Mipmap pyramid. Since Mipmap pyramid is established from the original texture image by 2×2 downsampling, its each pyramid level has the same texture scaling ratio in both row direction and column direction. So in row sampling, while the sampling level is determined by S_y, the sample value is also affected by the scaling in column direction. There is the similar problem to column sampling.

In order to make row sampling and column sampling completely independent to each other, we build a new image pyramid called Dipmap pyramid. The Dipmap pyramid includes a row pyramid for row sampling and a column pyramid for column sampling. The row pyramid is obtained by 1×2 downsampling of the original texture image until image height is one, while column pyramid is obtained by 2×1 downsampling of the original texture image until image width is one, as shown in Figure 3. In the row pyramid, there is only texture scaling in row direction, the similar to the column pyramid.

C. Main steps of Dipmap

The main steps of Dipmap algorithm are

1) : Establish Dipmap pyramid.



(a) (b) (c)

Fig. 2. (a) A single sampling, choose $S = \max(S_x, S_y)$, image overblurring, (b) A single sampling, choose $S = \min(S_x, S_y)$, image aliasing, (c) The double sampling scheme with Mipmap pyramid, image balancing.

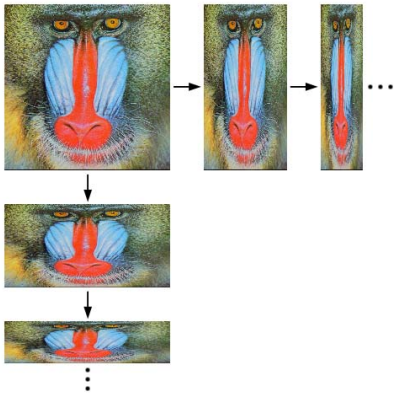


Fig. 3. Dipmap pyramid

2) : Compute the corresponding relationship between screen pixels and texture pixels. The same as Mipmap, given a screen pixel (x, y), its corresponding screen pixel (u, v), row scaling ratio S_y and column scaling ratio S_x are computed here.

3) : Choose pyramid level for sampling. Unlike Mipmap chooses a single pyramid for sampling, Dipmap chooses $L_x = \log_2(S_x)$ for row sampling and $L_y = \log_2(S_y)$ for column sampling separately.

4) : Generate screen pixel value. For a screen pixel, after L_x and L_y are determined, we can take row sampling in row pyramid and column sampling in column pyramid respectively. Then screen pixel value is generated by averaging row sample and column sample. This step is called texture filtering.

III. Experiments

In experiments, we implement the texture mapping in the way of software simulation. Mipmap, Ripmap, Dipmap and FAST are chosen as the texture filtering algorithms. We use C++ programming language on a AMD dual core 2.91GHz processor with 2GB ARM.

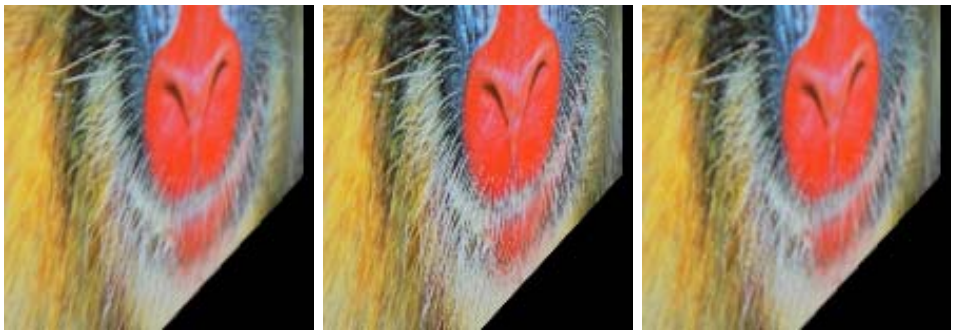
Two models including a rectangle and a ellipsoid which can be arbitrarily rotated and translated in three-dimensional space have been rendered. We choose the typical checkerboard image, a pane image and a natural image baboon as the texture images and render some scenes of the two models in different positions. For a screen image pixel, if it is affected by the scene, its value will be computed from texture image, otherwise it will be forced to zero. We first compare image quality among these screen images rendered using different texture filtering methods.

A. Image quality

Figure 8, 9 and 10 obviously show that all images generated by Mipmap are blurry while Ripmap, Dipmap, FAST can get higher image quality. Compared with Ripmap and FAST, Dipmap can get higher image quality on the aspects of image overall contrast and texture details. Observing Figure 4, baboon beard rendered by Dipmap is more clearer than these rendered by Ripmap and FAST. Shown in Figure 5 and 6, images rendered by Dipmap are brighter than these rendered by Ripmap. Images rendered by FAST have visible aliasing on the lines. In FAST, for the multisampling of a pixel, the samples are distributed evenly in the pixel quarter using the jittered sampling scheme. Since the jittered sampling scheme is random, the rendered screen images maybe different in each rendering process,

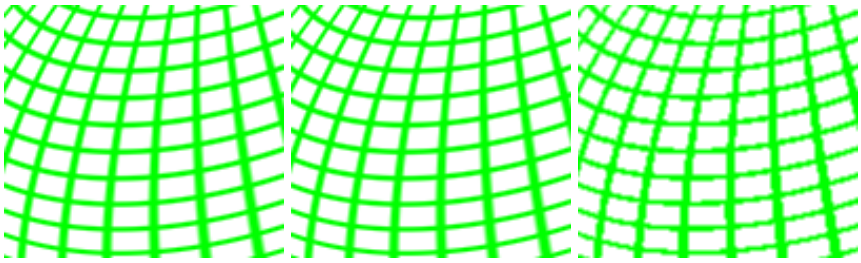
Generally, pyramid level L is a decimal number, so there are two levels, the lower level $L_f = \lfloor L \rfloor$ and the higher level $L_c = \lceil L \rceil$ can be selected for sampling. In our experiments, for Mipmap, the trilinear interpolation is used which includes two bilinear interpolations in two levels and one linear between two levels; For Ripmap, four bilinear interpolations in four levels and one bilinear interpolation between four levels are used. For Dipmap, a row sampling or a column sampling is usually equal to a Mipmap sampling. However, we find that, for the row sampling in Dipmap, using trilinear interpolation between the lower and the higher pyramid levels or bilinear interpolation in a single pyramid level may obtain almost the same image quality. Especially when the texture scaling ratio becomes larger, bilinear interpolation in the signal level can get better result than trilinear interpolation between two levels. There is the same situation for the column sampling. As shown in Figure 7, (a) is rendered using a single pyramid level, the lower pyramid level; (b) is rendered using both lower and higher levels. Observing images (a)(b), at their bottoms in which pixels have smaller texture scaling ratios, the bears of baboon in both images have the same visual affect; at their tops in which pixels have bigger texture scaling ratios, the eyes of baboon in image (a) is obviously clearer than those in image (b).

Considering of both image quality and filtering efficiency, we use the bilinear interpolation in the single pyramid level, the lower pyramid level, both for row sampling and column sampling. Just sampling in the lower level maybe cause



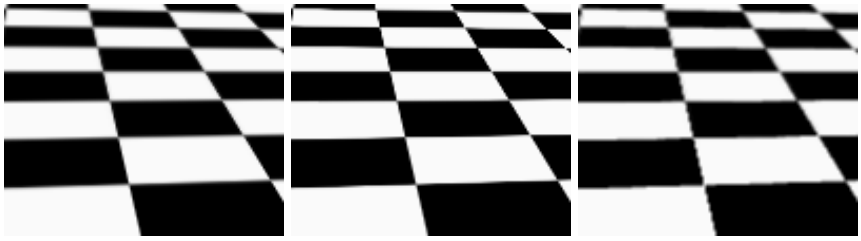
(a) (b) (c)

Fig. 4. Magnified images from the marked rectangles of Figure 8(b)(c)(d) rendered by algorithms. (a) Ripmap, (b) Dipmap, (c) FAST.



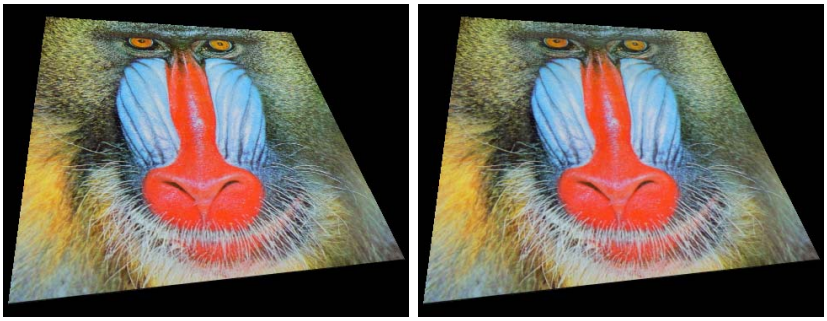
(a) (b) (c)

Fig. 5. Magnified images from the marked rectangles of Figure 9(b)(c)(d) rendered by algorithms. (a) Ripmap, (b) Dipmap, (c) FAST.



(a) (b) (c)

Fig. 6. Magnified images from the marked rectangles of Figure 10(b)(c)(d) rendered by algorithms. (a) Ripmap, (b) Dipmap, (c)FAST.



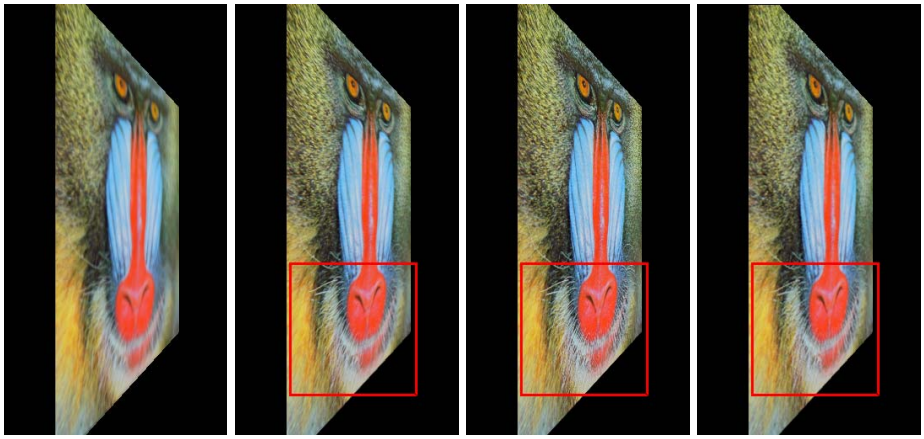
(a) (b)

Fig. 7. A model rendered with baboon image by Dipmap. (a) Sampling in the lower pyramid level with bilinear interpolation, (b) Sampling in both lower and higher pyramid levels with trilinear interpolation.

aliasing in screen image, however the bilinear interpolation is able to decrease the aliasing, because it has the function of blurring. So it is rational to sample in the lower pyramid level using the bilinear interpolation.

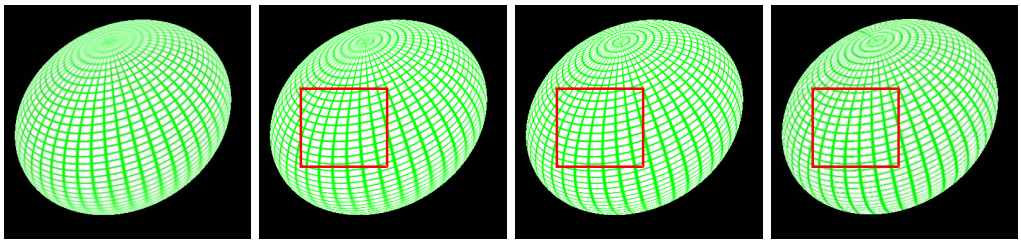
B. Filtering efficiency

The filtering process of Mipmap is equal to a trilinear interpolation, for each screen pixel its efficiency is invari-



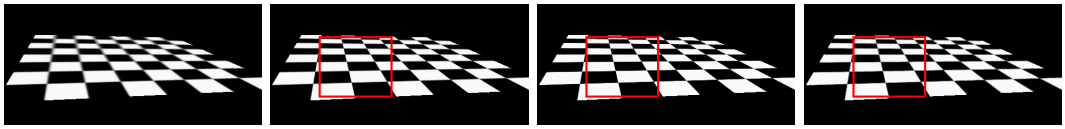
(a) (b) (c) (d)

Fig. 8. A model rendered with baboon image by algorithms. (a) Mipmap, (b) Ripmap, (c) Dipmap, (d) FAST.



(a) (b) (c) (d)

Fig. 9. A model rendered with pane image by algorithms. (a) Mipmap, (b) Ripmap, (c) Dipmap, (d) FAST.



(a) (b) (c) (d)

Fig. 10. A model rendered with checkerboard image by algorithms. (a) Mipmap, (b) Ripmap, (c) Dipmap, (d) FAST.

TABLE I

FILTERING EFFICIENCY

Figure	Pixel number	Time	Mipmap	Ripmap	Dipmap	FAST
8	61674	total time(ms)	32.44	69.34	38.30	123.13
		single time(us)	0.53	1.12	0.62	2.00
9	81509	total time(ms)	38.54	85.26	43.07	453.87
		single time(us)	0.47	1.05	0.53	5.57
10	46904	total time(ms)	25.32	52.09	28.12	103.01
		single time(us)	0.54	1.11	0.60	2.20

ant. For Dipmap, a row sampling process is a trilinear interpolation, the same to column sampling. Since screen pixel value is generated by averaging row sample and column sample, the filtering process of Dipmap includes two trilinear interpolations and a linear interpolation. However, in experiments, we use a bilinear interpolation in a single pyramid level for both row sampling and column sampling. The efficiency of two bilinear interpolations and the linear interpolation between row sampling and column sampling are equal to the efficiency of one trilinear interpolation. So in theory, Dipmap implemented in our experiments and Mipmap have the same filtering efficiency which is invariant for each screen pixel. Generally, to generate a screen pixel value, Ripmap needs four bilinear interpolations in four levels and a bilinear interpolation between these four levels, so its filtering efficiency is about half of Mipmap and Dipmap, which is invariant too. Due to the need of multisampling in the pixel square and the number of samples varying with the scaling ratio, the filtering efficiency of FAST is variant to each screen pixel and very lower than Dipmap.

Table I shows the running time of Mipmap, Ripmap, Dipmap and FAST .The data in this table is obtained by

taking the average of one hundred measurements. The item of pixel number is the number of totally rendered pixels in screen image, total time is the time used to render all these screen pixels, single time is the average time used to render a pixel. It obviously shows that the filtering efficiency of Mipmap and Dipmap is almost the same and nearly invariant. Ripmap is about twice slower than Dipmap, nearly invariant. FAST filtering efficiency is very lower and variant for each screen pixel. These experimental data verify our theoretical analysis.

IV. Conclusion and discusses

In this paper, we have developed a new texture filtering technique called Dipmap which includes the double sampling scheme and the Dipmap pyramid. Dipmap can obtain higher image quality than the exiting algorithm, such as trilinear Mipmap, Ripmap and FAST, while being nearly as efficient as the trilinear Mipmap.

Being similar to Ripmap, the drawback of Dipmap is that the Dipmap pyramid also requires more texture memory than Mipmap pyramid. Compared with the original texture image, Mipmap pyramid needs more than one third texture memory, while Ripmap pyramid requires four times and Dipmap requires three times. However, just using the double sampling scheme with Mipmap pyramid can also deliver higher image quality than the traditional Mipmap. Compared with Ripmap, Dipmap pyramid needs less texture memory than Ripmap pyramid. And, Dipmap not only delivers higher image quality than Ripmap, but also is more efficient.

For Dipmap, one way to decrease texture memory requirement is the serial sampling, first generate row image and then generate column image in two scanlines. In the future, our aim is to find a more rational and effective method to handle this problem.

References

[1] P. S. Heckbert, "Survey of texture Mapping," *IEEE Comput. Graphics Appl.*, vol. 6, no. 11, pp. 56-67, 1986.

[2] N. Green and P. S. Heckbert, "Creating raster omnimax images from multiple perspective views using the elliptical weighted fIlter," *IEEE Comput. Graphics Appl.*, vol. 18, no. 3, pp. 21-27, 1986.

[3] L. Williams, "Pyramidal parametrics," *Comput. Graphics* vol. 17, no. 3, pp. 1-11, 1983.

[4] P. S. Heckbert, "Fundamentals of texture mapping and image warping," *EECS Department, University of California, Berkeley*, UCB/CSD-89-516, 1989.

[5] T. A. Moller and E. Haines, *Real-time rendering(second edition,Chinese version)*, Beiing : Peking University Press, 2002, pp.77-78.

[6] F .C. Crow, "Summed-area tables for texture mapping," *Comput. Graphics*, vol. 18, no. 3, pp. 207-212, 1984.

[7] A. Schilling, G. Knittel and W. Strasser, "Texram: a smart memory for texturing," *IEEE Comput. Graphics Appl.*, vol. 16, no. 3, pp. 32-41, 1996.

[8] J. P. Ewins, M. D. Warcus, M. White and P. F. Lister, "Mip-map Level selection for texture mapping," *IEEE Trans. Visual Comput.*, vol, 4, no. 4, pp. 317-329, 1998.

[9] J. McCormack, R. Perry, K. I. Farkas, and N. P. Jouppi, "Feline: Fast elliptical lines for anisotropic texture mapping," *Proc. SIGGRAPH*, 1999, pp. 243-250.

[10] B. Q. Chen, F. Dachille and A. Kaufman, "Footprint area sampled texturing," *IEEE Trans. Visual Comput.*, vol. 10, no. 2, pp. 230-240, 2004.

[11] D. Shreiner, M. Woo, J. Neider and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2.1, six Edition (Chinese version)*, Beijing : China Machine Press, 2009, pp. 247-253.

[12] J. F. Bline and M. E. Newell, "Texture and reflection in computer generatedimages," *Commun. ACM*, vol.19, no.10, pp. 542-546, 1976.

[13] E. A. Feibush, M. Levoy and R. L. Cook, "Synthetic texturing using digitalfilters," *Comput. Graphics*, vol. 14, no. 3, pp. 294-301, 1980.