I.J. Image, Graphics and Signal Processing, 2011, 3, 33-40
Published Online April 2011 in MECS (http://www.mecs-press.org/)

Wavelet and Blend maps for texture synthesis

Du Jin-Lian Wang Song Meng Xianhai
Beijing University of Technology
Beijing China
dujinlian@bjut.edu.cn

***Abstract*— blending is now a popular technology for large realtime texture synthesis .Nevertheless, creating blend map during rendering is time and computation consuming work. In this paper, we exploited a method to create a kind of blend tile which can be tile together seamlessly. Note that blend map is in fact a kind of image, which is Markov Random Field, contains multiresolution signals, while wavelet is a powerful way to process multiresolution signals, we use wavelet to process the traditional blend tile. After our processing steps, the result blend tile become smooth and suitable for tiling, with no important features lost. Using this kind blend tile, many computation resources for computing blend map during texture synthesizing is saved. The experimental results shows that our method may successfully process many traditional blend tiles.**

Index Terms*— texture synthesis; wavelet; Multiresolution analysis; blend map

I. INTRODUCTION

Textures are extensively used to increase the realism of rendered objects. It is especially important for the realtime rendering of terrain. In this case, the use of large textures often results low performance during rendering. As alternatives, a number of techniques including [Ashikhmin's 2001, Wei and Levoy 2000][1,2], have been developed for procedural texture synthesis. Given a sample texture, these techniques can create a similar texture that fits the target surface naturally and seamlessly. To increase the visual richness of the synthesis texture, blend maps and blending algorithms are used during synthesizing. However, create large blending tiles will consume computation resources and also consume extra texture memory which is very precious now. This paper introduces a method to produce small blend tile which can be tiled together with no seams along the boundaries of two tiles. So the blend tile could be used repeatedly to produce large texture with no need to blend the sharp boundaries. The blend tile is produced by wavelet analysis, so the processing steps and mathematics of our approach are discussed firstly. Advantages of using small blend tiles and its impact on performance are illustrated in the paper.

Related works about texturing terrain or other surfaces using texture tiles and blending technology are discussed in next section, followed by discussion of our method.

II. RELATED WORK

In recent years, a great deal of works on texture synthesis based on texture tiles and blending technology has been proposed. Here we simply review some up-to-date and typical approaches.

Texture splatting[3] is the typical technology which use linear blending to smooth the transition between textures tiles and it was used widely in game terrain rendering. The limit of this method is that it needs large alpha maps for blending and linear blending often produced undesirable artifacts.

To improve the quality of synthesis texture, many researchers have contributed their wisdom and effort. [Malik et al.1999][4] proposed conception of texton channel to specify the features of a texture map and the texton map was applied successfully to control procedural texture synthesis by [Zhang et al.2003;Wu and Yu 2004][5,6]. The work of Lai et al.[7] introduced a novel method to blend textures. Their algorithm present identifies the features of an image and provides a visually pleasing border between tiled textures on the terrain. The technology is limited by the number of textures that are precomputed, thus the visual richness of terrain may be reduced. The textures that blend between regions also require extra texture memory, over and above the tiles already in use. Alexandre hardy[8] introduced blend maps to increase the control and observed richness of tiled textures. The technology construct blend map by identifying important features in a texture for these features are often highlighted under certain lighting conditions. The resulting texture produced by the technology appears more realistic in many cases. However, there are too many coefficients need to be define in this method so it is not easy for user.

In our opinion, if the blend map can be tiled seamlessly, the computing work for blending textures will be reduced largely, because there is no need to synthesizing blend map or feature map. So our work is try to produce blend map which can tile together —blend tile with Wavelet analysis.

Our approach is based on the observation that many blend maps, which are Markov Random Field, contains multiresolution signals, that means the map is refinable. While wavelet analysis is a powerful way to analyze and construct functions[9,10,11]. It is therefore natural to use wavelet analysis to process the blend map, remove high

Supported by Scientific Research Common Program of Beijing Municipal Commission of Education under Grant Nos. KM200710005002。

frequencies from the map, smooth the remainder. Then the result noise is suitable for tiling and will not cause aliasing and discontinuities.

In the following section we present the overview for processing steps of our approach and the mathematics for deriving this approach is presented in section 4. Practical implementation and results are discussed in section 5.

I. OVERVIEW OF OUR ALGORITHM

To provide a context for the remainder of the paper and to simplify the algorithm, we present a high-level overview. The essence of our algorithm consists of the following three steps illustrated in Fig.1:

Create the blend map R filled with important features of the textures which suitable for your application.(fig.1a)

Downsample R to create the half-size image Rd.(fig.1b)

Upsample Rd to a full size image Rn(fig.1c).

Tile Rn to construct large blend map for your application.

Rn is created by downsample R to get the low frequencies and refine it to a full size. Thus Rn contains smoothed low-frequency part of R.

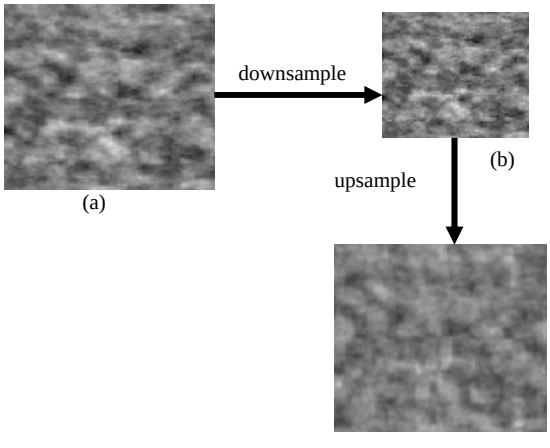


Figure1. (a).Blend map R of a texture..(b).Half size image R_d. (c).Full size image R_n.

III. THEORY

Wavelet analysis, known as multiresolution analysis, has emerged a powerful way to extract the different frequencies information from original signal and enrich them at the same time. We use this feature of wavelet analysis to process the blend map, and provide the mathematical underpinnings for our algorithm. First, we work in one dimension, then extend to 2 dimensions.

A. Upsampling and Downsampling

In the multiresolution opinion, much function can be represented as a weighted sum of basis function, as following;

$$F(x) = \sum_i f_i \phi(x-i) \tag{1}$$

Where $\phi(x)$ is the basis function and $\phi(x-i)$ is a translated version of $\phi(x)$ by integer amount i. where the f_i are the coefficients of the representation.

If we call the vector space formed by $\phi(x)$ and it's translated versions the resolution 0 space and denote by S^0:

$$S^0 : \left\{ F(x) \mid F(x) = \sum_i \phi(x-i) \right\} \tag{2}$$

A larger resolution 1 space S^1 of function can be represented by scaling down the width of $\phi(x)$ by factor of 2. Function $G(x)$ in S^1 take the following form:

$$S^1(x) : G(x) = \sum_i g_i \phi(2x-i) \tag{3}$$

In wavelet analysis idea, all the functions in S^0 are also in S^1, that means $S^0 \subset S^1$. This guarantees that S^1 enriches the space S^0. and $\phi(x)$ can be written in terms of $\phi(2x-k)$:

$$\phi(x) = \sum_k h_k \phi(2x-k) \tag{4}$$

Thus given the coefficients f_i that represents $F(x)$ in S^0, there exist coefficients f_i^1 that represents that function exactly in S^1:

$$F(x) = \sum_i f_i^1 \phi(2x-i) \tag{5}$$

Using the equation 4 and equation 5, it is straightforward to show that:

$$f_i^1 = \sum_k h_{i-2k} f_k \tag{6}$$

The equation 6 represents an upsampling filter, which is in fact used in step 3 of our algorithm.

Although the upsampling filter established above can guarantee that every member of S^0 can be represented exactly as a member of S^1. The converse is not true: not every function $G(x)$ in S^1 can be represented exactly in the lower resolution space S^0, there is some lost detail. But the loss of detail can be minimized in a least squares sense by wavelet analysis. Assume $G^0(x)$ is the least squares best approximation to

$G(x)$ in S^0, then given the coefficients g_i for $G(x)$, the coefficients g_i^0 can be determined using standard result from wavelet analysis:

$$g_i^0 = \sum_k a_{k-2i} g_k \tag{7}$$

a_k is the downsampling filter used in step 2 of our algorithm.

When extend above equation to two dimensions, the upsampling filter f_i^1 represented in equation 6 and downsampling filter g_i^0 should be written in the following form respectively:

$$f_{i,j}^1 = \sum_{kx,ky} h_{i-2kx,\, j-2ky} f_{kx,ky} \tag{8}$$

$$g_{i,j}^0 = \sum_{kx,ky} a_{kx-2i,ky-2j} g_{kx,ky} \tag{9}$$

B. constructing blend tile

In this section, a more detailed algorithm presented in section 2 is described. In our opinion, each blend map can be regard as the function $R(x)$ formed by weighted sum of basis function as equation 1. $\phi(x)$ is a wavelet basis function, such as uniform quadratic B-spline basis function, sincs and so on. We use the uniform quadratic B-spline basis function and you can also use other refinable functions, such as sincs, according to your application. Steps to build R_n is as following:

Create small blend tile R using a texture or other method, such as Perlin noise in S^1, $R = \{...r_{i,j}...\}$. Defines a function $R(x,y)$ in S^1. Using the basis function $B(x,y)$, $R(x,y)$ can be written as equation 8:

$$R(x,y) = \sum_{i,j} r_{i,j} B(2x-i) B(2y-j) \tag{10}$$

Compute R_d by downsampling R

As shown in equation 9, $R(x,y)$ can be downsampling by downsampling filter $a_{kx,ky}$ to get $R_d(x,y)$ in S^0:

$$R_d(x,y) = \sum_{i,j} r_{i,j}^d B(x-i) B(y-j) \tag{11}$$

And r^d is computed from the coefficient r:

$$r_{i,j}^d = \sum_{kx,ky} a_{kx-2i,ky-2j} r_{kx,ky} \tag{12}$$

Then we get $R_d = \{...r_{i,j}^d...\}$.

Compute R_n by upsampling R_d

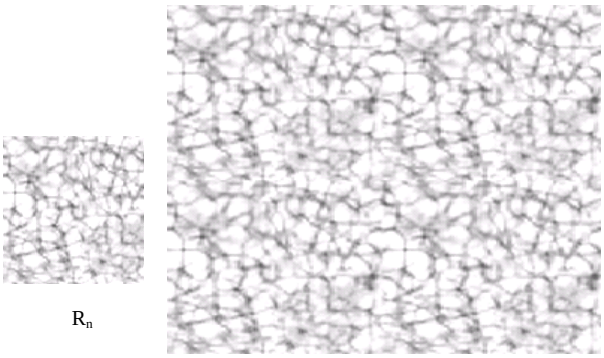
By using equation 12, we get the R_d containing low frequencies signal represented in S^0. However, R_d looks still rough, not smooth enough. Thus we smoothen R_d by upsampling it to achieve the target blend image R_n in S^1. $R_n = \{...r_{i,j}^n...\}$,and $r_{i,j}^n$ can be written as following:

$$r_{i,j}^n = \sum_{kx,ky} h_{i-2kx,\, j-2ky} r_{kx,ky}^d \tag{13}$$

In this step, no important detail is lost, because all the members in R_d is also represented exactly as members in R_n.

C. construct large blend pattern by R_n

After creating blend tile R_n, we could define a large space and tile it with R_n to create blend map for this space. As illustrated in Fig2. The images on left column are the blend tile processed by wavelet and images on the right column are the blend map tiled by it. The downside of using R_n blend tile is that it will introduce low-frequency repeating patters into the map. Fortunately, this repetition is not always objectionable, In fact, the repetition is beneficial to generate the repeating patter with obvious structures for some applications. For other applications, the repeating pattern can be eliminated by using stochastic function to disturb the signal inner each tile but keep the boundary not change. Only when it needs different tiles in some area, blend method should be adopted to eliminate the discontinuities along the boundaries between different tiles.



R_n

blend map

Figure 2. Create blend map by tiling R_n

V. IMPLEMENTATION

A Create original blend tiles

There are a numbers of methods for creating original blend tiles. One way is get the blend tile from texture you will use, which will preserved many features of the texture, as shown in fig3. The typical steps are described in alexandre haroy's paper. Another way is use Perlin noise, which is widely used in texture generating. The advantage of using Perlin noise is that the result image has strong random nature.

B Implementatal Results

The size of blend tile for all examples in our implementation is 128*128, and the size of target blend pattern is 256*256 or 512*512. Fig.3 shows the target patterns generated by blend tiles processed by our method(we name it repeatable noise tile) and Fig.4 shows the blend results of two textures with repeatable blend tile.

When we look at the examples in Fig3, we'll find that the target patterns generated by original blend tile shows discontinuities on the boundaries of two tiles.

Especially in figure a1,a3and a4, the discontinuities is very obviously. In figure a2, discontinuities is not very intensely. One reasons is that the main part of these blend tile contains high-frequencies noises which result in coarse pattern. And the coarse pattern diminishes discontinuities. The other reason is that the feature of the result pattern makes it difficult to find the discontinuities, such as a2. Use original blend tile to generate large blend map often make the target pattern looks very rough which is not satisfied result for some application, this is in fact the aliasing caused by higher frequencies. So other methods are needed to blend or eliminate the aliasing, but it will consume computation resources and time.

however, use the repeatable blend tile to tile large blend map, the target patterns looks smoothly and there are nearly no discontinuities between two tiles. See b1,b2,b3 and b4 in Fig.3. This shows that our method can make original coarse blend tiles become smooth，especially the boundary will not contain high frequency nose, thus it is easily use it for realtime texture synthesizing with tiling technology.

C. Advantages discussion

Among the results, the first two original blend tiles, original blend tiles in a1 and a2, are created by textures directly with the method similar to Alexadre hardy [8].original blend tile in a1 are created by texture stone and original blend tile in a2 are created by texture pebble show in fig.4. these two blend tiles all preserve the important features of their textures. With these important features, the target texture will looks more realistic: The grass can't growth on the pebbles and stone, it should growth around the pebbles and stone, or growth from the cracks. After processed by our algorithm, the result tile still preserves the features of these textures. This is illustrated by blending results in c1 and c2 of Fig4.

The other two original blend tiles are produced by Perlin noise. Perlin noise has been very important for creating textures in computer graphics, for example marble , wood grain and cloud patterns and so on[12,13,14]. But it is hard to use perlin noise tile to accelerate the rendering speed. Because Perlin constructs patterns from bands of noise, that is to say each perlin band is intended to be band-limited, but each band actually contains a much wider ranges of frequencies. These weak band limits lead to some serious problems. Near the Nyquist limit, a Perlin band contains both frequencies that are low enough to be representable and frequencies that are high enough to be unrepresentable. The unrepresentable frequencies will cause aliasing artifacts. These kind aliasing has been a constant source of frustration for shader writer to get satisfied render target. More important, when use perlin noise tile to generate large texture, the aliasing will generate sharp boundary between two tiles in render target(see fig.5).

However, perlin noise is refinable, so we can use wavelet to process it, remove high frequencies from the noise, smooth the remainder. Then the result noise is suitable for tiling and will not cause aliasing and discontinuities, see b3 and b4 in fig.3.

In terrain rendering, many terrain textures are unstructured, stochastic. Thus Perlin noise blend tile will be a powerful way to produce rich ,realistic unstructured terrain textures. In this opinion, our algorithm is a low-cost method to generate unstructured texture. Of course, if need to produce texture with strong random nature, stochastic function may be used to disturb the noise blend tile during blending, this is not difficult to realize.

As for run-time cost for generating large texture pattern, we have not compare our algorithm with other texture synthesizing algorithms, that is our next work. However, because the blend tile is pre-computed and not very large(generally 64*64 or 128*128 is enough), just read from out storage at the beginning of texture creating process, so there is no time-consumption for generating blend tile, at the same time there is no computation consumption for generate large blend map because the blend tile can be used repeatedly. Thus our method is faster than traditional texture synthesizing method based blending. At last but not least, use blend tile will need less memory than previous texture synthesis algorithms. If we textured a large area of terrain with similar appearance constructed by several matters, for example stone, grass and soil. Then only several type small blend tiles are needed. In sum, our algorithm is an effective way to prepare blending data for procedural texture synthesis. In fact, our method cal also be used in any procedural texture generating application with tiling technology.

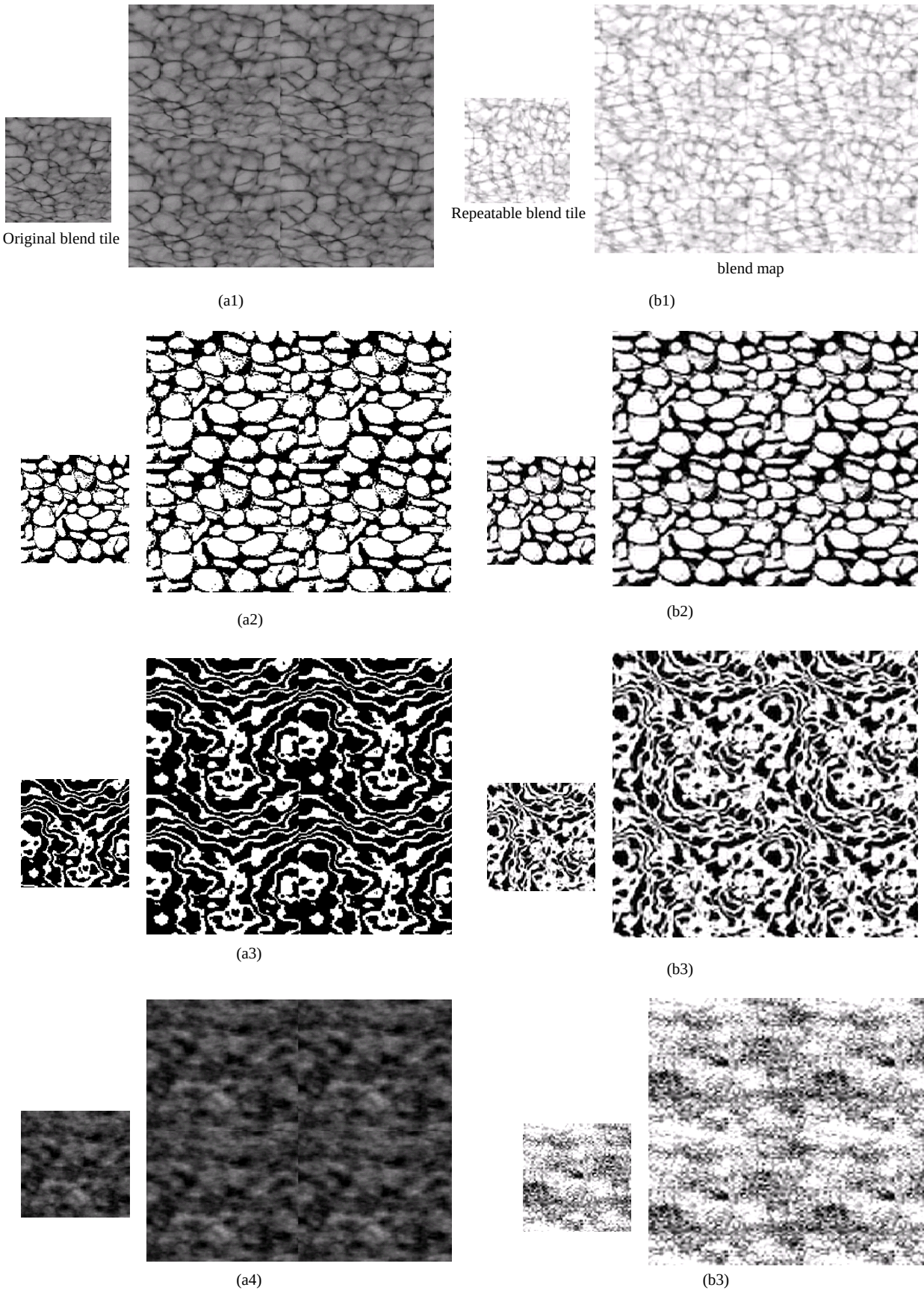


Figure 3. examples of our algorithm. On the left column (a1,a2,a3,a4)are the original blend tiles and maps tiled by them on the right column (b1,b2,b3,b4)are the repeatable blend tiles produced by our algorithm and result patterns tiled by repeatable blend tiles.

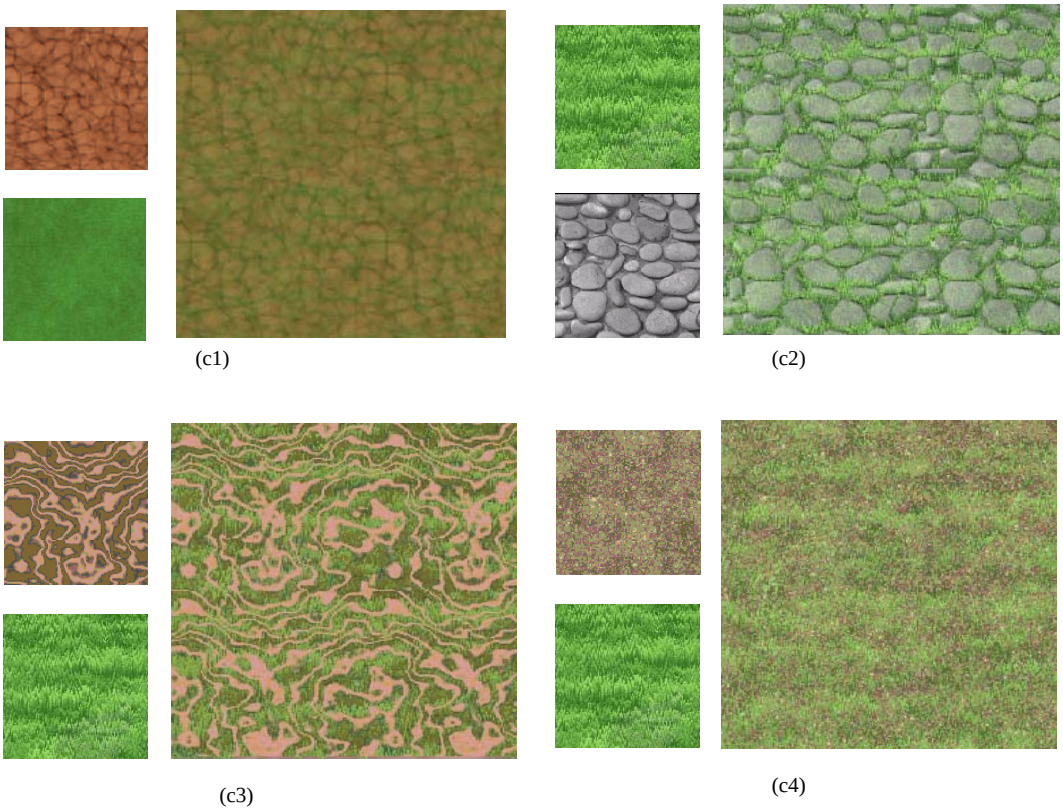


Figure 4. blending results of two textures with the repeatable blend tiles produced by our algirithm

V. SUMMARY

Our mainly work is processing traditional blend tile by wavelet analysis to generate new blend tile suitable for tiling for large scale texture synthesis. The processing calculation is:

- Create a blend tile from a texture or Perlin noise.
- Downsampling the blend tile to get a tile of 1/2 size.
- Upsampling the 1/2 size tile to new tile of full size.
- (Optional)Adjust the tile borders to accommodate tile meshing(if using different tiles).

After processed by above steps, the final blend tile looks smoothly. Use this kind of blend tile to create large blend map, the aliasing and discontinuities along the boundaries is smoothed away and the target pattern become smooth and clear. This means that the blend tile processed by our approach can be used with tiling technology during texture synthesis, which may accelerate the synthesizing speed. although use different tiles to create texture should adopt blending or other technologies to eliminate boundary discontinuities, it is easier to control the final outcome than using traditional blend tiles, because the higher frequencies are eliminated by our processing method. Our next work include using this method for interactive terrain rendering, comparing the performance of our algorithm with other texture synthesizing algorithms, such as Alexadre hardy' algorithm, and find effective method for transition of different blend tiles

REFERENCES

[1] ASHIKHMIN, M.. Synthesizing natural textures. ACM Symposium on Interactive 3D Graphics,,2001, 217.226.

[2]. WEI, L. Y., AND LEVOY, M. Fast texture synthesis using treestructured vector quantization. In Proc. SIGGRAPH 2000, 479.488.

[3] Bloom, C. Texture splatting. 2000. http://www.cbloom.com/3d/techdocs/splatting.txt.

[4] Malik, J., Belongie, S., Shi, J., and Leung, T. Textons, contours and regions: Cue integration in image segmentation. In Seventh International Conference on Computer Vision (ICCV'99). 1999,Vol. 2. 918–925.

[5] Zhang, J., Zhou, K., Velho, L., Guo, B., and Shum, H.-Y. Synthesis of progressively-variant textures on arbitrary surfaces.ACM Trans. Graph, 22, 3, 2003. 295–302.

[6] Wu, Q. and Yu, Y. Feature matching and deformation for texture synthesis. ACM Trans. Graph. 2004 23, 3, 364–367.

[7] Lai, Y.-Y., Tai, W.-K., Chang, C.-C., and Liu, C.-D. Synthesizing transition textures on succession patterns. In GRAPHITE'05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia. ACM Press, New York, NY, USA, 2005. 273–276.

[8] ALEXANDRE HARDY, DUNCAN ANDREW KEITH MC ROBERTS. Blend Maps: Enhanced Terrain Texturing. http://portal.acm.org/citation.cfm?id=1216269.

[9] CHUI,C.K.An introduction to wavelets. Academic Press Professional, Inc., San Diego, CA, USA.1992.

[10] STOLLNITZ,E.,DEROSE,T., AND SALESIN,D. Wavelets for Computer Graphics. Morgan Kaufmann Publishers,1996.

[11] ROBERT L.COOK, TONY DEROSE. Wavelet Noise. In SIGGRAPH'2005: Proceedings of the 32th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 2005.pp.803-811.

[12] PERLIN K.,An image synthesizer. In SIGGRAPH'02: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA,1985,pp. 287-296.

[13] PEACHEY,D.,Building procedural textures. In Texturing and Modeling: A Procedural Approach, third ed. Morgan Kaufmann Publishers Inc., ch.2.2003.

[14] LEWIS,J.P. Agorithms for solid noise synthesis. In SIGGRAPH'89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques, ACM Press, New York, NY, USA, 1989.pp.263-270.

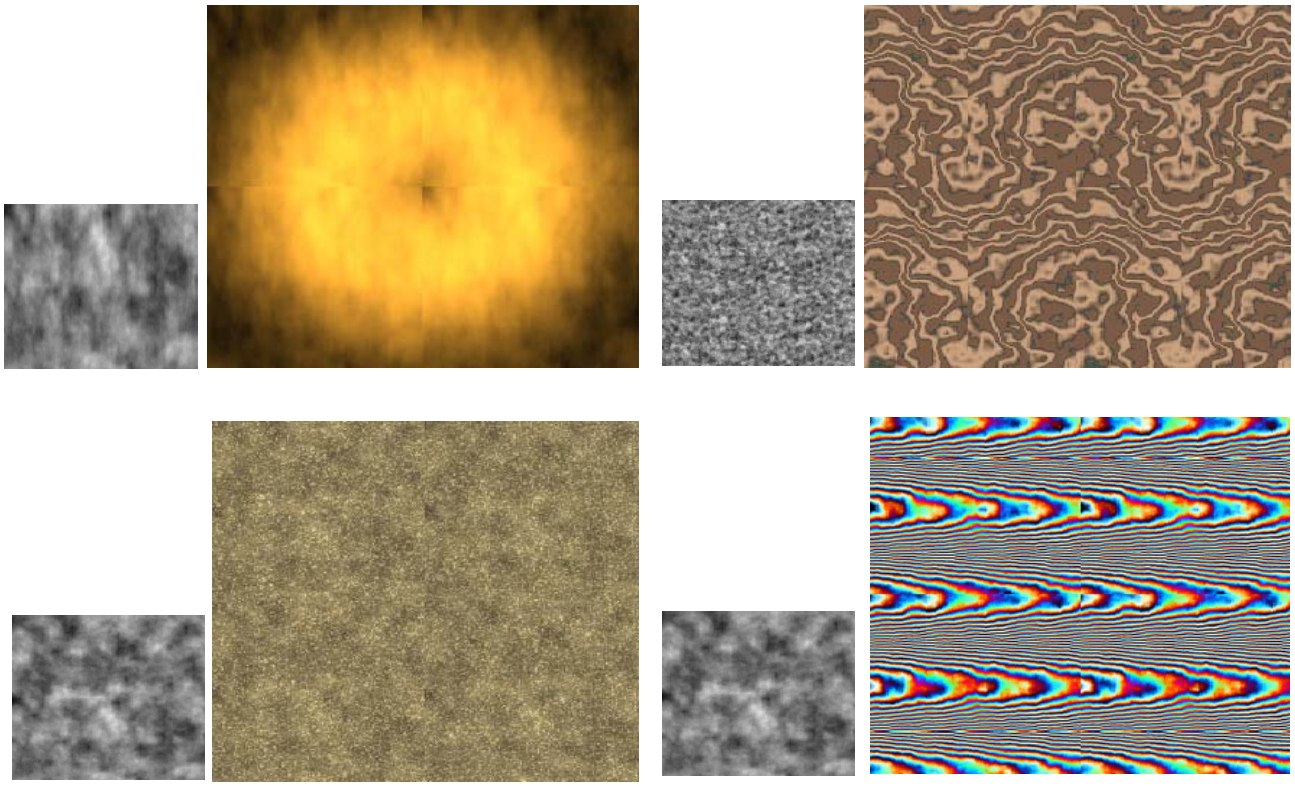


Figure 5. images created with Perlin noise tile. The left small gray images are Perlin noise tile, and the right big images are generated by some functions which use the map tiled by Perlin noise tile as inputs. You may find there are obviously sharp aliasing along the boundaries between two tiles.

Jinlian Du, was born on october 27.1972 in Chifeng, Innermongoli Republic of China. She received her M.Sc in system engeneering from Innermongoli University of Science and Technology in 1998 and her PH.D. in computer science from Dalian University of Technology in 2003. respectively, she is currently an associate professor in the Department of Computer Science at Beijing University of Technology, Beijing, Republic of china. Her research interests include realtime rendering, texture synthesis, physical-based modeling, gaming and complex data management.

Song Wang was born on Feberary 10.1986 in BaZhou, Hebei, Republic of China. He received his B.Sc. in computer science and thechnology from Northeastern University at Qinhuangdao in 2009. Now he is a graduate student of computer science and technology at Beijing University of technology. His research interests include texture synthesis, realtime rendering.

Xianhai Meng. was born on May 17, 1986 in Xiaoyi, Shanxi , Republic of China. He received his B.Sc in Software Engineering from East China Jiaotong University in 2010. Now he is studying for his B.S.c in Beijing University of Technology, Beijing, Republic of china. His research interests include physical-based modeling and gaming.

