Packing Square Tiles into One Texture

Philippe Decaudin* Fabrice Neyret*

GRAVIR/IMAG-INRIA , Grenoble, France

Abstract

This paper deals with the packing of square tiles of the same size into one texture. Texture size is constrained by the graphics hardware. In particular, width and height resolutions must be powers of two.
To cover the whole texture and avoid space loss, common schemes pack a number of tiles that is a power of two. We show that numbers of tiles like 5, 13, 17, 25, 34 and others can also be reached without wasting memory. To achieve this, our scheme takes advantage of texture rotation and the wrapping capability of texture-addressing, which gives a torus topology to the texture space.

Keywords
textures, tiling.

1. Introduction

Sets of square texture tiles of the same size are often used in Computer Graphics, especially for real-time 3D applications (VR, video games, etc.). For instance, such tiles are often assembled to map large surfaces like terrains. Animated textures for real-time applications are also often represented by a set of textures, one of them being picked at each frame. [Sta97, NC99, CSHD03, DN04] also use sets of tiles to achieve aperiodic texture mapping. These tiles can also be used as a basis to generate procedural textures by programming the graphics hardware [LN03].

It is often important to gather the tiles together in one large texture. This technique avoids performance penalty due to graphic context switching while rendering. It is also easier to manage one large texture instead of a set a small ones when managing the application resources (loading, saving, conversions, etc.). These arguments remain true even for non-square tiles. For instance, texture *atlases* are often packed in one texture as well [MYV93].

However graphic boards and graphic libraries impose constraints on texture sizes [SWN*03]:

- The width and height of textures must be power of two ($2^a \times 2^b$). Therefore, the ratio between the width and the height is also a power of two. Some recent graphic boards allow *non-power of two* texture sizes, but with restrictions limiting their use. For such "non-power of two" textures, MIP-mapping is not allowed, and not all compressed texture formats are available [Mic03].
- The width and height cannot exceed a maximum value. Common maximum value of today's hardware is 2048 or 4096. This restriction is another argument for not wasting space.

In the following we will only consider square tiles of same size, and simply refer to them as *tiles*.

These texture constraints make it difficult to pack an arbitrary number N of tiles in one texture without too much space loss. If N is a power of two, the tiles can fit into a constrained texture without space loss (see fig. 1). But if N is in-between these values, the space left empty in the texture

* e-mails: Firstname.Name@imag.fr
web: http://www-evasion.imag.fr/Publications/2004/DN04b/
http://www.antisphere.com

can be considerable. For instance, to pack 5 tiles, one might choose a texture that can contain up to 8 tiles, leaving 3 of them empty. It implies that 37% of the texture is unused. The gap between successive powers of two increases very fast, making the average space loss growing consequently with N.

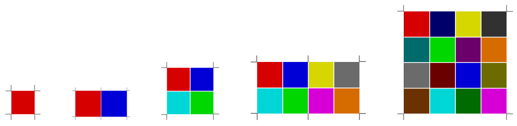


Figure 1: *Common packing of* 2^n *tiles.*

This short paper presents a scheme that allows N to take other values than powers of two tiles without space loss. Not all numbers can be reached, but we show that numbers like 5, 13, 17, 25, 34 and others can be reached. This reduces the distance between two reachable numbers, thus greatly reducing the space loss for other values of N.

Our scheme relies on two properties of texture-addressing:

- The texture has a torus topology (opposite borders cycle), because texture addresses wrap. This is done by setting the `GL_REPEAT` mode in OpenGL, or `D3DTADDRESS_WRAP` mode in DirectX.
- Rotations of the texture can be easily addressed. This is done by setting a texture matrix, or by modifying the texture coordinates of the mesh on which the texture is mapped.

Note that we do not use the capability of addressing stretched texture. Stretching the tiles can be a solution to fill the whole texture. But this is equivalent to leaving the tiles unstretched and having empty space.

Lot of work has been done on tilings and their applications [GS86]. Mathematicians have studied the problem of packing tiles, and especially square tiles [Fri98, Jen95]. Best known packing of an arbitrary number of tiles might result in complex scheme, as shown on figure 2 for 10 and 17 tiles, and space is still lost. But the wrapping capability to address textures changes the problem. A more suitable packing scheme for non-power of two numbers of tiles can be found. Using our scheme, 10 and 17 tiles can fit exactly into a square texture, as seen on figure 3*(e)* and 3*(g)*. This scheme is described in section 2.

Our discussion is focused on the number of tiles that can fit into a constrained texture, independently of their resolution. But the resolution of tiles and the texel resampling due to tiles rotation must also be taken into account to determine the final resolution of the texture. This is discussed in section 3.

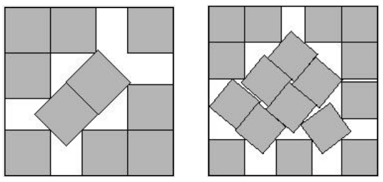


Figure 2: *Best known packing of 10 and 17 tiles without wrapping (from [Fri98]).*

2. Packing tiles

This section presents the principle used to build schemes for packing tiles into a texture other than the common power-of-two ones shown on figure 1. It also presents how to combine them to increase the count of reachable numbers of tiles, and therefore partially fill the gap between those numbers.

In the following, we will use the notation $\times n$ to denote a scheme that packs n tiles.

2.1. Common $\times 2$ and $\times n^2$ packing schemes

Figure 3*(a)left* shows the common way of packing two tiles in a texture. The result texture is not square, but constraint on ratio between width and height is enforced.

Nevertheless, two tiles can be packed into a *square* texture, thanks to wrapping and a 45° rotation. This scheme is shown on figure 3*(a)right*. Such a packing scheme can be useful to avoid the building of non-square textures if needed.

This $\times 2$ scheme can be combined with another $\times n$ schemes to pack $2n$ tiles. *E.g.,* for $n = 2$, we get the $\times 4$ scheme of figure 3*(b)*.

Regular subdivisions of a square texture result in n^2 tiles. For instance, figure 3*(c)* shows the trivial $\times 9$ scheme.

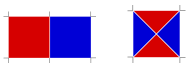
2.2. $\times(n^2 + k^2)$ packing scheme

Let consider the packing of *five* tiles.

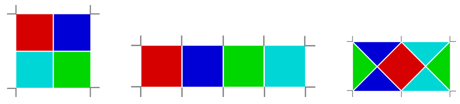
This can be obtained by subdividing a square by drawing lines from the center of a border to an opposite corner as shown on figure 3*(d)*. Considering wrapping, we get five regions, and it can easily be proven that they are squares and have equal sizes.

By combining this scheme with the previous $\times 2$ scheme, one can pack $2^p 5$ tiles. *E.g.,* scheme $\times 10$ (*i.e.* $2^1 5$) is shown on figure 3*(e)*. Scheme 3*(e)left* (resp. 3*(e)right*) is obtained by combining schemes 3*(d)* and 3*(a)left* (resp. 3*(a)right*).

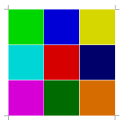
By combining $\times 5$ with itself and with $\times 2$, we can pack $2^{p_2} 5^{p_5}$ tiles, where p_2 and p_5 are any positive integers. Therefore 25, 50, 100, 125,... tiles can be packed lossless.



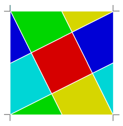
(a) $\times 2$: packing 2 tiles



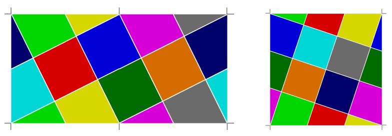
(b) $\times 4$: packing 4 tiles



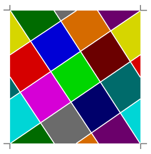
(c) $\times 9$: packing 9 tiles



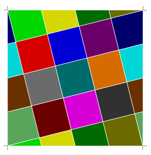
(d) $\times 5$: packing 5 tiles



(e) $\times 10$: packing 10 tiles



(f) $\times 13$: packing 13 tiles



(g) $\times 17$: packing 17 tiles

Figure 3: *Packing schemes (resolution not conserved).*

Generalization:

To build a $\times(n^2+k^2)$ packing scheme where n and k are two positive integers with $k \leq n$, we start from a $n \times n$ square. We subdivide it with two sets of parallel slanted lines (figure 4): one of slope $\frac{n}{k}$ going through points $(i,0)$, and one of slope $-\frac{k}{n}$ going through points $(0,i)$, where i is an integer.

We prove that this construction results in n^2+k^2 tiles:

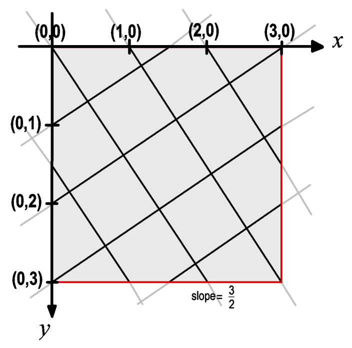
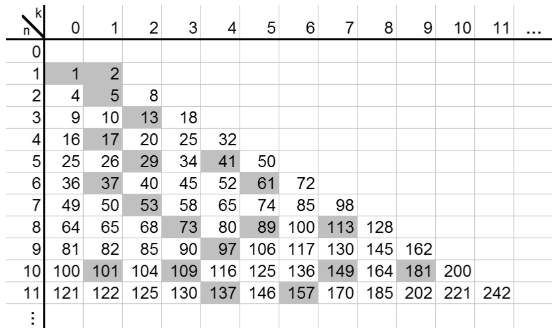


Figure 4: *Construction of the n^2+k^2 scheme (here, $n=3$ and $k=2$).*

Consider the line of slope $\frac{n}{k}$ which go through the upper-left corner of the texture. Its angle α with the x-axis is such as $\tan\alpha = \frac{k}{n}$. The construction projects the n subdivisions of a texture border onto this line, which shrinks them by $\cos\alpha$. So, in relation to the initial $n \times n$ non-slanted grid, each tile will be shrinked by $\cos^2\alpha$. Thus, there are $1/\cos^2\alpha$ more tiles in the slanted grid than in the initial one, with $\tilde{n}^2 = n^2/\cos^2\alpha$.
Since $1/\cos^2\alpha = 1+\tan^2\alpha$, $\tilde{n}^2 = n^2(1+(\frac{k}{n})^2) = n^2+k^2$, we have built n^2+k^2 tiles.

This defines schemes to pack n^2+k^2 tiles, with $k \leq n$. This provides a set $\{m_i\}$ of directly reachable numbers. Then, by combining these schemes, any tile number N of the form $N = \prod_i {m_i}^{p_i}$, where $\{p_i\}$ are any positive integers, can be reached.



n \ k	0	1	2	3	4	5	6	7	8	9	10	11	...
0													
1	**1**	**2**											
2	4	**5**	8										
3	9	10	**13**	18									
4	16	**17**	20	25	32								
5	25	26	**29**	34	**41**	50							
6	36	**37**	40	45	52	**61**	72						
7	49	50	**53**	58	65	74	85	98					
8	64	65	68	**73**	80	**89**	100	**113**	128				
9	81	82	85	90	**97**	106	117	130	145	162			
10	100	**101**	104	**109**	116	125	136	**149**	164	**181**	200		
11	121	122	125	130	**137**	146	**157**	170	185	202	221	242	
⋮													

Figure 5: *Values of N for all the possible schemes n^2+k^2. Remarkable values of N are in grey: they are prime numbers and cannot be decompose in product of other values, and then they cannot be obtained by combining other schemes.*

Figure 5 shows the values of N that can be reached by applying the n^2+k^2 scheme.

3. Discussion

The resolution of tiles and the texel resampling due to the tiles rotation must also be taken into account to determine the final resolution of the texture.

Tiles rotation and texels resampling:

In the $\times(n^2+k^2)$ scheme with $k>0$, tiles are stored rotated into the texture. Thus, little distortion happens when the tiles are mapped unrotated. In practice, if the sampling of the rotated tiles is done properly when they are stored, nearly no data is lost, and the distortion remains quite unnoticeable when they are mapped. This mapping operation is performed accurately by current graphics hardware.

Determining the texture resolution:

The final resolution of the texture is constrained. Width and height must be powers of two. Let's consider that the resolution of tiles is given. We might want to store them with the best possible resolution within the texture.

First, a scheme is chosen to pack the number of tiles with as less space loss as possible. Since tiles resolution is given, this results in a texture size that is not necessarily a power of two. Therefore, the texture must be uniformly scaled to the nearest power-of-two size, or the upper if one doesn't want that tiles size is scaled down. This can be considered an extra space loss, but this also occurs with common packing schemes.

4. Conclusions and future work

We have shown that N equal square tiles can be packed into a texture without empty space between tiles, if N can be decompose into products $\prod_i {m_i}^{p_i}$ where p_i is positive integer, and m_i is integer of the form $m_i = n_i^2 + k_i^2$ (where n_i and k_i are two positive integers).

Note that this is quite similar to the prime factor decomposition of N. Our decomposition is slightly constrainted by the fact that *some* primes can only have even power (*e.g.,* 3, 7, 11, 19, 23). Values of N which do not fit this decomposition must be rounded to the next allowed value.

This allows many more possible schemes to pack tiles than with the common power-of-two schemes, and it is easier to pick a good packing scheme that avoids space loss.

Schemes based on the same idea could also be defined for 3D textures. As wrapping is also available for those textures, cubic tiles could then be efficiently packed into a constrained 3D texture. And the amount of space saved could be consequential.

Acknowledgment
We wish to thank Adrien Treuille for rereading this paper.

References

[CSHD03] COHEN M., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Transaction on Graphics (SIGGRAPH '03 Conf. Proceedings) 22*, 3 (2003), 287–294. 1

[DN04] DECAUDIN P., NEYRET F.: Rendering forest scenes in real-time. In *Proceedings of Eurographics Symposium on Rendering* (2004). `http://www-evasion.imag.fr/Publications/2004/DN04/`. 1

[Fri98] FRIEDMAN E.: Packing unit squares in squares: A survey and new results. *The Electronic Journal of Combinatorics DS7* (Mar. 1998), 1–24. `http://www.combinatorics.org/Surveys/ds7.html`. 2

[GS86] GRÜNBAUM B., SHEPHARD G. C.: *Tilings and patterns*. W. H. Freeman & Co., 1986. ISBN 0716711931. 2

[Jen95] JENNINGS D.: On packings of squares and rectangles. *Discrete Mathematics 138*, 1-3 (1995), 293–300. 2

[LN03] LEFEBVRE S., NEYRET F.: Pattern based procedural textures. In *Proceedings of ACM SIGGRAPH 2003 Symposium on Interactive 3D Graphics* (2003), ACM, ACM Press. `http://www-evasion.imag.fr/Publications/2003/LN03/`. 1

[Mic03] MICROSOFT: DirectX 9 reference manual, `D3DPTEXTURECAPS_NONPOW2CONDITIONAL` reference, 2003. `http://msdn.microsoft.com/directx`. 1

[MYV93] MAILLOT J., YAHIA H., VERROUST A.: Interactive texture mapping. In *SIGGRAPH '93 Conference Proceedings* (1993), pp. 27–34. 1

[NC99] NEYRET F., CANI M.-P.: Pattern-based texturing revisited. In *SIGGRAPH '99 Conference Proceedings* (Aug. 1999), ACM SIGGRAPH, Addison Wesley, pp. 235–242. `http://www-imagis.imag.fr/Membres/Fabrice.Neyret/publis/SIG99/`. 1

[Sta97] STAM J.: *Aperiodic texture mapping*. Research Rep. R046, European Research Consortium for Informatics and Mathematics (ERCIM), 1997. `http://www.ercim.org/publication/technical_reports/046-abstract.html`. 1

[SWN*03] SHREINER D., WOO M., NEIDER J., DAVIS T., OPENGL ARCHITECTURE REVIEW BOARD: *OpenGL Programming Guide (4th Edition), Chapter 9: Texture Mapping*. Addison Wesley, 2003. ISBN 0321173481. 1