rpTextures: Systematic Layering for Large Texture Generation

Austin E. MacKay   Jonathan D. Denning
Taylor University, Indiana USA

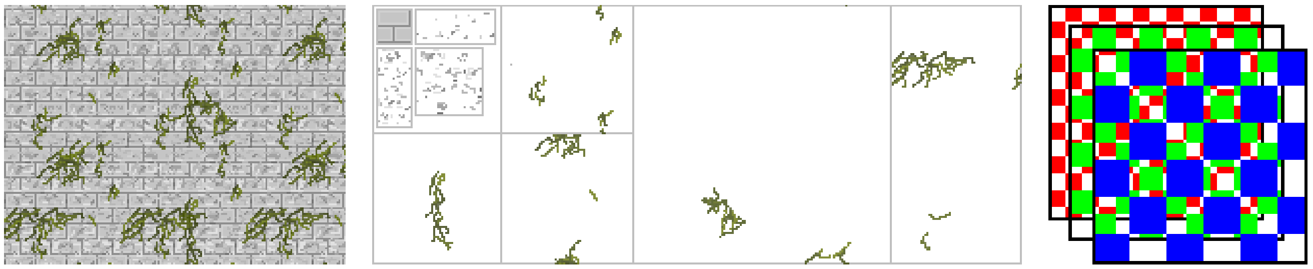

Figure 1: (left) Mossy Stone Brick texture, generated from eight layers, has a period of 2432×2432. The largest layer is 128×128. (middle) All eight layers packed into a single 320×128 graphic texture (borders added to show individual layers). (right) Tiled layers of sizes 5×5 (red), 7×7 (green), and 11×11 (blue) generate a texture with a period of 5·7·11=385 in width and height.

Abstract

We used systematic layering of variously sized layers to quickly create large, seemingly non-repeating textures. This leads to significantly more control for artists to create large visual scenes without the need for large teams to create massive textures. Our method maintains the visual appeal of seamless and non-repeating design, as well as uses little memory and fast rendering.

Introduction

Compelling video game textures are difficult to create. Small, tiled textures are quick to produce and render, but the repeated appearance can remove the player from their immersion experience. As game worlds become larger and often more realistic, the textures need to grow in resolution and fidelity to compensate. Currently textures are created by methods of generating filtered noise or by having an artist go through and paint every pixel. Painting large textures is time consuming, requires nontrivial compression, and novel texture management systems. Methods for more automated texture generation include filtering blue noise [Stam 1997], aperiodic tiling [Parzer 2013], and texture compression (epitome) [Wang et al. 2008], but these methods are difficult to generalize across a variety of categories of texture structures.

We present a method of creating large-scale, seamless, and complex textures that is artist friendly (painted using conventional methods) and generalizes across texture categories. Our method involves a systematic layering of variously-sized textures which balances randomization and easy customization, and is fast to render. We also report on the useful design principles. Our work extends and formalizes the work of Alex Walker [2011].

Technical Details and Implementation

Repetitious textures have visual features that appear periodically. A tiled texture has a fixed period based on its size. The key insight to our work is in creating layers of different sizes that repeat at different periods. By choosing the sizes of the layers, we can control the effect of repeating to generate noisy textures (e.g., grass, dirt, sand) or structures textures (e.g., brick, tiles). The final

SIGGRAPH '17 Posters, July 30–August 03, 2017, Los Angeles, CA, USA
ACM 978-1-4503-5015-0/17/07
10.1145/3102163.3102219

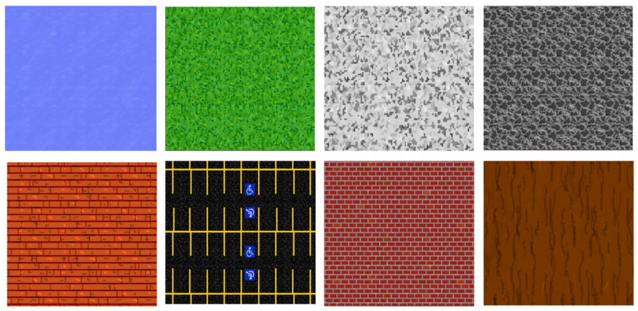

Figure 2: Example textures: water, grass, gravel, stone, wood planks, parking lot, brick wall, tree bark.

texture repeats at a size that is the least common multiple of the layer sizes. Choosing layer sizes that are relatively prime will result with the texture repeating only at the product of the sizes. The size of the non-repeated texture grows roughly exponentially with the number of layers. For example, layers of widths 5, 7, and 11 will generate a non-repeating texture of width 385 (Fig. 1). Adding a fourth layer of width 13 increases generated texture width to 5005. Prime sizes are ideal for generating textures with very little structure. To generate structured textures, we chose sizes that have a greatest common divisor proportional to the size of the intended structure. For example, a brick 8 pixels wide would use layers of width 5·8=40, 7·8=56, and 8·11=88 generates a texture of width 3080. Adding a fourth layer of width 8·13=104 generates a texture of width 40040. Using layers with sizes that are multiples of relatively prime numbers, artists can create massive, random textures using conventional methods.

We created a prototype web application using WebGL to test how expressive our method is for creating textures. The shader produces the final color by covering or adjusting HSV values for each layer sampled at the texture coordinate mod the layers size. We asked an artist to create textures from many different categories, such as grass, dirt, bricks, and stone (Fig. 2).

Current and Future Work

We present a range of low-resolution texture types and discuss design principles and strategies for creating rpTextures for video games. For future work, we plan to run studies to determine general rules for creating textures that avoid repetition, and we plan to compare the performance of our method against methods common in industry. Currently, we focus on using textures for diffuse reflection of surface. We believe this work can easily extend to control other surface properties, such as specularity, emission, normal perturbation, and displacement. We believe that this can also extend this method to other domains, such as audio.

References

PARZER, S. 2013. Irrational Image Generator. Diploma thesis (Vienna Univ. Technol., Vienna).

STAM, J. 1997. Aperiodic Texture Mapping. European Research Consortium for Informatics and Mathematics.

WALKER, A. 2011. The Cicada Principle and Why It Matters to Web Designers. www.sitepoint.com/the-cicada-principle-and-why-it-matters-to-web-designers/.

WANG, H., WEXLER, Y., OFEK, E., HOPPE, H. 2008. Factoring repeated content within and among images. *ACM TOG* 27, 3.

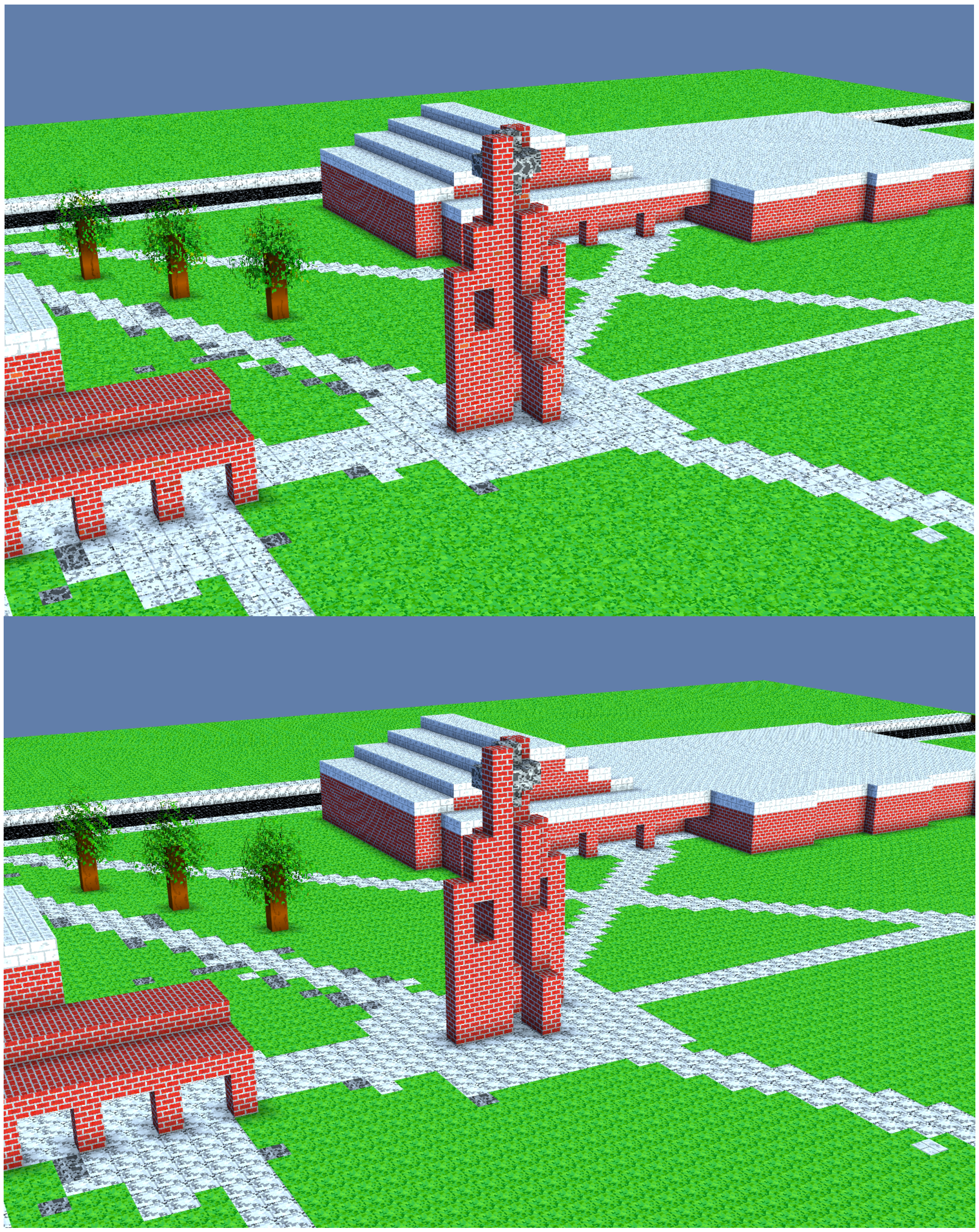

Figure 3: Large 3D scenes using rpTextures (top) and 16x16 tiled textures (bottom). Although the textures in both images are scaled the same, rpTextures avoid the repetition seen using tiled textures.