Wang Cubes for Video Synthesis and Geometry Placement (posters_0150)

Peter G. Sibley* Brown University
Philip Montgomery† Brown University
G. Elisabeta Marai‡ Brown University

Abstract

We present an extension of Cohen's Wang Tiles to three dimensions:*Wang Cubes*. Cubes are filled with video or Poisson distributed points to perform realtime video synthesis or geometry placement.

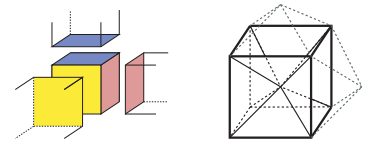
1 Motivation

Video synthesis from a sample is useful for generating dynamic backgrounds for games or special effects but costly in terms of storage and runtime. Randomly positioning non-overlapping 3D geometry is useful for simulations and games but also costly. We propose Wang Cubes where we only store 32 cubes and generate, at runtime, large amounts of synthesized video, or Poisson distributed points.

2 Methods

Cohen et al. [2003] introduced a fast and simple stochastic algorithm to generate an aperiodic tiling of the plane with as few as eight Wang Tiles (oriented squares with color associated edges). Cohen et al. used these tilings for texture synthesis and 2D geometry placement.

We extend these applications to the 3D case, where cubes with colored faces replace tiles. 32 cubes are sufficient to tile space. The extended tiling algorithm iterates through the space, placing a cube at each point. To select a cube, we pick a random cube whose top, left, and front faces match the colors of the three neighbors (Figure 1a). The 32 cubes contain either Poisson ball distributed points or video data and are tiled at runtime to generate large stretches of 3D geometry or video sequences.



(a) (b)

Figure 1: Wang Cubes. (a) Aligning face colors. (b) Assembling six octahedra to form a cube.

We use dart-throwing to fill each cube with Poisson distributed points. Several iterations of Lloyd's relaxation are applied to prevent points near boundaries from violating the minimum distance constraint in tiling. To fill the cubes with video data, we cut six octahedra from the video stream and stitch them together. Then, the result is trimmed into a cube (Figure 1b). Each original octahedron is associated with a face color. A *xy* plane in this cube corresponds to a frame of video and the *z* axis corresponds to time.

Stitching the octahedra together requires finding a minimal cutting surface between the two neighboring octahedra. We use the Kwatra et al. [2003] approach, by formulating this as a min-cut graph problem and solving it via the Ford-Fulkerson max flow algorithm.

*e-mail: pgs@cs.brown.edu
†e-mail: pgm@cs.brown.edu
‡e-mail: gem@cs.brown.edu

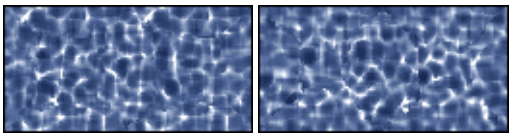


Figure 2: Vertical slices from two tiled Wang cubes. Note that the middle seam of each frames is invisible.

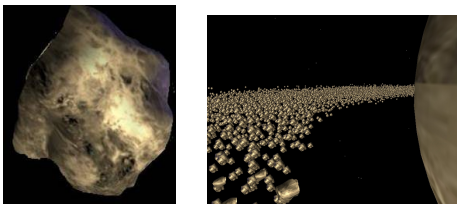


Figure 3: (a) Single asteroid model. (b)Saturn Asteroid Belt, 5959 asteroid instances placed using tiling of 3972 cubes each with 15 Poisson distributed points. Note, this took only seconds to generate.

3 Results

We constructed a cube set (64^3 voxels per cube) from a video of simulated shallow pool caustics. Two vertical slices through two cubes tiled horizontally are show in Figure 2. Note how the vertical seam in the middle of each frame is invisible. In order to keep our computation feasible, we constrained the cuts to lie near the intersecting triangles of the octahedra. We have noticed temporal artifacts in the videos, a growing and shrinking square-discontinuity. We believe these are caused by constrained cuts and small cube sizes.

As a geometry placement application, we modeled the asteroid belt of Saturn with 5958 astroids constructed from 3972 tiled cubes with 15 points in each cube. The asteroids are placed according to a Poisson distribution in this large area (Figure 3b). Using cubes only took 15 minutes to precompute, and under 20 seconds to tile. Note that filling the same region with dart throwing is simply infeasible. These tests were performed on an AMD Athlon XP 1800 with 512MB of memory.

4 Future Work

For video synthesis, we restricted the space searched for a min-cut surface, sacrificing quality of the cut for faster execution. Implementing known randomized max-flow algorithms to approximate the cut could yield much lower preprocessing, which would allow for less constrained cuts and eliminate temporal artifacts.

References

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3, 287–294.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003 22*, 3 (July), 277–286.

Wang Cubes for Fast Geometry Placement & Video Synthesis

Peter G. Sibley†, Philip Mongomery, G. Elisabeta Marai
Brown University

BROWN

1. Abstract

We present an extension of Cohen's Wang Tiles to three dimensions: Wang Cubes. Cubes are filled with video or Poisson distributed points to perform realtime video synthesis or geometry placement. Video synthesis from a sample is useful for generating dynamic backgrounds for games or special effects but costly in terms of storage and runtime. Randomly positioning non-overlapping 3D geometry is useful for simulations and games but also costly. We propose Wang Cubes where we only store 32 cubes and generate, at runtime, large amounts of synthesized video, or Poisson distributed geometry

2. Methods

Cohen et al. introduced a fast and simple stochastic algorithm to generate an aperiodic tiling of the plane with as few as eight Wang Tiles (oriented squares with color associated edges). Cohen et al. used these tilings for texture synthesis and 2D geometry placement.

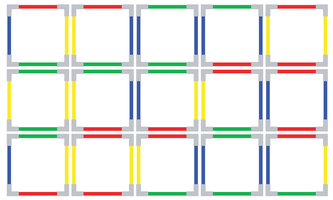


Figure 1: A valid tiling of Wang Tiles , from Cohen et al.

We extend these applications to the 3D case, where cubes with colored faces replace tiles. 32 cubes are sufficient to tile space. The extended tiling algorithm iterates through the space, placing a cube at each point (Figure 2). The 32 cubes contain either Poisson ball distributed points or video data and are tiled at runtime to generate large stretches of 3D geometry or video sequences.

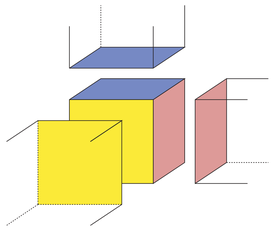


Figure 2: Aligning face colors of Wang Cubes

To place geometry data, we use dart-throwing to fill each cube with Poisson distributed points. Several iterations of Lloyd's relaxation are applied to prevent points near boundaries from violating the minimum distance constraint in tiling.

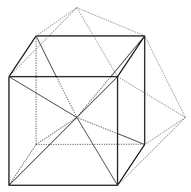


Figure 3: Assembling six octahedra of video to form one cube.

To fill the cubes with video data, we cut six octahedra from the video stream and stitch them together through the graph cut method of Kwatra et al. Then, the result is trimmed into a cube (Figure 3). Each original octahedron is associated with a face color. A xy plane in this cube corresponds to a frame of video and the z axis corresponds to time (Figure 4).

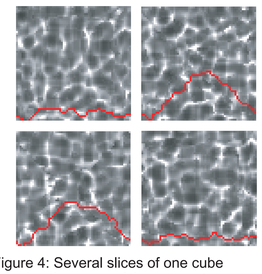


Figure 4: Several slices of one cube showing the seam of the bottom tetrahedron.

†{pgs,pgm,gem}@cs.brown.edu

3. Geometry Results

Figure 6: Single Asteroid.

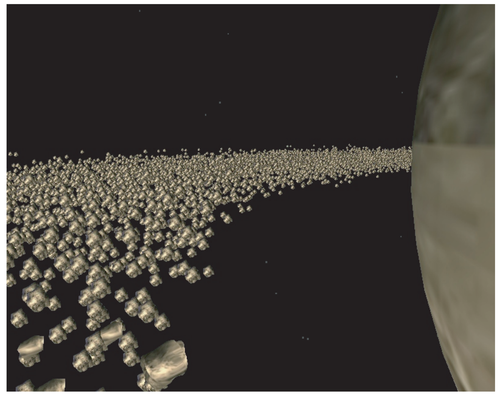


Figure 7: Saturn Asteroid belt, 5959 asteroid instances placed using tiling of 3972 cubes each with 15 Poisson distributed points. Note, this took only seconds to generate. Filling the same region with dart throwing is simply infeasible.

As a geometry placement application, we modeled the asteroid belt of Saturn with 5958 asteroids (Figure 6) constructed from 3972 tiled cubes with 15 points in each cube. The asteroids are placed according to a Poisson distribution in this large area (Figures 7 and 8). It only took 15 minutes to precompute the cubes, and under 20 seconds to tile them. Note that filling the same region with dart throwing is simply infeasible. Teapot geometry and sheep billboard distributions are shown in Figures 9 and 11. These tests were performed on an AMD Athlon XP 1800 with 512MB of memory.

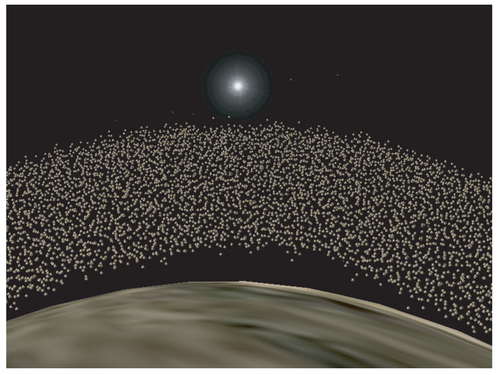


Figure 8: An overhead of the Saturn Asteroid belt.

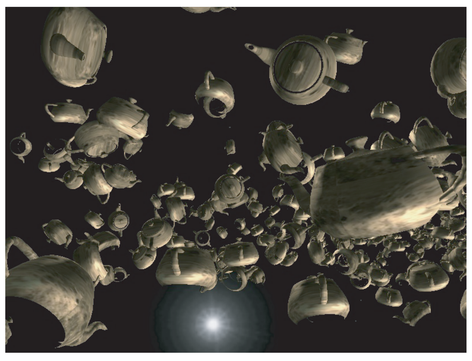


Figure 9: An infinite teapot field. Note the teapot instances are neither colliding nor have a regular position pattern.

4. Video Results

We constructed a cube set (64*64*64 voxels per cube) from a video of simulated shallow pool caustics. Three vertical slices through two cubes tiled horizontally are shown in Figure 10. Note how the vertical seam in the middle of each frame is invisible. An infinite caustics pool (both in space and time) could be generated in this manner.

In order to keep our computation feasible, we constrained the cuts to lie near the intersecting triangles of the octahedra. We have noticed temporal artifacts in the videos, a growing and shrinking square-discontinuity. We believe these are caused by constrained cuts and small cube sizes.

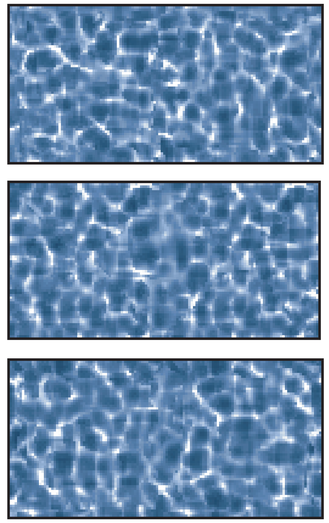


Figure 10: Vertical slices from two tiled Wang cubes. Note that the vertical middle seam of each frame is invisible

5. Discussion and Future Work

For video synthesis, we restricted the space searched for a min-cut surface, sacrificing quality of the cut for faster execution. Also, because of computational constraints we could only use quite small cube sizes (64*64*64). Implementing known randomized max-flow algorithms to approximate the cut could yield much lower preprocessing, which would allow for less constrained cuts and eliminate temporal artifacts. Incorporating newer texture synthesis techniques could produce better quality cubes.

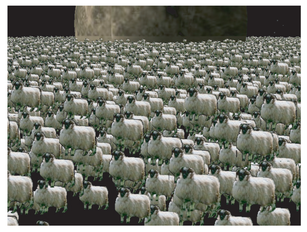


Figure 11: A sheep belt instead of an asteroid belt.

References

Cohen, M.F., Shade, J.,Hiller,S., and Deussen,O. 2003.
Wang tiles for image and texture generation. *ACM Trans. Graph.* 22,3,287-294.
Kwatra,V., Schödl,A.,Essa,I. Turk,G., and Bobick,A. 2003.
Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. 22,3,277-286.*

SIGGRAPH2004