Texturing: Procedural Texture Generation

Outline

- Perlin noise
- Texture synthesis
Noise as a Texture Generator

- Easiest texture to make: Random values for texels
  - $\text{noise}(x, y) = \text{random}()$
- If $\text{random}()$ has limited range (e.g., $[0, 1]$), can control maximum value via 
  \textbf{amplitude}
  - $a \times \text{noise}(x, y)$
- But the results usually aren’t very exciting visually

3-D Noise (aka “Perlin” noise)

- 3-D or solid texture has value at every point $(x, y, z)$
- Simple solid texture generator is noise function on lattice:
  - $\text{noise}(x, y, z) = \text{random}()$
- For points in between, we need to interpolate
3-D Noise Interpolation

- $f(x, y, z)$ can be evaluated at non-lattice points with a straightforward extension of 2-D bilinear interpolation (to **trilinear interpolation**)
- Other interpolation methods (quadratic, cubic, etc.) also applicable
- All of these are approximations of smoothing filters such as Gaussians

![3-D Noise Interpolation Diagram](image)

3-D Noise Texturing: Examples

- Original object
- Noise with trilinear smoothing
- Triquadratic noise

![3-D Noise Texturing Examples](image)
Noise Frequency

• By selecting larger spaces between lattice points, we are increasing the magnification of the noise texture and hence reducing its frequency.

Original noise
Smoothed noise

Smoothed noise at different magnification levels

turb(x, y, z) = Σ a_i * noise_i(x, y, z)

Typically:
- Magnification doubles at each level (octave)
- Amplitude drops by half (this is adjustable)
3-D Noise Texturing: Example

- Smoothed noise only
- Turbulence (8 octaves)

Marbling

- Use turbulence to perturb sine function via phase shift
- \( \text{marble}(x, y, z) = \sin(f \cdot (x + a \cdot \text{turb}(x, y, z))) \)
  where
  - \( f \) = marble pattern frequency
  - \( a \) = marble noise amplitude
Perlin Noise: Applications

- 2-D noise:
  - Traditional “wrappable” textures; good for:
    - Clouds
    - Water
  - Height maps—e.g., fractal terrain
- 3-D noise
  - Solid textures such as marble, wood
- 4-D noise
  - Animated solid textures

Texture Synthesis

- Suppose we have a sample of a natural pattern with which we would like to texture an object
  - Other applications: In-painting—filling in unwanted/missing sections of textures (aka images)
- Easy solutions that aren’t good enough:
  - Just stretch what we have over entire object
    - Resolution too low
  - Use “repeat” wrapping mode
    - Border artifacts too jarring, pattern too repetitious
- Idea: Try to replicate visual characteristics of texture sample without duplicating it exactly in order to “grow” what we have
Image Quilting (Efros & Freeman, SIGGRAPH 2001)

• Idea: Make new texture with series of “quilt squares” taken from sample texture
Image Quilting algorithm

1. Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap)
2. For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block.
3. Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. Repeat.

Image Quilting: Details

- Block size should be “big enough to capture the relevant structures in the texture, but small enough so that the interaction between these structures is left up to the algorithm.”
- Overlap typically 1/6 side length of block
- Error is sum of squared differences between pixel values
- Error tolerance = All block candidates with error within 0.1 times error of best candidate
Image Quilting: Examples

Minimal error boundary cut

overlapping blocks \[ \rightarrow \] vertical boundary

\begin{align*}
\text{overlap error} & \quad \rightarrow \\
2 & \quad \rightarrow \\
\text{min. error boundary}
\end{align*}

Use Dijkstra's algorithm to find least-cost path across "error surface"
Texture Transfer

- An additional constraint allows direct application of textures to objects in images
- Define a **correspondence map** with an image the same size as the one we are synthesizing
- **Key idea**: The quilt square we are laying down must not only “fit” via error tolerance with neighbors, but it must also be similar to correspondence map at same location
- Modification to original algorithm is that new error is **blend** of old error and correspondence map error:
  - Error = $\alpha$ * block matching error + $(1 - \alpha)$ * squared difference between correspondence map and quilt square at this location

Texture Transfer: Example

**Input image**

**Correspondence map**

**Standard quilting texture synthesis**

**Quilting texture synthesis with texture transfer**
Texture Transfer

Example texture

One last example

Correspondence map
courtesy of A. Efros