Rotationally Invariant Textures

Background

- Also known as isotropic toroidal texture patterns
- Needed for the texture tiles so that a viewer cannot perceive sharp edges
- Basic building blocks for creating large aperiodic textures

Properties

- All textures should be square and of the same size, in both height and width
- The textures should be of same brightness and contrast
- The textures should contain only one type of terrain (isotropic)
- Textures representing the same terrain type should fit against each other with no visible seam
- Textures should be at the same scale, that is, comparable features should appear to be of the same size
- From here on, we'll call such textures as *tiles*

Building tiles

- Consider an isotropic image (tree_subimage.jpg, extracted from tree.jpg)
- Input provided by an image of uniform terrain type
- Input image should be, preferably, larger than the output size desired
- Must specify the size of the desired texture tile, preferably square
- Algorithm to build the tiles
  - Extract the base pattern from the input image
    - Base pattern is the sub-image of the specified size extracted from the center of the given image (tree_sub.jpg)
    - Center of the sub-image and given image are coincidental
    - Extraction is simply by selecting the sub-image from the given image, as per the desired width and height of output tile
  - Remove the straight line seams when the extracted tiles are rendered
    - Cut part of left and bottom edges of the extracted tile
    - Fill the cut part with new pixels that will achieve the blend
    - Cutting of the edges should not proceed along straight lines but along wavy lines so that no straight line seams are visible
    - Achieved by *minimum resistance cut line*
  - Minimum resistance cut line
    - Line along an edge formed by randomly traversing the pixels along the edge under the constraint that the line is continuous in 8-neighborhood of each pixel
    - Used to remove some areas along the edges
    - Area along the left edge is then filled to ensure smooth blending with the right part of the tile
    - Similarly, area along the bottom edge is filled to ensure smooth blending with the top edge
    - Resulting tile is rotationally invariant as it blends from top to bottom and right to left
  - Drawing minimum resistance cut line

        - Draw a wavy line that will cover some area along a specified direction
        - Drawing line along the left edge
            - Start at bottom left corner and grow the line towards top right corner
            - At each pixel, look at the pixel's neighbors to find the pixel thatis closest to the current pixel by some specified criterion
            - Two possible criteria include Euclidean distance and distance in terms of luminance value
            - Look at the pixels in 8-neighborhood; but that can make the line go in *any* direction
            - Since we want to control the direction of the line's growth, we must constrain our search to a subset of pixels in the 8-neighborhood
            - The next pixel to be selected must be one of the three pixels in the 8-neighborhood that are towards top, top left, and left
            - Newly selected pixel becomes the current pixel in the next iteration and we continue to grow the line until we reach the top or right edge of the tile
            - Avoid long stretches of straight lines by limiting the number of pixels that can be traversed in horizontal or vertical direction
            - This parameter can be specified as a configurable to control how deep the cut line can go from the edge
            - A similar cut line is drawn by starting at the top left corner and growing towards bottom right
            - The second line stops to grow as soon as it crosses the first line
            - The first line fromintersection point to the top/right edge is then deleted
        - Drawing line along the bottom edge
            - The line along bottom edge follows the same logic as the one along the left edge
            - The first line is drawn similar to the one above
            - The main difference is that the scan for the neighboring pixel now goes anticlockwise starting at the pixel towards right
            - The second cut line is drawn from bottom right towards the top left and stops on intersecting with the first line
            - The first line from the intersection point to right/top edge is then cleared
    - Filling in the left and bottom edges
        - Fill in the area towards the left of the cut line by picking up the area that is just to the right of the extracted tile
        - The area used for filling in is taken from the part that was ignored, or thrown away when we extracted the smaller tile of required dimensions
        - This area is contiguous to the extracted texture's right edge which makes our job easy
        - Similarly, the area towards the bottom is filled in with pixels extracted from the top of extracted texture
    - Blending the pixels along cut line
        - There is now a sharp transition point given by the pixels on the cut lines along left and bottom edges
        - We can make this sharp transition blend smoothly by applying a weighted linear interpolation scheme
        - Blending is achieved by extending the texture on both sides of cut line – the original pixels from extracted pattern extended to the right of the line and the pixels from the overlay texture being extended to the left of the cut line
        - Weighting scheme
            - Blending along the left edge proceeds by scanning each pixel in every row to determine the location of cut line
            - At the cut line, assign equal weight to both pixels – in overlay and well as extracted texture
            - To the left of cut line, assign more weight to pixels from base texture and less weight to the overlay pixels
            - Revert the weights on the right side of cut line
            - Pixels along the bottom edge cut line are handled similarly

Building Multiple Instances of Texture Tiles of Same Type

- Multiple instances of tiles we created above will show the periodicity when rendered against each other
- Need to create textures that will
  - Tile against each other
  - Contain the same type of texture/terrain
  - Retain the other image characteristics such as color, intensity, and contrast as we go from one tile to the next
- Minimum resistance cut line
  - Create a minimum resistance cut line on all four sides of a tile-able texture that has been created
    - This ensures that the edges of the tile will match
  - Extract the middle part and fill it with the texture extracted from the original big tile from a randomly selected area
  - `<A HREF=http://www.cs.umsl.edu/˜sanjiv/classes/cs6420/images/lt.jpg>Example 1</A>`